Representation Learning Based Predicate Invention on Knowledge Graphs

Man Zhu[1], **Pengfei Huang**[2], **Lei Gu**[3], **Xiaolong Xu**[1], **Jingyu Han**[1]

[1]School of Computer Science, Nanjing University of Posts and Telecommunications, China
[2]College of Integrated Circuits, Nanjing University of Aeronautics and Astronautics, China
[3]School of Computer Science and Engineering, Nanjing University of Science and Technology, China
mzhu@njupt.edu.cn

Abstract

The recognition of whether or not a predicate should be invented is an important problem in the domain of predicate invention. Despite its significance, existing research has yet to fully harness the rich data available in knowledge graphs. In this paper, we introduce a novel problem formulation, ReLPI (Representation Learning for Predicate Invention in Knowledge Graphs), which constitutes a significant advancement in this domain. To address the core issues of ReLPI, we devise a scoring function that informs the learning process. By optimizing embeddings towards this scoring function, we endow them with semantic meaning, crucial for capturing the nuances of predicate presence patterns. Furthermore, we present SEmPI (Semantic Embeddings for Predicate Invention), a framework that leverages predicate (relation) embeddings as a trainable medium. SEmPI uncovers latent patterns governing predicate occurrences in knowledge graphs, enabling the invention of novel predicates grounded in these discovered patterns. This approach represents a significant step forward in leveraging data-driven methods for predicate invention in knowledge graphs. We evaluate the proposed approach on FB15k and DRKG datasets, and the results demonstrate the effectiveness of SEmPI in discovering new predicates.

Introduction

The quality of learning within a knowledge base is constrained by its existing vocabulary. In cases where the vocabulary is restricted for a learning task, or when the task heavily depends on the available vocabulary in a complex manner, inventing predicates can lead to a more concise and precise model (Kramer 2020). Predicate invention (PI) that targets on the automatic invention of new auxiliary predicate symbols using existing vocabulary (Cropper and Dumančić 2022) is considered important since the founding of inductive logic programming. From another perspective, contemporary AI predominantly emphasizes inductive learning based on existing knowledge and deductive learning relying on existing vocabulary. Therefore, PI is also regarded as crucial to reach human-level AI, as it enables the rational "creation" of new knowledge built upon existing knowledge (Russell 2019). A significant hurdle within the field of Predicate Invention (PI) is undoubtedly the identification of the appropriate moments and underlying reasons for the creation of novel predicates, as highlighted by (Cropper and Dumančić 2022). To date, the potential of the extensive data within knowledge graphs remains largely untapped by existing research efforts.

PI has its roots deeply embedded in logics. In (Cropper and Morel 2021b), PI is said to be *necessary* when without it no solution can be found. Here, a solution refers to a hypothesis that "clarifies" both positive and negative examples in the knowledge base. Furthermore, PI is considered *useful* when it enables the learning of a superior solution. It is not hard to understand that PI is closely linked to logics, so it is straightforward that a large body of studies on PI utilize logic operators or employ schema-driven approaches to invent new predicates (Kramer 2020). These works inherit the merit of logics, thus are generally sound but challenging to scale.

In recent years, there has been a marked surge in the availability of structured data. This vast amount of structured data encompasses detailed conceptualizations of entities and their intricate relationships, enabling us to delve deeper into the patterns that underlie the presence of predicates within this data. This, in turn, offers us valuable insights into the model of predicate invention, which is grounded in conceptualization. To illustrate this concept, we present an exemplary scenario in Figure 1, demonstrating the predictability of an anonymous predicate's emergence based on the predicate invention model.

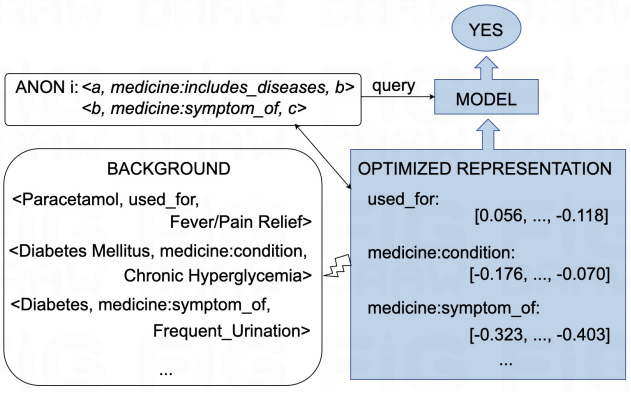

Figure 1: An illustrative scenario of ReLPI.

In this paper, we make three key contributions: 1) We formally introduce the ReLPI problem (Representation learning based predicate invention on knowledge graphs), which is crucial for predicate invention. This addresses the issue of uncovering patterns in the numerous predicates present in knowledge graphs, as illustrated in Figure 1. 2) To tackle ReLPI, we present SEmPI, an approach that harmonizes semantics and embeddings. This enables classifiers to be trained, leveraging representation learning centered around low-dimensional embeddings. As discussed in (Yang et al. 2015), embeddings demonstrate excellent scalability and reasoning capabilities for validating novel facts based on existing knowledge bases. 3) Through experiments, we demonstrate the efficacy of our framework, establishing a benchmark for the ReLPI problem.

