Generative Representation Learning on Hyper-relational Knowledge Graphs via Masked Discrete Diffusion

Jaejun Lee [1] **Seheon Kim** [1] **Joyce Jiyoung Whang** [2]

Abstract

Hyper-relational knowledge graphs (HKGs) effectively represent complex facts. While inferring new knowledge in HKGs is a critical problem, current methods cast it as a simple link prediction, assuming that nearly all entities and relations within a fact are known, leaving only a single blank to be filled. However, this restricted assumption may not hold in real-world scenarios in which multiple, or even all, constituent components of a fact may be missing simultaneously. To bridge this gap, we introduce a task called fact generation: generating a valid hyper-relational fact from an arbitrarily masked query, i.e., completing a partially observed fact or generating a fact from scratch. We propose KREPE, the first generative representation learning method for HKGs that learns to model the probability distributions of missing components conditioned on the local fact components and global structure of HKGs via a masked discrete diffusion. KREPE models both the intra-fact dependencies by contextual message passing and inter-fact correlations by aggregating stochastically sampled contexts. KREPE seamlessly unifies link prediction and fact generation within a single training framework, achieving state-of-the-art performance on standard HKG link prediction benchmarks and outperforming LLM-based baselines in generating novel and correct facts.

1. Introduction

Knowledge graphs (KGs) structure human knowledge by representing facts as triplets. However, the simple triplet format is insufficient to capture the multifaceted nature of real-world facts. To address this limitation, knowledge bases such as Wikidata (Vrandečić & Krötzsch, 2014) and YAGO (Suchanek et al., 2007) extend the triplets into hyper-relational facts by adding auxiliary key-value pairs, known as qualifiers, forming hyper-relational knowledge graphs (HKGs). Since real-world HKGs are often incomplete, link prediction has been proposed as an HKG-completion task, where each problem is defined as predicting a single missing element in a fact, given all the other elements. As illustrated in Figure 1(a), a link prediction method outputs a ranked list of candidates for the target missing element, assuming that all the other entities and relations within the fact are known.

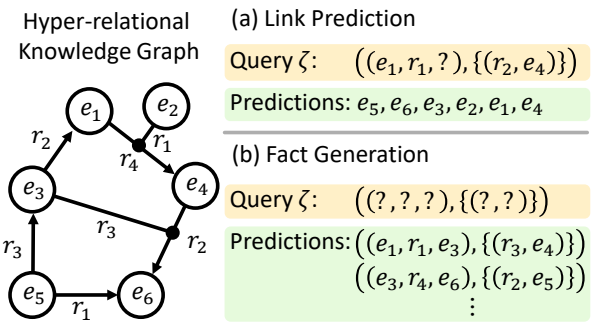

Figure 1. Comparison of link prediction and fact generation tasks on HKGs. Link prediction retrieves a single missing element by ranking candidates, whereas fact generation constructs valid facts from arbitrarily masked inputs, including fully masked ones.

Some link prediction methods assess the plausibility of each fact formed by filling the missing entry with a candidate component (Rosso et al., 2020; Guan et al., 2020), and other methods employ the transformer architecture to predict a missing component in an incomplete fact (Chung et al., 2023; Wang et al., 2021). However, these methods focus on processing facts individually and do not sufficiently consider the complex dependencies between facts within the HKG structure. While approaches combining GNN-based encoding with Transformer decoding have been proposed (Galkin et al., 2020; Luo et al., 2023), they use the graph structure in a limited capacity, for instance, by ignoring relations or qualifiers in updating representations. Recently, MAYPL (Lee & Whang, 2025) has been introduced to comprehensively encode the HKG structure into representation learning.

[1]School of Computing, [2]Department of AI Computing, KAIST, Daejeon, South Korea. Correspondence to: Joyce Jiyoung Whang <jjwhang@kaist.ac.kr>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

Despite these architectural advancements, a fundamental limitation persists across existing works: they are restricted to the task of link prediction, which is essentially a passive "fill-in-the-blank" problem relying on the rigid assumption of a single missing element. This assumption is inconsistent with real-world scenarios, where the extent of missing information in queries varies unpredictably. In practice, we are rarely presented with a nearly complete fact requiring a single component to be filled, e.g., $\big((h, r, ?), \{(k_1, v_1)\}\big)$. Instead, the challenge often lies in generating valid knowledge from highly incomplete observations. To accommodate the various levels of missing information in real-world queries, we propose a task named *Fact Generation*. Unlike link prediction, which infers a specific missing component based on an almost-complete query with a single mask, fact generation aims to construct a valid hyper-relational fact from an arbitrarily masked query. As shown in Figure 1(b), our task extends even to generating complete facts from a fully masked input $\big((?, ?, ?), \{(?, ?)\}\big)$.

In this work, we formulate fact generation as the task of modeling the conditional probability distribution of masked components given a query and the HKG structure. We present KREPE (Contextual H**K**G **REP**resentation Learning via Masked Discret**E** Diffusion), which leverages contextual message passing to compute entity and relation representations and employs masked discrete diffusion to estimate the underlying distribution of masked components. Specifically, our contextual message passing updates each component to encode its surrounding structure, i.e., its context within each fact, capturing intra-fact dependencies. Simultaneously, a bi-level noising strategy is employed to simulate diverse conditional generation scenarios for both the query and the observed structure. KREPE is trained under a masked discrete diffusion framework that treats generation as an iterative reconstruction process, thereby learning the joint conditional distribution of multiple masked components within a query. Since link prediction is a special case of fact generation involving a single masked component, KREPE seamlessly unifies entity prediction, relation prediction, and fact generation within a single representation learning framework. Our main contributions are summarized as follows:

- We propose *Fact Generation*, expanding the scope of HKG completion from passive ranking of candidates for single blanks to direct generation of missing facts.
- We present KREPE, the first generative representation learning method for HKGs incorporating masked discrete diffusion to explicitly model the joint conditional probability distributions of missing components in a query. Our code and datasets are available at `https://github.com/bdi-lab/KREPE`.
- KREPE achieves state-of-the-art performance on three standard link prediction benchmarks, demonstrating that our generative objective enhances the quality of learned representations.
- KREPE significantly outperforms a diverse set of competitive LLM-based baselines in fact generation, validating that modeling the probability distributions conditioned on both inter-fact and intra-fact dependencies is effective for generating novel and correct facts.

2. Related Work

HKG Representation Learning Most HKG representation learning methods (Wang et al., 2021; Lee & Whang, 2025) are tailored for link prediction. Scoring-based methods such as HINGE (Rosso et al., 2020) and NeuInfer (Guan et al., 2020) rank all possible candidates for a single missing position. These approaches are computationally intractable for fact generation involving multiple missing components, as the combinatorial search space grows exponentially. StarE (Galkin et al., 2020) and Hy-Transformer (Yu & Yang, 2021) encode an incomplete fact into a vector to predict a single target, and HyperFormer (Hu et al., 2023) requires the neighbor information of observed entities, restricting them to single-element inference. Although HAHE (Luo et al., 2023) introduces a multi-position prediction task, it is constrained by predefined mask patterns, allowing at most one missing component per triplet or qualifier pair. As such, existing methods are limited to predicting single missing elements or to handling fixed, limited patterns of masks and lack the capability to support knowledge generation in real-world scenarios with varying mask patterns.

LLMs for KGs. Although focused on KGs rather than HKGs, efforts have been made to adapt large language models (LLMs) for KG completion, particularly for link prediction (Xu et al., 2024; Guo et al., 2024; Jiang et al., 2024). A common approach to adapting LLMs to KG link prediction is a *retrieve-then-rerank* strategy (Liu et al., 2025b;a; Lin et al., 2025), where a pre-trained KG model retrieves candidates for a single-blank query, which are subsequently refined by an LLM. For instance, KICGPT (Wei et al., 2023) utilizes RotatE (Sun et al., 2019) to get top-k candidates for a given query and employs an LLM to re-rank them. However, these methods require a backbone KG link prediction model to provide candidates, and thus cannot be directly applied to fact generation, as existing models for KGs cannot handle multi-masked queries. While MuKDC (Li et al., 2024a) synthesizes triplets using LLMs, it is a data augmentation technique to mitigate sparsity in few-shot settings, rather than a generative model for inferring novel facts during inference. Obviously, we also distinguish our work from text-to-KG construction methods (Mo et al., 2025; Luo et al., 2024); while they aim to convert unstructured text corpora into structured knowledge, our objective is to generate new knowledge, which requires inference capability on HKGs fundamentally distinct from simple information extraction.

Diffusion Models for (H)KGs Denoising diffusion probabilistic models (Ho et al., 2020) have become a dominant regime for generative modeling on continuous data, such as images. To extend them to discrete data, discrete diffusion methods (Austin et al., 2021), such as masked discrete diffusion, have been developed for tasks like text generation. While several diffusion-based methods have been proposed in the KG domain (Long et al., 2024b;a), they are conditional denoising diffusion models that generate a continuous embedding vector for a missing entity conditioned on a query, rather than generating the fact itself. DiffusionE (Cao et al., 2024) utilizes heat diffusion on Riemannian manifolds to reformulate message passing for link prediction, which fundamentally differs from the generative denoising diffusion. HDiff (Luo et al., 2025) applies denoising diffusion to HKGs to refine continuous fact representations for link prediction. While existing methods use diffusion merely to refine continuous embeddings for candidate ranking, KREPE leverages the diffusion process to model the joint probability distribution of missing components in a query, enabling the generation of facts from arbitrarily masked queries.

Triple Set Prediction Triple set prediction (Zhang et al., 2024) is the task of predicting missing triplets in which all elements are unknown. However, this task is defined for KGs and does not account for scenarios where partial information about missing triplets is available. In contrast, fact generation is defined for HKGs and aims to predict valid facts from arbitrarily masked queries, including fully masked ones. Thus, triple set prediction is a special case of fact generation. Notably, GPHT (Zhang et al., 2024), a model for triple set prediction, cannot be directly applied to fact generation. GPHT reduces the number of candidate triplets and then ranks the remaining candidate triplets for prediction. However, for hyper-relational facts, the candidate space grows exponentially with the number of qualifiers, making such ranking-based prediction impractical.

3. Problem Definition

We formally define HKGs (Definition 3.1) and introduce the tasks of link prediction (Definition 3.2) and fact generation (Definition 3.3) on HKGs.

Definition 3.1 (Hyper-relational Knowledge Graph)**.** A hyper-relational knowledge graph (HKG) is defined as $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$, where $\mathcal{V}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively, and $\mathcal{H}$ denotes the set of hyper-relational facts. A hyper-relational fact $\xi \in \mathcal{H}$ comprises a primary triplet and a set of qualifiers, denoted as $\xi = \big((h, r, t), \mathcal{Q}_\xi\big)$. The primary triplet (h, r, t) consists of a head entity $h \in \mathcal{V}$, a primary relation $r \in \mathcal{R}$, and a tail entity $t \in \mathcal{V}$. The set of qualifiers is defined as $\mathcal{Q}_\xi = \{(k_i, v_i) \mid k_i \in \mathcal{R}, v_i \in \mathcal{V}\}_{i=1}^{n_\xi}$, where $n_\xi \geq 0$ is the number of qualifiers in ξ, and each qualifier pair (k_i, v_i) consists of a qualifier relation k_i and a qualifier entity v_i.

