Journal of Machine Learning Research 22 (2021) 1-6 Submitted 5/19; Revised 12/20; Published 2/21

Pykg2vec: A Python Library for Knowledge Graph Embedding

Shih-Yuan Yu* SHIHYUAY@UCI.EDU
Sujit Rokka Chhetri* SCHHETRI@UCI.EDU
Department of Electrical Engineering and Computer Science, University of California-Irvine

Arquimedes Canedo ARQUIMEDES.CANEDO@SIEMENS.COM
Siemens Corporate Technology, Princeton

Palash Goyal PALASHGO@USC.EDU
Department of Computer Science, University of Southern California

Mohammad Abdullah Al Faruque ALFARUQU@UCI.EDU
Department of Electrical Engineering and Computer Science, University of California-Irvine

Editor: Andreas Mueller

Abstract

Pykg2vec is a Python library for learning the representations of the entities and relations in knowledge graphs. *Pykg2vec*'s flexible and modular software architecture currently implements 25 state-of-the-art knowledge graph embedding algorithms, and is designed to easily incorporate new algorithms. The goal of *pykg2vec* is to provide a practical and educational platform to accelerate research in knowledge graph representation learning. *Pykg2vec* is built on top of PyTorch and Python's multiprocessing framework and provides modules for batch generation, Bayesian hyperparameter optimization, evaluation of KGE tasks, embedding, and result visualization. *Pykg2vec* is released under the MIT License and is also available in the Python Package Index (PyPI). The source code of *pykg2vec* is available at `https://github.com/Sujit-O/pykg2vec`†.

Keywords: Knowledge Graph Embedding, Representation Learning

1. Introduction

In recent years, Knowledge Graph Embedding (KGE) has become an active research area and many authors have provided reference software implementations. However, most of these are standalone implementations and therefore it is difficult and time-consuming to: (i) find the source code; (ii) adapt the source code to new datasets; (iii) correctly parameterize the models; and (iv) compare against other methods. Recently, libraries such as PyKEEN (Ali et al., 2018), OpenKE (Han et al., 2018) and AmpliGraph (Costabello et al., 2019) provide unifying frameworks for a set of KGE methods, allowing researchers to test KGE methods on multiple benchmarks and their datasets. However, these libraries impose preset hyperparameters that may only work for specific benchmarks, algorithms, or even pipeline implementations. For new datasets, where the corresponding *golden* hyperparam-

*Shih-Yuan Yu and Sujit Rokka Chhetri contributed equally to this article.
†The master branch is the PyTorch version and the *tf2-master* branch is the legacy TensorFlow 2.0 version.

eters may not have been found, it still requires manual trial-and-error runs and inspections to adapt these KGE methods to new applications.

To overcome the limitations identified above, we propose *pykg2vec*, a Python library with 25 state-of-the-art KGE methods (Nickel et al., 2011; Bordes et al., 2014, 2013; Socher et al., 2013; Fan et al., 2014; Wang et al., 2014; Yang et al., 2014; **?**; Lin et al., 2015; Ji et al., 2015; Nickel et al., 2016; Xiao et al., 2016; Trouillon et al., 2016; Dettmers et al., 2018; Shi and Weninger, 2017; Sun et al., 2019). Table 1 compares the features of *pykg2vec* against similar frameworks. The goals of *pykg2vec* are as follows. (a) Provide access to the latest and state-of-the-art KGE implementations. Compared to other libraries, we provide the most KGE methods. (b) Automate the discovery of golden hyperparameters. *pykg2vec* is the only KGE library providing built-in automation for golden hyperameter discovery using Bayesian optimization. (c) Deliver a modular and flexible software architecture and KGE pipeline that is both educational and of practical use for researchers. We provide a set of utilities to inspect the training and resulting embeddings, and to export the results for inspection using other tools. *pykg2vec* is released under the MIT License and also available in Python Package Index (PyPI)[†].

	Pykg2vec(v0.0.51)	OpenKE(latest)	AmpliGraph(v1.0.3)	PyKEEN(v0.0.25)
# of Available methods	25	9	4	10
# of Benchmark Datasets	8	6	5	✗
Built-in Hyperparameter discovery	✓	✗	✗	✗

Table 1: Feature comparison between *pykg2vec*, OpenKE, AmpliGraph and PyKEEN

2. Knowledge Graph Embedding Methods

A knowledge graph contains a set of entities $\mathbb{E}$ and relations $\mathbb{R}$ between entities. The set of facts $\mathbb{D}^+$ in the knowledge graph are represented in the form of triples (h, r, t), where $h, t \in \mathbb{E}$ are referred to as the *head* (or *subject*) and the *tail* (or *object*) entities, and $r \in \mathbb{R}$ is referred to as the *relationship* (or *predicate*). The problem of KGE is in finding a function that learns the embeddings of triples using low-dimensional vectors while preserving structural information, $f : \mathbb{D}^+ \rightarrow \mathbb{R}^d$. One general principle is to enforce the learning of entities and relationships to be compatible with the information in $\mathbb{D}^+$. The representation choices include, for example, deterministic point (Bordes et al., 2013) or complex number (Trouillon et al., 2016). Under the Open World Assumption (OWA), a set of unseen *negative* triplets, $\mathbb{D}^-$, are sampled from *positive* triples $\mathbb{D}^+$ by either corrupting the head or tail entities. Then, a scoring function, $f_r(h, t)$ is defined to reward the positive triples and penalize the negative triples. To aggregate the scores, various loss functions can be utilized such as a pair-wise margin based (Bordes et al., 2013), point-wise logistic (Trouillon et al., 2016), or binary cross-entropy multiclass loss (Dettmers et al., 2018). Finally, an optimization algorithm is used to minimize or maximize the loss. KGE methods are often evaluated in terms of their capability of predicting the missing entities in negative triples $(?, r, t)$ or $(h, r, ?)$, or predicting whether an unseen fact is true or not. The evaluation metrics include the rank of the answer in the predicted list (mean rank), the ratio of answers ranked top-k in the list (hit-k ratio), and the mean of rank's reciprocal (mean reciprocal rank).

