World Wide Web (2022) 25:1037–1065
https://doi.org/10.1007/s11280-021-00972-6

# HOPLoP: multi-hop link prediction over knowledge graph embeddings

Varun Ranganathan[1] · Denilson Barbosa[1]

Received: 1 February 2021 / Revised: 20 July 2021 / Accepted: 20 October 2021 /
Published online: 11 December 2021

## Abstract

Large-scale Knowledge Graphs (KGs) support applications such as Web search and personal assistants and provide training data for numerous Natural Language Processing tasks. Nevertheless, building KGs with high accuracy and domain coverage remains difficult, and neither manual nor automatic efforts are up to par. Link Prediction (LP) is one of many tasks aimed at addressing this problem. Its goal is to find missing links between entities in the KG based on structural by exploiting regularities in the graph structure. Recent years have seen two approaches emerge: using KG embeddings, and modelling complex relations by exploiting correlations between individual links and longer paths connecting the same pair of entities. For the latter, state-of-the-art methods traverse the KG itself and are hampered both by incompleteness and skewed degree distributions found in most KGs, resulting in some entities being overly represented in the training set leading to poor generalization. We present HOPLoP: an efficient and effective multi-hop LP *meta* method that performs the equivalent to path traversals on the KG embedding space instead of the KG itself, marrying both ideas. We show how to train and tune our method with different underlying KG embeddings, and report on experiments on many benchmarks, showing both that HOPLoP improves each LP method on its own and that it consistently outperforms the previous state-of-the-art by a good margin. Finally, we describe a way to interpret paths generated by HOPLoP when used with TransE.

**Keywords** Link Prediction · Knowledge Graph Embeddings · Multi-hop reasoning

This article belongs to the Topical Collection: *Special Issue on Large Scale Graph Data Analytics*
Guest Editors: Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang

✉ Denilson Barbosa
denilson@ualberta.ca

Varun Ranganathan
vrangana@ualberta.ca

[1] Department of Computing Science, University of Alberta, Edmonton, AB, Canada

## 1 Introduction

Open and large-scale Knowledge Graphs (KGs) built from the Web, such as Wikidata [69], Freebase [6], YAGO [62], to name just a few, power important tasks such as question answering [45] and automated reasoning [55]. These tasks require KGs that are both *accurate* and *comprehensive*, with extensive coverage of topics of interest to users.

Building KGs that are accurate and complete is difficult. On one hand, manually constructing KGs lead to accuracy that is high enough to support these tasks but requires experts and takes too much time [3]. For example, developing WordNet, a manually constructed KG with information about 155,287 English words and phrases, which are linked via semantic relations such as *Hypernymy* and *Synonymy*, took 27 years. And yet, it covers only a small fraction of the ever-changing English language. On the other hand, KGs constructed automatically with the help of Information Extraction (IE) tools [43] applied to Web-scale text corpora cover many topics but are less accurate as the state-of-the-art is not sophisticated enough to understand the nuances of natural language [2, 49].

Even when they succeed, IE methods can only capture facts explicitly mentioned in the text. For example, consider the following excerpt from an article in Wikipedia describing the movie “The Terminator”:[1]

> The Terminator is a 1984 science fiction action film released by Orion Pictures, co-written and directed by James Cameron and starring Arnold Schwarzenegger, Linda Hamilton and Michael Biehn. It is the first work in the Terminator franchise. ...

At the time of writing, the best IE methods applied to that text would find facts mentioned explicitly such as (James Cameron, Director Of, The Terminator) and (The Terminator, Genre, Science Fiction), but would miss facts that are implied, such as (James Cameron, Directs Genre Of Movies, Science Fiction). We need a way to close that gap, for example by exploiting correlations between ($x$, Directs Genre Of Movies, $z$) edges and paths of the form ($x$, Director Of, $y$), ($y$, Genre, $z$) connecting the same pairs of entities.

### 1.1 Building knowledge graphs

Building accurate and comprehensive KGs is often done through a combination of *extraction* methods that capture facts explicitly expressed in the source, and *inference* methods that can derive implicit facts missed by the extractors. A promising strategy for inferring missing facts that emerged in recent years is called Link Prediction (LP). The field of LP has been dominated by methods based on KG embeddings [8, 42, 56], since they provide an elegant solution to the incompleteness problem. By controlling the dimensionality of the embedding space, these models are able to generalize to unseen facts [20]. However, these consider only two entities and one relation at a time to make a prediction. Thus, they cannot model complex relations and reason over multiple relations to solve our problem.

**Multi-hop inference for link prediction** There is strong evidence in the literature showing that leveraging graph traversals as features can help with LP as it allows *correlations* between paths and direct relations [35] to be exploited. *Multi-hop* link prediction algorithms leverage structural information gathered through exploring the graph looking for paths

[1] https://en.wikipedia.org/wiki/Terminator_(franchise)#The_Terminator_(1984)

connecting pairs of entities involved in a given relation [36]. Although these algorithms greatly help with LP, existing methods perform the traversals on the KG itself and are thus constrained by its deficiencies. KGs are notoriously incomplete and have highly skewed degree distribution of nodes, resulting in some nodes being over represented in training and leading to models that do not generalize well. For example, at the time of writing, about half of all Citizenship (wdt:P27) facts in Wikidata involve just 10 countries, with the United States (wd:Q30) alone having 13.7% of all facts. LP methods that exploit graph traversals are likely to attend to those paths related to the United States disproportionately more than other countries.

In passing, we also note that this data skew also bias embedding-based LP methods towards the so-called *supernodes* which are over-represented during training. What is worse, the evaluation of LP methods is based on removing individual relations (i.e., single edges) from the KG and attempting to re-construct them. This results in supernodes appearing disproportionately more in the validation sets used in the literature. As a result, many authors report impressive, yet misleading, results [56].

### 1.1.1 Motivation

We hypothesize that the performance of a LP algorithm can be improved if it is allowed to leverage graph traversals that are *not* constrained by the KG itself. To that end, we develop a method in which graph "traversals" and "paths" are defined over the *embedding* space where the entities are represented.

To verify our hypothesis, we introduce a simple yet efficacious multi-hop link prediction framework called HOPLoP which is, in effect, a meta-algorithm that can be used with any underlying embedding method. HOPLoP is an end-to-end differentiable multi-hop LP framework that learns to traverse a KG in the embedded space while distinguishing between existent and non-existent links of the KG. By doing so, HOPLoP is able to recognize and account for errors in the embedded representation of the KG and create appropriate decision boundaries, which leads to performance boosts in LP.

Figure 1 illustrates a "traversal" on an embedding space resulting from a geometric method such as TransE [7]. Recall that in such an embedding, relations (represented as dashed lines in the figure) act as constraints on the positioning of the entities which tend to be clustered together by type. Assume we are given many pairs of directors and the genres of their movies; for the sake of the example, assume we are given the fact that James Cameron directs Science Fiction movies but not Drama movies. HOPLoP's pathfinder will search for both paths associated with positive examples (shown in green above) and paths associated with negative examples (red). As illustrated in the figure, a "traversing" path of length $k$ amounts to applying the $k$ corresponding vectors to the point in space representing the starting entity. In the case of TransE, we do that by adding vectors. Note that the end-point of each application may not correspond to an actual entity in the graph, but that is not a concern: HOPLoP learns to traverse such a partial representation of the KG to accurately distinguish between positive and negative examples.

**Contributions** Our contributions are as follows:

1. We propose an end-to-end differentiable multi-hop link prediction framework for LP over large KGs. End-to-end differentiability improves computational efficiency, allowing for simpler optimization algorithms such as Gradient Descent [57].

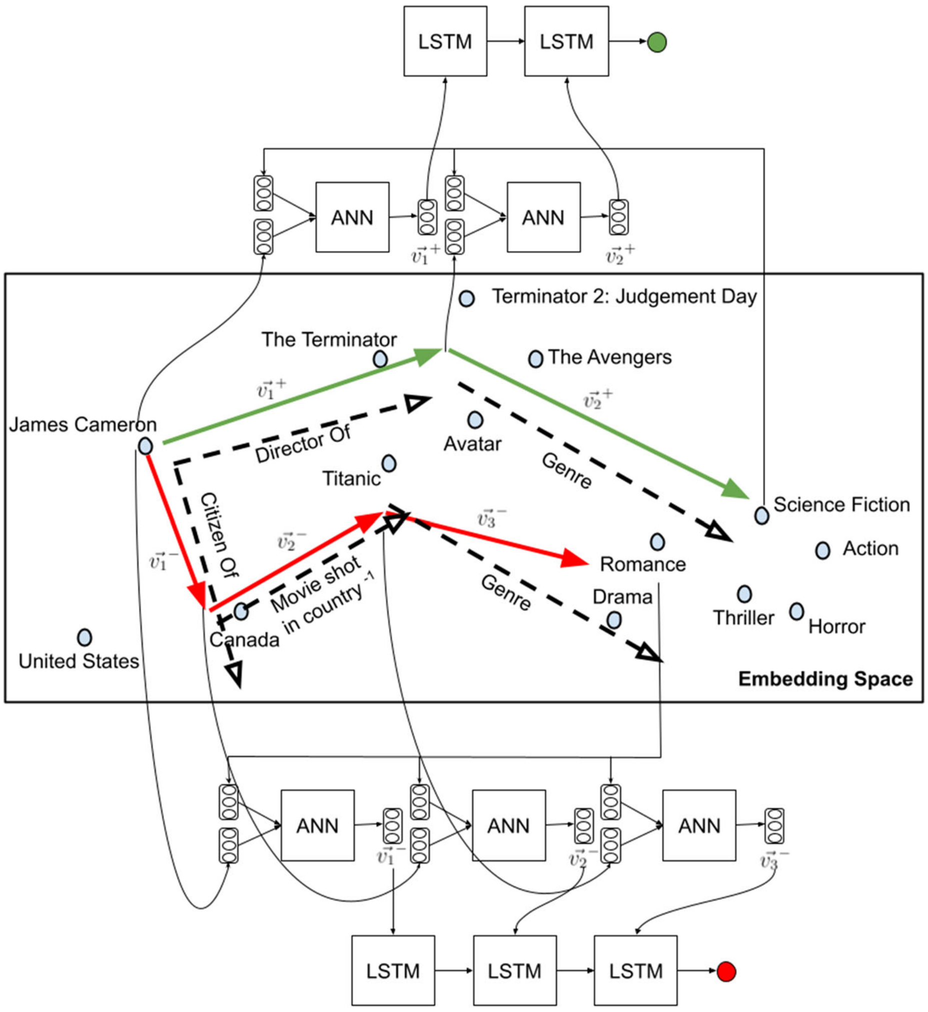

**Figure 1** Illustration of how a trained HOPLoP model may learn correlations between

2. We introduce the idea of traversing over the embedded space instead of the discrete KG space to solve the LP task. Doing so mitigates issues related to incompleteness and skewed degree distributions for nodes.
3. We evaluate our method on 2 well known benchmarks, and introduce 2 new ones for the task of multi-hop LP. The evaluation is done on 2 tasks: entity prediction and relation prediction, and the results show our method consistently outperform previous methods across datasets and metrics.
4. We provide a method to interpret reasoning paths of HOPLoP based on the geometric embedding method TransE and show that the such interpretations are meaningful.

