# Incorporating Symbolic Knowledge in Knowledge Graph Embeddings

**Shizhe Chen**
alszch@ucla.edu

## Abstract

In this research report, we look at how to constrain knowledge graph embeddings using first-order logic sentences - specifically answering "can we incorporate semantic *a priori* knowledge about the knowledge graphs when learning embeddings?" While our work is not the first to investigate this topic, we believe this is the first attempt at modeling semantic knowledge as a differentiable loss function using the probabilistic databases framework. While no experimental results are reportable at the time of SRP199 submission, we introduce the concepts that enable this first order logic semantic loss function, set up our experiment, and discuss what a positive result would look like.

## 1 Introduction

Knowledge graph embeddings (KGE) enable the prediction of unseen facts by learning a representation of the graph. That is, given graph $G = (V, E)$, where node $v \in V$ is an entity object and $l : (v_1, v_2) \in E$ is $l$ type directed edge from object $v_1$ to $v_2$, we wish to learn independent representations for all $v \in V$ and all $l$ edge types such that the representations match as close as possible to the original graph $G$. For example fact "Stewart is Jon's ancestor" can be encoded as a directed `Parent` type edge from the `Stewart` node to the `Jon` node in a family tree knowledge graph, written in short-hand as `Parent(Stewart, Jon)`. With trained KGE, we can perform *knowledge graph completion* - also known as *triple classification* - to identify edges that are missing from the input graph, but should exist according to some semantic pattern hopefully extracted by the KGE. This is done by

$$\forall h \in V, t \in V, R \in \text{Types}(E).\text{Score}(R(h,t)) > u \implies R(h,t) \in G$$

for some real-valued threshold $u$.

Additionally, we can perform *link prediction* on any KGE, which tests the embedding against a test edge $R(h,t)$. We perform link prediction by "querying" the KGE *given edge type R and starting node h, predict the most likely ending node* $\hat{t}$. This is implemented as a sorting of all $t \in V$ using the index $\text{Score}(R(h,t_i))$. Evaluating how well the KGE predicts $t$ is done by simply inverting the position ("reciprocal rank") $p$ in $[\hat{t}]$ where $\hat{t}[p] = t$. To calculate a score given a set of test cases, we can calculate *Mean Reciprocal Rank* over a set of queries:

$$\text{Mean Reciprocal Rank} = \text{avg}(\text{rank}^{-1})$$

In fact, a variant of link prediction known as *negative sampling* is used as the standard loss function in training a KGE. Negative sampling relies on the open world hypothesis, which assumes that edges which do not exist in graph $G$ are false, and should be counted negatively if included in the KGE. Negative sampling has a paradoxical flaw: it directs the values of a KGE by assuming edges which do not exist in the input graph are false, and yet later we used the trained KGE *to assert that certain edges that are not in the input graph are true*.

Preprint. Under review.

Table 1: Sample KG Embedding Models Extended from CP

| Name | Domain | Embeddings per Node | Scoring |
|---|---|---|---|
| ComplEx | $\mathbb{C}$ | 1 | $\text{Re}(\langle n_{e_1} v_r n_{e_2} \rangle)$ |
| DistMult | $\mathbb{R}$ | 1 | $\langle n_{e_1} v_r n_{e_2} \rangle$ |
| SimplE | $\mathbb{R}$ | 2 | $\frac{1}{2}(\langle h_{e_1} v_r t_{e_2} \rangle + \langle h_{e_2} v_{r^{-1}} h_{e_1} \rangle)$ |

Our paper proposes to mitigate this flaw by incorporating *a priori* knowledge about the knowledge graph as an additional loss signal during training. This way, the values of the embeddings are guided not only by the open world hypothesis, but also by supplied rules (semantic knowledge) which we independently believe are true about the KG. We accomplish this by using the results of a previous paper to reinterpret the values of embeddings as probabilities in probabilistic databases, then modeling our supplied rules as first order logic sentences so that we can run lifted inference on the sentences with respect to the probabilistic databases, yielding a single probability to be subtracted from the true probability of 1 (since we always believe our input semantic knowledge is true). While no results have been derived, we describe our experiment setup, and discuss what results could show that our loss function is having its intended effect of guiding the KGE regardless of the input knowledge graph.

