Augmenting and Tuning Knowledge Graph Embeddings

Robert Bamler[*]
Department of Computer Science
University of California, Irvine
Irvine, CA 92617

Farnood Salehi[*]
École Polytechnique Fédérale
de Lausanne (EPFL),
Switzerland

Stephan Mandt
Department of Computer Science
University of California, Irvine
Irvine, CA 9261

Abstract

Knowledge graph embeddings rank among the most successful methods for link prediction in knowledge graphs, i.e., the task of completing an incomplete collection of relational facts. A downside of these models is their strong sensitivity to model hyperparameters, in particular regularizers, which have to be extensively tuned to reach good performance [Kadlec et al., 2017]. We propose an efficient method for large scale hyperparameter tuning by interpreting these models in a probabilistic framework. After a model augmentation that introduces per-entity hyperparameters, we use a variational expectation-maximization approach to tune thousands of such hyperparameters with minimal additional cost. Our approach is agnostic to details of the model and results in a new state of the art in link prediction on standard benchmark data.

1 INTRODUCTION

In 2012, Google announced that it improved the quality of its search engines significantly by utilizing knowledge graphs [Eder, 2012]. A knowledge graph is a data set of relational facts represented as triplets (*head*, *relation*, *tail*). The *head* and *tail* symbols represent real-world entities, such as people, objects, or places. The *relation* describes how the two entities are related to each other, e.g., '⟨head⟩ *was founded by* ⟨tail⟩' or '⟨head⟩ *graduated from* ⟨tail⟩'.

While the number of true relational facts among a large set of entities can be enormous, the amount of data points in empirical knowledge graphs is often rather small. It is therefore desirable to complete missing facts in a knowledge graph algorithmically based on patterns detected in the data set of known facts [Nickel et al., 2016a]. Such link prediction in relational knowledge graphs has become an important subfield of artificial intelligence [Bordes et al., 2013, Wang et al., 2014, Lin et al., 2015, Nickel et al., 2016b, Trouillon et al., 2016, Wang and Li, 2016, Ji et al., 2016, Shen et al., 2016, Xiao et al., 2017, Shi and Weninger, 2017, Lacroix et al., 2018].

A popular approach to link prediction is to fit an embedding model to the observed facts [Kadlec et al., 2017, Nguyen, 2017, Wang et al., 2017]. A knowledge graph embedding model represents each entity and each relation by a low-dimensional semantic embedding vector. Over the past six years, these models have made significant progress on link prediction [Bordes et al., 2013, Yang et al., 2015, Nickel et al., 2016b, Trouillon et al., 2016, Lacroix et al., 2018]. However, Kadlec et al. [2017] pointed out that these models are highly sensitive to hyperparameters, specifically the regularization strength. This is not surprising since even large knowledge graphs often contain only few data points per entity (i.e., per embedding vector), and so the regularizer plays an important role. Kadlec et al. [2017] showed that a simple baseline model can outperform more modern models when using carefully tuned hyperparameters.

In addition to being highly sensitive to the regularization strength, knowledge graph embedding models also need vastly different regularization strengths for different embedding vectors. Knowledge graph embedding models are typically trained by minimizing some function ℓ_{hrt} of the embedding vectors for each triplet fact (h, r, t) (short or *head*, *relation*, and *tail*) in the training set $\mathbb{S}$. One typically adds a regularizer with some strength $\lambda > 0$ as follows,

$$\sum_{(h,r,t)\in\mathbb{S}} \Big[\ell_{hrt}(\mathbf{E},\mathbf{R}) + \frac{\lambda}{p}\Big(||\mathrm{E}_h||_p^p + ||\mathrm{E}_t||_p^p + ||\mathrm{R}_r||_p^p\Big)\Big]. \quad (1)$$

Here, E_h, E_t, and R_r is the embedding for entity h, entity t, and relation r, respectively. Boldface $\mathbf{E}$ and $\mathbf{R}$ is shorthand for all entity and relation embeddings, respectively, and one typically uses a p-norm regularizer with $p \in \{2, 3\}$.

It was pointed out by Lacroix et al. [2018] that Eq. 1 implicitly scales the regularization strength proportionally to

[*]joint first authorship.

the frequency of entities and relations in the data set $\mathbb{S}$ since the regularizer is inside the sum over training points. This implies vastly different regularization strengths for different embedding vectors since the frequencies of entities and relations vary over a wide range (Figure 1). As we show in this paper, the general idea to use stronger regularization for more frequent entities and relations can be justified from a Bayesian perspective (for empirical evidence, see [Srebro and Salakhutdinov, 2010]). However, the specific choice to make the regularization strength *proportional* to the frequency seems more like a historic accident.

Rather than imposing a proprotional relationship between frequency and regularization strength, we propose to augment the model family such that each embedding E_e and R_r has its individual regularization strength λ_e^{E} and λ_r^{R}, respectively. This replaces the loss function from Eq. 1 with

$$\sum_{(h,r,t)\in\mathbb{S}} \ell_{hrt}(\mathbf{E},\mathbf{R}) + \sum_e \frac{\lambda_e^{\mathrm{E}}}{p}||\mathrm{E}_e||_p^p + \sum_r \frac{\lambda_r^{\mathrm{R}}}{p}||\mathrm{R}_r||_p^p. \quad (2)$$

Here, the last two sums run over each entity e and each relation r exactly once (there is only one sum over entities since the same entity embedding vector E_e is used for an entity e in either head or tail position).

The loss in Eq. 2 contains a macroscopic number of hyperparameters $\{\lambda_e^{\mathrm{E}}\}$ and $\{\lambda_r^{\mathrm{R}}\}$: over 16,000 in our largest experiments. It would be impossible to tune such a large number of hyperparameters with traditional grid search, which scales exponentially in the number of hyperparameters. To solve this issue, we propose in this work a probabilistic interpretation of knowledge graph embedding models. The probabilistic interpretation admits efficient hyperparameter tuning with variational expectation-maximization [Dempster et al., 1977, Bernardo et al., 2003]. This allows us to optimize over all hyperparameters in parallel, and it leads to models with better predictive performance.

Besides improving performance, our approach also has the potential to accelerate research on new knowledge graph embedding models. Researchers who propose a new model architecture currently have to invest considerable resources into hyperparameter tuning to prove competitiveness with existing, highly tuned models. Our cheap large-scale hyperparameter tuning speeds up iteration on new models.

In detail, our contributions are as follows:

- We consider a broad class of knowledge graph embedding models containing ComplEx [Trouillon et al., 2016] and DistMult [Yang et al., 2015]. We interpret these models as generative models of facts with a corresponding generative process (Figure 2).

- We first *augment* these models by introducing separate priors for each entity and relationship vector. In a nonprobabilistic picture, these correspond to regularizers. This augmentation makes the models more flexibile, but it introduces thousands of new hyperparameters (regularizers) that need to be optimized.

- We then show how to efficiently *tune* such augmented models. The large number of hyperparameters rules out both grid search with cross validation and Bayesian optimization, calling for gradient-based hyperparameter optimization. Gradient-based hyperparameter optimization would lead to singular solutions in classical maximum likelihood training. Instead, we propose variational expectation-maximization (EM), which avoids such singularities.

- We evaluate our proposed hyperparameter optimization method experimentally for augmented versions of DistMult and ComplEx.[1] The high tunability of the proposed models combined with our efficient hyperparameter tuning method improve the predictive performance over the previous state of the art.

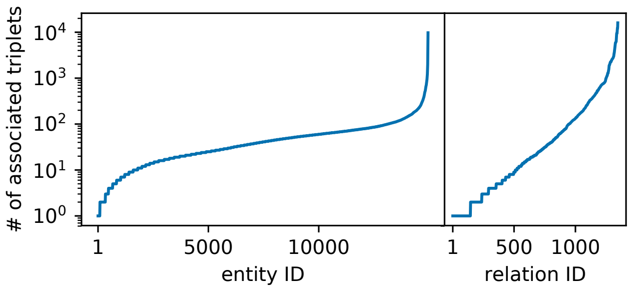

Figure 1: Number of training points (triplet facts) for each entity (left) and relation (right) in the FB15K data set (we sorted entity and relation IDs by frequency). The large variation on the y-axis motivates entity/relation-dependent regularization strengths as proposed in Eq. 2.