## 2 Preliminaries

### 2.1 Knowledge graphs

As customary, we define a Knowledge Graph (KG) as a labeled, directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{L})$ consisting of a set of entities $\mathcal{E}$ which correspond to unique objects, a set of relations $\mathcal{R}$ which are labels applied to links and a set of links $\mathcal{L} \subseteq \mathcal{E} \times \mathcal{E} \times \mathcal{R}$. Each link connects one so-called "query entity" $e_q \in \mathcal{E}$ to a "target entity" $e_t \in \mathcal{E}$, assigning a relation $r \in \mathcal{R}$ to that pair, resulting in a triple represented as $(e_q, r, e_t)$.

Often, KGs store facts of the world in a format usable by humans and machines [49]. They seamlessly organize and integrate data from multiple sources, based on an ontology, to capture information about entities of interest in a given domain or task (e.g. people, places or events), and forge connections between them. Each link in the KG represents a fact, which is usually captured from natural language.

To remain current, KGs need to be constantly updated with new facts. This is done through two approaches: Knowledge Base Population (KBP) [26] and Knowledge Base Completion (KBC). In this paper, we focus on a special case of the latter (KBC), called Link Prediction.

### 2.2 Link prediction

Given a *link* in the form of a triple $(e_q, r, e_t)$, the Link Prediction (LP) task aims at producing a score indicative of the *likelihood* for the link existing in the KG. Surveying the literature uncovers 2 variants of the LP task: Relation Prediction and Entity Prediction, discussed next.

**Relation prediction** The relation prediction task focuses on predicting which relations hold between query entity $e_q$ and a target entity $e_t$, ranking them by decreased likelihood. It can be used to answer question like "how is James Cameron related to Avatar?", which is often denoted by the incomplete triple (James Cameron, ?, Avatar) in the literature.

**Entity prediction** Entity prediction is concerned with ranking target entities $e_t$ that would be linked to a given query entity through a given relation label. It is used to answer questions like "who directed the movie Avatar?", which is often denoted by the incomplete triple (Avatar, Director Of$^{-1}$, ?). Note the use of the relation inverse in this case, to indicate that we want to "traverse" that edge of the graph backwards.

We show how to perform both entity and relation prediction using HOPLoP. Motivated readers may refer to extensive surveys [27, 49, 56, 70] on LP methods applied on KGs.

### 2.3 Knowledge graph embeddings

KG embedding algorithms learn to map discrete entities and relationships in a KG onto vector representations that capture the latent structure of the KG [70]. The key intuition involves optimizing embedded representations of discrete components such that, the score predicted for a link that exists in the KG is *generally* higher than a score predicted for a link that is not present in the KG. Given a link, expressed as triple $(e_q, r, e_t)$, a KG embedding model takes as input $\mathbf{e_q}$, $\mathbf{r}$, $\mathbf{e_t}$, and computes a score representing the likelihood that the link exists in the KG. By controlling the dimensionality of the embedding space, we observe that these models generalize to unseen facts [20, 47, 48]. The resultant embedded representation of the KG is more-complete but less accurate than the discrete KG [8].

To verify our hypothesis introduced in Section 1, that KGs mapped onto an embedded space can be traversed by HOPLoP, we experiment with three KG embedding models: TransE [7], ComplEx [66], and TuckER [4]. TransE was introduced by Bordes et al. [7] and is the first and the most competitive translational distance KG embedding model [51]. TransE views each relation label $r$ as a vector which maps, via a translation, the query entities to their target entities in the vectorial space. By representing entities and relations in the same vector space, TransE produces embeddings that are easier to interpret and is also less computationally expensive. Tensor factorization models [32], such as RESCAL [48] and DistMult [73], attempt to decompose a binary tensor representation of a KG. These models learn to generate embeddings that assign higher scores to triples that represent links present in the KG. Trouillon et al. [66] introduced ComplEx to improve asymmetric relation modelling. By enabling the representation of real and imaginary numbers, entities can behave differently based on their position (query or target) in a link. TuckER [4] is a fully expressive bilinear tensor factorization model based out of the Tucker decomposition [67]. We include TuckER in our experiments since it achieves state-of-the-art performance in LP [4].

### 2.4 Neural architectures

Neural Networks are based on a collection of connected units or nodes called artificial neurons [19]. These neural architectures are used in conjunction with an optimization objective and a learning algorithm to achieve state-of-the-art performance on various tasks [37]. The performance of neural networks are backed by the Universal Approximation Theorem [12] and the development of parallelized computation.

Recurrent neural networks [28, 58] are a class of deep neural networks that learn to model temporal dynamic behaviour by utilizing an internal memory component that stores the state of the network. Due to problems such as vanishing and exploding gradients [50], RNNs have been superseded by gated networks such as LSTMs [24] and GRUs [11]. LSTM networks learn to ignore noisy temporal patterns while paying a soft-attention to important patterns of the sequence, making them more powerful that vanilla RNNs. For this reason, we parameterize HOPLoP with an LSTM, but other architectures would work.

## 3 Related work

### 3.1 Path ranking algorithm

The Path Ranking Algorithm (PRA) [36] tackles the *entity prediction* variant of the LP task using an inference mechanism based on Random Walk with Restart (RWR) [35] that ranks related target entities. Each PRA model is specific to a relation (label) and is trained over a set of query entities and predefined paths to find a distribution over all target entities reachable through the paths and for which the relation holds. Given a set of query entities and pre-defined paths, a relation-specific PRA model performs a “weighted random walk” over all paths to predict links for that relation. The training phase allows PRA to tune the weight associated with each path such that paths that are more *likely* to reach the required target entities will be scored or ranked higher.

**Composition of KG embeddings** Frameworks following PRA used continuous representations of the KG, which helped the model understand the global structure of the graph [18]. Neelakantan et al. [47] introduced Path-RNN to tackle relation prediction by reasoning about composition of binary relations connected in a path. Similar to PRA, one Path-RNN model was used to model one relation. For a given task, Path-RNN uses PRA, in a preprocessing step, to generate relational paths relevant to the task. Embedded representations of relations, in a PRA path, were composed using a RNN, which predicted whether two entities were linked via a particular relation. Das et al. [13] extended the work of Neelakantan et al. [47] to introduce a multi-task framework called Single-Model that outperforms Path-RNNs by including entity type information in the LP process. Guu et al. [20] tackles the entity prediction problem by introducing a novel compositional training objective that produced a structural regularization effect in the LP process.

**Differences to HOPLoP** Approaches that train models in a supervised fashion over paths collected using PRA are affected by large fan-out areas caused by certain entities known as supernodes [61], making the LP process inefficient. In contrast, our framework does not utilize PRA to collect relational paths. Instead, it learns to traverse an embedded representation of the KG by modifying the representation for the query entity such that, after the traversal process, the resultant embedding represents the target entity embedding. This sequence of modifications is analysed by an LSTM network, which estimates the probability with which that particular relation holds between the query and the target entity.

HOPLoP shares similarities to the work of Guu et al. [20], with some key differences. First, their method explicitly requires paths to be fed into their model, whereas HOPLoP does away with this requirement and searches for paths on its own. Second, they operate on a discrete state space since pre-computed paths are given as input, whereas HOPLoP traverses a continuous embedding space and is not restricted by the links in the discrete graph. Third, their approach looks to retrain KG embeddings to provide a form of structural regularization, whereas HOPLoP utilizes separate neural architectures that learn to traverse over a constant embedding space. Fourth, their approach is only compatible with compositional KG embeddings such as RESCAL [48], DistMult [73] and TransE [7], whereas HOPLoP can work with any KG embedding method.

## 3.2 DeepPath and reinforcement learning

To overcome the bottleneck caused by supernodes, subsequent approaches sought out techniques from Reinforcement Learning (RL) [64]. Xiong et al. [72] introduced DeepPath, which is the first algorithm that applies RL for link prediction. DeepPath [72] tackles the problem of relation prediction by combining KG embeddings and reinforcement learning that learn to traverse the graph, in search of paths that support the relation of interest. DeepPath's training phase is broken down into two parts. The first involves training the mathematical model of the RL algorithm using standard supervised learning and paths collected by a Breath First Search (BFS) procedure. During the second phase of training, DeepPath uses the REINFORCE [71] algorithm to efficiently search for paths based on a reward function that takes into account the accuracy, diversity and efficiency of a traversed path. At each time-step, the RL agent outputs the relation to traverse, which is used by the environment to discretely search for the "next-hop" entity. The KG environment operates on a discrete space and provides the agent with rewards based on accuracy, efficiency and diversity. As experience is gained, the RL algorithm aims to improve the rewards it receives.

Das et al. [14] introduced MINERVA to tackle the entity prediction task. Their model utilizes a policy network to find a chain of links that connects the query entity with a target entity such that the underlying relation is expressed. Rather than pre-training the RL model using labeled data, MINERVA is an “RL-only” solution that uses a LSTM network to generate a path embedding at each step. Path embeddings at each time-step are used to predict the link to be taken at that time-step. MINERVA’s environment uses the discrete KG to search and extend the path towards the target entity. Since the environment assigns equal rewards to all paths that reach a required target entity, it ignores the quality of the path.

Lin et al. [39] aim to improve the quality of the path with a reward shaping strategy that gives a full positive reward ($+1$) if the agent reaches the correct target entity. However, if the agent does not reach the correct target entity, the reward is fractional based on a function of a pre-trained KG embedding model. The underlying assumption is that the KG embedding method is more complete than the discrete KG and contains information about links that are not present in the KG. Shen et al. [60] introduced M-Walk, which consists of a RNN network and Monte Carlo Tree Search guiding the search to pick entities that help the agent move towards the target entity.

**Variational inference for LP** Ranganathan and Subramanyam [52] describe two phases in the multi-hop LP process: Path-finding and Path-reasoning. Path-finding is the process of searching for a path connecting two entities. Path-reasoning is the process of evaluating whether a traversal path represents an underlying relationship. Approaches prior to Chen et al. [10] did not facilitate adequate interactions between the two phases of multi-hop LP. For instance, DeepPath and MINERVA can be interpreted as enhancing the Path-finding step while compositional reasoning [13, 47] algorithms can be interpreted as enhancing the Path-reasoning step. DeepPath is trained to find paths more efficiently between two given entities while being agnostic to whether the link exists between the two entities. MINERVA learns to reach target nodes given a source entity-relation pair while being agnostic to the quality of the searched path. Compositional reasoning models learn to predict the underlying relation given paths, while being agnostic to the path-finding procedure. Chen et al. [10] introduced DIVA to tackle the relation prediction problem using neural architectures coupled with the variational auto-encoder algorithm [31] for training and inference to cope with complex link connections in a KG.