Related Works

Works related to ours fall into two main categories: representation learning and predicate invention. Here, representation learning specifically refers to learning embeddings of entities in knowledge bases. In this section, we discuss these works respectively.

Knowledge Representation. Recent years we have seen a large body of works on learning low-dimensional embedding of entities and relations so that downstream tasks take these representations as input can carry on. Several surveys make more detailed discussions on this direction of researches (Ji et al. 2021)(Bianchi et al. 2020)(Cao et al. 2024). Most representation learning methods focus on link prediction, or KB completion, and the quality is generally based on estimators such as kernel density estimator (Bordes et al. 2011). Early studies (Bordes et al. 2013)(Wang et al. 2014)(Yang et al. 2015) mainly use real-valued point-wise space including vector, matrix and tensor space. Particularly, TransE (Bordes et al. 2013) embedded entities in a Euclidean vector space targeting $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, where $\mathbf{h}, \mathbf{r}$, and $\mathbf{t}$ are embeddings of subject, predicate, and object in a triple respectively. TransH (Wang et al. 2014) was proposed to tackle the difficulties of TransE in dealing with various kinds of relations, such as $1-n$, $n-1$, $n-n$, and reflexive, by adopting relation-specific hyperplanes. In this way, the distances are measured on projected vectors $||\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp||_2^2$. TransR (Lin et al. 2015) learn entity and predicate embeddings in different spaces, and the score function in TransR is defined by projected vectors of entities. DistMult (Yang et al. 2015) is a representative of learning model based on bilinear/linear transformations. In DistMult, relations are represented by matrices, and instead of additions, multiplications are used $\mathbf{h} * \mathbf{M_r} \approx \mathbf{t}$. DistMult is a well-performing knowledge graph embedding model (Kadlec, Bajgar, and Kleindienst 2017).

Additionally, other kinds of space including complex vector space, Gaussian space, and manifold are also utilized. ComplEx (Trouillon, Welbl, and Riedel 2017) is a simple approach based on complex embeddings, where only Hermitian dot product is used. RotatE (Sun et al. 2019) models the triples as rotations from the heads to the tails in the complex vector space, namely $\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$. Rather than using shallow parameter spaces, ConvE (Dettmers et al. 2018) utilized multi-layer convolutional network model to add more expressiveness with fewer parameters.

Multiple studies have fused the process of embedding learning with the application of rules. IterE employed a linear map assumption for embedding learning, so that rules can be derived based on the assumption (Zhang et al. 2019). ALPs (Dumancic et al. 2020) learned to encode relational data based on logic programs, which are then decoded to improve the performance of relational learning. IterE (Zhang et al. 2019) tried to combine embedding representation and rules to do knowledge graph reasoning. TransO (Li et al. 2023) is an approach that learn knowledge representation based on ontology information. INGRAM (Lee, Chung, and Whang 2023) answers queries of unknown predicates through aggregating weights from embeddings of adjacent entities at inference time.

Predicate Invention. The goal of predicate invention is for a system to automatically invent new auxiliary predicate symbols (Cropper and Dumančić 2022). It is one of the core challenges of Inductive Logic Programming (ILP) (Dumancic, Meert, and Blockeel 2016). Examples of ILP systems that support predicate invention are Metagol (Cropper, Morel, and Muggleton 2020), HEXMIL (Kaminski, Eiter, and Inoue 2018), Popper (Cropper and Morel 2021a). Among them, we can distinguish between reformulation approaches, demand-driven approaches and clause-refinement approaches, where a reformulation of an existing theory is done in any case to express the theory more compactly, in contrast to demand-driven approaches, that aim to complete a learning task when the given vocabulary is insufficient. Clause-refinement approaches aim to discriminate between positive and negative instances to refine a theory by which instances are incorrectly covered (Kramer 2020).

On the other hand, there are approaches that view this problem in other angles. For example, POPPI (Cropper and Morel 2021b) formulates PI as an answer set programming problem, which extends "learning from failures" by introducing redundancy constraints. NOPI (Cerna and Cropper 2024) combines negation and predicate invention to learn logic programs. (Silver et al. 2023) adopts PI in efficient planning to eliminate labor requirements. (Hocquette and Muggleton 2021) studies PI in meta-interpretive learning based on a bottom-up procedure so that background knowledge can be generalized.

We approach PI from another perspective, concentrating on the semantic connections between anonymous predicates and those already established, as informed by the statistical patterns discerned from knowledge graphs.

Preliminary

In this section, we introduce notions that are closely related to this paper.

Knowledge Graph

Definition 1 (Knowledge graph)**.** *A knowledge graph, denoted as $\mathcal{G}$, is a collection of structured facts typically in the form of $< s, p, o >$ triples $\subseteq E \times P \times (E \cup L)$, where E is a set of entities, P a set of predicates and L a set of literals.*

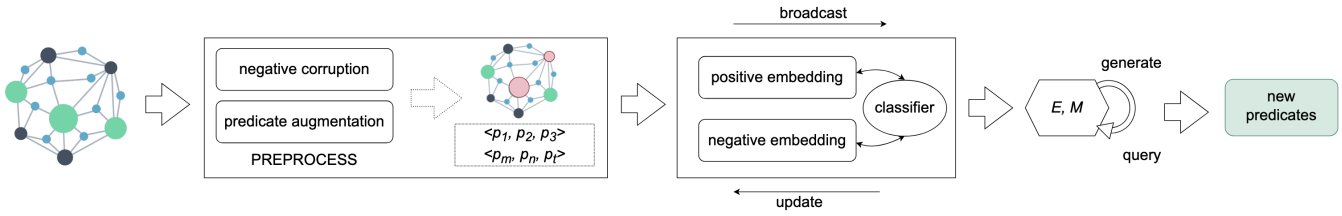

Figure 2: SEmPI (Semantic Embeddings for Predicate Invention) framework.