Definition 3.2 (Link Prediction on HKGs)**.** Given an HKG $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$, let ξ be a valid hyper-relational fact composed of entities from $\mathcal{V}$ and relations from $\mathcal{R}$. Let ζ be an incomplete hyper-relational fact derived from ξ by masking a single component y. Link prediction is the task of inferring the missing component y. Link prediction is categorized by the type of y: if $y \in \mathcal{V}$, the task is *entity prediction*, and if $y \in \mathcal{R}$, the task is *relation prediction*.

Definition 3.3 (Fact Generation on HKGs)**.** Given an HKG $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$, let ξ be a valid hyper-relational fact composed of entities and relations from $\mathcal{V}$ and $\mathcal{R}$, respectively. Let ζ be an incomplete hyper-relational fact derived from ξ by masking an arbitrary subset of its components. Fact generation is the task of generating a valid fact ξ' from ζ by replacing the masked components with entities in $\mathcal{V}$ and relations in $\mathcal{R}$. Note that ξ' is not necessarily identical to ξ.

4. KREPE: Contextual HKG Representation Learning via Masked Discrete Diffusion

We propose KREPE, which utilizes contextual message passing to compute representations and is trained via masked discrete diffusion to gain generative capabilities. To update entity and relation representations, KREPE aggregates messages from the facts they belong to while explicitly excluding their own information from those facts' messages; this exclusion forces their representations to rely solely on the surrounding "context" in the facts. The representation of a masked component is also derived from its context and compared against all candidate representations to compute its probability distribution. Figure 2 illustrates the training and inference pipeline. During training, KREPE partitions the training set into an observed set and a target set; facts in the observed set are used to compute entity and relation representations, while the facts in the target set are masked and serve as queries. In contrast, the entire training set serves as the observed set during inference. The probability distributions of the masked components are used to compute the diffusion loss during training and to perform both discriminative and generative tasks during inference. Implementation details of KREPE are provided in Appendix A.

4.1. Contextual Message Passing

To compute representations for entities and relations, KREPE employs contextual message passing, in which the message from a fact to each of its constituent components is computed using the fact's components, excluding the target component itself. The remaining components that do not include the target component within a fact are termed the "context" of the target component. By removing the component's own representation from the messages, KREPE prevents the trivial retention of self information while en-

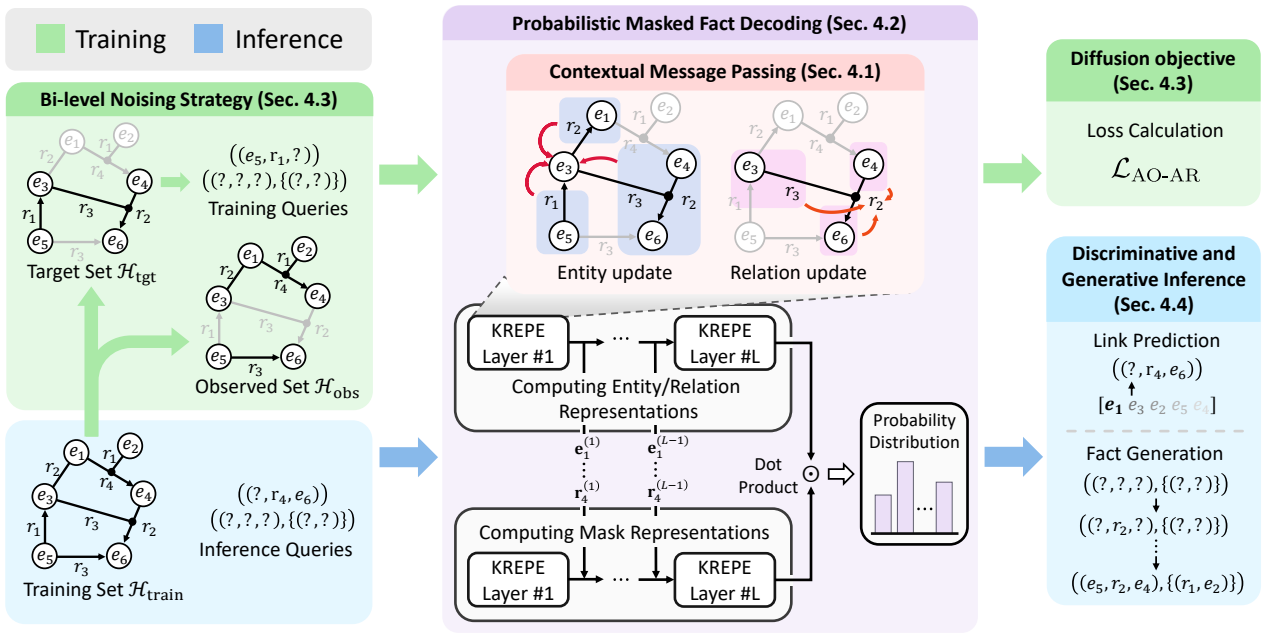

Figure 2. Overview of KREPE. During training, the training set is partitioned into an observed set and a target set, where the facts in the target set are masked to serve as queries. Representations of the entities, relations, and masks are updated via contextual message passing, integrating both the local contexts within the facts and the relationship between facts. These representations are used to approximate the probability distribution of the masks. The estimated probability distribution is used for the diffusion objective. During inference, the training set acts as the observed set, and the resulting probability distribution is used for both discriminative and generative inference tasks.

forcing reliance on the other components.

Let $\mathbf{p}^{(l)}, \mathbf{e}^{(l)} \in \mathbb{R}^d$ denote the representation of a relation p and an entity e at the l-th layer, respectively, where d denotes the dimension, $l \in \{1, \ldots, L\}$, and L is the number of layers. The representation of an entity e is initialized by $\mathbf{e}^{(0)} = \mathbf{z}_{\text{ENT}}$, where $\mathbf{z}_{\text{ENT}} \in \mathbb{R}^d$ is a shared learnable token for all entities. Similarly, the representation of a relation p is initialized by $\mathbf{p}^{(0)} = \mathbf{z}_{\text{REL}}$, where $\mathbf{z}_{\text{REL}} \in \mathbb{R}^d$ is a shared learnable token for all relations.

Let us decompose a fact into a set of its constituent relation-entity pairs. For example, a hyper-relational fact $\xi = \big((h, r, t), \mathcal{Q}_\xi\big)$ is decomposed into the set of pairs $\{(r, h), (r, t), (k_1, v_1), \cdots, (k_{n_\xi}, v_{n_\xi})\}$, where $(k_i, v_i) \in \mathcal{Q}_\xi$. Then, each pair is associated with a specific role: head, tail, or qualifier, based on the position of the entity, i.e., head entity, tail entity, or qualifier entity. For example, the role of (r, h) is head, (r, t) is tail, and (k_i, v_i) is qualifier. Let (p, e) denote a generic relation-entity pair associated with a role $\rho \in \{\texttt{head}, \texttt{tail}, \texttt{qual}\}$ in ξ, where p is a relation and e is an entity. The representation $\mathbf{z}^{(l)}_{(p,e)}$ of the pair (p, e) at the l-th layer is defined as follows:

$$\mathbf{z}^{(l)}_{(p,e)} = W^{(l)}_{\rho}[\mathbf{p}^{(l-1)}; \mathbf{e}^{(l-1)}] \tag{1}$$

where $W^{(l)}_{\texttt{head}}, W^{(l)}_{\texttt{tail}}, W^{(l)}_{\texttt{qual}} \in \mathbb{R}^{d \times 2d}$ are the projection matrices for the respective roles and $[;]$ denotes concatenation. The representation of a fact ξ at layer l, denoted as $\mathbf{z}^{(l)}_{\xi}$, is obtained by summing its constituent pairs:

$$\mathbf{z}^{(l)}_{\xi} = \mathbf{z}^{(l)}_{(r,h)} + \mathbf{z}^{(l)}_{(r,t)} + \sum_{i=1}^{n_\xi} \mathbf{z}^{(l)}_{(k_i, v_i)} \tag{2}$$

To update a component within a fact, we compute a *context message* that encodes all the other components in the fact. This ensures that the update is conditioned on the surrounding structure of the component. The context messages $\mathbf{m}^{(l)}_{\xi \to e}$ and $\mathbf{m}^{(l)}_{\xi \to p}$ from fact ξ to its components e and p, respectively, in the constituent pair (p, e) are computed by:

$$\mathbf{m}^{(l)}_{\xi \to (p,e)} = \text{MLP}^{(l)}\left(\frac{\mathbf{z}^{(l)}_{\xi} - \mathbf{z}^{(l)}_{(p,e)}}{n_\xi + 1}\right) \tag{3}$$

$$\mathbf{m}^{(l)}_{\xi \to e} = W^{(l)}_{\text{ENT}}[\mathbf{m}^{(l)}_{\xi \to (p,e)}; \mathbf{p}^{(l-1)}] \tag{4}$$

$$\mathbf{m}^{(l)}_{\xi \to p} = W^{(l)}_{\text{REL}}[\mathbf{m}^{(l)}_{\xi \to (p,e)}; \mathbf{e}^{(l-1)}] \tag{5}$$

where MLP denotes a multi-layer perceptron and $W^{(l)}_{\text{ENT}}$, $W^{(l)}_{\text{REL}} \in \mathbb{R}^{d \times 2d}$ denote the projection matrices. The message $\mathbf{m}^{(l)}_{\xi \to (p,e)}$ from ξ to (p, e) represents the composition of all the other pairs within ξ. The context message $\mathbf{m}^{(l)}_{\xi \to e}$ is formed by projecting the concatenation of this message with the representation of the paired relation p. This allows the context of e within ξ to incorporate all complementary components of e. Similarly, $\mathbf{m}^{(l)}_{\xi \to p}$ is computed by projecting

the concatenation of $\mathbf{m}^{(l)}_{\xi\to(p,e)}$ with $\mathbf{e}^{(l-1)}$.

Finally, the representation of an entity e is updated from $\mathbf{e}^{(l-1)}$ to $\mathbf{e}^{(l)}$ by aggregating the context messages from all facts involving e using a multi-head attention mechanism followed by an MLP (Vaswani et al., 2017). A similar process is applied to update the relation representations using distinct parameters. By iteratively aggregating the context messages across layers, the final representations encode the structural dependencies between the facts.

4.2. Probabilistic Masked Fact Decoding

KREPE is designed to generate facts from an incomplete fact with an arbitrary number of unknown components. Formally, an incomplete fact ζ is defined as a fact where one or more, potentially all, of its components are unknown, i.e., masked. Each masked entity and relation is initialized with a learnable vector, $\mathbf{x}_{\text{ENT}} \in \mathbb{R}^d$ and $\mathbf{x}_{\text{REL}} \in \mathbb{R}^d$, respectively.

