Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Non-Parametric Estimation of Multiple Embeddings for Link Prediction on Dynamic Knowledge Graphs

Yi Tay,[1] **Luu Anh Tuan,**[2] **Siu Cheung Hui**[3]
[1, 3] Nanyang Technological University
School of Computer Science and Engineering, Singapore
[2] Institute for Infocomm Research, Singapore

Abstract

Knowledge graphs play a significant role in many intelligent systems such as semantic search and recommendation systems. Recent works in this area of knowledge graph embeddings such as TransE, TransH and TransR have shown extremely competitive and promising results in relational learning. In this paper, we propose a novel extension of the translational embedding model to solve three main problems of the current models. Firstly, translational models are highly sensitive to hyperparameters such as margin and learning rate. Secondly, the translation principle only allows one spot in vector space for each golden triplet. Thus, congestion of entities and relations in vector space may reduce precision. Lastly, the current models are not able to handle dynamic data especially the introduction of new unseen entities/relations or removal of triplets. In this paper, we propose Parallel Universe TransE (puTransE), an adaptable and robust adaptation of the translational model. Our approach non-parametrically estimates the energy score of a triplet from multiple embedding spaces of structurally and semantically aware triplet selection. Our proposed approach is simple, robust and parallelizable. Our experimental results show that our proposed approach outperforms TransE and many other embedding methods for link prediction on knowledge graphs on both public benchmark dataset and a real world dynamic dataset.

Introduction

Knowledge Graphs (KG) are represented in the form (*head,relation,tail*). Recently, representational learning on KGs which involves learning vector representations of entities (head or tail) and relations in a knowledge graph has become extremely popular. A simple and effective model in this line of work is the translational model in which each tail entity vector (denoted as **t**) is represented as a translation relation vector (denoted as **r**) from its head entity vector (denoted as **h**) for each golden triplet. Popular embedding models based on the translation principle include TransE (Bordes et al. 2013), TransH (Wang et al. 2014) and TransR (Lin et al. 2015) which learn embeddings using a margin-based objective function. Given the learned vector representations of entities and relations, one can compute the probability of all triplet permutations existing in the KG. This is also known as the classic knowledge base completion task where we augment existing facts with new ones generated by link prediction. In addition, link prediction is also useful in search, recommendation and other IR tasks.

Despite promising results, translational embedding methods suffer from several major weaknesses. Firstly, embedding methods are known to suffer from sensitivity to hyperparameters. For example, the learning rate and margin can critically impact performance. The difficulty in parameter tuning can inevitably lead to sub-optimal solutions. With only one embedding space, there can be only one global configuration of hyperparameters. Secondly, by the translation principle, the objective of $\|h + r - t\| = 0$ may be too geometrically restrictive since the golden spot is only one point in vector space. TransH and TransR have proposed relation-specific vector/matrix projections to alleviate this problem but come with an extra computational cost and complexity. Empirically, we also found that the performance in these extension models are often poor unless they are initialized with pre-trained TransE embeddings. In addition, the current methods lack adaptability to dynamic data. There is no way to incrementally update the existing models especially for triplet deletion and inclusion of new entities since they require a fixed index during training. Thus, all other models have to be retrained. In conjunction with the problem of hyperparameter sensitivity, the optimal hyperparameters are likely to change given a sizable update which requires retuning again after each update. This is costly and impractical for applications in dynamic domains.

In this paper, we propose puTransE (Parallel Universe TransE), an online and robust adaptation of TransE to solve the above mentioned problems. Our proposed approach explicitly generates multiple embedding spaces via semantically and structurally aware triplet selection scheme and non-parametrically estimates the energy score of a triplet. The intuition for our approach is that, in every parallel universe embedding space, we impose a constraint on triplets in terms of count and diversity such that each embedding space observes the original knowledge graph from a different view. We outline the key advantages of our approach as follows:

- Our approach eliminates the need for hyperparameter tuning of TransE. This is because we no longer depend on one global configuration. We show that our approach is more robust in general and across dynamic updates.