**Differences to HOPLoP** Unlike previous multi-hop approaches, which traverse a discrete representation of the KG and suffer from incompleteness and skewed degree distributions, HOPLoP is allowed to traverse anywhere while boosting performance of LP. While HOPLoP and DIVA aim to improve the interaction between the two subtasks of link prediction process, HOPLoP operates entirely over an embedded space. By operating over an embedded space, HOPLoP is not affected by the incompleteness problem since it operates over a more-complete representation of the KG and has the ability to create traversal functions that can account for errors in the embedded space. Operating over an embedded space also enables end-to-end differentiability. This allows HOPLoP to be optimized efficiently using gradient descent methods [57], unlike variational inference based optimization methods, which are computationally expensive [17].

## 3.3 Other approaches

Departing from multi-hop LP, there are methods that aim to embed conjunctive graph queries [23, 53, 54], by mapping a subset of logic to efficient geometric operations in an embedding space, by jointly optimizing embeddings for components in the KG with predefined logic computation graphs. Similar to problems with previous multi-hop link prediction approaches, these algorithms requires queries (similar to paths) as input. HOPLoP does not require any pre-processing steps, and automatically learns to find paths. R-GCN [59] utilizes a relational graph convolutional network (GCN) to model entities as nodes of a graph neural network (GNN). The nodes are embedded in high-dimensional space and each node embedding contains information of a 1-hot neighborhood of facts around that entity. Similar to HOPLoP, R-GCN "hops" over a entities, but different from HOPLoP, R-GCN does this by evolving all node embeddings at each hop, making this method computationally expensive. In contrast, HOPLoP simply adds a translation vector to it's current position in the embedding space, to move to the next-hop entity.

## 4 HOPLoP: multi-hop link prediction over knowledge graph embeddings

Our motivating hypothesis is that the performance in multi-hop LP can be improved if the KG representation is more complete than its discrete form and allows traversals that are not affected by supernodes in the KG. Traversing a more complete representation of a KG can uncover correlations between relations and paths that help with the LP task. HOPLoP's path-finder is a one-hop function that can perform multiple traversals to anywhere in the embedded space, but we *control* its traversal from the source to the target entity.

Now, because we are traversing the embedded space, we have a much larger space of paths from the source to the target entity. DL architectures, such as LSTMs, can form appropriate compressed representations for all these paths, which then can be used by a simple logistic regression function that performs the link prediction. Moreover, since KG embeddings are neural representations of entities, a neural path-finder [72] can utilize this information to model the relationship between the 2 entities using *paths* in the space that connect the source to the target entity.

Since all operations (traversal as a sequence of translations and neural architectures) are connected and differentiable, we can use standard optimization methods based on gradients and back-propagation to tune the parameters of the model. This will be preferred over using computationally expensive techniques such as, RL (which usually requires pre-training with labeled data), and variational inference techniques [17]. Through backpropagation, a logistic regression model can propagate error gradients to the path-reasoner, which in turn can propagate errors to the path-finder, facilitating adequate interactions between the path-finder and path-reasoner [10].

### 4.1 Task

Specifically, we tackle the problem of *relation prediction* $(e_q, ?, e_t)$, i.e., finding the most likely relation between the given query and target entity. To promote differentiability, we convert this problem into the stochastic setting by predicting the probability that a relation $\mathcal{R} = r$ exists between any two entities $e_q, e_t$:

$$\Pr(\mathcal{R} = r | \mathcal{Q}, \mathcal{T}) \tag{1}$$

where $\mathcal{Q}$ and $\mathcal{T}$ are random variables representing all query and target entities respectively.

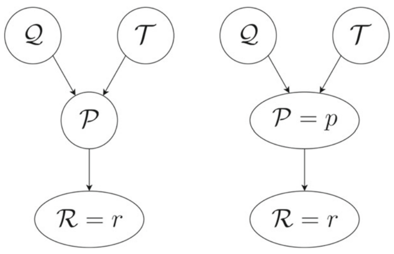

**Figure 2** Modelling our LP task. Left: original task taking "paths" into account. Right: once a path $p \in \mathcal{P}$ is traversed, the parent random variables $\mathcal{Q}$ and $\mathcal{T}$ no longer influence the child random variable $\mathcal{R}$

HOPLoP is allowed to explore "paths" connecting (the representations) of $e_q, e_t$ and defined in the embedded space, in which following a link is equivalent to following a *traversal vector*, which may or may not correspond to to an actual link in the KG. A traversal path $p$ is a sequence of traversal vectors $p = (\mathbf{v_1}, ..., \mathbf{v_H})$ where $H = |p|$ is the length of the sequence. Each traversal vector characterizes a "hop" from one point in the embedding space to another point in the same embedding space. In the next subsection, we describe the path-finder whose job it is to attain a path $p$.

Let $\mathcal{P}$ be a random variable, indicative of all paths in the embedded KG space. On incorporating path information $\mathcal{P}$ in the LP process, we have:

$$\Pr(\mathcal{R} = r|\mathcal{Q}, \mathcal{T}) = \Pr(\mathcal{R} = r|\mathcal{Q}, \mathcal{T}, \mathcal{P}) \tag{2}$$

Since each path is generated based on the query and the target entities, the random variable $\mathcal{P}$ is a *child* of random variables $\mathcal{Q}$ and $\mathcal{T}$ in our modelling. Figure 2 illustrates the idea. Therefore, once $\mathcal{P}$ is known, $\mathcal{Q}$ and $\mathcal{T}$ no longer influence any descendants of $\mathcal{P}$ [22]. Therefore, we arrive at:

$$\Pr(\mathcal{R} = r|\mathcal{Q}, \mathcal{T}, \mathcal{P} = p) = \Pr(\mathcal{R} = r|\mathcal{P} = p) \tag{3}$$

## 4.2 Model

In the previous subsection, we have simplified and divided our task into 2 parts: generating a path or "path-finding" and reasoning about a path to predict a link or "path-reasoning". To solve this problem of LP, we employ two neural network architectures [37]. The path-finder is a single-layered fully-connected neural network, consisting of a hidden layer with 1000 ReLU-activated neurons, that learns to traverse the continuous representation of the KG. The output layer contains neurons corresponding to the dimension of the KG embedding space. Due to the shallow nature of our path-finder, we simply refer to a single-(hidden)-layered fully connected neural network as an Artificial Neural Network (ANN). This neural network accepts, as input, concatenated vector representations of the query and target entity. It outputs a translation vector, which is added to the query entity embedding to represent it's current position in the embedding space. The point representing the current position may or may not refer to an entity in the KG. HOPLoP's current position is now concatenated with the target entity embedding and fed into the neural network, to traverse another point in the embedded space. This process continues until a maximum number of hops $H$ is reached.

Our framework also includes a path-reasoner, which is an LSTM [24] network that analyses the sequence of translation vectors. The LSTM network is single-layered, consisting of LSTM cells corresponding to the entity embedding dimension. The hidden state vector of the LSTM at the end of the traversal can be interpreted as the path embedding. This path embedding is sent to a sigmoid-activated neuron that performs logistic regression. The LSTM network takes, as input, the sequence of translation vectors, outputted by the path-finder. The output aims to predict the probability that the relation $\mathcal{R} = r$ holds between the query entity and the target entity.

To summarize, our path-finder traverses the embedding representation of the KG to form a path $\mathcal{P} = p$ as given by:

$$p = (\mathbf{v_1}, ..., \mathbf{v_H}) \tag{4}$$

where $\mathbf{v_i}$ is defined as:

$$\mathbf{v_i} = ANN_r \left( \left[ e_q + \sum_{h=1}^{i-1} \mathbf{v_h}; e_t \right] \right) \tag{5}$$

where $ANN_r$ represents the one-hop function modeled by the ANN and [; ] represents the concatenation operation. At hop $i = 0$, $v_0$ since undefined, since HOPLoP has not started the traversal process. At hop $i = 1$, $v_1 = ANN_r([e_q; e_t])$; at hop $i = 2$, $v_2 = ANN_r([e_q + v_1; e_t])$; at hop $i = 3$, $v_3 = ANN_r([e_q + v_1 + v_2; e_t])$; at the end of the traversal, $v_H = ANN_r([e_q + v_1 + v_2 + ... + v_H; e_t])$

Finally, our link prediction problem is formulated by:

$$\Pr(\mathcal{R} = r | \mathcal{P} = p) = \sigma(u_r^T \times LSTM_r(p) + b_r) \tag{6}$$

where $u$ is a trainable vector that are multiplied with the last hidden state of the LSTM, and $b$ represents a trainable bias. The result is squashed using the sigmoid function $\sigma$ to obtain a value between 0 and 1 that represents the probability that relation $r$ connects the query entity $e_q$ to the target entity $e_t$. Subscript $r$ indicates relation-specific parameters.

## 4.3 Training

The path-finder is an ANN that aims to traverse the KG, from the query entity to the target entity. To learn this traversal function, we control the path-finder by employing a distance-to-target (D2T) loss. At the end of the traversal, we expect that the path-finder should reach the target entity position. This loss can be given as follows.

$$L_{D2T} = \sum_{d=1}^{D} \left( \mathbf{e_{t_d}} - \left( \mathbf{e_{q_d}} + \sum_{i=1}^{H} \mathbf{v_{i_d}} \right) \right)^2 \tag{7}$$

where $\mathbf{e_t}$ is a embedding representing the target entity, $\mathbf{e_q}$ is a embedding representing the query entity, $\mathbf{v_i}$ is the traversal vector, outputted by the neural network, at the $i^{th}$ hop, $d$ represents the element at that position in the vector and $D$ is the embedding dimension.

