EmbedKB: A Framework for Implementing and Evaluating Embedding Models for Knowledge Bases

Zafarali Ahmed
School of Computer Science
McGill University

Charles C. Onu
School of Computer Science
McGill University

Abstract

Knowledge bases capture information about entities in the world and relationships between them. The development of natural language systems that can generalize to unseen entities and relationships is crucial, considering that no single database can embody all knowledge about the world. Several embedding models which encode entities and relationships as dense vectors have been proposed, alongside tasks for evaluating the goodness of the resulting representations. In this work, we implement and open source *EmbedKB* - a framework for developing and evaluating embedding models. We use this framework to implement five distinct embedding models from the literature. We further evaluate them on two out of three tasks that are essential for KBs to pass. The relative performances of the models were consistent with the literature especially that models with a bilinear transformation tend to perform better on tasks.

1 Introduction

A knowledge base (KB) is a graph where the *nodes* correspond to entities and the *edges* correspond to relationships. A relationship between entities can be represented as a triple *(head, relationship, tail)* or *(h, r, t)*. For example (Avatar, Directed By, James Cameroon). A KB typically contains several thousands of such triples.

In order to augment natural language processing (NLP) systems with world knowledge, knowledge base embedding (KBE) models (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Jenatton et al., 2012) have been proposed. These models encode entities and relationships as continuous vectors that can be leveraged in various NLP tasks. For example, (Yang and Mitchell, 2017) used external KBs for improved machine reading and (Ahn et al., 2016) augmented a neural language model to generate factual sentences.

KBE models primarily differ in their score function f, which is defined such that the score $f(h, r, t)$ of a plausible triple (h, r, t) is smaller than that $f(h', r', t')$ of an implausible triple (h', r', t'). Each model makes different assumptions about the form of the score function and consequently defines different arithmetic operations over the embeddings of entities and relationships. The work of (Yang et al., 2014) showed that score functions of some KBE models can be unified under a general framework.

In this work, we implement the framework and re-derive five existing KBE models. We benchmark the models on KB evaluation tasks of triple classification and entity prediction on the publicly available FB15K-237[1] (Toutanova et al., 2015). All implementation is in the Python programming language and *Tensorflow* (Abadi et al., 2015), a numerical computation library. The relative performances of the models were consistent with the literature. We open source *EmbedKB* so that going forward, it will facilitate rapid development of new models and tasks.

The rest of this paper is organized as follows: In section 2, we summarise the literature on embedding models. In section 3, we present our approach in de-

[1] https://www.microsoft.com/en-us/download/details.aspx?id=52312

tail. In section 3.3 we discuss some key characteristics that KBEs must satisfy. In section 4, we present benchmark results on the models we implemented. Finally, in section 5, we provide an interpretation of our results and possible paths for future work.

2 Related Work

Many KBE frameworks have been proposed in the literature (Bordes et al., 2011; Bordes et al., 2013; Socher et al., 2013; Jenatton et al., 2012; Dettmers et al., 2017). These models make different assumptions and evaluate their respective models on different tasks and datasets. (Yang et al., 2014) introduced a *general framework* which demonstrated that certain models can be derived as a combination of linear and bilinear products (as discussed in Section 3.2).

In essence, our paper is more of a review than an introduction of a new model. Recently, (Nguyen, 2017) did a literature review of the available KBE models and the two tasks we consider in this paper. However, this paper only reports metrics from the original citations. Here, in addition to providing the framework from (Yang et al., 2014) we also implement and report the metrics on the FB15K-237 dataset (Toutanova et al., 2015).

3 Method

3.1 Preliminaries

The main component of a KBE is the score function $f(h, r, t)$ which quantifies the *implausibility* of a triple. The lower the score the more likely the triple is a true fact. The model must learn vector embeddings for the head, v_h, and tail, v_t and encode the relationship r via a vector, matrix or tensor v_r. Many assumptions can be made about the interaction between entities and relationships which characterise different forms of f. For example, one could assume that the relationships are encoded in vectors, v_r that translate in embedding space such that $v_h + v_r \approx v_t$. From this we can derive a score function $f(h, r, t) = ||v_h + v_r - v_t||$. This is the assumption of the TransE model (Bordes et al., 2013). The scoring functions of the other models are shown in Table 1.