- By implementing a constrained triplet selection scheme in each embedding space, we are also able to alleviate the congestion problem.
- puTransE is easy to train, parallelize and orchestrate over multiple machines. Our approach is also simple and robust.
- puTransE is able to cope with dynamic knowledge graphs. Since we create new embedding spaces once new data arrives, we are able to adapt to new data. puTransE also allows new entities and relations to be added on the fly.
- Our approach is able to perform fast learning on large web-scale knowledge graphs. We will show that we are able to extract meaningful predictions in less than a minute.

Finally, we evaluate the performance of our approach for incremental learning on knowledge graphs by experimenting on a time-aware multi-relational dataset based on real world social networks. We show that puTransE outperforms TransE and many other embedding models on the task of link prediction.

Related Work

Link Prediction on knowledge graphs have attracted intense research focus in recent years. Embedding methods that learn latent features are generally considered to be the state-of-the-art. There are a myriad of models for doing so. For the sake of simplicity, we simply classify them under Translational Models and Others (Non-Translational Models).

Translational-based Models

In this section, we introduce TransE, TransH and TransR. Before we proceed, we formally introduce the notations used in this paper. We denote a triplet as (h, r, t) and likewise their vectors as **h**, **r** and **t** respectively. The scoring energy function is represented by $E_r(h, t)$ and the training objective is that $\|h + r - t\| = 0$ for each golden triplet.

TransE In TransE (Bordes et al. 2013), a relation **r** is represented in vector space as a translation from **h** and **t**. In other words, the vector $(\mathbf{h}+\mathbf{r})$ is close to **t** if a triplet (h, r, t) exists in the knowledge graph. Therefore, the energy score function of TransE is as follows:

$$E_r(h,t) = -\left\|h + r - t\right\|_{L1/L2} \tag{1}$$

TransE is simple and efficient and has performed extremely well given its simplicity. However, its training objective is geometrically restrictive. We refer to this phenomena as *over-congestion in vector space*. This is because for each pair $(h+r)$ or $(r-t)$, only one entity can be accommodated in the golden spot of $\|h + r - t\| = 0$. Without a doubt, this flaw will cause problems especially in obtaining strictly accurate prediction results. Furthermore, TransE cannot handle complex relation types such as $1 - to - N$, $N - to - 1$ and $N - to - N$.

TransH and TransR In attempts to fix the flaws of TransE, TransH (Wang et al. 2014) was proposed. In TransH, the vector representation of each entity is dependent on the relation-specific hyperplane. The energy score function of TransH is defined as:

$$E_r(h,t) = -\left\|h_\perp + r - t_\perp\right\|_{L1/L2} \tag{2}$$

where $h_\perp = w_r^\top \mathbf{h} w_r$ and $t_\perp = w_r^\top \mathbf{t} w_r$. Naturally, constraints such as $\|w_r\| = 1$ and $\left\|w_r^\top d_r\right\| / \|d + r\|_2 \leq \epsilon$ are enforced to ensure that d_r (the translation vector), $h_\perp$ and $t_\perp$ are on the hyperplane.

TransE and TransH both proposed embedding entities and relations in the same vector space. However, TransR (Lin et al. 2015) proposed that entities and relations should exist in different vector spaces. TransR utilizes a relation-specific projection matrix M_r to project entities into a relation specific subspace. The energy scoring function of TransR is defined as:

$$E_r(h,t) = -\left\|M_r h + r - M_r t\right\|_{L1/L2} \tag{3}$$

where M_r is a relation-specific matrix projection. TransR has shown better performance as compared to TransE and TransH. However, TransR does not scale well due to its expensive matrix-vector operations. Furthermore, there is extra memory cost in storing M matrices if $|r|$ is large.

Generally, the current Translational Models lack scalability and adaptability. Even in TransE which has the lowest computational costs, larger scale KGs involving more triplets will simply cause the congestion problem to worsen. Therefore, TransE is *intrinsically* not scalable despite its efficiency. Furthermore, there is no present way to adapt to dynamic data. KGs may change over time due to learning new facts, combining with other KGs from related domains or simply change naturally in a volatile domain. Finally, these models often require extensive hyperparameter tuning. This makes adapting to dynamic KGs a harder problem, i.e., besides retraining the model, an extensive parameter tuning process has to be undertaken once there is substantial update to the knowledge graph.