For example, $<$ New York, country, USA $>$ is a fact in a knowledge graph. It asserts that New York is located in a country named USA. New York and USA are entities, and country is a predicate. We have chosen to consistently employ the term "predicate" to represent both "relation" and "predicate", thereby eliminating any ambiguity that may have arisen from our previous interchangeable usage.

Rule

Now, we introduce the notion of a "rule" that is closely related to the proposed approach. A triple $< s, p, o >$ in $\mathcal{G}$ is taken as a ground *atom*, and is always written in $p(s, o)$ form, where $s, o \in E$, and $p \in P$. Based on this form, we can further generalize s, o by variables, and write a horn clause in its rule form, such as

$$\text{areaServed}(x, z) \leftarrow \text{serviceLocation}(x, y) \wedge \text{country}(y, z)$$

which can be read as "if x provides service in a location y, and y is a city in z, then x's service areas contain z". Here 'areaServed' is referred to as a **transitive predicate**, as it can be deduced through the transition of 'serviceLocation' and 'country'. Take this rule as an example, x, y, z are *variables*; areaServed, serviceLocation, and country are *predicates*. In this rule, x, z are universally quantified which means x, z can be substituted by every entity in E; y is existentially quantified meaning that at least an entity can be found in E to substitute y, because it only appears in the body of the rule. Given a rule, it is first instantiated with concrete entities in set E, resulting in a set of ground rules. Take this rule as an example, it might be instantiated with concrete entities of "ACME", "New York", and "USA", giving the ground rule areaServed(ACME, USA)$\leftarrow$ serviceLocation(ACME, New York) $\wedge$ country(New York, USA). A ground rule can then be interpreted as a complex formula, constructed by combining ground atoms with logical connectives (e.g. $\wedge$ and $\leftarrow$). In order to get explanations for the rule, *variables* x, y, z can be *substituted* by entities and predicates in a knowledge graph $\mathcal{G}$. Roughly speaking, a rule *holds* with respect to a knowledge base iff it is true for all ground rules. Since the description of this notion is much complex, and is beyond our focus, please refer to (Muggleton and Raedt 1994) for a more detailed explanation if needed.

Problem Formalization

We are working with a knowledge graph, denoted as $\mathcal{G}$, which is a structured representation of information where nodes represent entities and edges represent predicates between these entities. We transform the problem of predicate invention in knowledge graphs into query answering with predicate representations over a classification model. The classification model is trained to predict the presence of a predicate based on the characteristics of the current knowledge graph. Here we formalize the problem being researched in this paper as follows:

Definition 2 (Representation learning based predicate invention on knowledge graphs (ReLPI))**.** *Given a knowledge graph $\mathcal{G}$, learn representations $E = \{\mathbf{e}_p\}$ for the predicates $p \in P$ in $\mathcal{G}$, and train a classification model $\mathcal{M}$, such that new predicates p' can be represented by vectors within E, and by querying $\mathcal{M}$, the invention decision of p' is known. This training process involves minimizing a loss function L, which measures the discrepancy between the model's predictions and the true semantic representations of p':*

$$\arg\min_{\mathcal{M},E} L(\mathcal{M}, E, p').$$

where L is a predefined loss function.

The task at hand is to learn a set of vector representations, labeled as $E = \{\mathbf{e}_p\}$, for each predicate p that exists within the set of all predicates P in the knowledge graph. The purpose of these vector representations is to capture the semantic meaning of each predicate in a way that is computationally friendly for machine learning models. We are also aiming to train a model, referred to as $\mathcal{M}$, which will be responsible for generating these vector representations for not just the existing predicates in $\mathcal{G}$, but also for any new predicates, denoted as p', that may arise.

The training of model $\mathcal{M}$ is driven by a loss function L that measures how well the model can represent the semantics of new predicates p' using the learned vector space E. The ultimate goal is to adjust the parameters of both the model $\mathcal{M}$ and the vector representations E in such a way that the loss function is minimized. This means that the difference between the model's generated vector for a new predicate and the actual semantic representation of that predicate is as small as possible.

In essence, we are looking for the optimal combination of model parameters and vector representations that best captures the semantic relationships within the knowledge graph and can generalize to new, unseen predicates with minimal error, as defined by the loss function L.

Proposed Method

Here, we describe SEmPI (as shown in figure 2), an approach to solve the ReLPI problem. We introduce in three

parts: the encoding of predicates, the associated loss function, and the whole framework of SEmPI.

Semantic Embedding

In this paper, we learn representations for predicates in a manner that preserves their underlying logical structures. The logical structures that govern the interactions between predicates function both implicitly in the underlying fabric of the knowledge graph and in the discernible patterns of how these predicates manifest. In the following, we analyze relationships between rules and predicate representations.