During training, we use the max-margin based loss as the objective:

$$L = \sum_{(h,r,t)} \sum_{(h',r',t')} \max(\gamma + f(h, r, t) - f(h', r', t'), 0)$$

where γ is a margin. (h', r', t') represent false triples generated by "corrupting" true triples (h, r, t) in the data base. This corruption can be done by substituting h or t of a true triple with a randomly selected entity.

3.2 The General Framework for Knowledge Base Embedding Models

(Yang et al., 2014) introduced the general framework for KBE models. For completeness, we briefly present this framework here.

The score functions of some KBE models can be expressed as a combination of linear g_r^a and bilinear g_r^b components, defined as:

$$g_r^a(v_h, v_t) = A_{r,1} v_h + A_{r,2} v_t$$

and

$$g_r^b(v_h, v_t) = v_h^T B_r v_t$$

Where A_r and B_r are relationship-dependent parameters. In table 1 we show how score functions of some models break down into linear and bilinear components.

3.3 Model Evaluation

KBs can provide reasoning capabilities to current NLP systems by incorporating outside knowledge. We argue that useful KBEs should have a few key characteristics:

1. It must be able to evaluate the truthfulness of a statement. For example, "London is the capital of the UK" should be classified as true, compared to "London is the capital of the US".
2. We should be able to query, rank and obtain information from it. For example, if we ask for "Cities in Canada", we should get a list of cities that satisfy (?, CityOf, Canada).
3. It should allow complex multi-hop reasoning. For example, given that a model knows "San Francisco is in California" and "California is a state in USA", a KB should be able to reason that "San Francisco is in the country USA".

Table 1: Re-derivation of score functions under the general framework.

Model	Original Score Function	Re-derived Score function
Structured Embedding	$\|\|W_{r,1}v_h - W_{r,2}v_t\|\|_1$	$-\|\|g^a_r(v_h, v_t)\|\|_1$
Single Layer	$v_r^T \tanh\left(W_{r,1}v_h + W_{r,2}v_t + b_r\right)$	$u_r^T \tanh\left(g^a_r(v_h, v_t)\right)$
TransE	$\|\|v_h + v_r - v_t\|\|_2$	$2g^a_r(v_h, v_t) - 2g^b_r(v_h, v_t) + \|\|v_r\|\|_2^2$, $A_{r,1} = v_r$, $A_{r,2} = v_r$
Bilinear	$v_h M_r v_t$	$g^b_r(v_h, v_t)$, $B_r \in \mathbb{R}^{n \times n}$
Neural Tensor Network	$v_r^T \tanh\left(v_h^T M_r v_t + W_{r,1}v_h + W_{r,2}v_t + b_r\right)$	$u_r^T \tanh\left(g^a_r(v_h, v_t) + g^b_r(v_h, v_t)\right)$, $B_r \in \mathbb{R}^{n \times n \times m}$

The first two characteristics are widely benchmarked in the literature as the triple classification task (Socher et al., 2013) and entity/link prediction task respectively (Bordes et al., 2013). The third characteristic is more difficult to evaluate. In (Guu et al., 2015), the authors use composable operators (for example, $v_{r_1 \to r_2} = v_{r_1} + v_{r_2}$ in TransE) in the training objective. This allows one to query the KB by passing in the head entity and subsequent relations. In our example: "Which Country is San Francisco in?" would become a query for the form (San Francisco, State, Country). In (Das et al., 2017), the authors cast the problem under the reinforcement learning framework (Sutton and Barto, 1998) and learned the rules to traverse the vector space in a more effective way.

In this work, we do not consider the last task. We implement and evaluate the triple classification and entity prediction tasks as a first step towards the goal of having representative KBEs.

3.3.1 Triple Classification

The triple classification (Socher et al., 2013) task aims to predict whether an entire triple (h, r, t) is correct or not. A triple is classified as true if the implausibility score is less than a relation-specific threshold, i.e., $f(h, r, t) < \theta_r$, and false otherwise. To determine the thresholds, we first create false triples by corrupting either the head or the tail of true triples with entities that are plausible for that relation[2]. We select the threshold for a given relation by evaluating several values in 0.01 increments from the minimum to maximum score. The score at which the average accuracy on the validation set peaks is selected as optimal threshold and used to report performance on the test set.