Other Models

Besides Translational Models, there are many other methods that can be used in similar applications. However, earlier embedding methods like Unstructured (Bordes et al. 2013), Structured Embeddings (Bordes et al. 2011), Semantic Matching Energy (Glorot et al. 2013), Latent Factor Model (Jenatton et al. 2012) have all been surpassed by Translational Models in recent years. The most complex and expressive models include RESCAL (Nickel, Tresp, and Kriegel 2011), a collective Matrix Factorization approach that represents KGs as multi-dimensional arrays (tensors) and the Neural Tensor Network (Socher et al. 2013) which models the second-order correlations using a non-linear neural network. However, these complex models take a long time to train. Thus, it is difficult to adopt them in dynamic domains despite good performance. Due to the lack of space, we refer interested readers to their respective papers.

Parallel Universe TransE

In this section, we introduce puTransE, a novel online and robust embedding approach for link prediction on knowledge graphs. Algorithm 1 and Figure 1 outline the procedure for learning puTransE.

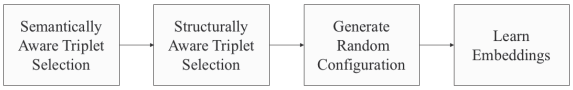


Figure 1: Proposed Approach puTransE

Triplet Selection Scheme

The crux of puTransE is to learn multiple embedding spaces where each embedding space implements a triplet constraint both in count and diversity. For each embedding space $\Delta_i \in \Delta$, we denote χ_i as the set of observed triplets for embedding space Δ_i. Let β_i be the triplet constraint in embedding space Δ_i. Note that β_i is often much smaller than n_t, the total number of triplets. Let E_i and R_i be the set of entities and relations in Δ_i.

Semantically-Aware Triplet Selection In order to enable effective relational learning despite the constraints on triplet count, the selected triplets (and entities that form them) should be *semantically relevant* to each other. To do so, we sample a relation r from $\mathbf{R}$ and generate a set E_r^f of all entities containing r as either an outgoing or incoming edge. (lines 3-4) r is regarded as the *semantic focus* of Δ_i.

Structurally-Aware Triplet Selection Next, we adopt the bidirectional Random Walk (RW) Model using the entities selected as starting nodes. (Lines 7-13) The use of RW is analogous to events unfolding in the canonical interpretation of the parallel universe theory. In puTransE, each parallel embedding space contains a slightly different event sequence and thus, observes the knowledge graph from a different view. This is exactly why we name our model *Parallel Universe TransE*, taking inspiration from the canonical interpretation of Parallel Universe. Alternatively, this can also be seen as diversifying features which is prevalent in ensemble methods such as Random Forests (Breiman 2001). There are several advantages to adopting RW. First, in each embedding space, entities and relations are *structurally relevant* to each other as they are connected by the walk. Second, RW is simple, efficient and easily parallelizable.

Balance between Semantics and Structure Naturally, $E_r^f > \beta_i$ may occur. Therefore, we have to strike a balance between semantics and structure. Here we set a hyperparameter $\theta \in [0, 1]$ to control the balance between semantics and structure. In short, $\theta \times \beta_i$ is allocated to semantics. In practice, we found that a value between 0.25 and 0.5 works well and does not critically affect performance.

Generating Random Configuration Instead of relying on one global configuration, we can simply assign a different hyperparameter configuration to each embedding space. For each Δ_i, we randomly generate values for not only the original hyperparameters of TransE (margin μ and learning rate) but also for θ, β_i and number of training epochs.

Learning puTransE

The training process of puTransE minimizes the following margin based loss function:

$$L = \sum_{\Delta_i \in \Delta} \sum_{\xi \in \Delta_i} \sum_{\xi' \notin \Delta_i} max(0, E_l(\xi) + \gamma - E_l(\xi')) \quad (4)$$

where ξ is a triplet that exists in parallel embedding space Δ_i and ξ' is a triplet that does not exist within Δ_i. The training of embeddings within each embedding space is the same as TransE albeit restricted to the set of triplets $\chi_i \in \Delta_i$, and the set of entities and relations E_i and R_i respectively. $E_l(\xi)$ is the local energy score of the triplet in its embedding space.