The paper is structured as follows: Section 2 summarizes a large class of knowledge graph embedding models and presents our probabilistic perspective on these models in terms of a generative probabilistic process. Section 3 describes our algorithm for hyperparameter tuning. We present experiments in Section 4, compare our method to related work in Section 5, and conclude in Section 6.

2 GENERATIVE KNOWLEDGE GRAPH EMBEDDING MODELS

In this section, we introduce our notation for a large class of knowledge graph embedding models (KG embeddings) from the literature (Section 2.1), and we then generalize these models in two aspects. First, while conventional KG embeddings typically share the same regularization strength across all entities and relationship vectors, we lift

[1]Source code: https://github.com/mandt-lab/knowledge-graph-tuning

this constraint and allow each embedding vector to be regularized differently (Section 2.2). Second, we show that the loss functions of conventional KG embeddings as well as our augmented model class can be obtained as point estimates of a probabilistic generative process of the data (Section 2.3). Drawing on this probabilistic perspective, we can optimize all hyperparameters efficiently using variational expectation-maximization (Section 3).

2.1 CONVENTIONAL KG EMBEDDINGS

We introduce our notation for a large class of knowledge graph embedding models (KG embeddings) from the literature, such as DistMult [Yang et al., 2015], ComplEx [Trouillon et al., 2016], and Holographic Embeddings [Nickel et al., 2016b].

Knowledge graphs are sets of triplet facts (h, r, t) where the 'head' h and 'tail' t both belong to a fixed set of N_e entities, and r describes which one out of a set of N_r relations holds between h and t. KG embeddings represent each entity $e \in [N_\text{e}]$ and each relation $r \in [N_\text{r}]$ by an embedding vector E_e and R_r, respectively, that lives in a semantic embedding space $\mathbb{V}$ with a low dimension K. A model is defined by a real valued score function $f(\text{E}_h, \text{R}_r, \text{E}_t)$. One fits the embedding vectors such that f assigns a high score to observed triplet facts $(h, r, t) \in \mathbb{S}$ in the training set $\mathbb{S}$ and a low score to triplets that do not appear in $\mathbb{S}$.

Examples. We give examples of the two models that reach highest predictive performance to the best of our knowledge. For more models, see [Kadlec et al., 2017].

In the DistMult model [Yang et al., 2015], the embedding space $\mathbb{V} = \mathbb{R}^K$ is real valued, and the score is defined as

$$f(\text{E}_h, \text{R}_r, \text{E}_t) = \sum_{k=1}^{K} \text{E}_{hk}\text{R}_{rk}\text{E}_{tk} \qquad \text{(DistMult)} \qquad (3)$$

where, e.g., $\text{E}_{hk} \in \mathbb{R}$ is the k^th entry of the vector E_h.

The ComplEx model [Trouillon et al., 2016] uses a complex embedding space $\mathbb{V} = \mathbb{C}^K$, and defines the score

$$f(\text{E}_h, \text{R}_r, \text{E}_t) = \sum_{k=1}^{K} \text{Re}[\text{E}_{hk}\text{R}_{rk}\bar{\text{E}}_{tk}] \qquad \text{(ComplEx)} \qquad (4)$$

where $\text{Re}[\,\cdot\,]$ denotes the real part of a complex number, and $\bar{\text{E}}_{tk}$ is the complex conjugate of $\text{E}_{tk} \in \mathbb{C}$.

Tail And Head Prediction. Typical benchmark tasks for KG embeddings are 'tail prediction' and 'head prediction', i.e., completing queries of the form $(h, r, ?)$ and $(?, r, t)$, respectively, by ranking potential completions (h, r, t) by their score $f(\text{E}_h, \text{R}_r, \text{E}_t)$. Most proposals for KG embeddings train a single model for both tail and head prediction. Thus, the loss function is given by Eq. 1, where $\ell_{hrt}(\mathbf{E}, \mathbf{R})$ is a sum of two terms to train for tail and head prediction, respectively. While early works (e.g., [Bordes et al., 2013, Wang et al., 2014, Yang et al., 2015]) trained by maximizing a margin over negative samples, the more recent literature [Kadlec et al., 2017, Liang et al., 2018] suggests that the softmax loss leads to better predictive performance,

$$\begin{aligned}\ell_{hrt}(\mathbf{E}, \mathbf{R}) = &- f(\text{E}_h, \text{R}_r, \text{E}_t) + \log\left(\sum_{t'=1}^{N_\text{e}} e^{f(\text{E}_h, \text{R}_r, \text{E}_{t'})}\right) \\ &- f(\text{E}_h, \text{R}_r, \text{E}_t) + \log\left(\sum_{h'=1}^{N_\text{e}} e^{f(\text{E}_{h'}, \text{R}_r, \text{E}_t)}\right).\end{aligned} \qquad (5)$$

Here, the first line (with the sum over tails t') is the softmax loss for tail prediction, while the second line (with the sum over heads h') is the softmax loss for head prediction.

2.2 REGULARIZATION IN KG EMBEDDINGS

Knowledge graph embedding models are highly sensitive to hyperparameters, especially to the strength of the regularizer [Kadlec et al., 2017]. This can be understood since even large knowledge graphs typically contain only few data points per entity. For example, the FB15K data set contains 483,000 data points, but 88% of all entities $e \in [N_\text{e}]$ appear fewer than 100 times as head or tail of a training point. Moreover, the amount of training data varies strongly across entities and relations (see Figure 1), suggesting that the regularization strength for embedding vectors E_e and R_r should depend on the entity e and relation r.

The loss function for conventional KG embeddings in Eq. 1 regularizes all embedding vectors with the same strength λ. We propose to replace λ by individual regularization strengths λ_e^E and λ_r^R for each entity e and relation r, respectively, and to fit models with the loss function in Eq. 2. It generalize Eq. 1, which one obtains for

$$\lambda_e^\text{E} = \lambda n_e^\text{E}; \quad \lambda_r^\text{R} = \lambda n_r^\text{R}; \quad \text{(conventional models)} \qquad (6)$$

where n_e^E and n_r^R denote the number of times that entity e or relation r appears in the training data, respectively. The proposed augmented models described by Eq. 2 are more flexible as they do not impose a linear relationship between $n_{e/r}^\text{E/R}$ and the regularization strength $\lambda_{e/r}^\text{E/R}$.

The downside of the augmented KG embedding models is that one has to tune a macroscopic number of hyperparameters $\{\lambda_e^\text{E}\}$ and $\{\lambda_r^\text{R}\}$: more than 16,000 in the popular FB15K data set. Tuning such a large number of hyperparameters would be far too computationally expensive in a conventional setup that fits point estimates by minimizing the loss function. For point estimated models, it is well known that one cannot fit hyperparameters to the training data as this would lead to overfitting (see also Supplementary Material). To avoid overfitting, knowledge graph

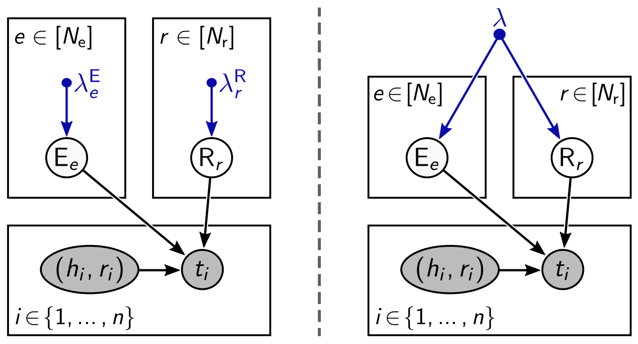

Figure 2: Two different generative processes of triplet facts $\{(h_i, r_i, t_i)\}_{i=1}^n$. Hyperparameters (regularizer strengths) highlighted in blue. Left: proposed class of models augmented with individual regularizer strengths $\lambda_{e/r}^{E/R}$. Right: probabilistic interpretation of conventional models (Eq. 1). The conventional models fix the relative strength of regularizers by reducing all $\lambda_{e/r}^{E/R}$ to a single scalar λ.

embedding models are conventionally tuned by cross validation on heldout data. This requires training a model from scratch for each new hyperparameter setting. Cross validation does not scale beyond models with a handful of hyperparameters, and it is expensive even there (see, e.g., [Kadlec et al., 2017, Lacroix et al., 2018]).