Embeddings are low-dimensional representations for entities and predicates in a knowledge graph $\mathcal{G}$, demonstrating remarkable efficacy in tackling diverse challenges, such as knowledge graph completion (Ji et al. 2021). In this work, we specifically opt for embeddings as the representational framework for predicates, leveraging their ability to capture semantics in an efficient and scalable manner.

In SEmPI (Semantic Embeddings for Predicate Invention), entities within the set E are represented as vectors, specifically designated by $\mathbf{s}$ for subjects and $\mathbf{o}$ for objects, respectively. Predicates from the set P are encoded as diagonal matrices, denoted by M_p. To facilitate semantic computations leveraging these representations, we aspire to achieve a **learning objective** where the multiplication of a subject vector with its corresponding predicate matrix approximates the object vector, formalized as $\mathbf{s} \cdot M_p \approx \mathbf{o}$. This approach enables the system to comprehend and manipulate the relationships between predicates in a semantically meaningful manner, which we will discuss.

1. **Transitive predicate**. The embedding of a transitive predicate can be approximately represented based on the embeddings of the predicates that facilitate the transition of entities. Suppose a rule $p(a, c) \leftarrow p_1(a, b) \wedge p_2(b, c)$ holds in $\mathcal{G}$. M_{p_1}, M_{p_2} represent the embeddings of p_1, p_2 respectively, which satisfy the learning objective mentioned above. Then the embedding M_p for the predicate p in the head of the rule can be approximately represented by the composition of the embeddings M_{p_1} and M_{p_2}, denoted as $M_{p_1} \cdot M_{p_2}$, where the operation '$\cdot$' signifies a suitable composition function that captures the semantic interplay between p_1 and p_2 within the context of the composite rule p. In this paper, the operation '$\cdot$' specifically refers to matrix multiplication. The justification for this claim is clear and direct. Given that M_{p_1} and M_{p_2} represent the embeddings of p_1 and p_2, respectively, adhering to the learning objective, and $\mathbf{e}_a$, $\mathbf{e}_b$, and $\mathbf{e}_c$ are the corresponding embeddings of a, b, and c, it is evident that $\mathbf{e}_a$ transformed by M_{p_1} approximates $\mathbf{e}_b$, and similarly, $\mathbf{e}_b$ transformed by M_{p_2} approximates $\mathbf{e}_c$. Consequently, it is logical to conclude that sequentially applying M_{p_1} and M_{p_2} to $\mathbf{e}_a$ yields an approximation of $\mathbf{e}_c$, a.k.a., $\mathbf{e}_a \cdot M_{p_1} \cdot M_{p_2} \approx \mathbf{e}_c$. Reconsider the rule, it is clear that $< a, p, c > \in \mathcal{G}$, and consequently $\mathbf{e}_a \cdot M_p \approx \mathbf{e}_c$. If we further generalize this claim, we can derive an approximation for M_p in terms of the multiplication of M_{p_1}, M_{p_2}, ..., and M_{p_n} pertaining to the rule $p(b_0, b_n) \leftarrow p_1(b_0, b_1) \wedge p_2(b_1, b_2) \wedge \ldots \wedge p_n(b_{n-1}, b_n)$.
2. **Conjunctive predicate**. Suppose a rule $p(a, b) \leftarrow p_1(a, b) \wedge p_2(a, b)$ holds in $\mathcal{G}$, and let M_{p_1}, M_{p_2} represent the embeddings of p_1, p_2 respectively, , which adhere to the aforementioned learning objective. Then the embedding M_p for the predicate p in the head of the rule can be approximately represented by averaging the embeddings M_{p_1} and M_{p_2}, denoted succinctly as $\frac{1}{2}(M_{p_1} + M_{p_2})$. Moreover, if $p(a, b) \leftarrow p_1(a, b) \wedge p_2(a, b) \wedge \ldots \wedge p_n(a, b)$ holds, and $M_{p_1}, M_{p_2}, \ldots, M_{p_n}$ along with M_p are embeddings of $p_1, p_2, \ldots, p_n$, and p respectively, which have been learnt to satisfy the previous learning objective, then M_p can be approximately calculated by taking the average of the embeddings in the body of the rule, denoted as $(\frac{1}{n}(M_{p_1} + M_{p_2} + \ldots + M_{p_n}), M_p)$. The justification for this claim follows straightforwardly from the explanation provided for transitive predicates.
3. **Reverse predicate**. If a predicate is the inverse of another predicate, for instance, $p(a, b) \leftarrow p'(b, a)$, then the embedding of p corresponds to the inverse of the embedding p'. To justify it, suppose $\mathbf{e}_a$, $\mathbf{e}_b$, and $M_{p'}$ are embeddings of a, b, and p', then $\mathbf{e}_b \cdot M_{p'} \approx \mathbf{e}_a$, and $\mathbf{e}_a \cdot M_{p'}^{-1} \approx \mathbf{e}_b$.

Loss Function

In the ReLPI problem, in addition to semantic embeddings, a pivotal requirement necessitates ensuring that the parameters encapsulated within the model $\mathcal{M}$ and the embeddings residing in the set E are optimized to minimize a loss function. This loss function is designed to precisely quantify and diminish the gap between the predicted occurrence (or invention decision) of predicates and their actual existence in the data. Specifically, it aims to align the model's predictions with the ground truth, fostering accuracy and reliability.