Algorithm 1 Learning Parallel Universe TransE (puTransE)

Input: Training Tuples T = $\{(h, r, t)\}$, sets E and R, embeddings dimension k, number of embeddings num
Output: Set of Generated Embedding Spaces ϕ

1: $\phi \leftarrow$ Initialize Empty Set for Embedding Spaces
2: **while** $num > 0$ **do**
3: $S_r \leftarrow$ sample relation from **R**
4: $V_r \leftarrow$ select semantically relevant entities
5: $\mu, lr, \beta, \theta, itr \leftarrow$ generate random hyperparameters
6: $\chi \leftarrow \emptyset$ // Initialize empty set for selected triplets
7: **while** $|\chi| \leq \beta$ **do**
8: **for** $v \in V_r$ **do**
9: $t \leftarrow$ form triplet with randomly selected neighbor $\eta(v)$
10: $\chi \leftarrow \chi \cup t$ // add selected triplet
11: **end for**
12: $V_r \leftarrow$ all entities collected in the last iteration
13: **end while**
14: $E_i, R_i \leftarrow$ all entities and relations in χ respectively
15: $e, r \leftarrow$ Initialize uniform $(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for $e \in E_i, r \in R_i$
16: **while** $itr > 0$ **do**
17: **for** $(h, r, t) \in \chi$ **do**
18: $(h', r,' t') \leftarrow S_{(h,r,t)}$ // Sample corrupted triplet
19: **end for**
20: update params w.r.t $\sum_{\xi \in \chi_i} \sum_{\xi' \in \chi_i} max(0, E_l(\xi) + \gamma - E_l(\xi'))$
21: $itr \leftarrow itr - 1$
22: **end while**
23: $\Delta \leftarrow (E_i, R_i)$ // Trained Parameters are saved as one embedding space
24: $\phi \leftarrow \phi \cup \Delta$ // Add Embedding Space to Set
25: $num \leftarrow num - 1$
26: **end while**

Non-Parametric Energy Estimation

In this section, we introduce the combination scheme for performing link prediction across parallel embedding spaces.

Non-Parametrically Estimated Global Energy For puTransE, we define the final global energy score (across all embedding spaces) of a triplet as:

$$G_r(h, t) = \max_{(h,r,t) \in \Phi} - \|h + r - t\| \quad (5)$$

where Φ is the set of all embedding spaces that contain h, r, t and $\|.\|$ is either the $l1$ or $l2$ norm. Algorithm 2 outlines the procedure for combining energy scores during test time.

Algorithm 2 Non-Parametric Energy Estimation

Input: Δ set of embedding spaces from trained puTransE, test tuples T = (e_i, r_i)
Output: Global Predictions G

$G \leftarrow \{\}$ // Initialize empty Dict of global energy scores
for $(e_i^t, r_i^t) \in T$ **do**
  for $\Delta_i \in \Delta$ **do**
    $E_i, R_i \leftarrow$ get embeddings from Δ_i
    if $e_i^t \in E_i$ and $r_i^t \in R_i$ **then**
      for $e_i \in E_i$ **do**
        $E_l(e_i^t, r_i^t, e_i) \leftarrow \left\| e_i^t + r_i^t - e_i \right\|_{L1/L2}$ //Local Score
        $G[e_i^t, r_i^t, e_i] \leftarrow \max(G[e_i^t, r_i^t, e_i], E_l(e_i^t, r_i^t, e_i))$
      end for
    end if
  end for
end for

In short, for each given test tuple, we loop through all $\Delta_i \in \Delta$. Whenever the entities/relations from the test tuple exists in the index of Δ_i, we calculate all scores within Δ_i that are relevant to the testing pair and add them to the global score. Lastly, we introduce how puTransE is able to handle dynamic knowledge graphs.

Proposed Architecture for Online Link Prediction Figure 2 shows the proposed architecture for Online Link Prediction. As in Figure 2, embedding spaces are created from left to right. The blue spaces denote the initial embedding spaces at a particular time step and the green spaces are generated after a KG update and likewise the red for update 2. Note that we are able to use a randomized configuration for each embedding space. Finally, on-demand, predictions are non-parametrically[1] estimated from embedding spaces.