Probabilistic models, by contrast, allow tuning of many hyperparameters in parallel using the empirical Bayes method [Dempster et al., 1977, Maritz, 2018]. We propose a probabilistic formulation of augmented KG embeddings in the next section, and we present a method for efficient hyperparameter tuning in these models in Section 3.

2.3 PROBABILISTIC KG EMBEDDINGS

We now present our probabilistic version of KG embeddings. The probabilistic formulation enables efficient optimization over thousands of hyperparameters, see Section 3.

Reciprocal Facts. The KG embedding models discussed in Sections 2.1 and 2.2 make a direct interpretation as a generative probabilistic process difficult. Training a single model for both head and tail prediction introduces cyclic causal dependencies. As will become clear below, the tail prediction part in Eq. 5 (first line on the right-hand side) corresponds to a generative process where the head h causes the tail t. However, the head prediction part (second line) corresponds to a generative process where t causes h.

To solve this issue, we employ a data augmentation due to Lacroix et al. [2018] that goes as follows. For each relation $r \in [N_r]$, one introduces a new symbol r^{-1}, which has the interpretation of the inverse of r, but whose embedding vector is not tied to that of r. One then constructs an augmented training set $\mathbb{S}'$ by adding the reciprocal facts,

$$\mathbb{S}' := \mathbb{S} \cup \{(t, r^{-1}, h)\}_{(h,r,t)\in\mathbb{S}} \tag{7}$$

One trains the model by minimizing the loss in Eq. 1 or Eq. 2, where the sum over data points is now over $\mathbb{S}'$ instead of $\mathbb{S}$, and where ℓ_{hrt} is given by only the first line of Eq. 5. When evaluating the model performance on a test set, one answers head prediction queries $(?, r, t)$ by answering the corresponding tail prediction query $(t, r^{-1}, ?)$. This data augmentation was introduced in [Lacroix et al., 2018] to improve performance. As we show next, it also has the advantage of enabling a probabilistic interpretation by establishing a causal order where h comes before t.

Generative Process. With the above data augmentation, minimizing the loss function in Eq. 2 is equivalent to point estimating the parameters of the probabilistic graphical model shown in Figure 2 (left). The generative process is:

- For each entity $e \in [N_e]$ and each relation $r \in [N_r]$, draw an embedding $E_e, R_r \in \mathbb{V}$ from the priors
$$p(E_e|\lambda_e^E) \propto e^{-\frac{\lambda_e^E}{p}||E_e||_p^p}; \quad p(R_r|\lambda_r^R) \propto e^{-\frac{\lambda_r^R}{p}||R_r||_p^p}. \tag{8}$$
Here, $p \in \{2, 3\}$ specifies the norm, and the omitted proportionality constant follows from normalization.
- Repeat for each data point to be generated:
  - Draw a head entity h and a relation r from some discrete distribution $P(h, r)$. The choice of this distribution has no influence on inference since h and r are both directly observed.
  - Draw a tail entity
  $$t \sim \text{Categorical}(\text{softmax}_t(f(E_h, R_r, E_t)) \tag{9}$$
  where f is the score (e.g., Eq. 3 or 4), and
  $$\text{softmax}_t(f(E_h, R_r, E_t)) = \frac{e^{f(E_h, R_r, E_t)}}{\sum_{t'} e^{f(E_h, R_r, E_{t'})}}.$$
  - Add the triplet fact (h, r, t) to the data set $\mathbb{S}'$.

This process defines a log joint distribution over $\mathbf{E}$, $\mathbf{R}$, and the data $\mathbb{S}'$, conditioned on the hyperparameters $\{\lambda_{e/r}^{E/R}\}$, which we denote collectively by the boldface symbol $\boldsymbol{\lambda}$,

$$\begin{aligned}\log p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda}) &= \\ &= \sum_{e\in[N_e]} \log p(E_e|\lambda_e^E) + \sum_{r\in[N_r]} \log p(R_r|\lambda_r^R) \\ &+ \sum_{(h,r,t)\in\mathbb{S}'} \big[\log P(h, r) + \log p(t|h, r, \mathbf{E}, \mathbf{R})\big].\end{aligned} \tag{10}$$

Using Eq. 9, it is easy to see that $\log p(t|h, r, \mathbf{E}, \mathbf{R})$ is the negative of the first line of Eq. 5. Thus, up to an additive

Algorithm 1: Conventional training of knowledge graph embedding models (stochastic gradient descent).

```
Input:  Augmented data set $\mathbb{S}'$, see Eq. 7.
        Model $p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})$ as defined by Eqs. 8-10.
        Hyperparameters $\boldsymbol{\lambda}$. Learning rate $\alpha$.
Output: Trained embedding vectors $\mathbf{E}$ and $\mathbf{R}$.
1 Initialize embedding vectors $\mathbf{E}$ and $\mathbf{R}$ randomly.
2 repeat
3     Draw a minibatch $\mathbb{B} \subset \mathbb{S}'$.
4     Calculate a minibatch estimate $\hat{L}_{\mathbb{B}}$ of the loss
       $L(\mathbf{E}, \mathbf{R}|\boldsymbol{\lambda}) := -\log p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})$.
5     Update $\mathbf{E} \leftarrow \mathbf{E} - \alpha \nabla_{\mathbf{E}} \hat{L}_{\mathbb{B}}$ and $\mathbf{R} \leftarrow \mathbf{R} - \alpha \nabla_{\mathbf{R}} \hat{L}_{\mathbb{B}}$.
  until convergence (see Section 3.2)
```

term that depends only on $\boldsymbol{\lambda}$, the log joint distribution in Eq. 10 is the negative of the loss function, and minimizing the loss over $\mathbf{E}$ and $\mathbf{R}$ is equivalent to a maximum a posteriori (MAP) approximation of the probabilistic model.

Figure 2 compares the generative process of the augmented KG embeddings proposed in Section 2.2 (left part of the figure) to the generative process for conventional KG embeddings that one obtains by setting λ_e^{E} and λ_r^{R} as in Eq. 6 (right). The augmented models are more flexible due to the large number of hyperparameters $\lambda_{e/r}^{\text{E/R}}$. We discuss next how the probabilistic interpretation allows us to efficiently optimize over this large number of hyperparameters.

3 HYPERPARAMETER OPTIMIZATION

We now describe the proposed method for hyperparameter tuning in the probabilistic knowledge graph embedding models introduced in Section 2.3. The method is based on variational expectation-maximization (EM). We first derive an approximate coordinate update equation for the hyperparameters (Section 3.1) and then cover details of the parameter initialization (Section 3.2).

Variational EM optimizes a lower bound to the marginal likelihood of the model over hyperparameters $\boldsymbol{\lambda}$, with model parameters $\mathbf{E}$ and $\mathbf{R}$ integrated out. As we show in the supplementary material, the naive alternative of simultaneously optimizing the original model's loss function over model parameters and hyperparameters would lead to divergent solutions. Variational EM avoids such divergent solutions by keeping track of parameter uncertainty. We elaborate on the role of parameter uncertainty in the supplementary material.

3.1 VARIATIONAL EM FOR KNOWLEDGE GRAPH EMBEDDING MODELS

Our proposed algorithm based on variational EM can easily be implemented in an existing model architecture by making a few modifications. Algorithm 1 shows the conventional way to train a knowledge graph embedding model using stochastic gradient descent (SGD). The log joint distribution in Eqs. 8-10 defines a loss function $L(\mathbf{E}, \mathbf{R}|\boldsymbol{\lambda}) := -\log p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})$ of the form of Eq. 2. In SGD, one repeatedly calculates an estimate $\hat{L}_{\mathbb{B}}$ of this loss function based on a minibatch $\mathbb{B}$ of training points, and one obtains gradient estimates by backpropagating through $\hat{L}_{\mathbb{B}}$.