## 2 Background

### 2.1 Knowledge Graph Embeddings

The KGE models that we inspect are all extended from *Canonical Polyadic Decomposition* [1], which extends singular value decomposition to higher order tensors. If we view a knowledge graph $G = (V, E)$ as a $n \times n \times i$ tensor $T$, such that $n = |V|$ and $i = |\text{types}(E)|$, CP decomposition of $T$ factorizes fact

$$R(e_1, e_2) \iff (h_{e_1}, v_R, t_{e_2})$$

where $h$, $v$ and $t$ are size $d$ embedding vectors for the head (originating) node, edge type (relation), and tail (destination node) respectively. Reconstructing the value of a fact given an embedding, i.e. the `Score` function is given by the sum of elements in the Hadamard product of vectors:

$$\text{Score}(R(e_1, e_2)) = \langle R, e_1, e_2 \rangle = \sum_i v_{R_i} h_{e_{1_i}} t_{e_{2_i}}$$

Models extended from CP decomposition(see Figure 1) make different choices for the domain, number, or scoring function of embeddings, but invariantly learn separate representations for entity nodes and relation edges. For our analysis, we work with the DistMult model, which has one $d$ size vector for each entity node, rather than two vectors for the head and tail occurrences of the node [8].

### 2.2 TractOR

We use results from previous work by Friedman and Van Den Broeck [2] to show the equivalence between a size $d$ DistMult embedding vector $r$ and $d$ tables in $d$ probabilistic databases. The TractOR model they proposed decomposes the edge $R(e_1, e_2)$ as three separate atoms in a probabilistic database:

$$E(e_1) \wedge T(R) \wedge E(e_2)$$

Here, $E$ represents a table labeled $E$ for embedding in a PDB, while $T$ represents a separate table labeled $T$ which stores edge types. The lookup of $e_1$ within the $E$ table returns a single probability, as does $T(R)$ and $E(e_2)$. TractOR is actually a size $d$ mixture of these simple models, so that for each node or relation type $i$, a size $d$ vector $r$ represents $i$, which is exactly what DistMult does. However, TractOR enables us to view the values in each embedding vector as probabilities in the tables of $d$

**Algorithm 1** $\textbf{Lift}^{\mathbf{R}}(\mathbf{Q}, \mathbf{P})$, abbreviated by $\mathbf{L}(Q)$

**Require:** UCQ $Q$ , prob. database $P$ with constants $T$.
**Ensure:** The probability $P_P(Q)$

**Step 0** *Base of Recursion*
  **if** Q is a single ground atom $t$ **then**
    **if** $\langle t : p \rangle \in P$ **then return** $p$ **else return** $0$
**Step 1** *Rewriting of Query*
  Convert Q to conjunction of UCQ: $Q_\wedge = Q_1 \wedge \cdots \wedge Q_m$
**Step 2** *Decomposable Conjunction*
  **if** $m > 1$ and $Q_\wedge = Q_1 \wedge Q_2$ where $Q_1 \perp Q_2$ **then**
    **return** $\mathbf{L}(Q_1) \cdot \mathbf{L}(Q_2)$
**Step 3** *Inclusion-Exclusion*
  **if** $m > 1$ but $Q_\wedge$ has no independent $Q_i$ **then**
    *(Do Cancellations First)*
    **return** $\sum_{s \subseteq [m]} (-1)^{|s|+1} \cdot \mathbf{L}\left(\bigvee_{i \in s} Q_i\right)$
**Step 4** *Decomposable Disjunction*
  **if** $Q = Q_1 \vee Q_2$ where $Q_1 \perp Q_2$ **then**
    **return** $1 - (1 - \mathbf{L}(Q_1)) \cdot (1 - \mathbf{L}(Q_2))$
**Step 5** *Decomposable Existential Quantifier*
  **if** Q has a *separator variable* $x$ **then**
    **return** $1 - \prod_{c \in T} (1 - \mathbf{L}(Q[x/c]))$
**Step 6** *Fail* (the query is #P-hard)

probabilistic databases. With some simple manipulation, we can show that DistMult and TractOR assign identical scores when performing link prediction tasks.

### 2.3 PDBs

Functionally, DistMult and TractOR assign identical scores to each $R(e_1, e_2)$ link. However, the choice of viewing each embedding vector as probabilistic database tables should not go under appreciated. Probabilistic databases enable uncertain reasoning by assigning each tuple of each table a probability. This can be viewed as an extension of predicate logic, where each predicate has its own table and each possible tuple of the predicate is assigned either 1 or 0. We can "query" PDBs on first order logic sentences, just as we can evaluate a first order logic sentence against its "predicate logic database" by combining the truth values of each requested tuple until the querying sentence resolves to true or false. Since PDBs use probability-ranged values for each tuple to return a probability value, we reduce the problem back to predicate logic evaluation by instantiating each tuple with a true/false value with the probability associated, counting the worlds in which the query sentence is true, and then dividing by all possible worlds. This is *Weighted First Order Model Counting* and be very inefficient because of difficulties counting the total number of worlds. Improving PDB querying requires exploiting symmetry of the query, akin to how an AND operator can be short circuited if the first operand returns false. The resulting improvement is formalized as the *Lifted Inference* algorithm (Algorithm 1). Typically when evaluating queries with lifted inference, one must take care in ensuring the query has symmetry which can be exploited, or the algorithm fails in Step 6. However, in the context of TractOR, it can be shown that any possible query will be efficiently evaluable, and query complexity will not be discussed further.