**Algorithm 1** Training HOPLoP.

```
Input: Dataset $D_r = \{.., (e_q, [.., e^+_{t_i}, e^-_{t_i}, ..]), ..\}$, Function HOPLoP$_r(ANN_r, LSTM_r,$
$LogR_r)$, Hops $H$, Loss functions $L_C$ and $L_{D2T}$, Initial Parameters $\Theta_r^{(t=0)}$

$\Theta_r^{(t)} \leftarrow \Theta_r^{(t=0)}$;
Repeat:
    $(e_q, e_t^+, e_t^-) \sim D_r$; // Sample a query entity, positive and negative target entity
    $p^+ \leftarrow []$; $p^- \leftarrow []$; // Path is initially empty
    $e_c^+ \leftarrow e_q$; $e_c^- \leftarrow e_q$; // Initial position is query entity
    for $h$ in $1...H$ do
        Append $v_h^+ \leftarrow ANN_r(e_c^+, e_t^+)$ to $p^+$ and $v_h^- \leftarrow ANN_r(e_c^-, e_t^-)$ to $p^-$;
        $e_c^+ \leftarrow e_c^+ + v_h^+$; $e_c^- \leftarrow e_c^- + v_h^-$;
    end for
    $o^+ = LogR_r(LSTM_r(p^+))$; $o^- = LogR_r(LSTM_r(p^-))$;
    $\nabla_{\Theta_r} L_C \leftarrow \frac{\partial(L_C(o^+,1)+L_C(o^-,0))}{\partial\Theta_r}(\Theta_r^{(t)})$; // Calculate $L_C$ gradients
    $\nabla_{\Theta_r} L_{D2T} \leftarrow \frac{\partial(L_{D2T}(e_c^+,e_t^+)+L_{D2T}(e_c^-,e_t^-))}{\partial\Theta_r}(\Theta_r^{(t)})$; // Calculate $L_{D2T}$ gradients
      /* Gradient proportions for parameters:
        $\delta_{LogR_r} \propto \nabla_{\Theta_r} L_C$;
        $\delta_{LSTM_r} \propto \delta_{LogR_r}$;
        $\delta_{ANN_r} \propto \nabla_{\Theta_r} L_{D2T} \times \delta_{LSTM_r}$; */
    $\Theta_r^{(t+1)} \leftarrow \Theta_r^{(t)} - \mu \times \nabla_{\Theta_r}(L_C + L_{D2T})$; // Update parameters
Until convergence; // Time-step increments by 1
Return $\Theta_r^{(t)}$;
```

We formulate the LP task as a classification problem. The model should learn to classify between positive and negative pairs of entities for a relation $r$. A positive entity pair involves two entities who are related via the relation $r$ and the triple $(e_q, r, e_t)$ represents a link, which is observed in the KG. To perform this classification task, we use the binary cross-entropy loss function. The classification loss can be given as follows.

$$L_C = -y \times \log(o) + (1 - y) \times \log(1 - o) \tag{8}$$

where $o = \Pr(\mathcal{R} = r|\mathcal{P} = p)$ is the value outputted by the LSTM network, and $y$ is a boolean variable that signifies if the relation $r$ holds between the source entity and the target entity. Since our model learns to classify between positive and negative entity pairs, following [72], we train the model with positive and negative examples in a 1:1 ratio.

The two objectives, described in (7) and (8), are jointly optimized using Gradient Descent [57]. The overall objective function can be given as follows.

$$\min_{\Theta} L = L_{D2T} + L_C \tag{9}$$

where $\Theta_r = \Theta_{ANN_r} \cup \Theta_{LSTM_r} \cup \Theta_{LogR_r}$ represents the parameters of the model. The differentiability of our framework allows us to jointly optimize the parameters of the path-finder and the path-reasoner.

Figure 3 describes both the path finding and reasoning processes in HOPLoP using the notation of Probabilistic Graphical Models (PGMs) [33]. The left side of the figure shows a PGM representing the path-finding process. The rectangle in the figure represents a plate

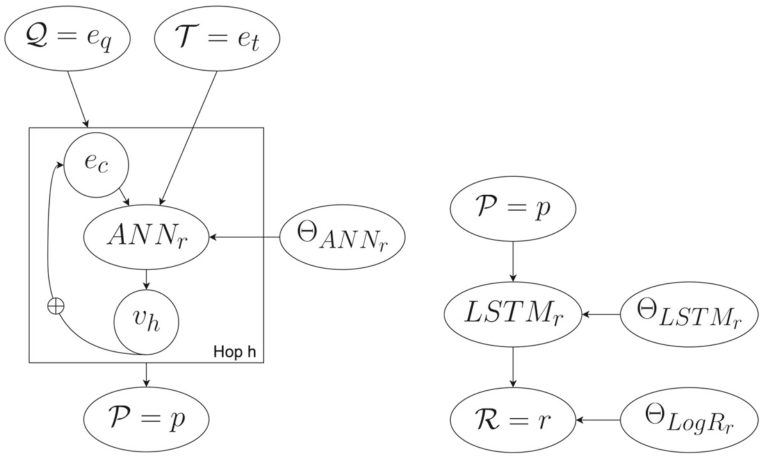

**Figure 3** Illustration of how we model the path-finder and a path-reasoner using PGMs

model, which is repeated for $H$ hops. $e_c$ represents the current position in the entity embedding space and $\oplus$ represents pointwise addition. Finding a path $p$ from $e_q$ to $e_t$ is performed by an ANN, whose parameters are controlled by $\Theta_{ANN_r}$. The right side of the figure shows a PGM diagram illustrating the path-reasoning process. The path is analyzed by the LSTM network, whose parameterized by $\Theta_{LSTM_r}$. The final hidden state of the LSTM is used as a path representation to perform logistic regression, parameterized by $\Theta_{LogR_r}$, to predict whether the relation exists between the two entities.

### 4.4 Discussion

HOPLoP framework essentially performs a logistic regression [25] which learns a decision boundary that can accurately distinguish positive and negative pairs of entities for a particular relation of interest. This classification is based on a vector representation of the traversed path, generated by the LSTM that analyses the sequence of translation vectors used to traverse from the query to the target entity. The traversal over an embedding space involves modifying the query entity embedding several times, through vector addition, such that, at the end of the traversal, the modified query entity should represent the target entity and the sequence of modifications applied on the source entity by the neural network should allow the LSTM to form effective path embeddings that characterizes the traversal process. Through backpropagation [58], the LSTM network controls how the neural network learns to traverse the embedding space, thus providing adequate interaction between the path-finding and path-reasoning processes.

## 5 Experiments and results

### 5.1 Datasets

To test our framework on the task of (multi-hop) link prediction, we evaluate on standard datasets such as NELL-995 [72] and FB15K-237 [65]. Dataset statistics have been provided in Table 1. Every dataset contains a set of tasks, relations of interest for which links must

**Table 1** Statistics of KG datasets used in experiments

| Dataset | #Entities | #Relations | #Triples | #Tasks |
|---|---|---|---|---|
| NELL-995 | 75,492 | 200 | 154,213 | 12 |
| FB15K-237 | 14,505 | 237 | 310,079 | 20 |
| WN18RR | 40,714 | 11 | 86,835 | 10 |
| YAGO3-10 | 123,143 | 37 | 1,079,040 | 23 |

be predicted. Each task is a query relation for which links must be predicted. These tasks were extracted from different domains like Sports, People, Locations, Film, etc. For each task, there is a set of query entities. Each query entity $e_q$ is related to a set of positive target entities $\{..., e_{t_i}^{+}, ...\}$ via a specific task $r$. The datasets also include several negative target entities $\{..., e_{t_i}^{-}, ...\}$ that are not linked to the query entity $e_q$ via the relation of interest $r$. Following Guu et al. [20] and Xiong et al. [72], negative target entities are picked from the same domain as the positive target entity, allowing for a fair evaluation of our framework. Following our movie example, the set of negative target entities will not contain entities such as Canada, Titanic, Football, etc., since these entities are out of the range of the relation Directs Genre Of Movies, whose range is equivalent to the range of Genre. Futhermore, following Lin et al. [38], we include the reverse relations for each relation in the KGs. Specifically, for each triple $(e_q, r, e_t)$, we add a triple $(e_t, r^{-1}, e_q)$ to the KG. This technique has known to improve the performance of KG embeddings in the LP task [46], allowing us to compare HOPLoP to stronger baseline KG embedding models. This technique helps previous multi-hop algorithms by enabling bi-directional traversal of a link, i.e., a RL agent is able to step backward in the KG [72]. This allows them to correct for mistakes in the graph, due to incompleteness, or mistakes in the traversal process.

To further evaluate our model, we introduce the WN18RR and YAGO3-10 [15] datasets for our task of relation prediction. These datasets have been previously used for entity prediction [29], in the literature of KG embeddings. To maintain consistency with well-known datasets in this literature of multi-hop LP, we pick several tasks from each dataset and generate negative target entities for facts involved in these relations:

#### 5.1.1 WN18RR [15]

We take advantage of the hierarchical structure of WordNet [44] to generate negative examples. Given a positive entity pair, we include all hyponyms of the hypernyms of the positive target entity as negative target entities for that query entity. This ensures negative target entities are from the same domain as the positive target entity. We pick 10 tasks from 11 relations from the KG such that the relation is involved in atleast 1% of all facts. This covers 99.9% of the WN18RR KG.

#### 5.1.2 YAGO3-10 [41]

Since YAGO is not hierarchically structured, we do not use the approach we used for the WN18RR dataset. Instead, we used Breadth First Search (BFS) to obtain negative examples. BFS traversed the links of the graph, while remaining agnostic to the relation, to reach entities that are connected to a query entity. We made sure that the negative target entities, picked by BFS, were in the range of the query relation. We used BFS rather than random sampling techniques to make the dataset more competitive, following the intuition that baseline KG embedding models will be able to distinguish between positive and negative entity pairs simply based on the distance between the entities [40]. We pick 23 tasks from 37 relations from the KG such that there are atleast 3,000 facts per relation to ensure adequate representation of the relation in embedded representation of the KG. This covers 99% of the YAGO3-10 KG. Following previous work, we extracted 10 negative target entities per positive target entity. For the relation hasGender in the YAGO3-10 dataset, there was only 1 negative target entity per positive target entity. This is because the hasGender relation is binary in that KG. The datasets, and codes for building them, are publicly available at https://github.com/U-Alberta/HOPLoP.

## 5.2 Experimental setup

Although our framework is agnostic to the KG embedding model, HOPLoP relies on a base KG embedding model to provide an embedding space to traverse over. We experiment with 3 popular KG embedding models as described in Section 2.3. In short:

- TransE [7] was the first, yet extremely competitive [51], translational distance LP model that capture relations between entities such that $\mathbf{e_q}+\mathbf{r} \approx \mathbf{e_t}$. ComplEx [66] and TuckER [4] are derived from tensor factorization methods (ComplEx: [48, 73], TuckER: [67]) which looks to decompose the binary tensor representation of the KG. The optimization goals between the two methods are very different: translational distance models look to minimize the distance, in euclidean space, between two related entities whereas tensor decomposition methods look to maximize the probability that a link exists in the KG through classification.
- Although ComplEx and TuckER are both tensor factorization methods, ComplEx represents embeddings in complex space and computes dot products for complex vectors such that positive links have a lower dot product value than negative links. This generates embeddings that generally have more negative values, due to the minimization procedure explained in Section 2.3, which is leveraged later by Ding et al. [16] to introduce sparsity and interpretability to the embeddings. On the other hand, TuckER decomposes the binary tensor representation of the KG $B \in \{0,1\}^{|\mathcal{E}|\times|\mathcal{R}|\times|\mathcal{E}|}$ into $B \approx \mathcal{W} \times_1 \mathcal{E} \times_2 \mathcal{R} \times_3 \mathcal{E}$, such that $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$, $\mathcal{E} \in \mathbb{R}^{|\mathcal{E}|\times d_e}$, $\mathcal{R} \in \mathbb{R}^{|\mathcal{R}|\times d_r}$, and the product of matrices should be result in higher dot product for positive link. The objective between two tensor factorization methods differs widely due to their view of their KG model.