Algorithm 2: Variational EM for knowledge graph embedding models with coordinate updates for $\boldsymbol{\lambda}$.

```
Input:  Augmented data set $\mathbb{S}'$, see Eq. 7.
        Model $p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})$ as defined by Eqs. 8-10.
        Initial values for $\boldsymbol{\mu}$, $\boldsymbol{\xi}$, $\boldsymbol{\lambda}$, see Section 3.2.
        Numbers $T_{\text{E}}$, $T_{\text{EM}}$ of E-steps and EM-steps.
        Learning rates $\alpha_{\boldsymbol{\mu}}, \alpha_{\boldsymbol{\xi}} > 0$ and $\alpha_{\boldsymbol{\lambda}} \in (0, 1]$.
Output: Optimized hyperparameters $\boldsymbol{\lambda}$, embedding
        vectors $\boldsymbol{\mu}^{\text{E/R}} \pm e^{\boldsymbol{\xi}^{\text{E/R}}}$ with uncertainties.
1 Initialize means $\boldsymbol{\mu}^{\text{E/R}}$ and log standard deviations $\boldsymbol{\xi}^{\text{E/R}}$
   around a pretrained model, see Section 3.2.
2 for $t \leftarrow 1$ to $T_{\text{E}} + T_{\text{EM}}$ do
3     Draw a minibatch $\mathbb{B} \subset \mathbb{S}'$.
4     Draw Gaussian noise $\boldsymbol{\epsilon}^{\text{E}}, \boldsymbol{\epsilon}^{\text{R}} \sim \mathcal{N}(0, I)$.
5     Calculate a minibatch estimate $\hat{L}_{\mathbb{B}}$ of the loss with
       injected noise, $L(\boldsymbol{\mu}^{\text{E}} + e^{\boldsymbol{\xi}^{\text{E}}} \odot \boldsymbol{\epsilon}^{\text{E}}, \boldsymbol{\mu}^{\text{R}} + e^{\boldsymbol{\xi}^{\text{R}}} \odot \boldsymbol{\epsilon}^{\text{R}} | \boldsymbol{\lambda})$
       ("$\odot$" denotes elementwise multiplication).
6     Update $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha_{\boldsymbol{\mu}} \nabla_{\boldsymbol{\mu}} \hat{L}_{\mathbb{B}}$.
7     Update $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \alpha_{\boldsymbol{\xi}} (\nabla_{\boldsymbol{\xi}} \hat{L}_{\mathbb{B}} - \mathbf{1})$.
8     if $t > T_{\text{E}}$ then
9         Update $\boldsymbol{\lambda} \leftarrow \left[(1 - \alpha_{\boldsymbol{\lambda}})\boldsymbol{\lambda}^{-1} + \alpha_{\boldsymbol{\lambda}} \hat{\boldsymbol{\lambda}}^{-1}\right]^{-1}$.
      end                                  ▷ M-step (see Eq. 17).
  end
```

Algorithm 2 shows the modifications that are necessary to implement hyperparameter optimization. We describe the algorithm in detail below. In summary, one has to: (i) inject noise into the loss estimate $\hat{L}_{\mathbb{B}}$ (lines 4-5); (ii) learn the optimal amount $\boldsymbol{\xi}$ of noise via SGD (line 7); and (iii) update the hyperparameters $\boldsymbol{\lambda}$ (line 9).

Variational Expectation-Maximization. Our probabilistic interpretation of knowledge graph embedding models allows us to optimization over all hyperparameters $\{\lambda_e^{\text{E}}\}$ and $\{\lambda_r^{\text{R}}\}$ in parallel via the expectation-maximization (EM) algorithm [Dempster et al., 1977]. This algorithm treats the model parameters $\mathbf{E}$ and $\mathbf{R}$ as latent variables that have to be integrated out. The EM algorithm alternates between a step in which the latent variables are integrated out ('E-step'), and an update step for the hyperparameters $\boldsymbol{\lambda}$ ('M-step'). We use a version of EM based on variational inference, termed variational EM [Bernardo et al., 2003], that avoids the integration step. We further derive an approximate coordinate update equation for the hyperparameters $\boldsymbol{\lambda}$, which lead to a significant speedup over gradient updates in our experiments.

Each choice of hyperparameters $\boldsymbol{\lambda}$ defines a different variant of the model. The marginal likelihood of the data,

$$p(\mathbb{S}'|\boldsymbol{\lambda}) = \int p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})\, d\mathbf{E}\, d\mathbf{R} \tag{11}$$

quantifies how well a given model variant describes the data $\mathbb{S}'$. Maximizing $p(\mathbb{S}'|\boldsymbol{\lambda})$ over $\boldsymbol{\lambda}$ thus yields the model variant that fits the data best. However, $p(\mathbb{S}'|\boldsymbol{\lambda})$ is unavailable in closed form as the integral in Eq. 11 is intractable.

To circumvent the problem of the intractable marginal likelihood, we use variational inference (VI) [Jordan et al., 1999]. Rather than integrating over the entire space of model parameters $\mathbf{E}$ and $\mathbf{R}$, we maximize a lower bound on the marginal likelihood. We introduce a so-called variational family of Gaussian probability distributions,

$$q_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{E}, \mathbf{R}) = q_{\boldsymbol{\mu}^{\mathrm{E}},\boldsymbol{\sigma}^{\mathrm{E}}}(\mathbf{E})\; q_{\boldsymbol{\mu}^{\mathrm{R}},\boldsymbol{\sigma}^{\mathrm{R}}}(\mathbf{R}) \tag{12}$$

with

$$q_{\boldsymbol{\mu}^{\mathrm{E}},\boldsymbol{\sigma}^{\mathrm{E}}}(\mathbf{E}) = \prod_{e\in[N_e]} \prod_{k=1}^{K} \mathcal{N}(\mathrm{E}_{ek}; \mu_{ek}, \sigma_{ek}^2) \tag{13}$$

and analogously for $q_{\boldsymbol{\mu}^{\mathrm{R}},\boldsymbol{\sigma}^{\mathrm{R}}}(\mathbf{R})$. Here, the means $\boldsymbol{\mu} \equiv (\boldsymbol{\mu}^{\mathrm{E}}, \boldsymbol{\mu}^{\mathrm{R}})$ and the standard deviations $\boldsymbol{\sigma} \equiv (\boldsymbol{\sigma}^{\mathrm{E}}, \boldsymbol{\sigma}^{\mathrm{R}})$ are so-called variational parameters over which we optimize.

Evoking Jensen's inequality, the log marginal likelihood is then lower-bounded by the *evidence lower bound* [Blei et al., 2017, Zhang et al., 2018], or ELBO:

$$\begin{aligned}
&\log p(\mathbb{S}'|\boldsymbol{\lambda}) \\
&\quad \geq \mathbb{E}_{\mathbf{E},\mathbf{R}\sim q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}\big[\log p(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda}) - \log q_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{E}, \mathbf{R})\big] \\
&\quad = -\mathbb{E}_{\mathbf{E},\mathbf{R}\sim q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}\big[L(\mathbf{E}, \mathbf{R}, \mathbb{S}'|\boldsymbol{\lambda})\big] + H[q_{\boldsymbol{\mu},\boldsymbol{\sigma}}] \\
&\quad =: \mathrm{ELBO}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma}).
\end{aligned} \tag{14}$$

Here, in the second step, we identified the log joint probability as the negative of the loss L of the corresponding point estimated model, and $H[q_{\boldsymbol{\mu},\boldsymbol{\sigma}}]$ is the entropy of $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$.

The bound in Eq. 14 is tight if the variational distribution $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is the true posterior of the model for given $\boldsymbol{\lambda}$. Since it is a lower bound, maximizing the ELBO over $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ minimizes the gap and yields the best approximation of the marginal likelihood. We thus take the ELBO as a proxy for the marginal likelihood, and we maximize it also over $\boldsymbol{\lambda}$ to find near-optimal hyperparameters.