## 3 Our Work

By viewing DistMult embeddings as mixtures of PDBs, TractOR allows evaluating arbitrary first order logic queries training the KGE. We take advantage of this flexibility to build an additional loss function of based on evaluating independently true semantic knowledge sentences, and show the construction of this loss function through the running example of transitive ancestry in a family tree knowledge graph.

**Step 1.** Convert the semantic knowledge "the ancestor of my ancestor is also my ancestor" into a first order logic sentence.

$$Q = \forall a, b, c. Ancest(a, b) \wedge Ancest(b, c) \implies Ancest(a, c)$$

**Step 2.** Decompose this sentence as TractOR decomposes relation edges, while consolidating duplicates:

$$Q = \forall a, b, c. (\neg T(Ancest) \vee \neg E(a) \vee \neg E(b) \vee \neg E(c)) \wedge (T(Ancest) \wedge E(a) \wedge E(c))$$

**Step 3.** Given a mixture of $d$ PDBs inputted through the current DistMult embeddings, run the lifted inference algorithm $L$ on each PDB and return the mean probability:

$$p_{eval} = \frac{1}{d} \sum_{i}^{d} L(Q, P_d)$$

**Step 4.** Return the cross entropy loss between $p_{eval}$ and the ground truth probability of 1:

$$\text{SemLoss}(\text{KGE}, Q) = -\log(p_e val)$$

**Step 5.** Combine with original negative sampling loss:

$$\text{Loss}(\text{KGE}) = \text{NegSampLoss}(\text{KGE}) + \text{SemLoss}(\text{KGE}, Q)$$

or, if evaluation against multiple semantic rules is desired, define a loss for each and combine:

$$\text{Loss}(\text{KGE}) = \text{NegSampLoss}(\text{KGE}) + \sum_{i} \text{SemLoss}(\text{KGE}, Q_i)$$

Since each step of the lifted inference algorithm is differentiable, we can find the partial derivatives for each element of the embedding through automatic differentiation to use in gradient descent.

However, we must fix one deficiency in TractOR before using it to construct semantic first order loss. Because TractOR does not differentiate between head and tail entities when factorizing link prediction queries, Step 2 above gives incorrect decompositions when any asymmetrical edge is involved. Observe that

$$\forall a, b, c. Ancest(a, b) \wedge Ancest(\textbf{c, b}) \implies Ancest(a, c) \quad \text{(changes highlighted in bold)}$$

would also give the decomposition shown in step 2 above, even though this clearly is not the transitivity relation. Fortunately, this is easy to fix: we borrow inspiration from SimplE [3], another CP-based KG embedding model. For a given fact $R(e_1, e_2)$, we instead decompose to three different tables in a PDB:

$$T(R) \wedge H(e_1) \wedge T(e_2)$$

Such that $H$ and $T$ are separate tables for Head and Tail positions of the entity node. To ensure both tables are populated with all nodes during link prediction, we additionally decompose a separate sentence:

$$T(R^{-1}) \wedge H(e_1) \wedge T(e_2)$$

where $R^{-1}$ represents an inverse edge originating from $e_2$. When decomposing semantic knowledge sentences, however, we only decompose to the former non-inverse sentence, as the inverse relation does not help us evaluate the probability the semantic knowledge sentence is true. With this minor change, we are set up to apply any semantic knowledge constraint during training.

### 3.1 Experiment Setup

We selected the CoDEx knowledge graph completion benchmark [10] as our framework for implementing and testing the efficacy of our loss function, due to concerns over test leakage in previous standard datasets FB15k and WN18 [9]. CoDEx has the added benefit of coming with 93 algorithmically (AMIE3) found first-order patterns from the dataset, which are readily incorporated as the semantic knowledge we wish to derive a loss signal from. In addition, CoDEx uses `LibKGE` for easy access to baseline models and implementation of TractOR with semantic loss.