To realize this objective, we formulate the parameter learning task as minimizing the loss function below:

$$L(\Omega) = \sum_{p \in P} log(1 + e^{(-y \cdot S(p, \mathcal{M}) \sum_{<s,p,o>} (\mathbf{e}_s \cdot M_p - \mathbf{e}_o)^2)}) \tag{1}$$

Here, P represents the comprehensive set of predicates, $S(p, \mathcal{M})$ is the score function of the classification model, which equals $(c(\mathcal{M}_p) - l(p))^2$. $c(\mathcal{M}_p)$ denotes the output from the classifier model indicating the predicted existence of predicate p given its embedding, while $l(p)$ is the binary label indicating the actual existence of p in the data. y is the label of the corresponding predicate. The first term of the loss function penalizes incorrect predictions of predicate existence, while the second term measures the discrepancy between the predicted relation embedding (given by the dot product of subject embedding $\mathbf{e}_s$ and the predicate-specific matrix M_p) and the observed object embedding $\mathbf{e}_o$ for each triple $< s, p, o >$ in the knowledge graph $\mathcal{G}$. This approach ensures that both the classification accuracy and the quality of the relation embeddings are optimized simultaneously.

The Learning Framework of SEmPI

In this part, we present the comprehensive learning framework titled SEmPI (Semantic Embeddings for Predicate Invention), which serves as an innovative approach to address

Algorithm 1: Predicate Augmentation

Input: $\mathcal{G}$, predicate set P, t_s, t_c
Output: predicate triples

cand$\leftarrow \emptyset$
for each p in P **do**
  Select the set of start relations $S = \{s : \mathcal{Y}_s \cap \mathcal{X}_r \neq \emptyset\}$
  Select the set of end relations $T = \{t : \mathcal{X}_t \cap \mathcal{Y}_r \neq \emptyset\}$
  for each possible relation sequence p_1 and p_2 that form rule r **do**
    if supp(r)$> t_s$ and conf(r)$> t_c$ **then**
      cand$\leftarrow < p_1, p_2, p >$
    end if
  end for
end for
return *cand*

the ReLPI problem. The end-to-end process is illustrated in figure 2, providing a clear representation of how SEmPI operates. We commence by outlining the architecture of the framework, followed by a detailed exposition of the key elements involved in each of its constituent steps, ensuring a thorough understanding of the methodology.

1. *Dataset preprocess*. There are two primary tasks in this step, specifically *predicate augmentation* and *negative corruption*. To reinforce the emphasis on the similarity between $\mathbf{e}_s \cdot M_p$ and $\mathbf{e}_o$, we augment the dataset through the incorporation of additional triples constructed by predicates. This augmentation process is guided by a rule induction-inspired procedure (Kumbhare and Chobe 2014), as outlined in algorithm 1. This process involves augmenting transitive predicates, enabling the natural learning of transitive embeddings.. In algorithm 1, $\mathcal{X}_r$ and $\mathcal{Y}_r$ denote the set of subject entities and object entities of predicate r respectively. A candidate is incorporated into the set if its support and confidence exceed the predefined thresholds. To generate negative examples for training the classification model and enrich the representation of the non-existence category, we introduce negative corruptions into the knowledge graph. Our strategy involves the following steps: Initially, we randomly select k predicates to corrupt and place them in a set R. For each $r \in R$, we fetch all triples associated with it, denoted as $T_r = \{< e, p, l > \mid < e, p, l > \in \mathcal{G}, p = r\}$. Subsequently, we corrupt either the subject e or the object l in each triple by replacing it with another randomly selected entity, following a $(0-1)$ distribution where θ represents the probability of replacing either e or l. The triples in the knowledge graph are corrupted by predicates according to parameters k and θ. Then the results after negative corruption and predicate augmentation are put into the knowledge graph.
2. *Model training*. As shown in figure 3 and algorithm 2, the parameters are learned in a propagate - update iterative manner. Positive and negative examples are embeddings with TRUE and FALSE class labels for exist and non-exist categories. In step 1, corrupted predicates are generated according to given parameters. While learning

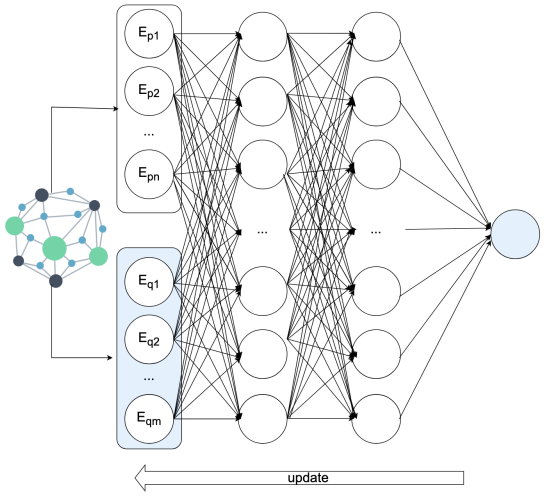

Figure 3: Learning procedure for ReLPI.