Gradient updates for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. We maximize the ELBO concurrently over both variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as well as over hyperparameters $\boldsymbol{\lambda}$. Updating the variational parameters is called the "E-step". Here, we use gradient updates using Black Box reparameterization gradients [Kingma and Welling, 2014, Rezende et al., 2014]. This has the advantage of being agnostic to the model architecture as long as the score $f(\mathrm{E}_h, \mathrm{R}_r, \mathrm{E}_t)$ (e.g., Eqs. 3-4) is differentiable, and it requires only few changes compared to the standard SGD training loop in Algorithm 1.

To make sure that the standard deviations are always positive, we parameterize them by their logarithms $\boldsymbol{\xi}$,

$$\boldsymbol{\sigma} = e^{\boldsymbol{\xi}} \qquad \text{(elementwise)}, \tag{15}$$

and we optimize over $\boldsymbol{\mu}$ and $\boldsymbol{\xi}$ using SGD. We obtain an unbiased estimate of the term $\mathbb{E}_{q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}[L]$ in Eq. 14 by drawing a single sample from $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ (lines 4-5 in Algorithm 2). The reparameterization gradient trick uses the fact that for, e.g., noise $\epsilon_{ek}^{\mathrm{E}} \sim \mathcal{N}(0,1)$ from a standard normal distribution, $\mu_{ek}^{\mathrm{E}} + \sigma_{ek}^{\mathrm{E}}\epsilon_{ek}^{\mathrm{E}}$ is distributed as $\mathcal{N}\big(\mu_{ek}^{\mathrm{E}}, (\sigma_{ek}^{\mathrm{E}})^2\big)$. The entropy part $H[q_{\boldsymbol{\mu},\boldsymbol{\sigma}}]$ of the ELBO (Eq. 14) can be calculated analytically. Up to an additive constant, it is given by the sum over all log standard deviations ξ_{ek}^{E} and ξ_{rk}^{R}. Thus, its gradient with respect to $\boldsymbol{\xi}$ has the constant value of one in each coordinate direction, which we denote by the bold face term "$\mathbf{1}$" on line 7 of Algorithm 2.

Coordinate updates for $\boldsymbol{\lambda}$. Optimizing the ELBO over $\boldsymbol{\lambda}$ leads to an improved set of hyperparameters provided that the ELBO is a good approximation of the marginal likelihood $p(\mathbb{S}'|\boldsymbol{\lambda})$. However, this is typically not the case at the beginning of the optimization when the variational distribution is still a poor fit of the posterior. We therefore begin the optimization with some number T_{E} of pure "E-step" updates during which we keep $\boldsymbol{\lambda}$ fixed. After T_{E} "E-steps", we alternate between "E" and "M" steps, where the latter update the hyperparameters $\boldsymbol{\lambda}$. In our experiments, we found that the optimization converged slowly when we used gradient updates for $\boldsymbol{\lambda}$. To speed up convergence, we therefore derive approximate coordinate updates for $\boldsymbol{\lambda}$.

To simplify the notation, we derive the update equation only for a single hyperparameter λ_e^{E}. Updates for λ_r^{R} are analogous. The only term in the ELBO (Eq. 14) that depends on λ_e^{E} is the expected log prior, $\mathbb{E}_{q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}[\log p(\mathrm{E}_e|\lambda_e^{\mathrm{E}})]$. Since this term is independent of the data we can write it out explicitly. The omitted proportionality constant in the prior (Eq. 8) is dictated by normalization. We find,

$$\log p(\mathrm{E}_e|\lambda_e^{\mathrm{E}}) = \frac{K'}{p}\log\left(\lambda_e^{\mathrm{E}}\right) - \frac{\lambda_e^{\mathrm{E}}}{p}||\mathrm{E}_e||_p^p + \text{const}, \tag{16}$$

where $K' = K$ for a real-valued embedding space $\mathbb{V} = \mathbb{R}^K$ (as in DistMult) and $K' = 2K$ if $\mathbb{V} = \mathbb{C}^K$ (as in ComplEx). Setting $\nabla_{\lambda_e^{\mathrm{E}}}\mathbb{E}_{q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}[\log p(\mathrm{E}_e|\lambda_e^{\mathrm{E}})]$ to zero we find that the regularizer strength $\hat{\lambda}_e^{\mathrm{E}}$ that maximizes the ELBO for given $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ satisfies

$$\frac{1}{\hat{\lambda}_e^{\mathrm{E}}} = \frac{1}{K'}\,\mathbb{E}_{q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}\big[||\mathrm{E}_e||_p^p\big] \tag{17}$$

In moderately high embedding dimensions K, we can approximate the right-hand side of Eq. 17 accurately by sampling from $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$. It is the expectation of the average of a large number of independent random variables, and therefore follows a highly peaked distribution. The update step on line 9 of Algorithm 2 uses a conservative weighted average between the current and the optimal value of $1/\lambda_e^{\mathrm{E}}$ with a learning rate $\alpha_{\boldsymbol{\lambda}} \in (0, 1]$. This effectively averages the estimates over past training steps with a decaying weight. Note that, for $\mathbb{V} = \mathbb{R}^K$, Eq. 17 has a closed form solution for $p = 2$ and $p = 3$, but we found it unnecessary in our experiments to implement specialized code for these cases.

Absence of overfitting. While the variational EM algorithm keeps track of uncertainty of model parameters, it fits only point estimates for the hyperparameters $\boldsymbol{\lambda}$. This is justified in our setup since there are much fewer hyperparameters than model parameters: each entity e and each relation r has an embedding vector E_e or R_r with $K = 2{,}000$ scalar components in our experiments, but only a single scalar hyperparameter λ_e^{E} or λ_r^{E}. We therefore expect much smaller posterior uncertainty for $\boldsymbol{\lambda}$ than for $\mathbf{E}$ and $\mathbf{R}$, which justifies point estimating $\boldsymbol{\lambda}$. Had we instead chosen a very flexible prior distribution with many hyperparameters per entity and relation, the EM algorithm would have essentially fitted the prior to the variational distribution, leading to an ill-posed problem. Judging from learning curves on the validation set, we did not detect any overfitting in variational EM.

3.2 PRE- AND RE-TRAINING

Variational EM (Algorithm 2) converges more slowly than fitting point estimates (Algorithm 1) because the injected noise increases the variance of the gradient estimator. To speed up convergence, we train the model in three consecutive phases: pre-training, variational EM, and re-training.

In the pre-training phase, we keep the hyperparameters $\boldsymbol{\lambda}$ fixed and fit point estimates $\mathbf{E}$ and $\mathbf{R}$ to the model using standard SGD (Algorithm 1). We found the final predictive performance (after the variational EM and re-training phases) to be insensitive to the initial hyperparameters. We use early stopping based on the mean reciprocal rank (see Eq. 18 in Section 4 below), evaluated on the validation set.

In the variational EM phase (Algorithm 2 and Section 3.1), we initialize the variational distribution $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ around the pre-trained model parameters $\mathbf{E}$ and $\mathbf{R}$. In detail, we initialize $\boldsymbol{\mu}^{\mathrm{E}} \leftarrow \mathbf{E}$ and $\boldsymbol{\mu}^{\mathrm{R}} \leftarrow \mathbf{R}$, and we initialize the components of $\boldsymbol{\sigma}$ with a value that is small compared to the typical components of $\boldsymbol{\mu}$ (0.2 in our experiments).

In the re-training phase, we fit again point estimates $\mathbf{E}$ and $\mathbf{R}$ with Algorithm 1, this time using the optimized hyperparameters $\boldsymbol{\lambda}$. We use the resulting models to evaluate the predictive performance, see results in Section 4.

Alternatively to re-training a point estimated model, one could also perform predictions by averaging predictive probabilities over samples from the variational distribution $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$. If $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is a good approximation of the model posterior then this results in an approximate Bayesian form of link prediction. In our experiments, we found that, in low embedding dimensions $K \lesssim 100$, predictions based on samples from $q_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ outperformed predictions based on point estimates. In higher embedding dimensions, however, the point estimated models from the re-training phase had better predictive performance. We interpret this somewhat counterintuitive observation as a failure of the fully factorized Gaussian variational approximation to adequately approximate the true posterior.