We perform hyperparameter tuning for those hyperparameters introduced by HOPLoP. To leverage computation gains from compiled static graphs generated by ML packages such Tensorflow [1], HOPLoP requires a pre-determined path length, which allows for compile time optimizations.[2] Therefore, we tune the maximum number of hops $H$ choosing from $\{1, 3, 5, 10, 15, 20\}$.

We provide a fair comparison between HOPLoP and baseline KG embedding models, we set the dimensionality of all embeddings to 100. We re-train for embeddings following the hyperparameters selected by the creators of the KG embedding. Embeddings and

[2]In the next section, we see that HOPLoP learns to *not* hop; this is non-trivial because we do not explicitly provide HOPLoP with feedback regarding when not to hop, nor do we set up any constraints in the traversal process.

network parameters were optimized using Adam [30] with the default hyperparameter settings (initial learning rate $\mu = 0.001$). Details regarding hyperparameter selection is available in the supplementary material.

We evaluate HOPLoP on 2 tasks: Relation prediction and Entity prediction. We describe how HOPLoP can be used to perform both entity and relation prediction while comparing it to the state-of-the-art in both tasks.

### 5.2.1 Relation prediction

The relation prediction task is commonly used to evaluate multi-hop link prediction algorithms [14, 36, 60, 72]. The reason for evaluating HOPLoP on relation prediction stems its mathematical derivation, making relation prediction the intrinsic evaluation method.

**Methodology** Given a test query entity $e_q$, a relation $r$ and a set of target entities $e_{t|q,r} = \{e_{t1}^{+}, e_{t2}^{+}, e_{t3}^{-}, e_{t4}^{-}, ...\}$, we compute the link prediction scores $\text{HOPLoP}_r(e_q, e')$ for each entity in $e' \in e_{t|q,r}$. $e_{ti}^{+}$ refers to a positive target entity, whereas $e_{ti}^{-}$ refers to a negative target entity. In the ideal situation, the scores for all positive target entities will be higher than all negative target entities, i.e., all positive target entities will be ranked higher than all negative target entities.

Following previous literature, our evaluation metric is the Mean Average Precision (MAP) score. Given a query relation, the MAP score is the mean AP across all query entities. AP is a strict metric since it penalizes when a negative target entity is ranked above a positive target entity. We evaluate HOPLoP in the relation prediction task on four datasets: NELL-995, FB15K-237, WN18RR and YAGO3-10.

### 5.2.2 Entity prediction

The entity prediction task is commonly used to evaluate KG embedding-based LP algorithms [29]. Although HOPLoP is designed for relation prediction, we describe a procedure for the extrinsic evaluation of HOPLoP on the more-common entity prediction task.

**Methodology** Given a test triple $(e_q, r, e_t)$, we calculate the probability that link exists, $HOPLoP_r(e_q, e')$ for all possible $e'$ where $e' \in \mathcal{E}$ is an entity, such that, $(e_q, r, e')$ does not represent a link in the KG, unless $e' = e_t$. This allows us to compute metrics in the filtered setting [7], which does not penalize the model for ranking other correct target entities higher than the correct target entity $e_t$ in question. We sort this list of scores in decreasing order and compute the rank for the correct target entity $e_t$. Since we do not train a HOPLoP model for every relation in the dataset, we consider the worst case prediction possible, i.e., the rank for the correct target entity $e_t$ at worst is $|\mathcal{E}|$.

Following previous literature, we compute 2 types of metrics: Hits@k (H@k) $k \in \{1, 3, 10\}$ and Mean Reciprocal Rank (MRR) and evaluate on two standard datasets: WN18RR and YAGO3-10.

## 5.3 Results

**Hypothesis** If a multi-hop LP algorithm is allowed to traverse the graph, unconstrained, then this will boost performance in the LP task. To this end, we proposed HOPLoP, which traverses the KG embedding space in a unconstrained yet controlled manner. In the context of HOPLoP, our hypothesis can be re-worded: If HOPLoP is trained, then it learns to uncover correlations between the controlled path traversals and the relation of interest, thus boosting LP performance. Table 5 observes that, without any training, HOPLoP does *not* perform as well as the baselines. This shows that the training process helps HOPLoP correlate traversal paths to a relation. This boost in LP performance can also be observed across all datasets and metrics.

### 5.3.1 Relation prediction

We compare HOPLoP with embedding-based algorithms [4, 7, 66], supervised path traversal approaches [36], reinforced path traversal approaches [14, 60, 72]. We also compare with DIVA [10], a probabilistic approach tackling the relation prediction task, but report their results in the captions of tables since their paper does not disclose MAP scores for each relation. Table 2 reports the results from experiments involving the NELL-995 dataset, a simple dataset for which many existing algorithms observe very high accuracies. We evaluate HOPLoP over the FB15K-237 dataset, which is considered to be more challenging arguably more relevant for real-world scenarios than the NELL dataset [10]. Results from this experiment have been reported in Table 3, showing that our approach performs significantly better than state-of-the-art approaches. From Tables 2 and 3, we can also observe that HOPLoP improves the performance of all KG embedding models. On the NELL-995 dataset, HOPLoP traverses over TransE's embedding space to boost performance by + 0.106 MAP. HOPLoP boosts performances of TransE, ComplEx, TuckER by + 0.137, + 0.050 and + 0.159 MAP on the FB15K-237 dataset. We also observe that less performant KG embedding models, such as TransE, are highly benefited by HOPLoP since their simple representation allows HOPLoP to easily understand the structure of the KG and create appropriately complex decision boundaries (Table 4).

Futhermore, we evaluate HOPLoP on two new datasets introduced for the task of multi-hop link prediction. We compare HOPLoP to baseline KG embedding models and observe

**Table 2** Performance of HOPLoP against baseline path-based and embedding-based approaches to the relation prediction task on the NELL-995 dataset

| Task | HOPLoP (TransE) | M-Walk | Minerva | DeepPath | PRA | TransE (Bernoulli) |
|---|---|---|---|---|---|---|
| AthletePlaysForTeam | **0.953** | 0.847 | 0.827 | 0.721 | 0.547 | 0.727 |
| AthletePlaysInLeague | **0.998** | 0.978 | 0.952 | 0.927 | 0.841 | 0.726 |
| AthleteHomeStadium | **0.930** | 0.919 | 0.928 | 0.846 | 0.859 | 0.798 |
| AthletePlaysSport | 0.929 | 0.983 | **0.986** | 0.917 | 0.474 | 0.805 |
| TeamPlaysSport | **0.980** | 0.884 | 0.875 | 0.696 | 0.791 | 0.759 |
| OrgHeadquaterCity | **0.956** | 0.950 | 0.945 | 0.790 | 0.811 | 0.912 |
| WorksFor | **0.993** | 0.842 | 0.827 | 0.699 | 0.681 | 0.901 |
| BornLocation | **0.965** | 0.812 | 0.782 | 0.755 | 0.668 | 0.744 |
| PersonLeadsOrg | **0.962** | 0.888 | 0.830 | 0.790 | 0.700 | 0.899 |
| OrgHiredPerson | **0.930** | 0.888 | 0.870 | 0.738 | 0.599 | 0.868 |
| Overall | **0.934** | 0.899 | 0.876 | 0.788 | 0.697 | 0.828 |

Values represent MAP scores. DIVA attained an overall MAP score of 0.886

**Table 3** Performance of HOPLoP against baseline path-based and embedding-based approaches to the relation prediction task on the FB15K-237 dataset

| Task | HOPLoP (TransE) | HOPLoP (ComplEx) | HOPLoP (TuckER) | DeepPath | PRA | TransE (Bernoulli) | ComplEx (Bernoulli) | TuckER |
|---|---|---|---|---|---|---|---|---|
| Team/Sport | 0.989 | 0.993 | **0.995** | 0.955 | 0.987 | 0.924 | 0.985 | 0.953 |
| Person/ PlaceOfBirth | **0.986** | 0.960 | 0.958 | 0.531 | 0.441 | 0.842 | 0.928 | 0.569 |
| Person/ Nationality | 0.964 | 0.977 | **0.981** | 0.823 | 0.846 | 0.849 | 0.900 | 0.934 |
| Film/Director | 0.654 | **0.679** | 0.671 | 0.441 | 0.349 | 0.534 | 0.553 | 0.604 |
| Film/WrittenBy | **0.994** | 0.978 | 0.972 | 0.457 | 0.601 | 0.770 | 0.783 | 0.789 |
| Film/Language | **0.971** | **0.971** | 0.935 | 0.670 | 0.663 | 0.720 | 0.698 | 0.780 |
| TvProgram/ Languages | 0.984 | **0.987** | 0.979 | 0.969 | 0.960 | 0.935 | 0.934 | 0.971 |
| CapitalOf/ Location | **0.906** | 0.851 | 0.838 | 0.783 | 0.829 | 0.599 | 0.905 | 0.560 |
| OrgFounder/ OrgFounded | 0.812 | 0.796 | 0.757 | 0.309 | 0.281 | 0.711 | **0.864** | 0.473 |
| Artist/Origin | **0.966** | 0.883 | 0.866 | 0.514 | 0.426 | 0.744 | 0.903 | 0.519 |
| Overall | **0.864** | 0.859 | 0.849 | 0.572 | 0.541 | 0.709 | 0.809 | 0.690 |

DIVA attained a MAP score of 0.598

a significant improvement in MAP scores. Specifically, on the WN18RR dataset, HOPLoP improves the performance of TransE, ComplEx, and TuckER by + 0.266, + 0.279, + 0.404 MAP respectively. Experiments on the YAGO3-10 dataset show that HOPLoP improves

**Table 4** Performance of HOPLoP against baseline embedding-based approaches for relation prediction on the WN18RR dataset

| Task | HOPLoP (TransE) | HOPLoP (ComplEx) | HOPLoP HOPLoP | TransE (Bernoulli) | ComplEx (Bernoulli) | TuckER |
|---|---|---|---|---|---|---|
| Hypernym | **0.968** | 0.860 | 0.865 | 0.556 | 0.496 | 0.472 |
| DerivationallyRelated | **0.993** | 0.977 | 0.909 | 0.955 | 0.953 | 0.460 |
| InstanceHypernym | 0.966 | 0.966 | **0.995** | 0.844 | 0.700 | 0.811 |
| AlsoSee | **0.938** | 0.841 | 0.844 | 0.411 | 0.301 | 0.264 |
| MemberMeronym | 0.706 | 0.643 | **0.898** | 0.152 | 0.112 | 0.112 |
| SynsetDomainTopic | 0.976 | 0.941 | **0.979** | 0.856 | 0.524 | 0.771 |
| HasPart | 0.832 | 0.738 | **0.906** | 0.345 | 0.382 | 0.307 |
| MemberDomainUsage | **0.571** | 0.518 | 0.482 | 0.388 | 0.201 | 0.288 |
| MemberDomainRegion | 0.669 | 0.652 | **0.696** | 0.468 | 0.507 | 0.237 |
| VerbGroup | 0.991 | **0.992** | 0.816 | 0.843 | 0.965 | 0.347 |
| Overall | **0.848** | 0.793 | 0.811 | 0.582 | 0.514 | 0.407 |

**Table 5** Performance comparison between untrained HOPLoP models, baseline KG embedding models and trained HOPLoP models

| Embedding dataset | TransE | ComplEx | TuckER |
|---|---|---|---|
| NELL-995 | 0.465 \| **0.828** \| **0.934** | – | – |
| FB15K-237 | 0.403 \| **0.709** \| **0.864** | 0.388 \| **0.809** \| **0.859** | 0.371 \| **0.690** \| **0.849** |
| WN18RR | 0.415 \| **0.582** \| **0.848** | 0.403 \| **0.514** \| **0.793** | 0.386 \| **0.407** \| **0.811** |
| YAGO3-10 | 0.422 \| **0.545** \| **0.908** | 0.396 \| **0.564** \| **0.861** | – |

Results are of the form untrained HOPLoP MAP | baseline KG embedding MAP | trained HOPLoP MAP

the performance of baseline KG embedding models TransE and ComplEx by +0.363 and +0.297 MAP respectively. Due to the enormous size of the YAGO3-10 dataset coupled with computational constraints, we do not run experiments with TuckER (Tables 5 and 6).