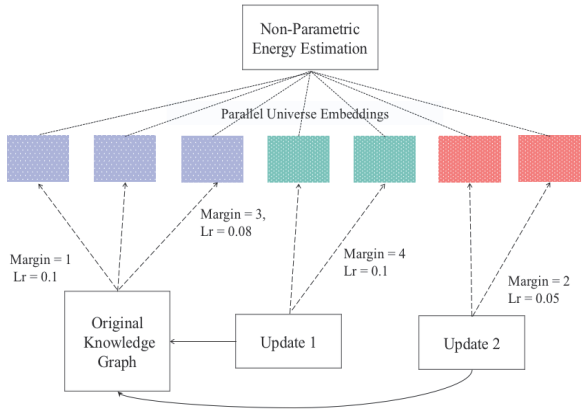


Figure 2: Proposed Architecture for Online Link Prediction

Handling Deletion of Triplets In practice, triplets can be either inserted or deleted during an online update of the KG. Deletion of triplets is one of the main reasons why current margin based embedding methods are fundamentally unable to handle online updates. Since deleted triplets were once facts in the KG and have been reinforced countless times as a positive example, it would be difficult for models to *unlearn* this information. puTransE handles this effectively because it has decoupled parallel embedding spaces. There are two ways to handle this in our approach. For domains with highly dynamic data, we simply maintain a constant flow of embedding spaces. Alternatively, we can simply choose to selectively negate the embedding spaces containing the deleted triplet at prediction time. However, this is an application dependent decision and is out of scope for this paper.

[1]The term *non-parametric* is only used loosely. puTransE, as a whole, is non-parametric in the sense where there are no fixed parameters.

Experiments

We evaluate our proposed puTransE on several experiments. We first evaluate our model on standard Link Prediction with benchmark static datasets. Second, we design a second experiment using a dynamic dataset which is constructed from real world data to test our approach's ability to handle incremental updates. Finally, using a web-scale knowledge graph, we show that our approach is able to extract meaningful predictions within a short period of time. All experiments were conducted on an Quad-core i7-6700 CPU@3.40GHz machine running Linux with 64GB of RAM.

Datasets

We introduce the datasets used in our experiments. Each dataset is aimed to test a certain quality of our approach.

- WN18 is a commonly used benchmark dataset constructed from WordNet (Miller 1995). WordNet is a large lexical knowledge graph where Entities in WordNet are synonyms which express distinct concepts and Relations in WordNet are conceptual-semantic and lexical relations.
- GS26k is a time-aware dataset constructed from real world social network data (Twitter, Foursquare, Instagram). We name our dataset GeoSocial (GS26K) since our dataset focuses on real time check-in data of users albeit in knowledge graph form. We use GS26k test the ability of puTransE to handle dynamic changes in the dataset. Therefore, GS26k comprises an initial set and three different snapshots denoted as Snapshot $\{1, 2, 3\}$. In GS26, entities are users, places, tweets, check-ins, posts and hash-tags. Examples of relations include *(hasVisitedLocation)*, *(lastVisitedPlace)*, *(currentlyVisitedMost)* and *(hasHashTag)*. We construct our train/test split in a time-aware manner according to month.
- YAGO (Hoffart et al. 2013) is a web-scale semantic KG that extracts data from a huge variety of sources. We use YAGO to test our approach's ability to generate predictions quickly on large KGs. Since we perform qualitative analysis on YAGO, we do not have test and validation sets.

Dataset	#Triplets	#Entities	#Relations	#Test	#Validation
WN18	112,581	40,943	18	5000	5000
GS26K	101,188	57023	9	1000	1000
Snapshot 1	165,487	26,000	10	1000	1000
Snapshot 2	319,638	60,640	10	1000	1000
Snapshot 3	787,034	83,559	10	1000	1000
YAGO	5,628,166	2,635,315	37	-	-

Table 1: Dataset Characteristics

Experiment 1 - Link Prediction

In this section, we briefly describe our experimental protocol, setup, datasets used as well as baselines for comparison.