4 EXPERIMENTAL RESULTS

We test the performance of the proposed model augmentation and the scalable hyperparameter tuning algorithm with two models and four different data sets. In this section, we report results using standard benchmark metrics and we compare to the previous state of the art. We also analyze the relationship between the optimized regularizer strengths and the frequency of entities and relations.

Model architectures and baselines. We report results for the DistMult model [Yang et al., 2015] and the ComplEx model [Trouillon et al., 2016]. We follow [Lacroix et al., 2018] for details of the model architecture: we use reciprocal facts as described at the end of Section 2.3, $(p = 3)$-norm regularizers, and an embedding dimension of $K = 2000$. We compare our results to the previous state of the art: [Dettmers et al., 2018, Kadlec et al., 2017] for DistMult and [Lacroix et al., 2018] for ComplEx.

Data sets. We used four standard data sets. The first two are FB15K from the Freebase project [Bollacker et al., 2008] and WN18 from the WordNet database [Bordes et al., 2014]. The other two data sets, FB15K-237 and WN18RR, are modified versions of FB15K and WN18 due to [Toutanova and Chen, 2015, Dettmers et al., 2018]. The motivation for the modified data sets is that FB15K and WN18 contain near duplicate relations that lead to leakage into the test set, which makes link prediction trivial for some facts, thus encouraging overfitting. In FB15K-237 and WN18RR these near duplicates were removed.

Metrics. We report two standard metrics used in the KG embedding literature: mean reciprocal rank (MRR) and Hits@10. We average over head and tail prediction on the test set $\mathbb{S}_{\text{test}}$, which is equivalent to averaging only over tail prediction on the augmented test set $\mathbb{S}'_{\text{test}}$, see Eq. 7.

All results are obtained in the 'filtered' setting introduced in [Bordes et al., 2013], which takes into account that more than one tail may be a correct answer to a query $(h, r, ?)$.

Table 1: Model performances (in 'filtered' setting; see Eqs. 18-19). *Lacroix et al. [2018] report performance metrics only with two decimals. In order to show three decimals, we reproduced their results using the code provided by the authors. When rounding to two digits, we recover all values reported in [Lacroix et al., 2018] except that the MRR for FB15K-237 is reported there as 0.37. The small discrepancy in the last decimal may be explained by different random seeds.

		data set →	WN18RR		WN18		FB15K-237		FB15K	
↓ model	↓ variant	metric →	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
DistMult	Yang et al. [2015] (orig.)		–	–	0.83	0.942	–	–	0.35	0.577
DistMult	Kadlec et al. [2017]		–	–	0.790	0.950	–	–	0.837	0.904
DistMult	Dettmers et al. [2018]		0.43	0.49	0.822	0.936	0.241	0.419	0.654	0.824
DistMult	Ours (after variational EM)		**0.455**	**0.544**	**0.911**	**0.961**	**0.357**	**0.548**	**0.841**	**0.914**
ComplEx	Trouillon et al. [2016] (orig.)		–	–	0.941	0.947	–	–	0.692	0.840
ComplEx	Lacroix et al. [2018]*		0.478	0.569	0.952	0.963	0.364	0.555	**0.857**	0.909
ComplEx	Ours (after variational EM)		**0.486**	**0.579**	**0.953**	**0.964**	**0.365**	**0.560**	0.854	**0.915**

When calculating the rank of the target tail t one therefore ignores any competing tails t' if the corresponding fact (h, r, t') exists in either the training, validation, or test set. More formally, the fitlered rank, denoted below as $\text{rank}_{\text{filt.}}(t|h, r)$, is defined as one plus the number of 'incorrect' facts (h, r, t') with the the given h and r for which $f(\text{E}_h, \text{R}_r, \text{E}_{t'}) \geq f(\text{E}_h, \text{R}_r, \text{E}_t)$. Here, candidate facts (h, r, t') are considered 'incorrect' if they appear neither in the training nor in the validation or test set.

Given a test set $\mathbb{S}'_{\text{test}}$ with reciprocal facts (Eq. 7), the mean reciprocal rank (MRR) is (see, e.g., [Yang et al., 2015]),

$$\text{MRR} = \frac{1}{|\mathbb{S}'_{\text{test}}|} \sum_{(h,r,t)\in\mathbb{S}'_{\text{test}}} \frac{1}{\text{rank}_{\text{filt.}}(t|h, r)}. \tag{18}$$

With 'Hits@10', we denote the fraction of test queries for which the filtered rank is at most 10,

$$\text{Hits@10} = \frac{|\{(h, r, t) \in \mathbb{S}'_{\text{test}} : \text{rank}_{\text{filt.}}(t|h, r) \leq 10\}|}{|\mathbb{S}'_{\text{test}}|}. \tag{19}$$

Quantitative results. Table 1 summarizes our quantitative results. The top half of the table shows results for the DistMult model. Our models with individually optimized regularization strengths significantly outperform the previous state of the art across all four data sets.

For the ComplEx model, the performance improvements are less pronounced (lower half of Table 1). This may be explained by the fact that the results in [Lacroix et al., 2018] were already obtained after large scale expensive hyperparameter tuning using grid search. By contrast, the hyperparameter search with our proposed method required only a single run per data set. Even for the largest data set FB15K, the variational EM phase took less than three hours on a single GPU. Despite the much cheaper hyperparameter optimization, our models slightly outperform the previous state of the art on three out of the four considered data sets, with only a small degradation on the fourth.

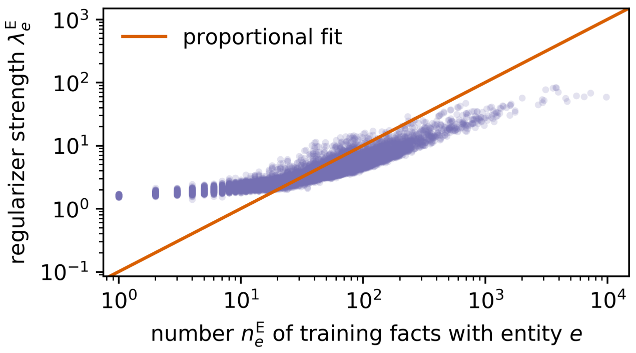

Figure 3: Relationship between the learned regularizer strengths λ_e^{E} and the frequency of each entity. The conventional proportional scaling (red line) overregularizes frequent entities and underregularizes infrequent ones.

Qualitative results. Finally, we study the relationship between optimized hyperparameters and frequencies of entities in the training data. Figure 3 shows the learned λ_e^{E} for all entities e as a function of the number of times n_e^{R} that each entity e appears in the training corpus of the FB15K data set. The red line is the best proportional fit $\lambda_e^{\text{E}} \propto n_e^{\text{E}}$ to the results, as is imposed by conventional models (Eq. 6).

Our findings confirm the general idea to use stronger regularization for entities with more training data. The Bayesian interpretation can explain this observation: a small amount of training data typically leads to high posterior uncertainty, which leads to small $\hat{\lambda}_e^{\text{E}}$ in Eq. 17. However, our results indicate that imposing a proportionality between λ_e^{E} and n_e^{E} would be a poor choice that significantly underregularizes infrequent entities and overregularizes frequent entities (note the logarithmic scale in Figure 3). Our empirical findings may inspire future theoretical work that derives an optimal frequency dependency of the regularization strength in tensor factorization models.

5 RELATED WORK

Related work to this paper can be grouped into link prediction algorithms and variational inference.

Link Prediction. Link prediction in knowledge graphs has gained a lot of attention as it may guide a way towards automated reasoning with real world data. For a review see [Nickel et al., 2016a]. Two different approaches for link prediction are predominant in the literature. In statistical relational learning, one infers explicit rules about relations (such as transitivity or commutativity) by detecting statistical patterns in the training set. One then uses these rules for logic reasoning [Friedman et al., 1999, Kersting et al., 2011, Niu et al., 2012, Pujara et al., 2015].