**Table 6** Performance of HOPLoP against baseline embedding-based approaches for relation prediction on the YAGO310 dataset

| Task | HOPLoP (TransE) | HOPLoP (ComplEx) | TransE (Bernoulli) | ComplEx (Bernoulli) |
|---|---|---|---|---|
| IsAffiliatedTo | 0.941 | **0.965** | 0.815 | 0.843 |
| PlaysFor | **0.955** | 0.904 | 0.778 | 0.814 |
| IsLocatedIn | **0.918** | 0.824 | 0.643 | 0.593 |
| HasGender* | **0.973** | 0.971 | 0.946 | 0.940 |
| WasBornIn | **0.883** | 0.766 | 0.415 | 0.390 |
| ActedIn | **0.861** | 0.839 | 0.369 | 0.397 |
| IsConnectedTo | **0.979** | 0.948 | 0.694 | 0.734 |
| HasWonPrize | 0.843 | **0.878** | 0.540 | 0.561 |
| Influences | **0.869** | 0.714 | 0.354 | 0.374 |
| DiedIn | **0.842** | 0.776 | 0.451 | 0.397 |
| HasMusicalRole | 0.923 | **0.935** | 0.646 | 0.538 |
| GraduatedFrom | 0.853 | **0.891** | 0.456 | 0.385 |
| Created | **0.916** | 0.837 | 0.523 | 0.604 |
| WroteMusicFor | 0.896 | **0.940** | 0.368 | 0.434 |
| Directed | **0.942** | 0.840 | 0.517 | 0.558 |
| ParticipatedIn | **0.790** | 0.770 | 0.476 | 0.552 |
| HasChild | **0.966** | 0.841 | 0.396 | 0.330 |
| HappenedIn | **0.955** | 0.877 | 0.600 | 0.628 |
| IsMarriedTo | **1.000** | 0.990 | 0.524 | 0.940 |
| IsCitizenOf | 0.934 | **0.975** | 0.634 | 0.548 |
| WorksAt | 0.968 | **0.997** | 0.581 | 0.621 |
| Edited | **0.988** | 0.948 | 0.328 | 0.426 |
| LivesIn | **0.967** | 0.965 | 0.484 | 0.356 |
| Overall | **0.908** | 0.861 | 0.545 | 0.564 |

Due to the biased nature of this task, a link prediction algorithm that always predicts male will achieve a MAP score of 0.963 in this task

**Table 7** Performance of HOPLoP(TransE) in the entity prediction task on the WN18RR dataset compared against state-of-the-art (multi-hop) LP algorithms

| Model | MRR | H@1 | H@3 | H@10 |
|---|---|---|---|---|
| M-Walk | 0.437 | 0.414 | 0.445 | – |
| ComplEx | 0.440 | 0.410 | 0.460 | 0.510 |
| MINERVA | 0.448 | 0.413 | 0.456 | 0.513 |
| TransE | 0.466 | 0.423 | – | 0.556 |
| TuckER | 0.470 | 0.443 | 0.482 | 0.526 |
| Reward Shaping (ComplEx) | 0.472 | 0.437 | – | 0.542 |
| CompGCN | 0.479 | 0.443 | 0.494 | 0.546 |
| ComplEx-N3 | 0.480 | – | – | 0.570 |
| HOPLoP(TransE) | **0.760** | **0.753** | **0.767** | **0.790** |

### 5.3.2 Entity prediction

We compare HOPLoP(TransE) with several state-of-the-art KG embedding models such as [7, 9, 15, 34, 63, 66, 68, 73] and multi-hop link prediction approaches [14, 39, 60] on WN18RR and YAGO3-10 dataset. From Tables 7 and 8 we see that HOPLoP(TransE) consistently outperforms previous state-of-the-art approaches by a good margin. On the WN18RR dataset and YAGO3-10, HOPLoP(TransE) achieves an error reduction (with respect to MRR) of 53.85% and 56.67% respectively.

## 5.4 Analysis

The main hypothesis of this paper is that the performance of a LP algorithm can be improved if it is allowed to leverage graph traversals that are not constrained by the KG. To verify this hypothesis, we introduced HOPLoP, a multi-hop link prediction framework that learns to traverse an embedding space, provided by a base KG embedding model. From Tables 2, 3, 4, and 6, we can observe that HOPLoP consistently outperforms baseline KG embedding models and previous state-of-the-art multi-hop link prediction algorithms. We support our hypothesis with the intuition that traversing an embedding space uncovers correlations between paths and relations facilitating better understanding of the global structure

**Table 8** Performance of HOPLoP(TransE) in the entity prediction task on the YAGO3-10 dataset compared against state-of-the-art KG embedding models

| Model | MRR | H@1 | H@3 | H@10 |
|---|---|---|---|---|
| DistMult | 0.340 | 0.240 | 0.380 | 0.540 |
| ComplEx | 0.360 | 0.260 | 0.400 | 0.550 |
| ConvE | 0.440 | 0.350 | 0.490 | 0.620 |
| RotatE | 0.495 | 0.402 | 0.550 | 0.670 |
| RefE | 0.577 | 0.503 | 0.621 | 0.712 |
| ComplEx-N3 | 0.580 | – | – | 0.710 |
| HOPLoP(TransE) | **0.818** | **0.817** | **0.818** | **0.820** |

**Table 9** Runtime of HOPLoP(TransE) compared to M-Walk and MINERVA on the WN18RR dataset

| Model | Training (hrs.) | Testing (sec./sample) |
|---|---|---|
| MINERVA ($S = 3$, $L = 100$) | 3 | $2 \times 10^{-2}$ |
| M-Walk ($S = 5$, $M = 128$) | 14 | $6 \times 10^{-3}$ |
| HOPLoP(TransE) ($H = 1$), fastest | 3.7* | $3.6 \times 10^{-3}$ |
| HOPLoP(TransE) ($H = 5$), best perf. | 0.5 | $5.9 \times 10^{-3}$ |
| HOPLoP(TransE) ($H = 20$), worst | 1.1 | $1.4 \times 10^{-2}$ |

$S$ indicates search horizons, $L$ indicates number of rollouts and $M$ indicates the number of MCTS simulations. * Early stopping mechanism with a high patience of 100 epochs did not allow quick termination of the training process

through KG embeddings. To verify this, Table 5 compares untrained versions of HOPLoP to baselines and the trained versions of HOPLoP to show that untrained HOPLoP models are not as performant as baseline KG embedding models and trained HOPLoP models. This shows that the training process helps HOPLoP discover correlations between paths and relations that help boost performance in the link prediction task. Futhermore, we show that HOPLoP is computationally inexpensive. In Table 9, we observe that the runtime of HOPLoP (Tensorflow-gpu) is lower than that of M-Walk (C++ & Cuda) and MINERVA (Tensorflow-gpu).

## 6 Interpretability of HOPLoP

In general, a limitation of operating over an embedding space is that reasoning paths cannot always be easily interpreted because they result from latent factors that are embedded in the KG embedding space [5] and the weights learned by HOPLoP. HOPLoP(TransE), however, operates on the TransE embedding space, where both entities and relations are represented [7], geometrically allowing one to interpret it's reasoning process by observing the translation vectors $\mathbf{v_1}...\mathbf{v_H}$. This is not possible in the case of ComplEx and TuckER since HOPLoP traverses the embedding space where only entities are represented. Following the intuition that similar relationships will be represented by similar vector representations [5], we use a function of the euclidean distance as a similarity measure between a translation vector $\mathbf{v_h}$ and an embedding for a relation. Let $s(\mathbf{v_h}, \vec{r_i})$ be the similarity function.

$$s(\mathbf{v_h}, \mathbf{r_i}) = -\sqrt{\sum_{d=1}^{D}(\mathbf{v_{h_d}} - \mathbf{r_{i_d}})^2} \tag{10}$$

where $D$ is the dimensionality of the vectors. Negating the euclidean distance allows us to produce a higher similarity score for more similar vectors.

**Methodology** Now, we shall describe our process for interpreting the traversal paths of HOPLoP(TransE). To generate interpretable relational paths for a given task $r$, we remove the relations $r$ and $r^{-1}$ from the graph. This forces beam-search [21] to find other single-hop[3] or multi-hop relational paths that may represent the task. For the task hasGender in

[3]This would imply that the single-hop relation $r'$ present in the graph is semantically similar to the task $r$.

the YAGO3-10 dataset, we do not remove the relation hasGender from the dataset since it's range {male, female} is connected to other entities only via the hasGender relation. We also add a "NO-OP" relation, represented by a 0-vector, to interpret the scenario where HOPLoP(TransE) does not hop to a new entity. At each hop, we compute the similarity between the translation vector $\mathbf{v_h}$ and all relation embeddings $\mathbf{r_i} \in R$ as given by (10). At each hop, the similarity scores are used to beam-search relational paths, which first explores relations in the graph whose embedded representations are most similar to the translation vector.

## 6.1 Example paths and their interpretation

We next give several examples of interpreted HOPLoP(TransE) paths, describing query relations from different KGs. In each case, a positive path is one that connects a positive pair of entities that belong to the relation while a negative path for the same relation connects a pair of entities that are not related as discussed before.