3.3.2 Entity Prediction

In the entity prediction task (Bordes et al., 2013), the goal is to predict the head or tail entity given the relationship type and the other entity. For example predicting the tail in (Avatar, DirectedBy, ?). We first take our target triple (h, r, t) and create a set of triples: $T = \{(h, r, e) : e \in E\}$ where E is the set of all entities in the KB. We then ask our model to rank T by evaluating $f(h, r, e)\ \forall\ (h, r, e) \in T$. We report *Hits@10* which is average proportion of times the correct triple was listed in the top 10. We also implement *mean rank* which is the mean rank of the target triple but do not report it in this paper. Many authors argue that this version of the task is unfair as there might be some entities in T that are valid by chance. Hence, the valid triples that are in the training KB are removed from T in the "Filtered" version of the task. The unfiltered version is known as "Raw".

3.4 EmbedKB

We implemented standard benchmarking tools for the tasks described in Section 3.3.1 and 3.3.2. In addition to this we implemented the general framework (Yang et al., 2014). This allows rapid prototyping of models that can be expressed as a combination of linear and bilinear products in a few lines of code. We stress that models need not be derived under the general framework to use EmbedKB. In fact, approaches related to convolution neural networks (Dettmers et al., 2017) and graph neural networks (Hamaguchi et al., 2017) can also

[2]From our literature review, it was unclear if and how the negative sampling scheme for calculating the thresholds differed from the training time scheme.

be implemented and benchmarked with the same code. The code is modularized so that new tasks can be created easily. We make this code available at https://github.com/zafarali/embedKB. Key modules include:

1. **Data tools**: handles importing and conversion of knowledge bases into relevant format for training.
2. **General framework**: provides a base class (to be inherited by new models) with pre-implemented linear and bilinear operators (Yang et al., 2014).
3. **Benchmark module**: provides pre-implemented evaluation tasks for models.

In addition to the above, researchers are able to leverage the tensorboard visualization toolkit to visualize embeddings and the evolution of the score values across epochs.

4 Results

4.1 Experimental Setup

A total of 5 models were derived and implemented with *EmbedKB*. The FB15K-237 knowledge base was used as data. It contains a total of 272,115 training, 17,535 validation and 20,466 test triples. There are 14,505 unique entities and 237 unique relations.

To allow for uniform comparison, the models were trained using the same hyper parameter settings. The values of these hyper parameters are summarized in table 2. Entity dimension represents the size of the embedding vector for head and tail entities v_h and v_t. Relation dimension is the size of the embedding vector for relations, v_r. Regularisation weight is the amount of regularisation applied to the parameters for the linear and bilinear transformations A_r and B_r. The next hyperparameter is the initial learning rate for the optimisation algorithm, AdaGrad (Duchi et al., 2011). We also fixed the number of epochs and the batch size to use for each iteration of optimization.

The models were evaluated on both the triple classification and entity prediction tasks.

4.2 Training

Figure 1 shows the loss as a function of number of epochs during the training of each model. The non-linear methods, NeuralTensorNetwork and SingleLayer, learn quickly compared to the linear methods which take more time to converge.

Table 2: Fixed hyper-parameter settings used to train knowledge base embedding models.

Hyper parameter	Value
Entity dimension	50
Relation dimension	50
Regularisation weight	0.001
Initial learning rate	0.01
Number of epochs	100
Batch size	32

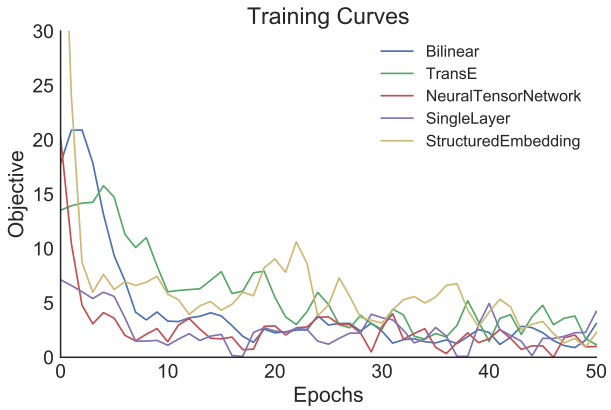

Figure 1: Validation learning curves for the first 50 epochs of training. We observe that the non-linear methods, SingleLayer and NeuralTensorNetwork, learn rapidly compared to the linear methods. (Shown is the smoothed validation curve of one replicate.)