Our work focuses on a complementary approach that builds on knowledge graph embedding models. This line of research started with the proposal of the TransE model [Bordes et al., 2013], which models relational facts as vector additions in a semantic space. More recently, a plethora of different knowledge graph embedding models based on tensor factorizations have been proposed. We summarize here only the path that lead to the current state of the art. Different models make different trade-offs between generality and effective use of training data.

Canonical tensor decomposition [Hitchcock, 1927] uses independent embeddings for entities in the head or tail position of a fact. DistMult [Yang et al., 2015, Toutanova and Chen, 2015], by contrast, uses the same embeddings for entities in head and tail position, thus making use of more training data per entity embedding, but restricting the model to symmetric relations. The ComplEx model [Trouillon et al., 2016] lifts this restriction by multiplying the head, relation, and tail embeddings in an asymmetric way. To the best of our knowledge, the current state of the art was presented by Lacroix et al. [2018], who improved upon the ComplEx model by introducing reciprocal relations and using a better regularizer.

The sensitivity of KG embeddings to the choice of hyperparameters, such as regularizer strengths, was first pointed out in [Kadlec et al., 2017]. A popular heuristic is to regularize each embedding every time it appears in a minibatch, thus effecitvely regularizing embeddings proportionally to their frequency [Srebro and Salakhutdinov, 2010, Lacroix et al., 2018]. In contrast, we propose to learn entity-dependent regularization strengths without relying on heuristics.

Vilnis et al. [2018] proposed a new model that is probabilistic in the sense that it assigns probabilities to the results of queries. However, in contrast to our proposal, the model is not a probabilistic generative model of the data set.

Variational Inference. Variational inference (VI) is a powerful technique to approximate a Bayesian posterior over latent variables given observations [Jordan et al., 1999, Blei et al., 2017, Zhang et al., 2018]. Besides approximating the posterior, VI also estimates the marginal likelihood of the data. This allows for iterative hyperparameter tuning, (variational EM) [Bernardo et al., 2003], which is the main benefit of the Bayesian approach used in this paper.

Our paper builds on recent probabilistic extensions of embedding models to Bayesian models, such as word [Barkan, 2017] or paragraph [Ji et al., 2017] embeddings. In these works, the words are embedded into a K-dimensional space. It has been shown that using a probabilistic approach leads to better performance on small data sets, and allows these models to be combined with powerful priors, such as for time series modeling [Bamler and Mandt, 2017, 2018, Jähnichen et al., 2018]. Yet, the underlying probabilistic models in these papers are very different from the ones considered in our work.

Bayesian Optimization. An alternative method that can be used for hyperparameter optimization is Bayesian optimization. However, Bayesian optimization does not scale to the large number of hyperparameters that we tune in this work. Most practical applications of Bayesian optimization (e.g., [Snoek et al., 2012, Wang et al., 2013]) tune only tens of hyperparameters, rather than ten thousands. This is because Bayesian optimization treats the model as a black box, which it can only train and then evaluate for a given choice of hyperparameters at a time. Each such evaluation contributes a single data point to fit an auxiliary model over the hyperparameters. By contrast, variational EM has access to gradient information to train all hyperparameters in parallel, and concurrently with the model parameters.

6 CONCLUSIONS

We augmented a large class of popular knowledge graph embedding models in such a way that every entity embedding and every relationship embedding vector has their own regularizer, and showed that it is possible to tune these potentially thousands of hyperparameters in a scalable way. Our approach is motivated by the observation that sharing a common regularization parameter across all embeddings leads to over-regularization.

We treat knowledge graph embeddings as generative probabilistic models, making them amenable to Bayesian model selection. We derived approximate coordinate updates for the hyperparameters in the framework of variational EM. We applied our method to generalizations of the DistMult and ComplEx models and outperformed the state of the art for link prediction. The approach can be applied to a wide range of models with minimal modifications to the training routine. In the future, it would be interesting to investigate whether tighter variational bounds [Burda et al., 2016, Bamler et al., 2017] may further improve model selection.

References

Robert Bamler and Stephan Mandt. Dynamic word embeddings. In *International conference on Machine learning*, 2017.

Robert Bamler and Stephan Mandt. Improving optimization in models with continuous symmetry breaking. In *International Conference on Machine Learning*, 2018.

Robert Bamler, Cheng Zhang, Manfred Opper, and Stephan Mandt. Perturbative black box variational inference. In *Advances in Neural Information Processing Systems*, pages 5079–5088, 2017.

Oren Barkan. Bayesian neural word embedding. In *Association for the Advancement of Artificial Intelligence*, pages 3135–3143, 2017.

JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, M West, et al. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 7:453–464, 2003.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2): 233–259, 2014.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Association for the Advancement of Artificial Intelligence*, 2018.

Jeffrey Scott Eder. Knowledge graph based search system, June 21 2012. US Patent App. 13/404,109.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Patrick Jähnichen, Florian Wenzel, Marius Kloft, and Stephan Mandt. Scalable generalized dynamic topic models. In *Artificial Intelligence and Statistics*, 2018.

Geng Ji, Robert Bamler, Erik B Sudderth, and Stephan Mandt. Bayesian paragraph vectors. *Symposium on Advances in Approximate Bayesian Inference*, 2017.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Association for the Advancement of Artificial Intelligence*, pages 985–991, 2016.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74. Association for Computational Linguistics, 2017.

Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational ai: logic, probability and computation. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, pages 1–9, 2011.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, 2018.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 689–698. International World Wide Web Conferences Steering Committee, 2018. ISBN 978-1-4503-5639-8.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Association for the Advancement of Artificial Intelligence*, volume 15, pages 2181–2187, 2015.

Johannes S Maritz. *Empirical Bayes Methods with Applications: 0*. Chapman and Hall/CRC, 2018.

Dat Quoc Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*, 2017.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104 (1):11–33, 2016a.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *Association for the Advancement of Artificial Intelligence*, volume 2, pages 3–2, 2016b.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

Jay Pujara, Hui Miao, Lise Getoor, and William W Cohen. Using semantics and statistics to turn data into knowledge. *AI Magazine*, 36(1):65–74, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. Implicit reasonet: Modeling large-scale structured relationships with shared memory. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2016.

Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Association for the Advancement of Artificial Intelligence*, volume 17, pages 1236–1242, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

Nathan Srebro and Ruslan R Salakhutdinov. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *Advances in Neural Information Processing Systems*, pages 2056–2064, 2010.

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

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

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Annual Meeting of the Association for Computational Linguistics*, 2018.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Association for the Advancement of Artificial Intelligence*, volume 14, pages 1112–1119, 2014.

Zhigang Wang and Juan-Zi Li. Text-enhanced representation learning for knowledge graph. In *International Joint Conference on Artificial Intelligence*, pages 1293–1299, 2016.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, 2013.

Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. Ssp: Semantic space projection for knowledge graph embedding with text descriptions. In *Association for the Advancement of Artificial Intelligence*, volume 17, pages 3104–3110, 2017.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Supplementary Material to "Augmenting and Tuning Knowledge Graph Embeddings"

Robert Bamler[*]
Department of Computer Science
University of California, Irvine
Irvine, CA 92617

Farnood Salehi[*]
École Polytechnique Fédérale
de Lausanne (EPFL),
Switzerland

Stephan Mandt
Department of Computer Science
University of California, Irvine
Irvine, CA 9261

The Role of Parameter Uncertainty in the Proposed Hyperparameter Optimization

Bayesian inference and the idea of measuring uncertainty are somewhat uncommon in the literature for knowledge graph embeddings. To clarify why uncertainty is important in the proposed hyperparameter optimization (Section 3.1 of the main text), we compare the method here to a more naive approach that will turn out to fail because it ignores uncertainty. We discuss the failure of the naive approach and the benefit of estimating parameter uncertainty first intuitively and then more formally.