Given an HKG $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$ and a masked fact ζ, KREPE treats each masked entity and relation as a vanilla entity and relation, respectively, and updates their representations via contextual message passing. In this process, ζ is not used to update the known components; ζ is used only to update its masked components. Specifically, the context message of a masked component x in ζ at the l-th layer is computed using the $(l-1)$-th layer's representations of known entities and relations in ζ and the $(l-1)$-th layer's representations of the other masked components in ζ, if any. This allows the representation of x to be conditioned on the context from ζ. Let $\mathbf{x}^{(L)}_{\text{mask}}$ denote the final representation of x. KREPE models the probability distribution of x on the candidate set $\mathcal{C}$, either $\mathcal{V}$ or $\mathcal{R}$ depending on x, as:

$$P_\theta(x \mid \zeta, G) = \underset{c\in\mathcal{C}}{\text{Softmax}}\left(\mathbf{x}^{(L)}_{\text{mask}} \cdot \mathbf{c}^{(L)}\right) \tag{6}$$

where $\mathbf{c}^{(L)}$ is the final representation of a candidate c computed using the facts in G. Our decoding strategy enables KREPE to estimate the probability distribution of missing components by integrating two complementary sources of information: (1) the local context of the masked component in the query, embedded in $\mathbf{x}^{(L)}_{\text{mask}}$, and (2) the global HKG structure, encoded across the candidate representations $\mathbf{c}^{(L)}$.

4.3. Generative Training via Masked Discrete Diffusion

To model the distribution of a masked component x in a query ζ conditioned on both the query ζ and the HKG G, i.e., $P_\theta(x|\zeta, G)$, KREPE utilizes masked discrete diffusion (Austin et al., 2021). However, unlike standard discrete diffusion where the corrupted input is often the only condition, fact generation in HKGs depends on both the local query and the global HKG structure. KREPE is trained to capture these dependencies via a bi-level noising strategy, which provides a diverse set of conditions by simultaneously varying both the query, i.e., the intra-fact level, and the structure of the observed HKG, i.e., the inter-fact level.

Bi-level Noising Strategy Bi-level noising simulates diverse conditional generation scenarios by varying both the observed structure and the query during training. First, to inject noise at inter-fact level, the HKG is dynamically partitioned every epoch via a process termed stochastic structure sampling. Specifically, the set of training facts $\mathcal{H}_{\text{train}}$ is split into an observed set $\mathcal{H}_{\text{obs}}$ and a target set $\mathcal{H}_{\text{tgt}}$, where KREPE computes entity and relation representations using only $\mathcal{H}_{\text{obs}}$. This strategy forces KREPE to output probability distributions conditioned on varying sets of observed facts. Next, at the intra-fact level, queries are generated by arbitrarily masking components within the facts in $\mathcal{H}_{\text{tgt}}$. Crucially, because $\mathcal{H}_{\text{tgt}}$ consists of facts excluded from $\mathcal{H}_{\text{obs}}$, KREPE is trained to generate facts missing in the currently observed facts, simulating the generation of new facts.

Diffusion objective The training objective of KREPE is the *any-order autoregressive* (AO-AR) objective, which is mathematically equivalent to the masked discrete diffusion objective (Ou et al., 2025). The masking distribution $\mathcal{D}_\xi$ is induced by first randomly selecting the number of masked components $n_{\text{mask}} \sim \mathcal{U}(\{1, \ldots, 2n_\xi + 3\})$ and subsequently sampling n_{mask} components without replacement from ξ. The objective is to minimize the negative log-likelihood of the ground-truth values for the masked components:

$$\mathcal{L}_{\text{AO-AR}} = \underset{\substack{\xi\sim\mathcal{H}_{\text{tgt}}\\ \mathcal{M}_\zeta\sim\mathcal{D}_\xi}}{\mathbb{E}}\left[-\sum_{(x,y)\in\mathcal{M}_\zeta} \log P_\theta(x=y \mid \zeta, (\mathcal{V}, \mathcal{R}, \mathcal{H}_{\text{obs}}))\right] \tag{7}$$

where $\mathcal{M}_\zeta$ denotes the set of pairs (x, y) comprising a masked component x and its ground-truth y in ζ. By optimizing this objective, KREPE is trained to reconstruct a masked query ζ in the target set using only the facts in $\mathcal{H}_{\text{obs}}$. As a result, KREPE learns to approximate the true probability distribution of masked components conditioned on the query and the observed structure, allowing it to generate plausible and novel facts absent from the training set. Note that KREPE learns the time-independent reparameterization of the absorbing discrete diffusion (Ou et al., 2025).

4.4. Discriminative and Generative Inference

During inference, KREPE computes the representations by utilizing all facts in $\mathcal{H}_{\text{train}}$ (Section 4.1). Given an incomplete fact, a forward pass generates mask representations to produce probability distributions for each mask (Section 4.2). These probability distributions are utilized to perform either discriminative or generative inference.

Link Prediction For discriminative link prediction (Definition 3.2), each query contains exactly one mask. KREPE

Table 1. Entity prediction results on WD50K and WikiPeople$^-$. Pri and All denote prediction on primary triplets and all positions including qualifiers, respectively.

		WD50K			WikiPeople$^-$		
		MRR	Hit10	Hit1	MRR	Hit10	Hit1
Pri	StarE (2020)	0.349	0.496	0.271	0.491	0.648	0.398
	GRAN (2021)	-	-	-	0.503	0.620	0.438
	Hy-Transformer (2021)	0.356	0.498	0.281	0.501	0.634	0.426
	HyperFormer (2023)	0.366	0.514	0.288	0.473	0.646	0.361
	HAHE (2023)	0.368	0.516	0.291	0.509	0.639	**0.447**
	ShrinkE (2023)	0.345	0.482	0.275	0.485	0.601	0.431
	HyNT (2023)	0.357	0.501	0.281	-	-	-
	HDiff (2025)	0.373	0.512	0.292	0.512	0.642	**0.447**
	FormerGNN (2025)	0.377	0.527	0.299	0.494	0.653	0.392
	MAYPL (2025)	0.381	0.544	0.297	**0.519**	0.657	0.444
	KREPE	**0.389**	**0.552**	**0.303**	**0.519**	**0.663**	0.437
All	HAHE (2023)	0.402	0.546	0.327	0.495	0.631	0.420
	HyNT (2023)	0.383	0.527	0.308	-	-	-
	HDiff (2025)	0.405	0.546	0.325	0.502	0.639	0.430
	MAYPL (2025)	0.411	0.572	0.326	0.521	0.659	**0.446**
	KREPE	**0.419**	**0.580**	**0.334**	**0.522**	**0.666**	0.439

Table 2. Entity prediction results on all entities within facts in WikiPeople.

	WikiPeople		
	MRR	Hit10	Hit1
RAE (2018)	0.172	0.320	0.102
NaLP (2019)	0.338	0.466	0.272
HypE (2020)	0.282	0.487	0.148
NeuInfer (2020)	0.350	0.467	0.282
RAM (2021)	0.380	0.539	0.279
S2S (2021)	0.372	0.533	0.277
PolygonE (2022)	0.431	0.568	0.334
HyConvE (2023)	0.362	0.501	0.275
MSeaKG (2023)	0.392	0.553	0.290
tNaLP (2023)	0.350	0.471	0.288
HCNet (2025)	0.421	0.565	0.344
HyCubE (2025b)	0.448	0.592	0.368
HJE (2024b)	0.450	0.582	0.375
GAHE (2025)	0.415	0.562	0.320
HySAE (2025a)	0.454	0.603	0.373
MAYPL (2025)	0.488	0.635	0.405
KREPE	**0.491**	**0.642**	**0.406**

leverages the computed probability distribution to rank the candidates and identifies the most likely missing component.

Fact Generation For fact generation (Definition 3.3), the goal is to generate a valid fact from an arbitrarily masked query via iterative decoding (Ou et al., 2025). At each step, probability distributions for all currently masked components are recomputed using the current state of query. A random subset of these masks is selected, which are replaced via probability-based top-p sampling, where p is a hyperparameter. This process continues until all masks are resolved or the maximum number of steps is reached. In the latter case, all remaining masks are sampled simultaneously. In our implementation, we employ a caching strategy that recomputes mask vectors only when a mask is replaced. This bounds the computation steps by the query length $|\zeta|$, resulting in an inference cost of $\mathcal{O}(|\zeta|^2 d^2 (L + |\mathcal{V}| + |\mathcal{R}|))$.

5. Experiments

For experiments, three HKG benchmark datasets are used: WD50K (Galkin et al., 2020), WikiPeople$^-$ (Wang et al., 2021), and WikiPeople (Guan et al., 2019). We follow the standard evaluation protocols (Galkin et al., 2020; Guan et al., 2019) for link prediction. For each dataset, KREPE is trained once using Eq. 7, and directly applied to three downstream tasks: entity prediction, relation prediction, and fact generation. In our tables, the best and second-best results are **boldfaced** and underlined, respectively. Dataset statistics and experimental details are provided in Appendix B, and hyperparameter settings are described in Appendix C. Additional experiments using BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) are reported in Appendix E.

Table 3. Relation prediction on WD50K and WikiPeople$^-$. Pri targets primary relations, whereas All includes qualifier relations.

		WD50K			WikiPeople$^-$		
		MRR	Hit10	Hit1	MRR	Hit10	Hit1
Pri	GRAN	-	-	-	0.957	0.976	0.942
	HAHE	0.916	0.964	0.885	0.957	0.978	0.941
	HDiff	0.950	0.985	0.924	0.973	0.991	0.958
	KREPE	**0.963**	**0.995**	**0.940**	**0.984**	**0.999**	**0.972**
All	HAHE	0.927	0.969	0.900	0.958	0.978	0.942
	HDiff	0.956	0.987	0.934	0.973	0.992	0.959
	KREPE	**0.968**	**0.995**	**0.947**	**0.984**	**0.999**	**0.972**

5.1. Link Prediction Results

We evaluate KREPE on two link prediction subtasks: entity prediction and relation prediction, where the Mean Reciprocal Rank (MRR) and Hits@k ($k = 1, 10$) are used as evaluation metrics. For all baseline methods, we report results from their original papers. Note that the number of baselines for relation prediction is limited, as only a few methods have been evaluated on this task.

Tables 1 and 2 show that KREPE outperforms entity prediction baselines on WD50K and WikiPeople while matching MAYPL (Lee & Whang, 2025) on WikiPeople$^-$. In relation prediction, reported in Table 3 and Appendix D, KREPE outperforms all baselines across all datasets. These results highlight that KREPE excels at discriminative tasks despite its generative objective. We attribute this to the nature of our

Table 4. Fact Generation on WD50K (denoted by WD), WikiPeople$^-$ (denoted by WP$^-$) and WikiPeople (denoted by WP). Accuracy is reported. *Scratch* denotes generating facts from scratch, *Targeted* denotes generating facts for a single known component, and *Arbitrary Masking* denotes generating facts starting from an arbitrary number of known components. N/A denotes not applicable.