AthletePlaysSport (NELL-995)

- **Positive**: AthletePlaysForTeam $\rightarrow$ TeamPlaysSport.
  Meaning: if an athlete $A$ plays for team $B$ and if team $B$ is known to play sport $C$, then athlete $A$ plays that sport $C$.
- **Negative**: PersonHasCitizenship $\rightarrow$ SportFansInCountry$^{-1}$
  Meaning: just because an athlete $A$ has citizenship in a country $B$ which contains fans of sport $C$, it does not imply athlete $A$ plays sport $C$, since a country may contain fans of multiple sports.

PersonBornInLocation (NELL-995)

- **Positive**: PersonHasCitizenship $\rightarrow$ CountryAlsoKnownAs
  Meaning: if a person $A$ has citizenship in a country $B$, it is highly probable that person $A$ was born in country $B$.
- **Negative**: PersonBelongsToOrganization $\rightarrow$ OrganizationAlsoKnownAs$^{-1}$ $\rightarrow$ AtLocation
  Meaning: if a person $A$ belongs to an organization $B$ and it is located at $C$, it does not mean person $A$ was born at location $C$, since a large number of people move for work.

Ethnicity/LanguagesSpoken (FB15K-237)

- **Positive**: Ethnicity/GeographicDistribution $\rightarrow$ Country/OfficialLanguage
  Meaning: if an ethnic group $A$ is from a country $B$ and country $B$'s has an official language $C$, it is highly probable that the ethnic group $B$ speaks language $C$.
- **Negative**: Ethnicity/People $\rightarrow$ Actor/DubbingPerformances/Language
  Meaning: if a dub actor $A$ performs in language $B$, it does not mean that the ethnic group $C$ of actor $A$ can speak the language $C$, since dub actors learn to speak multiple languages.

Event/Locations (FB15K-237)

- **Positive**: NcaaBasketballTournament/Team $\rightarrow$ SportsTeamLocation/Teams$^{-1}$
  Meaning: If an NCAA Basketball Tournament $A$ hosts a team $B$, and team $B$ plays at location $C$, this implies that event $A$ happened at location $C$.

- **Negative**: Film/FilmFestivals$^{-1}$ → NetflixGenre/Titles$^{-1}$ → Location/Contains$^{-1}$
  Meaning: If a Film Festival $A$ hosts Film $B$ and the film $B$ is part of the NetflixGenre $C$ and $C$ contains location $D$, it does not imply that event $A$ happened at location $D$.

hasGender (YAGO3-10)

- **Positive**: playsFor → isAffiliatedTo$^{-1}$ → hasGender
  Meaning: If an athlete $A$ plays for team $B$, another player $C$ is affiliated with team $B$ and player $C$ has gender $D$, then player $A$ also has gender $D$. This is because clubs form different teams for each gender.
- **Negative**: isMarriedTo$^{-1}$ → hasGender
  Meaning: If a person $A$ is married to another person $B$ who has gender $C$, person $A$ most likely does not have the gender $C$ of their spouse.

graduatedFrom (YAGO3-10)

- **Positive**: hasAcademicAdvisor → worksAt
  Meaning: If a person $A$ has an academic advisor $B$ who works at organization $C$, the likelihood of person $A$ graduating from $C$ is high.
- **Negative**: wasBornIn → isLocatedIn → isLocatedIn$^{-1}$
  Meaning: If a person $A$ was born in location $B$ and a particular school $C$ is located at $B$, it does not imply that person $A$ graduated from school $C$, since many students graduate from schools farm from their birth place.

## 6.2 Distribution of path lengths

Figure 4 presents the distribution of number of unique paths by path length of all tasks from the NELL-995 dataset. Note that the figure spans two pages. In this case, a unique path is a unique sequence of *both* relations and entities, i.e., $p = [e_q, v_1, e_1, ..., v_H, e_t]$. This examples the high number of unique paths observed, as compared to previous methods, which only look at relations in the path. We observe that the number of unique negative paths is higher than the number of unique positive paths. We also observe that the average path length for positive paths is consistently lower than the average path length for negative paths across all tasks.

These observations show the extent to which HOPLoP “explores” the embedded KG space in search for paths with strong support either way. Figure 4 also shows that some tasks have a fairly unique distributions, which indicates that HOPLoP can adapt to them. We observe that HOPLoP(TransE) utilizes the “NO-OP” relation, which does not change it current entity position. This explains the observation that path lengths rarely cross 10, since “NO-OP” is *not* a relation in the KG. We also observe that, for a few tasks, a substantial number of paths have been found exceeding path lengths 15. This could be attributed to “over-fitting”: since HOPLoP’s path-finder can traverse unconstrained, during the training phase, it learns complex traversal patterns that model the traversal process between only training pairs of entities. These complex traversal patterns lead to path embeddings that efficient encode training information only, leading to poor generalization performance of HOPLoP with higher hop $H$ value.

We observe that HOPLoP(TransE) finds more negative paths than positive paths. We also observe that path length rarely cross 10. This shows that HOPLoP(TransE) is utilizing the “NO-OP” operation, which is *not* counted as a relation in the KG. We also observe that, for a few tasks, a substantial number of paths have been found exceeding path lengths 15.

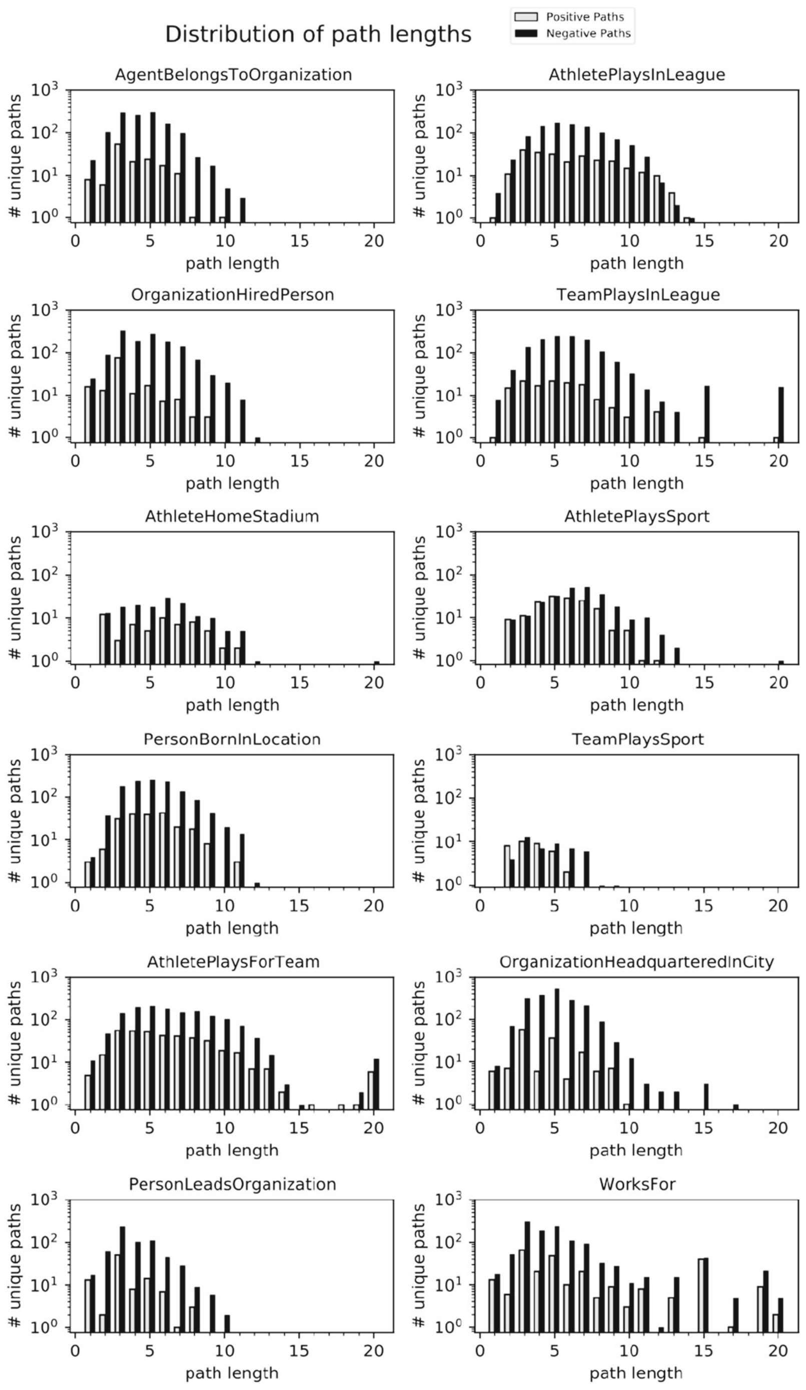

**Figure 4** Distribution of unique paths founds by path length for each task in the NELL-995 dataset. The y-axis is log-scaled. The light grey bars represent number of unique positive paths. The black bars represent number of unique negative paths

## 7 Conclusion and future work

In this paper, we hypothesized that the performance of a multi-hop link prediction algorithm may be improved if it traverses over a more-complete but inaccurate representation of the KG. We introduced HOPLoP, an end-to-end differentiable multi-hop framework that traverses over an embedding space to distinguish between existent and non-existent links of the KG. Upon performing standard LP evaluation practices, we observe that HOPLoP outperforms previous state-of-the-art multi-hop and KG embedding approaches across 4 datasets and 2 variants of the LP task. On the intrinsic relation prediction task, HOPLoP advances previous state-of-the-art methods by reducing errors by 46.53% on NELL-995 and 54.97% on FB15K-237. On the extrinsic entity prediction task, HOPLoP advances previous state-of-the-art methods with an error reduction of 53.85% on WN18RR and 56.67% on YAGO3-10. We also described a method to interpret HOPLoP(TransE)'s reasoning paths. Experiment codes, scripts and additional materials can be obtained at https://github.com/U-Alberta/HOPLoP.

### 7.1 Applicability of HOPLoP

Similar to all KG embedding methods, HOPLoP provides a score for any link, expressed as a triple $(e_q, r, e_t)$. The adoption of HOPLoP would be similar to any KG embedding method, but would require an existing embedding space. Motivated practitioners may use the scripts available in the supplementary material to generate an embedding space. To replace an existing embedding space in use would reap the highest benefits. HOPLoP can be trained to operate over any embedding space to directly replace that KG embedding method for LP, without any change in the training pipeline. Since HOPLoP uses separate parameters, it would *not* "fine-tune" the embeddings, which might be in use by different ML systems.

### 7.2 Future research directions

We believe this new approach of "generating" traversal paths over an embedded space can shed light onto a new approach for modeling sequential data. Path lengths are not limited by the size of the model, and thus, they may be the answer to expressing lengthy and $\infty$-length sequences. Paths can be created, on the go, by an hypothetical one-hop model, similar to HOPLoP, that receives its current state and a new input, and performs composition, to "move" to a new hidden state. This approach can be used to create representations for sequences, which then can be analyzed by a stronger "reasoning" model. By enabling end-to-end multi-hop reasoning over large-scale KGs, we can reason over different types of data (text, image, audio, video) in the same space, opening up several opportunities for a "data-centric" framework for machine learning.