Intuitive picture. The variational EM algorithm maximizes (a lower bound on) the marginal likelihood $p(\mathbb{S}'|\boldsymbol{\lambda})$, Eq. 14 of the main text. In a more naive attempt to hyperparameter tuning without cross validation, one might be tempted to skip the marginalization over model parameters $\mathbf{E}$ and $\mathbf{R}$. Instead, one might try to directly maximize the log joint probability $\log p(\mathbf{E},\mathbf{R},\mathbb{S}'|\boldsymbol{\lambda})$, i.e., minimize the loss $L = -\log p(\mathbf{E},\mathbf{R},\mathbb{S}'|\boldsymbol{\lambda})$, over $\mathbf{E}$, $\mathbf{R}$, and hyperparameters $\boldsymbol{\lambda}$. This approach, however, would lead to divergent solutions because the log joint probability is unbounded.

The log joint probability contains the log priors (first two terms on the right-hand side of Eq. 10 of the main text). Figure S1 shows the prior $p(\mathrm{E}_{ek}|\lambda^{\mathrm{E}}_e)$, Eq. 8 of the main text, of a single component E_{ek} of an entity embedding assuming, for simplicity, a real embedding space. With growing regularizer strength (increasing λ^{E}_e), the prior becomes narrower and narrower. As the peak narrows, it also grows higher due to the normalization constraint. In the limit $\lambda^{\mathrm{E}}_e \to \infty$, the prior collapses to an infinitely narrow and high δ-peak at zero.

A hyperparameter tuning method that ignores posterior uncertainty can exploit the unbounded growth of the maximum of the prior to send the log joint distribution to infinity (i.e., the loss $L \to -\infty$). Without any posterior uncertainty, one can set the model parameter E_{ek} precisely to

[*]joint first authorship.

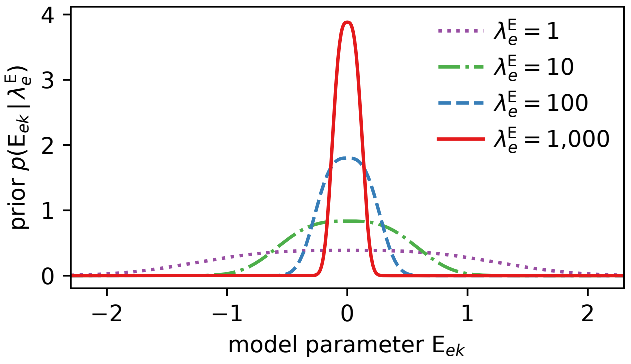

Figure S1: Prior (Eq. 8 of the main text with $p = 3$) for a single model parameter E_{ek}. As the prior gets more peaked for growing regularizer strength λ^{E}_e, the height of the peak grows unboundedly. An (incorrect) hyperparameter optimization method that ignores parameter uncertainty could end up exploiting this unboundedness and diverge to $\lambda^{\mathrm{E}}_e \to \infty$.

zero and then make the value of $p(\mathrm{E}_{ek}|\lambda^{\mathrm{E}}_e)$ arbitrarily large by sending $\lambda^{\mathrm{E}}_{ek} \to \infty$.

We prevent this collapse of the prior to a δ-peak by keeping track of parameter uncertainty. Admitting a nonzero uncertainty for E_{ek} no longer allows us to set E_{ek} precisely and deterministically to zero. Any slightly nonzero value of E_{ek} would have no support under a δ-peaked prior.

Formal derivation. We now formalize the above intuitive picture and show that the specific variational approximation chosen in Eqs. 12-13 indeed suffices to prevent any divergent solutions.

Assuming again a real embedding space, the log prior of a single model parameter E_{ek} is given by (cf., Eq. 16 of the main text)

$$\log p(\mathrm{E}_e|\lambda^{\mathrm{E}}_e) = \frac{1}{p}\Big[\log \lambda^{\mathrm{E}}_e - \lambda^{\mathrm{E}}_e |\mathrm{E}_{ek}|^p\Big] + \text{const.} \qquad \text{(S1)}$$

As discussed above, setting $\mathrm{E}_{ek} = 0$ and sending $\lambda^{\mathrm{E}}_e \to \infty$ sends the right-hand side of Eq. S1 to infinity. This can even be relaxed: the log prior diverges for $\lambda_e^{\mathrm{E}} \to \infty$ as long as we keep E_{ek} small enough, i.e., as long as $\mathrm{E}_{ek} = O((\lambda_e^{\mathrm{E}})^{-1/p})$. This is why maximizing the log joint distribution over $\boldsymbol{\lambda}$ leads to divergend solutions.

Instead of maximizing the log joint distribution, the variational EM algorithm maximizes the ELBO, Eq. 14 of the main text. Note first that the ELBO itself is bounded from above by zero: the ELBO is a lower bound on the marginal log likelihood $\log p(\mathbb{S}'|\boldsymbol{\lambda})$, where $p(\mathbb{S}'|\boldsymbol{\lambda}) \leq 1$ since it is a discrete probability distribution.

Further, the ELBO has a maximum at finite values for the variational parameters and hyperparameters. Maximizing the ELBO ensures that each model parameter is associated with a nonzero uncertainty $\sigma_{e/r}^{\mathrm{E/R}} > 0$ since the entropy term

$$H[q_{\boldsymbol{\mu},\boldsymbol{\sigma}}] = \sum_{e\in[N_{\mathrm{e}}]} \log \sigma_e^{\mathrm{E}} + \sum_{r\in[N_{\mathrm{r}}]} \log \sigma_r^{\mathrm{R}} + \text{const.} \tag{S2}$$

imposes an infinite penalty if any $\sigma_{e/r}^{\mathrm{E/R}} \to 0$. The entropy term in the ELBO thus has an additional regularizing effect. Combined with the other regularizing term in the ELBO, the expected log prior, we obtain for a given model parameter E_{ek} using Eq. S1, up to an additive constant,

$$\begin{aligned} &\mathbb{E}_{q_{\boldsymbol{\mu},\boldsymbol{\sigma}}}\left[\log p(\mathrm{E}_{ek}|\lambda_e^{\mathrm{E}}) - \log\left(q_{\sigma_{ek}^{\mathrm{E}},\mu_{ek}^{\mathrm{E}}}(\mathrm{E}_{ek})\right)\right] \\ &\quad= \frac{1}{p}\log\lambda_e^{\mathrm{E}} - \frac{\lambda_e^{\mathrm{E}}}{p}\mathbb{E}_{q_{\boldsymbol{\sigma},\boldsymbol{\mu}}}\left[|\mathrm{E}_{ek}|^p\right] + \log\sigma_{ek}^{\mathrm{E}} \qquad \text{(S3)} \\ &\quad= \frac{1}{p}\left[\log\left(\lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p\right) - \lambda_e^{\mathrm{E}}\,\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\left[\left|\mu_{ek}^{\mathrm{E}} + \sigma_{ek}^{\mathrm{E}}\epsilon\right|^p\right]\right] \end{aligned}$$

where, in the last equality, we made the dependency on σ_{ek}^{E} explict by reparameterizing the normal distributed random variable $\mathrm{E}_{ek} = \mu_{ek}^{\mathrm{E}} + \sigma_{ek}^{\mathrm{E}}\epsilon$ in terms of a standard-normal distributed variable ϵ.

Maximizing the right-hand side of Eq. S3 over μ_{ek}^{E} yields $\mu_{ek}^{\mathrm{E}} = 0$ by symmetry, thus simplifying the objective to

$$\begin{aligned} &\frac{1}{p}\left[\log\left(\lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p\right) - \lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p\,\mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\left[|\epsilon|^p\right]\right] \\ &= \frac{1}{p}\left[\log\left(\lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p\right) - \lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p c_p\right] \qquad \text{(S4)} \end{aligned}$$

with some numerical constant $c_p > 0$. The right-hand side of Eq. S4 is structurally similar to the right-hand side of Eq. S1, which had divergent solutions: both are a difference between a logarithmic and a linear term in λ_e^{E}. However, while in Eq. S1, we were able to send the logarithmic term to infinity and still keep the linear term bounded, this is not possible in Eq. S4, which has the same argument (up to a constant c_p) for the logarithmic and the linear term. The right-hand side of Eq. S4 has a maximum at a finite value for $\lambda_e^{\mathrm{E}}\left(\sigma_{ek}^{\mathrm{E}}\right)^p$. Thus, the variational EM algorithm avoids divergent solutions.