		Scratch			*Targeted*			*Arbitrary Masking*		
	LLM	WD	WP$^-$	WP	WD	WP$^-$	WP	WD	WP$^-$	WP
Iterative Prediction	N/A	0.262	0.273	0.201	0.270	0.367	0.343	0.447	0.395	0.453
Gibbs Sampling	N/A	0.299	0.369	0.230	0.199	0.291	0.270	0.390	0.418	0.397
Re-ranking	GPT 5.2	0.215	0.229	0.233	0.112	0.176	0.192	0.133	0.177	0.157
	Gemini 3 Pro	0.304	0.288	0.278	0.256	0.305	0.293	0.295	0.314	0.313
Neighbor Sets	GPT 5.2	0.175	0.124	0.109	0.397	0.309	0.355	0.351	0.497	0.339
	Gemini 3 Pro	0.351	0.198	0.204	0.445	0.394	0.361	0.488	0.475	0.409
Few-shot Facts	GPT 5.2	0.290	0.233	0.260	0.222	0.232	0.238	0.312	0.340	0.324
	Gemini 3 Pro	0.326	0.307	0.326	0.281	0.252	0.255	0.311	0.337	0.280
Random Facts	GPT 5.2	0.298	0.212	0.301	0.463	0.344	0.351	0.405	0.458	0.432
	Gemini 3 Pro	0.317	0.343	0.302	0.482	0.333	0.396	**0.604**	0.446	0.469
Autoregressive	GPT 5.2	0.248	0.145	0.228	0.250	0.213	0.301	0.248	0.257	0.234
	Gemini 3 Pro	0.265	0.154	0.238	0.190	0.175	0.239	0.423	0.350	0.364
KREPE	N/A	**0.717**	**0.855**	**0.777**	**0.508**	**0.600**	**0.591**	**0.604**	**0.649**	**0.621**

training: link prediction is encompassed within the masked discrete diffusion process, and the contextual message passing captures precise dependencies by explicitly conditioning on the complementary components.

5.2. Fact Generation Results

We define three settings for fact generation: (i) *Scratch*, generating a valid fact without any known component, representing the most extreme scenario of generation from scratch; (ii) *Targeted*, requiring the generation of a fact containing a single given component called a target; and (iii) *Arbitrary Masking*, generating a fact from an arbitrarily masked input, covering diverse scenarios including link prediction and generation from scratch. We use 1,000 fully masked queries for *Scratch*. For each of the *Targeted* and *Arbitrary Masking* setting, 1,000 facts in the test set for link prediction are sampled. Then, these facts are masked to retain a single component for *Targeted*, whereas arbitrary number of components are masked for *Arbitrary Masking*. To evaluate the validity of generated facts, we employ an LLM-as-a-judge framework (Zheng et al., 2023) with GPT-5.2 and report the accuracy as the ratio of correct facts. To align with the standard filtered setting in link prediction, we employ rejection sampling, where generated facts present in the training set are discarded and re-generated. If a model fails to generate a novel fact within 10 attempts, we penalize it by marking the result as incorrect. More details of fact generation experiments are in Appendix G.

Since existing HKG models are discriminative, they cannot be directly applied to generative tasks. Therefore, we design seven competitive baselines: two variations of discriminative HKG models and five LLM-based approaches tailored for fact generation. We employ the state-of-the-art HKG model, MAYPL, as the backbone of the baselines that require an HKG model. We use GPT 5.2 (OpenAI, 2025) and Gemini 3.0 Pro (Google, 2025) for the LLM-based methods. Details about these baseline methods are in Appendix I.

- *Iterative Prediction*: Missing components are randomly initialized. Subsequently, they are sequentially masked and replaced by an HKG model's prediction.
- *Gibbs Sampling:* Missing components are randomly initialized, followed by an Gibbs sampling process where each missing component is iteratively masked and replaced by resampling from the distribution predicted by an HKG model.
- *Re-ranking:* Missing components are initialized using a training fact that matches the known components of the query. Each missing component is sequentially masked, and an LLM selects the best candidate from the top-5 predictions of an HKG model.
- *Neighbor Sets:* An LLM generates each fact prompted by the neighbors of known components in the query.
- *Few-shot Facts:* An LLM generates each fact prompted by 30 training facts containing at least one known entity or relation from the query.
- *Random Facts:* An LLM is prompted to generate facts by utilizing 1,000 randomly sampled training facts.
- *Autoregressive:* An LLM generates components step-by-step, prompted to utilize training facts containing the previously predicted component for generation.

Table 4 presents the results for fact generation. Across all settings, KREPE consistently achieves state-of-the-art per-

Table 5. Fact Generation from scratch on WikiPeople$^-$. We report the Valid & Novel Rate (V&N Rate) and the expected number of generation attempts needed per valid and novel fact ($\mathbb{E}$[Gen.]). KREPE achieves the highest V&N Rate and the lowest $\mathbb{E}$[Gen.].

	LLM	V&N Rate(↑)	$\mathbb{E}$[Gen.](↓)
Iterative Prediction	N/A	0.204	4.90
Gibbs Sampling	N/A	0.184	5.44
Re-ranking	GPT 5.2	0.157	6.36
	Gemini 3.0 Pro	0.155	6.45
Neighbor Sets	GPT 5.2	0.110	9.11
	Gemini 3.0 Pro	0.136	7.37
Few-shot Facts	GPT 5.2	0.153	6.54
	Gemini 3.0 Pro	0.242	4.13
Random Facts	GPT 5.2	0.036	27.58
	Gemini 3.0 Pro	0.104	9.65
Autoregressive	GPT 5.2	0.089	11.27
	Gemini 3.0 Pro	0.091	10.99
KREPE	N/A	**0.351**	**2.85**

formance, demonstrating superior generative capabilities. This stems from the design of KREPE, which explicitly learns to generate plausible and new facts via contextual message passing and bi-level noising strategy. Unlike LLMs, which generates facts by modeling the likelihood of general text sequences, KREPE is directly optimized to model the probability distribution of missing components conditioned on the query and the global structure of an HKG. A critical distinction also lies in the novelty of the generated facts. While it is often indistinguishable whether LLMs are performing reasoning or merely retrieving memorized information from their massive pre-training corpora, the facts used to evaluate KREPE are clearly not in its training data. Thus, the accuracy reflects the capacity of KREPE to infer novel and correct facts. KREPE effectively performs conditional generation for incomplete facts, significantly outperforming LLMs in HKG-conditioned generation tasks.

Additional Metrics To comprehensively measure both accuracy and novelty, we introduce two supplementary metrics: the Valid & Novel Rate (V&N Rate), the proportion of correct and novel facts, and the expected number of generation attempts per valid and novel fact ($\mathbb{E}$[Gen.]). A higher V&N Rate and a lower $\mathbb{E}$[Gen.] indicate superior performance. Table 5 reports these metrics on WikiPeople$^-$ under the Scratch setting. KREPE maintains the best performance, confirming its capability to generate novel, accurate facts.

5.3. LLM-as-a-Judge Verification

LLM-as-a-Judge is a common alternative to human evaluation (Zheng et al., 2023). To validate our GPT-5.2 judge, we conduct two supplementary evaluations on WikiPeople$^-$ under the Scratch setting:

Table 6. Comparison of accuracy between the GPT-5.2 Judge and Multi-Judge Consensus (Multi) on the full set, and a comparison between the GPT-5.2 Judge and Human evaluation on a 10% subset. All evaluations are conducted for fact generation from scratch on WikiPeople$^-$. The trend remains consistent across different evaluators. GPT and Gemini denote GPT 5.2 and Gemini 3.0 Pro.

		Full set		*10% Subset*	
	LLM	GPT-5.2	Multi	GPT-5.2	Human
Iterative Prediction	N/A	0.273	0.254	0.24	0.25
Gibbs Sampling	N/A	0.369	0.343	0.33	0.25
Re-ranking	GPT	0.229	0.163	0.28	0.22
	Gemini	0.288	0.263	0.32	0.32
Neighbor Sets	GPT	0.124	0.107	0.15	0.11
	Gemini	0.198	0.160	0.17	0.17
Few-shot Facts	GPT	0.233	0.217	0.21	0.21
	Gemini	0.307	0.273	0.29	0.31
Random Facts	GPT	0.212	0.184	0.20	0.18
	Gemini	0.343	0.309	0.42	0.38
Autoregressive	GPT	0.145	0.126	0.18	0.14
	Gemini	0.154	0.141	0.18	0.18
KREPE	N/A	**0.855**	**0.821**	**0.86**	**0.83**

- *Multi-Judge Consensus*: We employ two additional LLM judges, Gemini 3.1 Pro (Google, 2026) and Grok 4.20 Reasoning (xAI, 2026). A generated fact is considered correct only if GPT 5.2, Gemini 3.1 Pro, and Grok 4.20 Reasoning reach a unanimous agreement.
- *Human Evaluation*: We sample a 10% subset of the generated facts and engage three human annotators. A fact is considered correct only upon unanimous agreement among all three humans.

Table 6 details the evaluation results, demonstrating that while strict unanimous criteria slightly lower the absolute scores, the overall model rankings remain highly stable. To formally verify this alignment, we calculate the Pearson correlation coefficients, where 1 implies an identical trend. The results reveal a near-perfect positive correlation: 0.997 between GPT-5.2 and Multi-Judge on the full set, and 0.987 between GPT-5.2 and Human on the 10% subset. These coefficients confirm that the scoring trends are almost identical, proving that our evaluation remains highly consistent across diverse LLMs and human judgment.

5.4. Ablation Studies

Table 8 presents the ablation studies of KREPE on WD50K and WikiPeople$^-$. **(i)** Removing context messages (Eq. 4, 5) and relying solely on fact representations (Eq.2) to update entities and relations (Section 4.1) leads to substantial degradation, confirming that excluding self information and ag-

Table 7. Qualitative examples of fact generation by KREPE on WD50K. Generated facts with zero, one, and two qualifiers for the head entity "Toy Story" and tail entity "A. Irving" are presented. Additionally, three facts generated from scratch are shown.

Query	Generation
$\big((\text{Toy Story}, ?, ?), \{(?, ?)\}_{i=1}^{n_\xi}\big)$	$\big((\text{Toy Story, genre, fantasy film}), \{\}\big)$
	$\big((\text{Toy Story, voice actor, T. Allen}), \{(\text{as, Buzz Lightyear})\}\big)$
	$\big((\text{Toy Story, nominated for, Oscars Best Score}), \{(\text{subject of, 68th Oscars}), (\text{nominee, R. Newman})\}\big)$
$\big((?, ?, \text{A. Irving}), \{(?, ?)\}_{i=1}^{n_\xi}\big)$	$\big((\text{She's Having a Baby, cast member, A. Irving}), \{\}\big)$
	$\big((\text{SAGA Performance Cast, winner, A. Irving}), \{(\text{for work, Traffic})\}\big)$
	$\big((\text{Who Framed Roger Rabbit, voice actor, A. Irving}), \{(\text{as, Jessica Rabbit}), (\text{role, singer})\}\big)$
$\big((?,?,?), \{(?,?)\}\big)$	$\big((\text{Star Wars: Episode IV, distributed by, InterCom}), \{(\text{country, Hungary})\}\big)$
	$\big((\text{L. Messi, participant in, 2008 Champions League}), \{(\text{team, F.C. Barcelona})\}\big)$
	$\big((\text{Hugo Award for Best Novel, winner, C. J. Cherryh}), \{(\text{for work, Cyteen})\}\big)$

Table 8. Ablation studies of KREPE on WD50K and WikiPeople$^-$. We report MRR for Entity Prediction on all positions (LP) and Accuracy for Fact Generation from scratch (FG).