Algorithm 2: SEmPI Training

Input: triple set T, corrupted predicate set P_{cor}, $\mathcal{G}$
Output: $\mathcal{M}$, $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{e}_p$ and $\mathbf{e}_{pcor}$

initialize a DNN model $\mathcal{M}$
initialize embeddings for entities and predicates $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{e}_p$ and $\mathbf{e}_{pcor}$
while not stop criterion
  learn or update a classifier $\mathcal{M}$ with $\{\mathbf{e}_p\}$, and $\{\mathbf{e}_{pcor}\}$
  for each $p \in P$, and P_{cor}
    for each $< s, p, o > \in \mathcal{G}$
      score$+ = ||\mathbf{e}_s \cdot M_p - \mathbf{e}_o||_2$
    end for
    score$=$ score $\times\ ||c(\mathcal{M}(p)) - l(p)||_2$.
  end for
  calculate loss function and update $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{e}_p$ and $\mathbf{e}_{pcor}$ according to the gradients
end while
return $\mathcal{M}$, $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{e}_p$ and $\mathbf{e}_{pcor}$

representations, negative examples are these corrupted predicates. To optimize the training process, a comprehensive stop criterion is employed, encompassing both an early stopping mechanism and a pre-established epoch threshold. We design a deep neural network based prediction model which computes the probability of a INVENT decision for a given (embedding, label) pair. The labels indicate whether the predicate denoted by the corresponding embedding exists or not, which is inherited from negative corruptions. The prediction model is a 4-layer multi-perceptron (MLP) architecture with a Rectified Linear Unit (ReLU) activation function.

3. *Prediction*. Inspired by (Cropper and Morel 2021b), where PI always leads to shorter programs, we generate candidate unknown predicates to simplify the knowledge graph. As discussed in the previous section, transitive, conjunctive, inverse embeddings can be calculated by simple matrix operations on embeddings learnt in step

2. Actually, the procedure to generate embeddings for the candidate unknown predicates is not the research problem trying to be solved in this paper. Any procedure that generate candidate unknown predicates can be used here. In this paper, we generate candidates based on the procedure described in (Yang et al. 2015). After the candidates are generated, we query for the INVENT decision by the classification model learnt in step 2, and the results show the prediction decision for invention.

Experiments

We have designed and executed a series of experiments to address the following inquiries: [Q1] Is the scoring function effective in capturing both semantics in embeddings and regularities underlying the existence of predicates? [Q2] Does SEmPI effectively classify the existence of predicates? [Q3] Can the model we have trained accurately predict the existence of unknown predicates?

Datasets. Our experiments have been thoroughly conducted on the widely recognized FB15k and DRKG datasets, offering a robust foundation for our analysis. In table 1, we present comprehensive statistics about these datasets.

Implementation and settings. We implement the whole framework based on Ampligraph (Costabello et al. 2019) using Python. The evaluations are carried on a server using 22 vCPU AMD 64-Core processor with 90GB RAM and 1 GPU RTX 4090 with 24GB VRAM. The operation system is Ubuntu 18.04.

To assess the prediction performance with a robust methodology, we implement a 10-fold cross-validation approach on the datasets. Prior to initiating the validation process, we refine the predicates by subjecting them to a filtering process. This entails verifying whether each predicate can be derived or reached via three primary routes: either through the transitive closure of two predicates, a conjunction of two predicates, or by inverting (taking the reverse) of another predicate. By incorporating this filtering step, we ensure that only meaningful and logically derived predicates contribute to the evaluation.

To create negative examples, we use a technique that involves randomly picking a predetermined number of predicates. After selecting these predicates, we intentionally modify (corrupt) the subject (head) and object (tail) of the triples that contain them, using a probabilistic approach based on a Bernoulli distribution with a specified probability p. This probability p determines how likely it is for an entity to be altered. Any predicate that undergoes this corruption process is labeled as FALSE, while those that remain unchanged are labeled as TRUE. This newly crafted dataset, comprising both the original and corrupted triples, is then utilized to refine the model's parameters through fine-tuning.

Dataset	#ent.	#pre.	#tri.
FB15k	14,951	1,345	592,213
DRKG	97,238	107	5,874,261

Table 1: Dataset statistics: number of entities, predicates, and triples

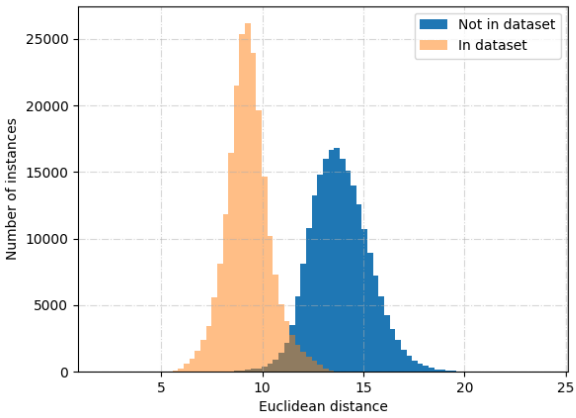

Figure 4: Triple count comparison based on distance between $\mathbf{s} \cdot M_p$ and $\mathbf{o}$ for triples [Present] and [Absent] in FB15k dataset.