We trained DistMult, SimplE, TractOR, and TractOR with semantic loss embeddings on the medium sized CoDEx-M dataset, due to its abundance of asymmetric edges. To evaluate the trained models, we run link prediction tests through the CoDEx framework, but constrain our test data to only include facts found in the semantic knowledge used. We also train three separate versions of TractOR with semantic loss, with 14, 28, and all 44 rules available to available to Codex-M. All models were trained with embedding size $d = 128$.

## 4 Results

Due to a server failure, this part will not be in the final SRP199 submission. However, since the scope of the project has changed since its original proposal at the beginning of the quarter, I believe it is reasonable to discuss what a positive result would look like, with full intention of adding results once available and discussing.

Results that show TractOR with semantic loss scoring higher on the constrained link prediction task as described above would show that our loss function is better. Specifically, we should see increasing semantic knowledge also increases link prediction score.

In addition, I should manually complete a full knowledge graph completion of CoDEx-M on each of the trained models and inspect whether the semantic loss models assigned higher scores to relations that were part of its input semantic knowledge.

## 5 Related Work and Future Improvements

### 5.1 Semantic Loss from Propositional Sentences

Our work was heavily inspired by a 2018 paper from Xu et al., which derived the semantic loss function for propositional loss and used it to enforce a one-hot encoding on the output layer of an MNIST classifier neural network [11].

### 5.2 Logically constraining KG embeddings using ILPs

Wang et al. [5][6] modeled first order sentences as the constraints of an ILP problem, with the objective of the ILP being the result of KG embedding. Their approach also seeks to constrain the embedding model during training, but require the output embeddings adhere to input rules. TractOR with semantic loss merely generates a loss signal to serve as a hint.

### 5.3 Extending loss function to other training tasks

*A priori* knowledge is commonly available in other machine learning tasks as well. The challenge of bringing our loss function to these methods is that while it may be easy to create a first order sentence encompassing this symbolic knowledge, there is no general way to link the entities and relations to learned parameters.

## References

[1] F. L. Hitchcock, "The Expression of a Tensor or a Polyadic as a Sum of Products," Journal of Mathematics and Physics, vol. 6, no. 1–4, pp. 164–189, 1927, doi: https://doi.org/10.1002/sapm192761164.

[2] T. Friedman and G. V. den Broeck, "Symbolic Querying of Vector Spaces: Probabilistic Databases Meets Relational Embeddings," arXiv:2002.10029 [cs], Jun. 2020, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/2002.10029.

[3] S. M. Kazemi and D. Poole, "SimplE Embedding for Link Prediction in Knowledge Graphs," p. 12. Accessed: Dec. 17, 2020. [Online]. Available: https://papers.nips.cc/paper/2018/file/b2ab001909a8a6f04b51920306046ce5-Paper.pdf.

[4] G. Bouchard and S. Singh, "On Approximate Reasoning Capabilities of Low-Rank Vector Spaces," p. 4. Accessed: Dec. 16, 2020. [Online]. Available: http://sameersingh.org/files/papers/logicmf-krr15.pdf.

[5] W. Y. Wang and W. W. Cohen, "Learning First-Order Logic Embeddings via Matrix Factorization," p. 7. Accessed: Dec. 17, 2020. [Online]. Available: http://www.cs.cmu.edu/ yww/papers/ijcai2016.pdf.

[6] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," SW, vol. 8, no. 3, pp. 489–508, Dec. 2016, doi: 10.3233/SW-160218.

[7] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge Graph Embedding: A Survey of Approaches and Applications," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724–2743, Dec. 2017, doi: 10.1109/TKDE.2017.2754499.

[8] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding Entities and Relations for Learning and Inference in Knowledge Bases," arXiv:1412.6575 [cs], Aug. 2015, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1412.6575.

[9] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D Knowledge Graph Embeddings," arXiv:1707.01476 [cs], Jul. 2018, Accessed: Dec. 18, 2020. [Online]. Available: http://arxiv.org/abs/1707.01476.

[10] T. Safavi and D. Koutra, "CoDEx: A Comprehensive Knowledge Graph Completion Benchmark," arXiv:2009.07810 [cs], Oct. 2020, Accessed: Dec. 18, 2020. [Online]. Available: http://arxiv.org/abs/2009.07810.

[11]J. Xu, Z. Zhang, T. Friedman, and Y. Liang, "A Semantic Loss Function for Deep Learning with Symbolic Knowledge," p. 10. Accessed: Dec. 17, 2020. [Online]. Available: http://proceedings.mlr.press/v80/xu18h/xu18h.pdf.