		WD50K		WikiPeople$^-$	
		LP	FG	LP	FG
(i)	w/o Context Msg.	0.408	0.552	0.507	0.578
(ii)	w/o Str. Samp.	0.296	0.545	0.409	0.681
(iii)	w/o Attention	0.408	0.673	0.466	0.779
(iv)	w/ Individual Init.	0.272	0.466	0.390	0.575
(v)	w/ LP Loss	0.415	0.038	0.516	0.002
(vi)	w/ Cos. Sim.	**0.419**	0.711	0.497	0.825
	KREPE	**0.419**	**0.717**	**0.522**	**0.855**

gregating surrounding contexts is crucial, especially for fact generation. **(ii)** Disabling stochastic structure sampling (Section 4.3) degrades performance, indicating that varying the observed set during training is essential. **(iii)** Replacing attention in contextual message passing with mean pooling(Section 4.1) leads to worse results, underscoring the importance of adaptively weighting facts. **(iv)** Using individual initializations for every entity and relation instead of our shared initialization (Section 4.1) reduces performance, suggesting that our approach outperforms the use of unique identifiers. **(v)** Replacing the AO-AR loss with the cross-entropy loss for link prediction yields near-zero generation accuracy. This proves that our objective unifies both capabilities without compromising discriminative performance. **(vi)** Replacing the dot product with cosine similarity in Eq. 6 degrades performance, highlighting the significance of the magnitude in the representation space. We also provide the runtime-quality tradeoff for denoising steps in Appendix F.

5.5. Qualitative Analysis of KREPE

Table 7 presents a qualitative analysis, illustrating facts generated by KREPE for given queries. First, we examine how the generated output varies with the number of qualifiers n_ξ when the head or tail entity is fixed. For instance, given the head entity "Toy Story" and $n_\xi = 2$, KREPE generates a fact representing the knowledge that the movie Toy Story was nominated for the Oscars for Best Score at the 68th Oscars, with the nominee R. Newman. When generating from scratch with a single qualifier pair, KREPE produces a diverse set of facts across various domains such as film, sports, and literature. These results demonstrate that KREPE generates both plausible and novel facts for a given query. Appendix H provides further insights, including the scratch generation results across varying lengths, a step-by-step analysis of the generation process, and case studies on the top-3 similar entities or relations for a target.

6. Conclusion and Future Work

We introduce *Fact Generation*, a novel task that advances beyond the traditional fill-in-the-blank link prediction in the HKG domain by aligning more closely with real-world scenarios where multiple components of a fact may be missing. We propose KREPE, the first HKG representation learning framework equipped with generative capabilities via a masked discrete diffusion objective. By employing contextual message passing to model conditional dependencies within a fact and stochastic structure sampling to capture inter-fact relationships, KREPE effectively learns the joint conditional distribution of masked components. Empirical results demonstrate the superiority of our approach: KREPE not only establishes a new state-of-the-art link prediction performance but also significantly outperforms LLM-based baselines in fact generation. For future work, we plan to extend our framework to inductive scenarios, applying it to inference graphs containing unseen entities and relations. Also, we intend to explore KREPE's generative capabilities across a wide range of applications that require structured reasoning and the discovery of new knowledge and facts.

Impact Statement

This paper introduces a framework for fact generation on HKGs. By enabling models to generate structurally valid and plausible facts beyond simple link prediction, our work has the potential to significantly accelerate the automated completion of large-scale knowledge bases in critical domains such as biomedicine, finance, and enterprise knowledge management. Unlike LLMs that may generate structurally inconsistent facts, our approach ensures adherence to the rigorous topology of HKGs, potentially offering a more reliable source for structured knowledge reasoning.

However, the advanced generative capabilities of KREPE also present potential risks regarding the dissemination of misinformation. Since the model is proficient at generating structurally coherent and plausible facts even from scratch, it could potentially be misused to automate the creation of fabricated knowledge. Malicious actors might leverage this capability to inject high-fidelity false information into broader information ecosystems, thereby corrupting downstream applications that rely on the factual integrity of the ecosystem. Furthermore, the model's ability to infer missing components without rigid constraints necessitates careful monitoring to prevent the unintentional reinforcement of biases present in the training data. Therefore, the deployment of such generative models requires robust verification mechanisms and human supervision to distinguish between genuine knowledge discovery and malicious or manipulated content.

Acknowledgements

This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korean government(MSIT) (70% from RS-2025-00559066), Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (24% from RS-2022-II220369, (Part 4) Development of AI Technology to support Expert Decision-making that can Explain the Reasons/Grounds for Judgment Results based on Expert Knowledge, 1% from RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain), and the InnoCORE program of the Ministry of Science and ICT (5% from AI Meta-Scientist, N10260110).

References

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pp. 17981–17993, 2021.

Cao, Z., Li, J., Wang, Z., and Li, J. DiffusionE: Reasoning on knowledge graphs via diffusion-based graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 222–230, 2024.

Cao, Z., Xu, Q., Yang, Z., He, Y., Cao, X., and Huang, Q. GAHE: Geometry-aware embedding for hyper-relational knowledge graph representation. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(7):1–26, 2025.

Chung, C., Lee, J., and Whang, J. J. Representation learning on hyper-relational and numeric knowledge graphs with transformers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 310–322, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Di, S. and Chen, L. Message function search for knowledge graph embedding. In *Proceedings of the ACM Web Conference 2023*, pp. 2633–2644, 2023.

Di, S., Yao, Q., and Chen, L. Searching to sparsify tensor decomposition for N-ary relational data. In *Proceedings of the Web Conference 2021*, pp. 4043–4054, 2021.

Fatemi, B., Taslakian, P., Vazquez, D., and Poole, D. Knowledge Hypergraphs: Prediction beyond binary relations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 2191–2197, 2020.

Galkin, M., Trivedi, P., Maheshwari, G., Usbeck, R., and Lehmann, J. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 7346–7359, 2020.

Google. Gemini 3.0 pro. https://gemini.google, 2025.

Google. Gemini 3.1 pro. https://gemini.google, 2026.

Guan, S., Jin, X., Wang, Y., and Cheng, X. Link prediction on N-ary relational data. In *Proceedings of The World Wide Web Conference 2019*, pp. 583–593, 2019.

Guan, S., Jin, X., Guo, J., Wang, Y., and Cheng, X. NeuInfer: Knowledge inference on N-ary facts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6141–6151, 2020.

Guan, S., Jin, X., Guo, J., Wang, Y., and Cheng, X. Link prediction on N-ary relational data based on relatedness evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 35(01):672–685, 2023.

Guo, L., Bo, Z., Chen, Z., Zhang, Y., Chen, J., Lan, Y., Sun, M., Zhang, Z., Luo, Y., Li, Q., Zhang, Q., Zhang, W., and Chen, H. MKGL: Mastery of a three-word language. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pp. 140509–140534, 2024.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 6840–6851, 2020.

Hu, Z., Gutiérrez-Basulto, V., Xiang, Z., Li, R., and Pan, J. Z. HyperFormer: Enhancing entity and relation interaction for hyper-relational knowledge graph completion. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 803–812, 2023.

Huang, X., Orth, M. R., Barcelo, P., Bronstein, M. M., and Ceylan, I. I. Link prediction with relational hypergraphs. *Transactions on Machine Learning Research*, 2025.

Jiang, P., Cao, L., Xiao, C., Bhatia, P., Sun, J., and Han, J. KG-FIT: Knowledge graph fine-tuning upon open-world knowledge. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pp. 136220–136258, 2024.

Lee, J. and Whang, J. J. Structure Is All You Need: Structural representation learning on hyper-relational knowledge graphs. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 33765–33783, 2025.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, 2020.

Li, Q., Chen, Z., Ji, C., Jiang, S., and Li, J. LLM-based multi-level knowledge generation for few-shot knowledge graph completion. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pp. 2135–2143, 2024a.

Li, Z., Wang, C., Wang, X., Chen, Z., and Li, J. HJE: Joint convolutional representation learning for knowledge hypergraph completion. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3879–3892, 2024b.

Li, Z., Wang, X., Zhao, J., Feng, F., Chen, Z., and Li, J. HySAE: An efficient semantic-enhanced representation learning model for knowledge hypergraph link prediction. In *Proceedings of the ACM Web Conference 2025*, pp. 86–97, 2025a.

Li, Z., Wang, X., Zhao, J., Guo, W., and Li, J. HyCubE: Efficient knowledge hypergraph 3d circular convolutional embedding. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1902–1914, 2025b.

Lin, Q., Zhao, T., He, K., Peng, Z., Xu, F., Huang, L., Ma, J., and Feng, M. Self-supervised quantized representation for seamlessly integrating knowledge graphs with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13587–13602, 2025.

Liu, Y., Yao, Q., and Li, Y. Role-aware modeling for N-ary relational knowledge bases. In *Proceedings of the Web Conference 2021*, pp. 2660–2671, 2021.

Liu, Y., Cao, Y., Lin, X., Shang, Y., Wang, S., and Pan, S. Enhancing large language model for knowledge graph completion via structure-aware alignment-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20970–20984, 2025a.

Liu, Y., Tian, X., Sun, Z., and Hu, W. Finetuning generative large language models with discrimination instructions for knowledge graph completion. In *Proceedings of the 23rd International Semantic Web Conference*, pp. 199–217, 2025b.

Long, X., Zhuang, L., Li, A., Li, H., and Wang, S. Fact embedding through diffusion model for knowledge graph completion. In *Proceedings of the ACM Web Conference 2024*, pp. 2020–2029, 2024a.

Long, X., Zhuang, L., Li, A., Wei, J., Li, H., and Wang, S. KGDM: A diffusion model to capture multiple relation semantics for knowledge graph embedding. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pp. 8850–8858, 2024b.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Luo, H., E, H., Yang, Y., Guo, Y., Sun, M., Yao, T., Tang, Z., Wan, K., Song, M., and Lin, W. HAHE: Hierarchical attention for hyper-relational knowledge graphs in global and local level. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8095–8107, 2023.

Luo, H., E, H., Yang, Y., Yao, T., Guo, Y., Tang, Z., Zhang, W., Peng, S., Wan, K., Song, M., Lin, W., Zhu, Y., and Luu, A. T. Text2NKG: Fine-grained N-ary relation extraction for N-ary relational knowledge graph construction. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pp. 27417–27439, 2024.

Luo, X., Zheng, R., Huang, J., and Yu, H. HDiff: Confidence-guided denoising diffusion for robust hyper-relational link prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7417–7434, 2025.