Experimental Setup We compare puTransE with many state-of-the-art methods in the task of link prediction on WN18. We compare with Unstructured, RESCAL, Structured Embeddings (SE), Semantic Matching Energy (SME), Latent Factor Model (LFM), TransE, TransH and TransR. We use the results reported in (Lin et al. 2015) directly since the dataset is the same. For puTransE, we define a reasonable range for each hyperparameter. We use a randomized margin of $\gamma \in [1, 4]$ and the initial learning rate in the ranges of $lr \in [0.01, 0.1]$. We set $\beta \in [500, 2000]$. Each embedding space is trained with AdaGrad (Duchi, Hazan, and Singer 2011) for a fixed number of iterations whereby the number of epoch for each embedding space is also set to a randomized range of $[50, 200]$. For puTransE, we stop generating embedding spaces once we have converged on the validation set (filtered hits@10).

Evaluation Metrics We follow the evaluation protocol of (Bordes et al. 2013) and report on two evaluation metrics.

- Mean Rank is the average position of all testing triplets.
- HITS@N is the number of testing triplets that appear within the top N ranks.

For both metrics, we take two settings, *raw* and *filter*. For the filter setting, we simply remove all triplets from the ranking that exist in the training set.

Experimental Results

Table 2 shows the results of our link prediction experiments.

Dataset	WN18			
Method	Mean Rank		Hits@10	
	Raw	Filter	Raw	Filter
Unstructured	315	304	35.3	38.2
RESCAL	1180	1163	37.2	52.8
SE	1011	985	68.5	80.5
SME (Linear)	545	533	65.1	74.1
SME (Bilinear)	526	509	54.7	61.3
LFM	469	456	71.4	81.6
TransE	263	251	75.4	89.2
TransH	318	303	75.4	86.7
TransR	232	219	78.3	91.7
puTransE	**39**	**29**	**88.1**	**94.9**

Table 2: Experimental Results for Link Prediction on WN18

On WN18, we see that puTransE[2] achieves state-of-the-art performance. In terms of precision, it has surpassed more complex models such as TransH and TransR. It is good to note that the precision of puTransE is much higher than that of TransE alone. The most notable increase in performance is in the metric of Mean Rank (Filter and Raw) where we reduce it to a mere 29. This is because our random walk model prunes the search space. We obtain the above results of our model with ≈ 5000 embedding spaces and noticed a direct correlation between number of embedding spaces and precision. This proves that our divide-and-conquer approach works, i.e., learning from local regions in knowledge graphs and then combining them is an effective method of relational learning. Note that the time taken (on our machine) to entirely train a single embedding space is $\approx 2 - 3s$ on WN18 which is comparable/faster than a single epoch of TransE and TransR.

Evaluation on Absolute Precision and Robustness The robustness of an embedding method is critical especially in dynamic domains since parameter tuning is a cost incurred that should be factored in practical applications. To observe the sensitivity of methods like TransE and TransR to hyperparameters, we conduct further experiments. Using the source code[3] of (Lin et al. 2015), we trained several models TransE and TransR of varying margin γ amongst $\{1, 2, 4\}$, learning rate amongst $\{0.1, 0.01, 0.001\}$ using a dimension of 50. Table 3 shows the best, average and worst result from each method on WN18. Generally, we found a huge drastic gap in performance in TransE/TransR if the hyperparameters are not tuned properly. This brings further merit to our model since our model provides a performance guarantee without requiring hyperparameter tuning. Hence, our model is more **robust**. Finally, we observe that our model increases the HITS@1 rate by almost 6 times as compared to TransE/TransR with optimal hyperparameters. This is due to a de-congesting effect since we have lesser triplets in each embedding space. We also note that our model produces an unnaturally high **raw** HITS@1 result. This is interesting because testing samples are being retrieved at higher priority over training samples/ground truths. This can be attributed by how we split the KG into smaller local subgraphs whereby there are less truths/triplets in each embedding space. This allows a higher chance for testing samples to be chosen over ground truths.