To comprehensively evaluate SEmPI, we benchmark it against variations of three prevalent embedding learning models: TransE, DistMult, and ComplEx. The variations are as follows: Firstly, we individually train each model to learn embeddings for the data. Following this, in the second step, we employ a Deep Neural Network (DNN) as a classifier to determine the existence of predicates within the data. In this section, the embedding learning models for predicate invention are named TransE-DNN, DistMult-DNN, and ComplEx-DNN. This two-step approach allows us to assess the performance of SEmPI in comparison to the other embedding learning based methods. In all the methods, we use the same DNN (deep neural network)-based classification model. The model is a 4-layer multi-perceptron architecture with a Rectified Linear Unit (ReLU) activation function, and which is optimized with Adam optimizer. We also tried existing PI approaches discussed in related works section, namely Metagol, HEXMIL, and Popper. They turned out to be effective in logic abundant smaller datasets, while ours is more applicable on fact abundant larger datasets.

To evaluate the performance of the model, we utilize precision, recall, and F1 measures as our key evaluation metrics.

Results

The key advantage of the scoring function introduced in this paper lies in its ability to capture semantic information encoded within the embeddings by minimizing the distance between $\mathbf{s} \cdot M_p$ and $\mathbf{o}$. To answer [Q1], Figure 4 provides a clear visualization of how the count of triples varies based on their presence or absence in the dataset, plotted against the Euclidean distance calculated using their respective embeddings on the x-axis. This figure illustrates the effectiveness of our scoring function in distinguishing between existing and non-existing triples by leveraging the semantic relationships captured in the embeddings.

In the evaluations, both entity and predicate embeddings have been set to a dimension of 200. The probability param-

Model	FB15k	DRKG
SEmPI	1528	9890
TransE	1437	9031
DistMult	1544	9356
complEx	2546	10323

Table 2: Comparisons of training time in each epoch of different models. The time is measured in milliseconds.

eter θ was set to 0.2 to achieve a balanced trade-off between mitigating noise issues and maintaining the learnability of the embeddings. We selected 500 predicates to corrupt in FB15k dataset, and 45 predicates to corrupt in DRKG. The results are shown in Figure 5. Figure 5 provides an insightful comparison of the performance metrics achieved by our proposed method, SEmPI, alongside TransE-DNN, DistMult-DNN, and ComplEx-DNN. The results underscore the superiority of SEmPI in terms of recall, demonstrating its ability to discover a wider range of relevant entities and predicates during the 10-fold cross-validation process. This enhanced recall performance can be attributed to SEmPI's unique capability to infer missing predicates that may not be explicitly present in the original datasets. However, it is important to note that this increase in recall comes at the cost of a slight decrease in precision, as some of the newly discovered predicates may not be perfectly aligned with the dataset, thereby not contributing to the precision score. Nonetheless, the overall performance of SEmPI highlights its potential to improve knowledge graph completion and enhance semantic representation. The training time of the whole framework is shown in table 2.

In order to answer [Q3], we follow the procedure of knowledge graph preparation described earlier in this paper to generate candidate predicates. We extract 610 possible candidates with 2 relation sequences from the FB15k dataset. We calculate their embeddings which are then put into the trained neural network for classification. We artificially evaluate the results given by the neural network model. Table 3 shows the results of prediction evaluations on the classification model learned by our framework according to the artificial evaluation. The precision of prediction of the SEmPI model reaches 0.82. Additionally, in table 4, we give

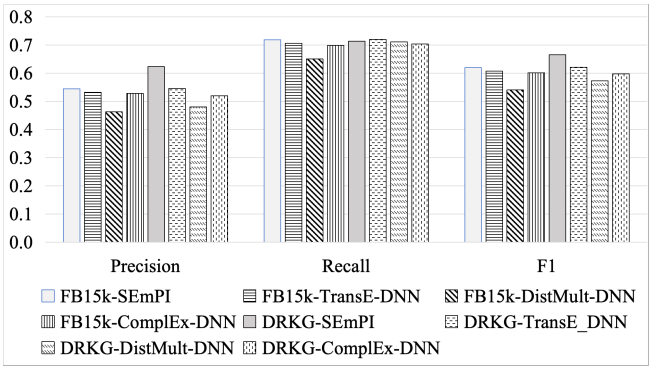

Figure 5: Prediction performance comparisons.

	accuracy	precision	recall	F1
FB15k	0.674	0.820	0.724	0.779

Table 3: Performance of prediction of the trained SEmPI model.

several examples of artificially categorized to be invented predicates of length 2.

Conclusion

In this paper, we delve into the challenge of predicate presence identification. Given the current deluge of data generation, we leverage the robust capabilities of knowledge graphs to address this problem. Our methodology commences with a rigorous formalization of the problem, followed by the introduction of an innovative framework that iteratively embeds entities and predicates (relations) within a semantic context. This approach enables us to develop a classifier capable of identifying previously unseen predicates, thereby enhancing the richness and accuracy of knowledge graphs. We comprehensively evaluate the performance of our classification model from two distinct perspectives. Firstly, we demonstrate the model's proficiency in preserving semantic integrity, ensuring that the embedded representations accurately reflect the underlying meaning of entities and predicates. Secondly, we rigorously assess the effectiveness of our approach in predicate invention by employing a rigorous 10-fold cross-validation scheme.