Mo, B., Yu, K., Kazdan, J., Mpala, P., Yu, L., Kanatsoulis, C., and Koyejo, S. KGGen: Extracting knowledge graphs from plain text with language models. In *Proceedings of the 39th Conference on Neural Information Processing Systems*, 2025.

OpenAI. Gpt-5.2. https://openai.com, 2025.

Ou, J., Nie, S., Xue, K., Zhu, F., Sun, J., Li, Z., and Li, C. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

Rosso, P., Yang, D., and Cudré-Mauroux, P. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*, pp. 1885–1896, 2020.

Su, J., Xu, J., Qiu, X., and Huang, X. Incorporating discriminator in sentence generation: a gibbs sampling method. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 5496–5503, 2018.

Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706, 2007.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. RotatE: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vrandečić, D. and Krötzsch, M. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57 (10):78–85, 2014.

Wang, C., Wang, X., Li, Z., Chen, Z., and Li, J. HyConvE: A novel embedding model for knowledge hypergraph link prediction with convolutional neural networks. In *Proceedings of the ACM Web Conference 2023*, pp. 188–198, 2023.

Wang, Q., Wang, H., Lyu, Y., and Zhu, Y. Link prediction on N-ary relational facts: A graph-based approach. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 396–407, 2021.

Wang, Y., Di, S., Wang, Z., Li, H., Teng, F., Xin, H., and Chen, L. Understanding the embedding models on hyper-relational knowledge graph. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 3177–3187, 2025.

Wei, Y., Huang, Q., Zhang, Y., and Kwok, J. KICGPT: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 8667–8683, 2023.

xAI. Grok 4.20 reasoning. https://grok.com/, 2026.

Xiong, B., Nayyeri, M., Pan, S., and Staab, S. Shrinking embeddings for hyper-relational knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13306–13320, 2023.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10524–10533, 2020.

Xu, D., Zhang, Z., Lin, Z., Wu, X., Zhu, Z., Xu, T., Zhao, X., Zheng, Y., and Chen, E. Multi-perspective improvement of knowledge graph completion with large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 11956–11968, 2024.

Yan, S., Zhang, Z., Sun, X., Xu, G., Li, S., Liu, Q., Liu, N., and Wang, S. PolygonE: Modeling n-ary relational data as gyro-polygons in hyperbolic space. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 4308–4317, 2022.

Yu, D. and Yang, Y. Improving hyper-relational knowledge graph completion. In *arXiv preprint*, 2021.

Zhang, R., Li, J., Mei, J., and Mao, Y. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1185–1194, 2018.

Zhang, W., Xu, Y., Ye, P., Huang, Z., Xu, Z., Chen, J., Pan, J. Z., and Chen, H. Start from zero: Triple set prediction for automatic knowledge graph completion. *IEEE Transactions on Knowledge and Data Engineering*, 36 (11):7087–7101, 2024.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, pp. 46595–46623, 2023.

A. Implementation Details of KREPE

We implement KREPE based on the Pre-LN architecture (Xiong et al., 2020) to ensure training stability, utilizing the Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2016) for non-linear activation. A scaled dot-product is employed to compute the final candidate probabilities as formulated in Eq. 6. Bias terms are included in all linear projections, and dropout is employed. Additionally, we set $\mathbf{z}_{\mathrm{ENT}} = \mathbf{x}_{\mathrm{ENT}}$ and $\mathbf{z}_{\mathrm{REL}} = \mathbf{x}_{\mathrm{REL}}$ for WikiPeople and WikiPeople$^-$. In terms of regularization, we exclude bias terms and LayerNorm parameters from weight decay following common practice. For WikiPeople and WikiPeople$^-$, weight decay is explicitly applied to the LayerNorm preceding the MLP layer. Our code and datasets are available at `https://github.com/bdi-lab/KREPE`.

We use Python 3.11 and PyTorch 2.6.0 with CUDA 11.8. KREPE is optimized via AdamW (Loshchilov & Hutter, 2019). We utilize an NVIDIA RTX A6000 for all datasets.

B. Details on the HKG Datasets

Table 9. Statistics of HKG datasets.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{H}_{\text{train}}\rvert$	$\lvert\mathcal{H}_{\text{valid}}\rvert$	$\lvert\mathcal{H}_{\text{test}}\rvert$
WD50K	47,155	531	166,435	23,913	46,159
WikiPeople$^-$	34,825	178	294,439	37,715	37,712
WikiPeople	47,765	193	305,725	38,223	38,281

In our experiments, we utilize three standard benchmarks: WD50K (Galkin et al., 2020), WikiPeople$^-$ (Wang et al., 2021), and WikiPeople (Guan et al., 2019). WD50K is an HKG dataset derived from Wikidata by extracting facts that contain entities that correspond to those in FB15K-237 (Toutanova & Chen, 2015). WikiPeople is another HKG dataset from Wikidata, collected by filtering facts involving humans, while WikiPeople$^-$ is a modified version of WikiPeople constructed by removing literals. The detailed statistics for each dataset are summarized in Table 9. $\mathcal{H}_{\text{train}}$, $\mathcal{H}_{\text{valid}}$, and $\mathcal{H}_{\text{test}}$ denote the training, validation, and test set, respectively.

On WD50K and WikiPeople$^-$, we select optimal hyperparameters using the validation set and report final results after retraining KREPE on the combined training and validation sets, by following the common practice for these datasets (Galkin et al., 2020; Wang et al., 2021). For WikiPeople, we directly utilized the model trained on the training set with the best validation performance. Following MAYPL (Lee & Whang, 2025), we transfer the hyperparameter configuration from WikiPeople$^-$ to WikiPeople, tuning only the number of training epochs based on validation performance.

C. Hyperparameter Settings of KREPE

We perform a grid search to determine the optimal hyperparameters for each dataset. Table 10 reports the best hyperparameter configurations for KREPE. We employ a cosine learning rate scheduler (Loshchilov & Hutter, 2017) over a total of 2,000 epochs, incorporating a linear warmup phase for the first 200 epochs. The peak learning rate and the minimum learning rate is denoted by $\eta_{\max}$ and $\eta_{\min}$, respectively. Validation is performed every 50 epochs to identify the best epoch. Additionally, we use a weight decay coefficient of 0.01 with AdamW for WD50K and 0.1 for WikiPeople and WikiPeople$^-$. For the stochastic structure sampling proposed in Section 4.3, the probability of sampling a fact from $\mathcal{H}_{\text{train}}$ to form the observed set $\mathcal{H}_{\text{obs}}$ is fixed at 0.7. To stabilize training, we apply gradient clipping with a maximum norm of 1.0. The number of attention heads is specified separately for entities (n_{ENT}) and relations (n_{REL}). B refers to the batch size. We use a dropout rate of 0.1 across all datasets.

Sampling Parameters As described in Section 4.4, we employ a probability-based top-p sampling for fact generation, where δ_{ENT} and δ_{REL} denote the probability thresholds for entity and relation sampling, respectively. Following (Ou et al., 2025), we adopt a log-linear noise schedule over 1,000 reconstruction steps to ensure that the AO-AR objective is equivalent to the absorbing diffusion loss (Ou et al., 2025). Additionally, we apply temperature scaling to control the stochasticity of generation, denoted by τ_{ENT} and τ_{REL}. For WD50K, the *Scratch* setting uses thresholds of $\delta_{\mathrm{ENT}} = 0.15$ and $\delta_{\mathrm{REL}} = 0.05$ with unit temperature ($\tau_{\mathrm{ENT}} = \tau_{\mathrm{REL}} = 1$). For the *Targeted* and *Arbitrary Masking* settings, we employ $\delta_{\mathrm{ENT}} = 0.7$, $\delta_{\mathrm{REL}} = 0.5$, $\tau_{\mathrm{ENT}} = 0.55$, and $\tau_{\mathrm{REL}} = 0.5$. For WikiPeople$^-$, we use $\delta_{\mathrm{ENT}} = \delta_{\mathrm{REL}} = 0.2$ and $\tau_{\mathrm{ENT}} = \tau_{\mathrm{REL}} = 1$ across all settings. For WikiPeople, we use $\delta_{\mathrm{ENT}} = 0.2$, $\delta_{\mathrm{REL}} = 0.1$, and $\tau_{\mathrm{ENT}} = \tau_{\mathrm{REL}} = 1$ across all settings.

Table 10. Best hyperparameter settings of KREPE for WD50K, WikiPeople$^-$, and WikiPeople.

	best epoch	$\eta_{\max}$	$\eta_{\min}$	d	L	n_{ENT}	n_{REL}	B
WD50K	1950	0.001	0.00001	128	16	4	4	2048
WikiPeople$^-$	2000	0.0004	0.00001	256	12	8	32	4096
WikiPeople	1750	0.0004	0.00001	256	12	8	32	4096

Table 11. Relation prediction on all relations in facts in WikiPeople.

	MRR	Hit10	Hit1
NaLP	0.735	0.938	0.595
NeuInfer	0.765	0.897	0.686
tNaLP	0.790	0.927	0.713
KREPE	**0.985**	**0.999**	**0.972**

D. Relation Prediction Results for WikiPeople

Table 11 reports the relation prediction results on WikiPeople. Consistent with the results on WD50K and WikiPeople$^-$, KREPE significantly outperforms all existing methods. As noted in Section 5.1, the baseline comparison is limited to a subset of those used for entity prediction, as relation prediction is less frequently evaluated in the literature.

E. Additional Baseline Results for WikiPeople$^-$

Table 12 reports the performance of BERT (Devlin et al., 2019), a masked language model, and BART (Lewis et al., 2020), an autoregressive transformer, on WikiPeople$^-$. We serialized the facts into sequences and performed end-to-end training for these architectures. We report the MRR for link prediction (LP) and the accuracy for fact generation from scratch (FG). The results show that KREPE significantly outperforms these baselines in both tasks.

Table 12. End-to-end training results using BERT and BART architecture for WikiPeople$^-$. We report the MRR for link prediction and the accuracy for fact generation from scratch.

	Link Prediction	Fact Generation
BERT	0.503	0.098
BART	0.477	0.181
KREPE	**0.522**	**0.855**

F. Runtime-Quality Tradeoff in Denoising Steps of KREPE

We provide the runtime-quality tradeoffs for fewer denoising steps on WikiPeople$^-$ for fact generation from scratch. We also report the time required to generate a single query with a batch size of 1. Table 13 shows that reducing steps decreases both runtime and accuracy. Yet, even with just 50 steps, the accuracy of KREPE (0.671) is nearly 2x higher than the best baseline, Gibbs Sampling (0.369), while being over 5$\times$ faster (0.53 sec vs. 2.81 sec). We note that KREPE supports efficient batch processing; for instance, generating a batch of 1,000 queries takes only 79 seconds, yielding a latency of $\sim$0.08 sec per query.

Table 13. Runtime-quality tradeoffs for varying denoising steps of KREPE on WikiPeople$^-$ for fact generation from scratch.

Steps	Time(sec/query)	Accuracy
50	0.530	0.671
100	0.550	0.684
250	0.647	0.708
500	0.757	0.768
1000	0.910	0.855

Table 14. Distribution of fact lengths in each dataset for the Scratch setting.