Method	**Mean Rank**		**Hits@10**		**Hits@1**	
	Raw	Filt	Raw	Filt	Raw	Filt
TransE (Worst)	982	967	32.5	34.6	1.3	1.6
TransE (Avg)	788	698	40.5	84.2	3.1	6.2
TransE (Best)	471	404	79.2	94.1	5.3	10.2
TransR (Worst)	20211	20203	0.02	0.02	0.0	0.0
TransR (Avg)	912	850	71.2	89.5	1.2	3.1
TransR (Best)	343	341	80.9	94.1	6.9	7.8
puTransE	**39**	**29**	**88.1**	**94.9**	**39.8**	**60.0**

Table 3: HITS@1 and Robustness on WN18

Experiment 2 - Evaluation on Real World Dynamic Dataset

In this section, we show the effectiveness of online learning of puTransE with an experiment that models a real world application of Place Recommendation using GS26K. Table 5

[2] Empirically, we also found that semantic and structural selection of triplets is mandatory. Random bagging of triplets that are irrelevant to each other does not learn anything useful, i.e, mean rank $\ggg$ 5k.

[3] Unfortunately, we were not able to reproduce the optimal results for TransR from (Lin et al. 2015) even with their code. We obtained a better HITS@10 result at the expense of Mean Rank. Meanwhile, The worst performance of TransR was produced with $\gamma = 4, lr = 0.1$ and 1440 mini-batches and trained with the best TransE model.

Time Step	Method	FMR	F-Hits@10	F-Hits@1	Training Time
GS26k (Base)	TransE	83	46.9	**20.4**	$\approx$ 20 mins +2 hrs
	TransR	223	**49.3**	18.2	$\approx$ 5 hrs + 40 hrs
	puTransE	**27**	43.5	10.2	$\approx$ 30 mins +0 hrs
Snapshot 1	TransE	3867	15.3	4.6	$\approx$ 20 mins
	TransR	12884	23.6	3.7	$\approx$ 5.5 hrs
	puTransE	**61**	**27.5**	**5.5**	$\approx$ 10 mins
Snapshot 2	TransE	9913	15.4	4.6	$\approx$ 35 mins
	TransR	14485	17.8	5.4	$\approx$ 7 hrs
	puTransE	**81**	**23.7**	**17.6**	$\approx$ 10 mins
Snapshot 3	TransE	21	58.3	19.4	$\approx$ 1 hr
	TransR	90	60.6	23.5	$\approx$ 8 hrs
	puTransE	**19**	**67.0**	**56.7**	$\approx$ 15 mins
Total	TransE	3233	35.3	12.0	$\approx$ 2 hrs + 2 hrs
	TransR	6921	37.8	12.7	$\approx$ 25 hrs + 40 hrs
	puTransE	**43**	**40.1**	**20.8**	$\approx$ 1hr

Table 4: Performance Results for Dynamic Link Prediction on GS26K

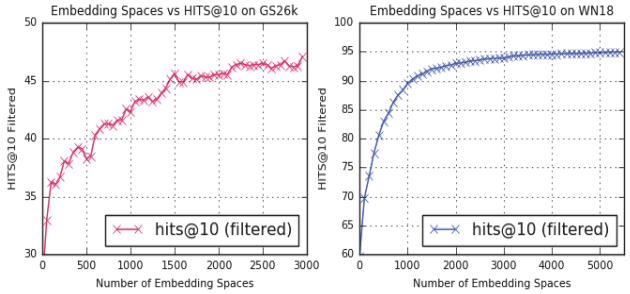
shows the details on the incremental update of each snapshot in GS26k.

	Date	#triplets	#Updates	#Add	#Del	Predict
Base	Jan-Feb	101,188	-	-	-	Mar-15
S1	Mar	165,487	64,299	64,299	0	Apr-15
S2	Apr	319,638	158,087	155,987	2100	May-15
S3	May-Jul	787,034	473,432	469,496	3936	Aug-Oct

Table 5: Characteristics of GS26K and Online Updates

Experimental Setup We evaluate our model on the task of predicting check-in of users in social media. The testing set comprises of triplets of relation *(hasVisitedLocation)* and aims to predict the check-ins of a user for the subsequent month. For this experiment, we only consider the filtered hits as it can be interpreted as new place recommendation problem. We compare our approach with TransE and TransR in terms of precision and runtime using the same metrics as Experiment 1. Since all the other models cannot handle incremental updates, we are only able to compare if we retrain each model at each update of the dataset. We tune the hyperparameters of TransE and TransR using grid search where we set the learning rate amongst $\{0.1, 0.01, 0.001\}$, and margin γ amongst $\{1, 2, 4\}$. For both TransE and TransR, we use a dimensionality of 50 and set the max epoch to 500. TransR is initialized with the embeddings from TransE. We tune the hyperparameters at the Base Set and use the same hyperparameters for training the subsequent snapshots. For puTransE, we use the same setting as in Experiment 1. We allow our model to learn 500 embedding spaces for each month to model a real world application. We report the Filtered Mean Rank, HITS@10, HITS@1 and time taken for each update step of GS26K.