4.3 Triple Classification (TC)

In Table 3, we show the accuracy on test sets for each model. Linear models like the Structured Embedding and Bilinear models performed best. Interestingly, their non-linear counterparts, Single Layer and Neural Tensor Network did not perform as well. We expect that this trend reflects the number of parameters that were learned for each model.

4.4 Entity Prediction (EP)

The trained models were also evaluated on the entity prediction task. In Table 4, we report the Hits@10 on the test set for each model. For simplicity, we do not show the mean rank even though it is output by our framework. In this task, the best model was the

Table 3: Model performance of the triple classification task on the test set. The model is fed a triple (h, r, t) and must classify it based on truthfulness according to a learned relation threshold: $f(h, r, t) < \theta_r$.

Model	**Accuracy**
Structured Embedding	70.55
Single Layer	67.97
TransE	68.77
Bilinear	72.08
Neural Tensor Network	69.94

Neural Tensor Network which performed well on both the raw and filtered version of the task. Surprisingly, despite performing well on TC, the Structured Embedding model performed worse on EP. The non-linear models performed well on EP unlike TC.

Table 4: Model performance of the entity prediction task on the test set. The model is fed a true triple (h, r, t) and is asked to rank a set of triples $\{(h, r, e)\ \forall\ e \in E\}$ using the score function f. Hits@10 is the proportion of times the true triple was in the top 10 of this list.

	Hits@10	
Model	**Raw**	**Filtered**
Structured Embedding	69.73	73.39
Single Layer	72.91	76.69
TransE	72.99	76.58
Bilinear	75.02	78.85
Neural Tensor Network	76.34	80.33

5 Discussion and Conclusion

The difference in performance of the linear and non-linear models on TC and EP suggests that the embeddings being learned have a characteristic that make them suitable for ranking and scoring rather than classification. This could be because EP is more closely related to training time specification since TC requires learning of some threshold parameters and non-linear models would exploit that. It is unclear if this is desired behavior. It could also suggest that multi-task training could make the learned embeddings more general.

The Bilinear model showed consistent performance between the two tasks. This seems to indicate that a simple model which can model relationships as a tensor is sufficient to learn a good generalization strategy. We leave this to be verified in future work.

We note that most of our results are consistent with those reported in the literature, and can be confirmed with further replicates. However, despite being a significant improvement to current benchmarking tools available in the community, EmbedKB could still be improved. In particular, it was unclear from our literature review what the correct corruption mechanism for training and evaluating the triple classification task were.

Another improvement would be to do a exhaustive hyperparameter search to ensure that the models can be compared at their best settings. Given that training each model for 100 epochs took on average 26 hours on a NVIDIA TITAN X GPU, this comparison was not possible in this work. We must evaluate the performance of each model on more datasets. This would ensure we can evaluate the generalization capability of the models. We can formulate this under the *meta-learning* objective of generalizing across tasks and datasets to derive a new metric.

In conclusion, we have released a framework that allows researchers to prototype and benchmark new knowledge base embeddings. We derived five popular models in the general framework of (Yang et al., 2014) and benchmarked on the above tasks. We also argue for the importance in such KBs in generalizing across *tasks* and *datasets* rather than just from training to testing data. This will ensure that the models will remain flexible enough for a wide variety of downstream tasks in natural language understanding and generation.

Statement of Contributions

ZA implemented the code scaffolding and downstream tasks in Tensorflow. ZA implemented the TransE and NTN models. CCO implemented the Bilinear, Single Layer and Structured Embedding models. CCO implemented scripts for training and benchmarking the models. Both authors contributed equally to experimental design and the manuscript.

Acknowledgements

We would like to thank Dr. Guillaume Rabusseau for helping us with bilinear products and Dr. Jackie Chi Kit Cheung for advice on how to design experiments and for guiding the discussions.

References

[Abadi et al.2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Ahn et al.2016] Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

[Bordes et al.2011] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

[Das et al.2017] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

[Dettmers et al.2017] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[Guu et al.2015] Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

[Hamaguchi et al.2017] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*.

[Jenatton et al.2012] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

[Nguyen2017] Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*.

[Socher et al.2013] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

[Sutton and Barto1998] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[Toutanova et al.2015] Kristina Toutanova, Danqi Chen, and Patrick Pantel. 2015. Representing text for joint embedding of text and knowledge bases.

[Yang and Mitchell2017] Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1436–1446.

[Yang et al.2014] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Learning multi-relational semantics using neural-embedding models. *arXiv preprint arXiv:1411.4072*.