	3	5	7	9	11	13
WD50K	864	89	40	4	2	1
WikiPeople$^-$	974	20	5	1	0	0
WikiPeople	884	67	40	7	2	0

Fact Verification Prompt Template

User:
You are an expert at verifying hyper-relational facts in a hyper-relational knowledge graph. Given a hyper-relational fact, you need to verify whether the fact is (1) semantically valid and (2) factually true. If you have understood your responsibility, respond "Yes". Otherwise, respond "No". Do not output anything except "Yes" or "No".

Model:
Yes

User:
The goal statement is: "`{fact}`". To verify the goal statement, you need to refer to other examples that may be similar or related to it. Some of the given examples are similar to the goal statement. You should draw analogies from them to understand the potential meaning of the goal statement. Other provided facts contain supplementary information; capture this extra information and mine potential relationships among them to support the verification. Please carefully read, analyze, and reflect on these examples. Identify the reasoning patterns demonstrated in these examples and retain any information that may help your verification task. While I provide examples, please remain silent until I ask you to respond.

Model:
I have received your instructions and the goal statement.

User:
Examples used for analogy:
Verify the fact "`{analogy_fact#1}`". The fact is semantically valid and factually true, so the verification result is Yes.
...

Examples used to supplement information:
Verify the fact "`{supplement_fact#1}`". The fact is semantically valid and factually true, so the verification result is Yes.
...

Keep thinking, but DO NOT give me any feedback.

Model:
I have analyzed the examples provided and will keep thinking without giving feedback.

User:
The goal is to verify the fact "`{fact}`". Based on the previous examples and your own knowledge, provide a verification result.

- If the statement is semantically valid and factually true, respond "Yes".
- If the statement is semantically valid but factually false, respond "Half".
- If the statement is semantically invalid, respond "No".

Your answer must be consistent with all provided examples, even if it conflicts with standard linguistic definitions.

DO NOT OUTPUT ANYTHING ELSE.

Model:
Yes/Half/No

Figure 3. The prompt used to evaluate the correctness of generated facts.

G. Dataset Construction and Evaluation Protocols for Fact Generation

Dataset Construction In Section 5.2, we defined three settings for fact generation: *Scratch*, *Targeted*, and *Arbitrary Masking*. Since the fact generation task requires queries to initiate the generation process, we construct 1,000 queries for each setting. We define the "base set" as the set of facts utilized to train the model for evaluation, which comprises the union of the training and validation sets for WD50K and WikiPeople$^-$, and strictly the training set for WikiPeople, following the standard conventions for these datasets (Galkin et al., 2020; Wang et al., 2021; Guan et al., 2019). For the *Scratch* setting, target lengths were sampled to match the distribution of fact lengths in the base sets. Table 14 shows the number of facts per length for each dataset. For the *Targeted* setting, we sampled 1,000 facts from the test set and masked all but one component (500 kept an entity, 500 kept a relation). For the *Arbitrary Masking* setting, we sampled 1,000 facts from the test set and randomly masked an arbitrary number of components (≥ 1).

Evaluation Protocol To ensure rigorous evaluation, we filtered out facts that are already present in the base sets or duplicates generated from the same query. The validity and the correctness of the generated facts were assessed as follows:

- **Incorrect:** Facts containing invalid entities or relations, or those that do not match the length of the query.
- **Correct:** Facts that exactly match the ground-truth. This includes matches in the test set for WD50K and WikiPeople$^-$, and matches in either the validation or test sets for WikiPeople.
- **LLM Verification:** For structurally valid but non-matching facts with the test set, GPT-5.2 acts as a judge. To provide information about the entities and relations within the generated fact, the judge is given a reference example fact for each component. We refer to the example facts for the relations as 'analogy facts' and the example facts for the entities as 'supplement facts', following (Wei et al., 2023). Figure 3 shows the prompt, which is adapted from (Wei et al., 2023). The judge classifies a fact as **Yes** if it is factually correct, **Half** if it is semantically valid but factually wrong, or **No** otherwise.

Accuracy is reported as the proportion of **Yes** labels, where **Correct** also counts as **Yes**.

H. Additional Qualitative Analysis of KREPE

Table 15 presents facts generated by KREPE on WD50K when varying numbers of qualifiers are specified. Consistent with the qualitative analysis in Section 5.5, KREPE produces correct and diverse facts for a given query with varying lengths.

Table 16 illustrates the step-by-step generation process of KREPE for a fact with two qualifiers. The process begins by predicting a qualifier entity, "31st Oscars", which establishes the topical context. This is followed by the primary relation, "awarded", narrowing the scope of the fact to awards. Subsequently, the head entity is generated, further resolving the fact to the movie "Gigi". Next, the model predicts the qualifier relation "subject of", linking it to the previously generated qualifier entity "31st Oscars". The second qualifier relation is then generated, followed by its associated entity. Finally, the tail entity is predicted to complete the fact.

Tables 17 and 18 present the top-3 nearest neighbors for the entity "Utah" and the relation "relative", respectively. We observe that the retrieved neighbors exhibit strong semantic similarity to the targets. For the entity "Utah", the top-3 similar entities are "Arizona", "New Mexico", and "Colorado"; notably, these are not only US states like "Utah" but are also its geographical neighbors. Similarly, for the relation "relative", the top-3 relations pertain to kinship and familial ties. These results suggest that the representations computed by KREPE capture meaningful semantic properties.

Table 15. Examples of facts generated by KREPE on WD50K, with zero, one, and two qualifiers.

Query	Generation
$\big((?,?,?), \{(?,?)\}_{i=1}^{n_\xi}\big)$	$\big$((Benjamin Franklin, occupation, writer), $\{\}\big)$
	$\big$((L. Messi, participant in, 2008 Champions League), $\{$(team, F.C. Barcelona)$\}\big)$
	$\big$((Gigi, awarded, Oscar Best Adapted Screenplay), $\{$(subject of, 31st Oscars), (winner, A. J. Lerner)$\}\big)$

Table 16. A step-by-step generation trajectory for the fact $\big$((Gigi, awarded, Oscar Best Adapted Screenplay), $\{$(subject of, 31st Oscars), (winner, A. J. Lerner)$\}\big)$ by KREPE on WD50K.

$\big((?, ?, ?), \{(?, ?), (?, ?)\}\big)$
$\big((?, ?, ?), \{(?$, 31st Oscars), $(?, ?)\}\big)$
$\big((?$, awarded, $?), \{(?$, 31st Oscars), $(?, ?)\}\big)$
$\big($(Gigi, awarded, $?), \{(?$, 31st Oscars), $(?, ?)\}\big)$
$\big($(Gigi, awarded, $?), \{$(subject of, 31st Oscars), $(?, ?)\}\big)$
$\big($(Gigi, awarded, $?), \{$(subject of, 31st Oscars), (winner, $?)\}\big)$
$\big($(Gigi, awarded, $?), \{$(subject of, 31st Oscars), (winner, A. J. Lerner)$\}\big)$
$\big($(Gigi, awarded, Oscar Best Adapted Screenplay), $\{$(subject of, 31st Oscars), (winner, A. J. Lerner)$\}\big)$

Table 17. Top 3 similar entities for target entity "Utah" based on the final representations from KREPE on WD50K.

Target		Similar Entities
Utah	#1	Arizona
	#2	New Mexico
	#3	Colorado

Table 18. Top 3 similar relations for target relation "relative" based on the final representations from KREPE on WD50K.

Target		Similar Relations
relative	#1	sibling
	#2	family
	#3	child

I. Details of the Proposed Baselines for Fact Generation

Details of the baselines for fact generation in Section 5.2 are provided here. For all LLM-based baselines, entity and relation IDs were converted to natural language descriptions for prompting. We utilized GPT-5.2 and Gemini 3.0 Pro with a default temperature of 1.0. For baselines utilizing MAYPL (Lee & Whang, 2025), we employed an instance of MAYPL retrained to support both entity and relation predictions. Note that we refer to the union of training and validation sets for "base sets" on WD50K and WikiPeople$^-$, while we strictly refer to the training set for WikiPeople.

Iterative Prediction This baseline represents a straightforward adaptation of a discriminative HKG model for generation. All missing components are randomly initialized and sequentially replaced with the top-1 prediction of MAYPL, obtained by masking the target component. We perform 11 cycles of updates following (Su et al., 2018), where the first 10 cycles serve as a burn-in period, and the state after the final cycle is taken as the output.

Gibbs Sampling Adapted from the text generation method in (Su et al., 2018), this baseline employs Gibbs sampling. Starting from a fact where missing components are randomly initialized while known components are preserved, we perform 11 cycles of updates. The first 10 cycles serve as burn-in, where each missing component is sequentially masked and replaced by probability-based random sampling from the top-5 predictions of MAYPL. The state after the final cycle was taken as the output.

Re-ranking This baseline adapts the retrieve-then-rerank strategy utilized by LLMs for KG link prediction. Specifically, it first utilizes a fact from the base set that satisfies the query to initialize the missing components. If no such fact exists, the missing components are randomly initialized. Then, for each originally missing component, MAYPL suggests the top-5 candidates, from which the LLM selects the best fit to construct the final fact. The prompts are adapted from (Wei et al., 2023); Figure 4 illustrates the prompt used across all settings.

Neighbor Sets Inspired by the neighbor-based augmentation strategy of MuKDC (Li et al., 2024a), this baseline utilizes all entities and relations sharing a fact with any known component within the query for generation. In the *Scratch* setting where no known components exist, a random entity is sampled to initiate the process. We provide the prompt templates for each setting in Figures 5, 6, and 7.

Few-shot Facts This baseline is similar to the *Neighbor Sets* baseline, but instead utilizes 30 training facts involving the known components as few-shot examples. Figures 8, 9, and 10 present the prompts utilized for the *Scratch*, *Targeted*, and *Arbitrary Masking* settings, respectively.

Random Facts The LLM is provided with a global pool of 1,000 facts randomly sampled from the base set and instructed to generate new facts using only the observed entities and relations. The prompt for the *Scratch* setting is shown in Figure 11, whereas Figure 12 and Figure 13 shows the prompt for the *Targeted* and *Arbitrary Masking* settings, respectively.

Autoregressive Generation proceeds autoregressively starting from the head entity. If no known components are given, a random entity is used as the starting head. The provided examples are dynamic: the first step uses 30 examples containing known components, while subsequent steps retrieve 30 examples involving the immediately preceding predicted element. The specific prompts for the *Scratch*, *Targeted*, and *Arbitrary Masking* settings are illustrated in Figures 14, 15, and 16, respectively.

We provide the implementation of the baseline methods for fact generation at `https://github.com/bdi-lab/KREPE`.

Re-ranking Prompt Template

User:
You are an expert for Knowledge Graph Completion tasks. Your goal is to perform link prediction. This involves filling in a missing element (denoted as [MASK]) in a hyper relational fact. The missing element could be an Entity or a Relation. Given a goal statement with a [MASK] and a list of candidate answers, you need to rank the candidates based on plausibility. If you understand your responsibility, respond "Yes". Otherwise, respond "No". Do not output anything except "Yes" and "No".