The lack of comprehensive studies surrounding the problem we address in this work highlights several promising avenues for future research. Here are two key directions we identify: Firstly, exploring a diverse array of loss functions can potentially unlock significant improvements. Secondly, while our model demonstrates a commendable ability to preserve semantic information, there remains ample opportunity to further refine our approach in this regard.

$p_1(a,b)$	$p_2(b,c)$
lang:countries_spoken_in	loc:official_language
loc:partially_contains	loc:second_level_divisions
martial_arts:martial_art	olympics:country
medicine:includes_diseases	medicine:symptom_of
medicine:used_to_treat	medicine:causes
music:artists	music:album
music:instruments_played	music:artists
olympics:country	olympics:sport
basketball:players	basketball:team
architecture:architect	peo:place_lived
award:honored_for	film:production_companies
peo:people_with_this_prof	business:title
film:language	loc:languages_spoken

Table 4: Examples of invented predicates. The URIs are abbreviated by the namespaces.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62472235.

References

Bianchi, F.; Rossiello, G.; Costabello, L.; Palmonari, M.; and Minervini, P. 2020. Knowledge Graph Embeddings and Explainable AI. ArXiv:2004.14843 [cs].

Bordes, A.; Usunier, N.; Garcia-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 2787–2795. Red Hook, NY, USA: Curran Associates Inc.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning Structured Embeddings of Knowledge Bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 25(1): 301–306. Number: 1.

Cao, J.; Fang, J.; Meng, Z.; and Liang, S. 2024. Knowledge graph embedding: A survey from the perspective of representation spaces. *ACM Computing Surveys*, 56(6): 1–42.

Cerna, D. M.; and Cropper, A. 2024. Generalisation through negation and predicate invention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 10467–10475.

Costabello, L.; Pai, S.; Le Van, C.; McGrath, R.; McCarthy, N.; and Tabacof, P. 2019. AmpliGraph: a library for representation learning on knowledge graphs. *Retrieved Oct*, 10: 2019.

Cropper, A.; and Dumančić, S. 2022. Inductive logic programming at 30: a new introduction. *Journal of Artificial Intelligence Research*, 74: 765–850.

Cropper, A.; and Dumančić, S. 2022. Inductive logic programming at 30: a new introduction. ArXiv:2008.07912 [cs].

Cropper, A.; and Morel, R. 2021a. Learning programs by learning from failures. *Machine Learning*, 110(4): 801–856.

Cropper, A.; and Morel, R. 2021b. Predicate Invention by Learning From Failures. ArXiv:2104.14426 [cs].

Cropper, A.; Morel, R.; and Muggleton, S. H. 2020. Learning Higher-Order Programs through Predicate Invention. *AAAI*, 34(09): 13655–13658.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). Number: 1.

Dumancic, S.; Guns, T.; Meert, W.; and Blockeel, H. 2020. Learning Relational Representations with Auto-encoding Logic Programs. ArXiv:1903.12577 [cs, stat].

Dumancic, S.; Meert, W.; and Blockeel, H. 2016. Theory reconstruction: a representation learning view on predicate invention. arXiv:1606.08660.

Hocquette, C.; and Muggleton, S. H. 2021. Complete bottom-up predicate invention in meta-interpretive learning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2312–2318.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2): 494–514.

Kadlec, R.; Bajgar, O.; and Kleindienst, J. 2017. Knowledge Base Completion: Baselines Strike Back. arXiv:1705.10744.

Kaminski, T.; Eiter, T.; and Inoue, K. 2018. Exploiting answer set programming with external sources for meta-interpretive learning. *Theory and Practice of Logic Programming*, 18(3-4): 571–588.

Kramer, S. 2020. A Brief History of Learning Symbolic Higher-Level Representations from Data (And a Curious Look Forward). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 4868–4876. Yokohama, Japan: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5.

Kumbhare, T. A.; and Chobe, S. V. 2014. An overview of association rule mining algorithms. *International Journal of Computer Science and Information Technologies*, 5(1): 927–930.

Lee, J.; Chung, C.; and Whang, J. J. 2023. InGram: Inductive knowledge graph embedding via relation graphs. In *International Conference on Machine Learning*, 18796–18809. PMLR.

Li, Z.; Liu, X.; Wang, X.; Liu, P.; and Shen, Y. 2023. Transo: a knowledge-driven representation learning method with ontology information constraints. *World Wide Web*, 26(1): 297–319.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Muggleton, S.; and Raedt, L. d. 1994. Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming*, 19-20: 629–679.

Russell, S. 2019. *Human compatible: Artificial intelligence and the problem of control*. Penguin.

Silver, T.; Chitnis, R.; Kumar, N.; McClinton, W.; Lozano-Pérez, T.; Kaelbling, L.; and Tenenbaum, J. B. 2023. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 12120–12129.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. ArXiv:1902.10197 [cs, stat].

Trouillon, T.; Welbl, J.; and Riedel, S. 2017. Complex Embeddings for Simple Link Prediction. In *ICML*.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. *AAAI*, 28(1).

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. ArXiv:1412.6575 [cs].

Zhang, W.; Paudel, B.; Wang, L.; Chen, J.; Zhu, H.; Zhang, W.; Bernstein, A.; and Chen, H. 2019. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The world wide web conference*, 2366–2377.