Experimental Results Table 4 shows the results of our dynamic link prediction experiments. The last column in Table 4 shows the time taken to train the model and $+n$ refers to the time taken to tune the hyperparameters. First, we see that TransE and TransR are unstable, i.e., we see that the Mean Rank results for TransE and TransR fluctuate drastically over the different snapshots of the dataset. This shows that we are **not** able to simply assume the same optimal hyperparameters apply given an update to the KG. On the other hand, our approach maintains the same consistency throughout snapshot updates and is therefore more **robust**. Next, we consider the size of each KG. TransE and TransR outperforms puTransE on smaller datasets (base set) but performs worst in comparison as the size of the dataset increases. This shows that TransE is *intrinsically* not scalable due to the congestion problem. On the other hand, our approach performs consistently well across all incremental updates especially in the metric of mean rank. Finally, note that the time taken to incrementally train puTransE is often very small ($\approx 10 - 20$ mins) and the time to entirely train an embedding space is about $2 - 5s$.



(a) On GS26k (b) On WN18

Figure 3: Effect of Embedding Spaces on Performance

Effect of Embedding Spaces Figure 3 shows the influence of creating new embedding spaces on WN18 and GS26K. We see that increasing the number of embedding spaces increases the precision. We show that we are able to collectively combine energy scores across embedding

spaces. This opens up possibilities even for aggregating energy scores across a variety of knowledge graphs.

Experiment 3 - Fast Learning on Web-Scale Knowledge Graphs (Qualitative Analysis)

In the last experiment, we evaluate puTransE qualitatively on its ability to perform fast learning on large knowledge graphs (YAGO). At $30s$, we sample 10000 triples and take the top 10 best global scores. Table 6 shows the results of our example predictions. Note that these are facts that do not exist in the original dataset. Therefore, we conclude that our approach is able to learn quickly even from extremely large datasets. This is an important feature because KGs are often very large.

Relation	Example Predictions in Global Top-10
livesIn	(Charles K.Kao,Germany), (Barack Obama, United States)
worksAt	(Adolf von Baeyer, Humboldt University),(Edward Witten, Princeton University), (Robert Bunsen, University of Gottingen)

Table 6: Qualitative Analysis Results

Discussion We noticed that within $30s$, our approach can learn patterns such as $graduatedFrom(x, y) \rightarrow worksAt(x, y)$ which seems to occur at a high chance in our dataset. Moreover, our approach is able to learn generalizations. For example, (Barack Obama, *livesIn*, Chicago) is the only triplet that exists originally containing information about where Barack Obama lives. Then, we are able to learn that living in Chicago implies that Barack Obama lives in the United States as well. We find this result remarkable as it shows that our approach is able to learn quickly even from large knowledge graphs.

Conclusion

In this paper, we propose a robust and online adaptation of the translational embedding model. puTransE is simple, easily parallelizable and handles dynamic updates to knowledge graphs effectively. The key discovery is that divide-and-conquer approaches are viable in relational learning. Our approach has not only outperformed many methods on the task of link prediction but is also suitable for dynamic domains.

References

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2787–2795.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Glorot, X.; Bordes, A.; Weston, J.; and Bengio, Y. 2013. A semantic matching energy function for learning with multi-relational data. *CoRR* abs/1301.3485.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia: Extended abstract. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 3161–3165.

Jenatton, R.; Roux, N. L.; Bordes, A.; and Obozinski, G. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 3176–3184.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2181–2187.

Miller, G. A. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

Nickel, M.; Tresp, V.; and Kriegel, H. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, 809–816.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. Y. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 926–934.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 1112–1119.