Model:
Yes

User:
To sort the candidate answers, you need to refer to other examples that may be similar or related to it. Some of the given examples are similar to the goal statement. You should draw analogies from them to understand the potential meaning of the goal statement. Other provided facts contain supplementary information; capture this extra information and mine potential relationships among them to help the sorting. Please carefully read, analyze, and reflect on these examples. Identify the reasoning patterns demonstrated in these examples and retain any information that may help your verification task. While I provide examples, please remain silent until I ask you to respond.

Model:
I have received your instructions and the goal statement.

User:
Examples used for analogy:
Predict the [MASK] from the given "{analogy_fact#1}". The answer is {ans}, so the [MASK] is {ans}.
...

Examples used to supplement information:
Predict the [MASK] from the given "{supplement_fact#1}". The answer is {ans}, so the [MASK] is {ans}.
...

Keep thinking, but DO NOT give me any feedback.

Model:
I have analyzed the examples provided and will keep thinking without giving feedback.

User:
The list of candidate answers is "{candidate_answer}" and the question is predict the [MASK] from the given "{masked_fact}". The goal is to verify the fact "{masked_fact}". Based on the previous examples and your own knowledge, determine the single most probable answer from the candidate list. Output ONLY the index of the best candidate based on the original list order.

Model:
0/1/2/3/4

Figure 4. The prompt used for fact generation across all settings in the *Re-ranking* baseline.

Neighbor Sets Prompt Template - Scratch

Role: You are an Expert Hyper-relational Fact Generator.

Task: Generate a hyper-relational fact with length {fact_length} containing the {entities_list} and {relations_list}

Constraint: Each fact must contain exactly {fact_length} elements. Only use the given entities and relations.

Output format: Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...

Figure 5. The prompt used for the *Neighbor Sets* baseline in the *Scratch* setting.

Neighbor Sets Prompt Template - Targeted

Role: You are an Expert Hyper-relational Fact Generator.

Available entities: {entities_list}
Available relations: {relations_list}

Task: Generate {num_candidates} DIFFERENT hyper-relational facts by completing the following MASKED fact. Each fact should be unique.

Masked Fact: {masked_fact}

Constraint:
- Each fact must contain exactly {fact_length} elements and have the same number of elements as the masked fact
- Position {anchor_idx} MUST be {anchor_element} (this is fixed and cannot be changed)
- Only use entities and relations that appear in the context facts above
- Generate {num_candidates} DIFFERENT facts

Output format: Provide exactly {num_candidates} facts, one per line:
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
...

Provide ONLY {num_candidates} completed facts, no explanations.

Figure 6. The prompt used for the *Neighbor Sets* baseline in the *Targeted* setting.

Neighbor Sets Prompt Template - Arbitrary Masking

Role: You are an Expert Hyper-relational Fact Generator.

Available entities: {entities_list}
Available relations: {relations_list}

Task: Generate {num_candidates} DIFFERENT hyper-relational facts by completing the following MASKED fact. Each fact should be unique.

Masked Fact: {masked_fact}

Constraints:
- Each fact must contain exactly {fact_length} elements and have the same number of elements as the masked fact
- The following positions are FIXED and CANNOT be changed:
{anchor_information}
- Only use entities and relations that appear in the available entities and relations above
- Generate {num_candidates} DIFFERENT facts

Output format: Provide exactly {num_candidates} facts, one per line:
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
...

Provide ONLY {num_candidates} completed facts, no explanations.

Figure 7. The prompt used for the *Neighbor Sets* baseline in the *Arbitrary Masking* setting

Few-shot Facts Prompt Template - Scratch

Role: You are an Expert Hyper-relational Fact Generator.

Here are {num_facts} existing facts for context:
{facts}

Task: Generate a NEW hyper-relational fact with length {fact_length}

Constraint: Each fact must contain exactly {fact_length} elements. Only use use entities and relations that appear in the facts above.

Output format: Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...

Figure 8. The prompt used for the *Few-shot Facts* baseline in the *Scratch* setting.

Few-shot Facts Prompt Template - Targeted

Role: You are an Expert Hyper-relational Fact Generator.

Here are {num_facts} existing facts for context:
{facts}

Task: Generate {num_candidates} DIFFERENT hyper-relational facts by completing the following MASKED fact. Each fact should be unique.

Masked Fact: {masked_facts}

Constraint:
- Each fact must contain exactly {fact_length} elements and have the same number of elements as the masked fact
- Position {anchor_index} MUST be {anchor_element} (this is fixed and cannot be changed)
- Only use entities and relations that appear in the context facts above
- Generate {num_candidates} DIFFERENT facts

Output format: Provide exactly {num_candidates} facts, one per line:
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
...

Provide ONLY {num_candidates} completed facts, no explanations.

Figure 9. The prompt used for the *Few-shot Facts* baseline in the *Targeted* setting.

Few-shot Facts Prompt Template - Arbitrary Masking

Role: You are an Expert Hyper-relational Fact Generator.

Here are {num_facts} existing facts for context:
{facts}

Task: Generate {num_candidates} DIFFERENT hyper-relational facts by completing the following MASKED fact. Each fact should be unique.

Masked Fact: {masked_facts}

Constraints:
- Each fact must contain exactly {fact_length} elements and have the same number of elements as the masked fact
- The following positions are FIXED and CANNOT be changed:
{anchor_information}
- Only use entities and relations that appear in the context facts above
- Generate {num_candidates} DIFFERENT facts

Output format: Provide exactly {num_candidates} facts, one per line:
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...
...

Provide ONLY {num_candidates} completed facts, no explanations.

Figure 10. The prompt used for the *Few-shot Facts* baseline in the *Arbitrary Masking* setting.

Random Facts Prompt Template - Scratch

Role: You are an Expert Hyper-relational Fact Generator.

Here are {num_facts} hyper-relational facts for context:
{facts}

Task: Generate {generate_size} NEW facts with target length {fact_length} based on this list.
Use entities, relations, and qualifier key-values only from the provided list.

Constraint: Each fact must contain exactly {fact_length} elements.

Output format: Subject ; Relation ; Object ; Qualifier relation ; Qualifier entity ; ...

Figure 11. The prompt used for the *Random Facts* baseline in the *Scratch* setting.

Random Facts Prompt Template - Targeted

Role: You are an Expert Hyper-relational Fact Generator.
Here are {num_facts} hyper-relational facts for context:
{facts}

Task: Complete the following {generate_size} MASKED facts. For EACH masked fact, generate {num_candidates} DIFFERENT hyper-relational facts by completing the following MASKED fact. Each fact should be unique.

Masked facts to complete:
{masked_fact}

Constraint:
- For each masked fact, maintain its exact length
- Keep non-MASK elements in their original positions
- Only use entities and relations from the context facts above
- Generate {num_candidates} different completions for EACH masked fact

Output format: For each masked fact, provide {num_candidates} completions, one per line:
Fact 1 - Completion 1: Subject ; Relation ; Object ; ...
Fact 1 - Completion 2: Subject ; Relation ; Object ; ...
...
Fact 1 - Completion {num_candidates}: Subject ; Relation ; Object ; ...
Fact 2 - Completion 1: Subject ; Relation ; Object ; ...
...

Provide ONLY completed facts in the format above, no explanations.

Figure 12. The prompt used for the *Random Facts* baseline in the *Targeted* settings.

Random Facts Prompt Template - Arbitrary Masking

Role: You are an Expert Hyper-relational Fact Generator.
Here are {num_facts} hyper-relational facts for context:
{facts}

Your task is to complete the following {generate_size} MASKED facts. For EACH masked fact, generate {num_candidates} DIFFERENT hyper-relational facts.
Each fact should be unique.

Masked facts to complete:
{masked_fact}

Constraints:
- For each masked fact, maintain its exact length
- Keep non-MASK elements in their original positions
- Only use entities and relations from the context facts above
- Generate {num_candidates} different completions for EACH masked fact

Output format: For each masked fact, provide {num_candidates} completions, one per line:
Fact 1 - Completion 1: Subject ; Relation ; Object ; ...
Fact 1 - Completion 2: Subject ; Relation ; Object ; ...
...
Fact 1 - Completion {num_candidates}: Subject ; Relation ; Object ; ...
Fact 2 - Completion 1: Subject ; Relation ; Object ; ...
...

Provide ONLY completed facts in the format above, no explanations.

Figure 13. The prompt used for the *Random Facts* baseline in the *Arbitrary Masking* settings.

Autoregressive Prompt Template - Scratch

User:
Role: You are an Expert Hyper-relational Fact Generator.

The HEAD entity has been selected: {head}
Here are facts where "{head}" appears:
{head_examples}

Task: Select ONE relation from the facts above that could connect to this head entity.
You must only use relations that appear in the provided facts.
Output ONLY the relation name, nothing else.

Model:
{relation}

User:
Current fact being built: {current_fact}
Here are facts where "{relation}" appears:
{relation_examples}

Task: Select ONE entity from the facts above that could connect to this relation.
You must only use entities that appear in the provided facts and the facts shown in previous steps.
Output ONLY the entity name, nothing else.

Model:
{tail}

User:
Current fact being built: {current_fact}
Here are facts where "{last_element}" appears:
{last_element_examples}

Task: Select ONE relation from the facts above that could connect to this entity.
You must only use relations that appear in the provided facts and the facts shown in previous steps.
Output ONLY the relation name, nothing else.

Model:
{qualifier_relation}

User:
Current fact being built: {current_fact}
Here are facts where "{qualifier_relation}" appears:
{qualifier_relation_examples}

Task: Select ONE entity from the facts above that could connect to this relation.
You must only use entities that appear in the provided facts and the facts shown in previous steps.
Output ONLY the entity name, nothing else.

Model:
{qualifier_entity}

...

Figure 14. The prompt used for the *Autoregressive* baseline in the *Scratch* setting.

Autoregressive Prompt Template - Targeted

User:
Role: You are an Expert Hyper-relational Fact Generator.

Current fact being built: {current_fact}

Here are facts where "{last_element}" appears:
{last_element_example}

Task: Select ONE {element_type} from the facts above that could fill the [FILLING NOW] position.
You must only use {element_type} that appear in the provided facts and the facts shown in previous steps.

Do not create same facts as in the examples.

Output ONLY the {element_type} name, nothing else.

Model:
{element}

...

Figure 15. The prompt used for the *Autoregressive* baseline in the *Targeted* setting.

Autoregressive Prompt Template - Arbitrary Masking

User:
Role: You are an Expert Hyper-relational Fact Generator.

Current fact being built: {current_fact}
Fixed positions (CANNOT change): {anchor_information}

Here are facts for context:
{context_fact}

Task: Select ONE {element_type} from the facts above that could fill the [FILLING NOW] position.
You must only use {element_type} that appear in the provided facts and the facts shown in previous steps.

Do not create same facts as in the examples.

Output ONLY the {element_type} name, nothing else.

Model:
{element}

...

Figure 16. The prompt used for the *Autoregressive* baseline in the *Arbitrary Masking* setting.