**Author Contributions** V. Ranganathan is the main author having contributed ideas, code, and experimentation. D. Barbosa supervised the research and has contributed ideas.

**Funding** This work was funded in part by grants from the Natural Science Research Council of Canada and an Alberta Innovates Graduate Student Scholarship.

**Availability of data and other material** https://github.com/U-Alberta/HOPLoP.

**Code availability** https://github.com/U-Alberta/HOPLoP.

## Declarations

**Conflict of Interests** The authors declare that they have no conflict of interest.

## References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, USA, OSDI'16, pp. 265–283 (2016)
2. Adnan, K., Akbar, R.: Limitations of information extraction methods and techniques for heterogeneous unstructured big data. Int. J. Eng. Bus. Manag. **11**, 1847979019890771 (2019). https://doi.org/10.1177/1847979019890771
3. Aggarwal, N., Shekarpour, S., Bhatia, S., Sheth, A.: Knowledge graphs: in theory and practice. In: Conference on Information and Knowledge Management, vol. 17 (2017)
4. Balazevic, I., Allen, C., Hospedales, T.: TuckER: tensor factorization for knowledge graph completion. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, pp. 5185-5194. https://doi.org/10.18653/v1/D19-1522 (2019)
5. Bianchi, F., Rossiello, G., Costabello, L., Palmonari, M., Minervini, P.: Knowledge graph embeddings and explainable ai. arXiv:200414843 (2020)
6. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA, SIGMOD '08, pp. 1247–1250. https://doi.org/10.1145/1376616.1376746 (2008)
7. Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Curran Associates Inc., Red Hook, NY, USA, NIPS'13, pp. 2787–2795 (2013)
8. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018). https://doi.org/10.1109/TKDE.2018.2807452
9. Chami, I., Wolf, A., Juan, D.C., Sala, F., Ravi, S., Ré, C.: Low-dimensional hyperbolic knowledge graph embeddings. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, pp. 6901-6914. https://doi.org/10.18653/v1/2020.acl-main.617 (2020)
10. Chen, W., Xiong, W., Yan, X., Wang, W.Y.: Variational knowledge graph reasoning. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, pp. 1823-1832. https://doi.org/10.18653/v1/N18-1165 (2018)
11. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014 (2014)
12. Csáji, B.C. et al.: Approximation with artificial neural networks. Faculty of Sciences, Etvs Lornd University, Hungary **24**(48), 7 (2001)
13. Das, R., Neelakantan, A., Belanger, D., McCallum, A.: Chains of reasoning over entities, relations, and text using recurrent neural networks. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, Valencia, Spain, pp. 132–141. https://www.aclweb.org/anthology/E17-1013 (2017)
14. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In: ICLR (2018)
15. Dettmers, T., Pasquale, M., Pontus, S., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the 32th AAAI Conference on Artificial Intelligence, pp. 1811–1818. arXiv:1707.01476 (2018)
16. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving knowledge graph embedding using simple constraints. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics

(Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, pp. 110–121. https://doi.org/10.18653/v1/P18-1011 (2018)
17. Gal, Y.: Uncertainty in Deep Learning. PhD thesis, University of Cambridge (2016)
18. Gardner, M., Talukdar, P., Krishnamurthy, J., Mitchell, T.: Incorporating vector space similarity in random walk inference over knowledge bases. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, pp. 397–406. https://doi.org/10.3115/v1/D14-1044 (2014)
19. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press. http://www.deeplearningbook.org (2016)
20. Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp. 318–327. https://doi.org/10.18653/v1/D15-1038 (2015)
21. Haeb-Umbach, R., Ney, H.: Improvements in beam search for 10000-word continuous-speech recognition. IEEE Transactions on Speech and Audio Processing **2**(2), 353–356 (1994). https://doi.org/10.1109/89.279287
22. Halpern, J.: Reasoning about Uncertainty. MIT Press (2017)
23. Hamilton, W.L., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding logical queries on knowledge graphs. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, NIPS'18, pp. 2030–2041 (2018)
24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
25. Hosmer, D., Lemeshow, S.: Applied Logistic Regression. Applied Logistic Regression, Wiley (2004)
26. Ji, H., Grishman, R.: Knowledge base population: successful approaches and challenges. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, Association for Computational Linguistics, HLT '11, pp. 1148–1158. https://aclanthology.org/P11-1115 (2011)
27. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: representation, acquisition and applications. arXiv:200200388 (2020)
28. Jordan, M.I.: Chapter 25 - serial order: a parallel distributed processing approach. In: Donahoe, J.W., Packard Dorsel, V. (eds.) Neural-Network Models of Cognition, Advances in Psychology, vol 121, North-Holland, pp. 471–495. https://doi.org/10.1016/S0166-4115(97)80111-2 (1997)
29. Kadlec, R., Bajgar, O., Kleindienst, J.: Knowledge base completion: baselines strike back. In: Proceedings of the 2nd Workshop on Representation Learning for NLP, Association for Computational Linguistics, Vancouver, Canada, pp. 69–74. https://doi.org/10.18653/v1/W17-2609 (2017)
30. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980, cite arxiv:1412.6980comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015 (2014)
31. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings. arXiv:1312.6114v10 (2014)
32. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X
33. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT press (2009)
34. Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of Machine Learning Research, vol. 80. pp. 2863–2872. http://proceedings.mlr.press/v80/lacroix18a.html (2018)
35. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010). https://doi.org/10.1007/s10994-010-5205-8
36. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, USA, EMNLP '11, pp. 529–539 (2011)
37. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature **521**(7553), 436–444 (2015)
38. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp. 705–714. https://doi.org/10.18653/v1/D15-1082 (2015)

39. Lin, X.V., Socher, R., Xiong, C.: Multi-hop knowledge graph reasoning with reward shaping. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, pp. 3243–3253. https://doi.org/10.18653/v1/D18-1362 (2018)
40. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press, AAAI'15, pp. 2181–2187 (2015)
41. Mahdisoltani, F., Biega, J., Suchanek, F.M.: YAGO3: a knowledge base from multilingual wikipedias. In: CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4–7, 2015, Online Proceedings, www.cidrdb.org. http://cidrdb.org/cidr2015/Papers/CIDR15_Paper1.pdf (2015)
42. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, pp. 3137–3143. https://doi.org/10.24963/ijcai.2019/435 (2019)
43. Mesquita, F., Cannaviccio, M., Schmidek, J., Mirza, P., Barbosa, D.: KnowledgeNet: a benchmark dataset for knowledge base population. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, pp. 749–758. https://doi.org/10.18653/v1/D19-1069 (2019)
44. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995). https://doi.org/10.1145/219717.219748
45. Moschitti, A., Tymoshenko, K., Alexopoulos, P., Walker, A.D., Nicosia, M., Vetere, G., Faraotti, A., Monti, M., Pan, J.Z., Wu, H., Zhao, Y.: Question Answering and Knowledge Graphs. Springer, pp. 181–212 (2017)
46. Nayyeri, M., Xu, C., Lehmann, J., Yazdi, H.S.: Logicenn: a neural based knowledge graphs embedding model with logical rules. arXiv:190807141 (2019)
47. Neelakantan, A., Roth, B., Mccallum, A.: Compositional vector space models for knowledge base completion. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Beijing, China, pp. 156–166. https://doi.org/10.3115/v1/P15-1016 (2015)
48. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, Omnipress, Madison, WI, USA, ICML'11, pp. 809–816 (2011)
49. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016). https://doi.org/10.1109/JPROC.2015.2483592
50. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, JMLR.org, ICML'13, pp. III–1310–III–1318 (2013)
51. Pinter, Y., Eisenstein, J.: Predicting semantic relations using global graph properties. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, pp. 1741–1751. https://doi.org/10.18653/v1/D18-1201 (2018)
52. Ranganathan, V., Subramanyam, N.: Sde-Kg: a stochastic dynamic environment for knowledge graphs. In: Cellier, P., Driessens, K. (eds.) Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, pp. 483–488 (2020)
53. Ren, H., Leskovec, J.: Beta embeddings for multi-hop logical reasoning in knowledge graphs. In: Neural Information Processing Systems (2020)
54. Ren, H., Hu, W., Leskovec, J.: Query2box: reasoning over knowledge graphs in vector space using box embeddings. In: International Conference on Learning Representations. https://openreview.net/forum?id=BJgr4kSFDS (2020)
55. Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press. https://www.sciencedirect.com/book/9780444508133/handbook-of-automated-reasoning (2001)
56. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge graph embedding for link prediction: a comparative analysis. ACM Trans. Knowl. Discov. Data **15**(2). https://doi.org/10.1145/3424672 (2021)
57. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv:160904747 (2016)

58. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
59. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., Navigli, R., Vidal, M.E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M. (eds.) The Semantic Web, Springer International Publishing, Cham, pp. 593–607 (2018)
60. Shen, Y., Chen, J., Huang, P.S., Guo, Y., Gao, J.: M-walk: learning to walk over graphs using monte carlo tree search. In: Bengio, S., Wallach. H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol. 31. pp. 6786–6797. https://proceedings.neurips.cc/paper/2018/file/c6f798b844366ccd65d99bc7f31e0e02-Paper.pdf (2018)
61. Raghavan, S., Garcia-Molina, H.: Representing web graphs. In: Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), pp. 405–416. https://doi.org/10.1109/ICDE.2003.1260809 (2003)
62. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, Association for Computing Machinery, New York, NY, USA, WWW '07, pp. 697–706. https://doi.org/10.1145/1242572.1242667 (2007)
63. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations. https://openreview.net/forum?id=HkgEQnRqYQ (2019)
64. Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. A Bradford Book Cambridge, MA, USA (2018)
65. Toutanova, K., Lin, V., Yih, W.T., Poon, H., Quirk, C.: Compositional learning of embeddings for relation paths in knowledge base and text. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, pp. 1434–1444. https://doi.org/10.18653/v1/P16-1136 (2016)
66. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, JMLR.org, ICML'16, pp. 2071–2080 (2016)
67. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
68. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.: Composition-based multi-relational graph convolutional networks. In: International Conference on Learning Representations. https://openreview.net/forum?id=BylA_C4tPr (2020)
69. Vrandečić, D.: Wikidata: a new platform for collaborative data collection. In: Proceedings of the 21st international conference on world wide web, pp. 1063–1064 (2012)
70. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017). https://doi.org/10.1109/TKDE.2017.2754499
71. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3-4), 229–256 (1992). https://doi.org/10.1007/BF00992696
72. Xiong, W., Hoang, T., Wang, W.Y.: DeepPath: a reinforcement learning method for knowledge graph reasoning. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, pp. 564–5730. https://doi.org/10.18653/v1/D17-1060 (2017)
73. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. arXiv:1412.6575 (2015)