[†]The authors appreciate the contribution from Xi Bai (Senior Software Engineer, Design & Engineering, BBC).

3. Software Architecture

Pykg2vec is built with Python and PyTorch that allows the computations to be assigned on GPUs (legacy TensorFlow version is also ready in a separate branch). Figure 1 shows the software architecture of *pykg2vec* and each building block will be described as follows.

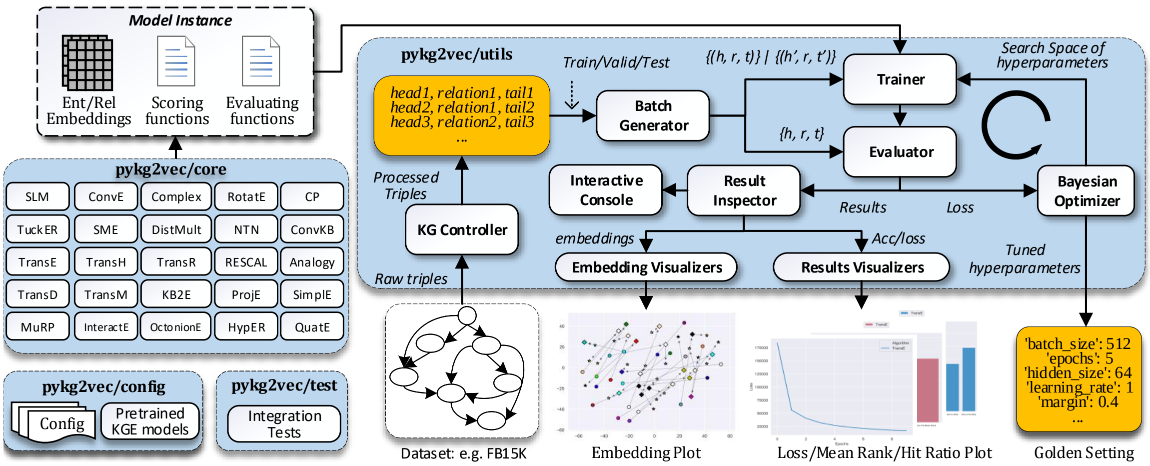

Figure 1: *pykg2vec* software architecture

The **KG Controller** module handles low-level parsing tasks such as finding the total unique set of entities and relations; creating ordinal encoding maps; performing train-test split; and caching the data on disk to optimize tasks that involve repetitive model testing. **Batch Generator** consists of multiple concurrent processes that adapt mini-batches of data to various KGE methods and perform data-processing for sampling negative samples. The batch generator runs independently to bring speedup for feeding the data to the training module running on the GPU. **Core Models** consists of KGE algorithms implemented as Python modules. Each module consists of a modular description of the inputs, outputs, loss function, embedding operations, and hyperparameter configuration. **Configuration** provides the necessary configuration to parse the datasets and also consists of the baseline hyperparameters for each KGE algorithm as presented in its original research paper. In addition, it provides the default search space for discovering golden hyperparameters. Ultimately, *pykg2vec* also provides access to pre-trained KGE models for users' convenience.

The **Trainer** module is responsible for taking an instance of the KGE model, the respective hyperparameter configuration, and input from the **Batch Generator** to train the algorithms. The **Evaluator** module performs link prediction and provides the respective metrics such as mean ranks or filtered mean ranks. Additionally, *pykg2vec* integrates the **Bayesian Optimizer** module that allows users to specify various kinds of search space for Bayesian hyperparameter optimization (Bergstra et al., 2011). This module uses the information from the past trials of evaluating KGE metrics on the validation set to update the next set of hyperparameters to explore, thus being more efficient than brute-force based grid-search approaches in finding a golden hyperparameter set. The **Result Inspector** module plots training loss and commonly used metrics in KGE tasks. To facilitate model analysis, *pykg2vec* provides utilities to visualize the embeddings of entities and relations using t-SNE based dimensionality reduction. Besides, *pykg2vec* exports the learned embeddings in standardized formats so users can also choose other tools such as *Embedding Projector* (Smilkov et al., 2016) in their analysis.

4. Usage Examples

The usage examples for *pykg2vec* are still evolving for better user experience, herein we only demonstrate two examples[‡]. Firstly, to train a KGE method, users can trigger the following commands. For each command, users can switch between various algorithms or adapt the settings to train on other benchmarks or even their own datasets.

```
$ pykg2vec-train(.exe) -h #print the manual for input arguments
$ pykg2vec-train(.exe) -mn [transe|transh|..] #train on TransE or others
$ pykg2vec-train(.exe) -mn transe -ds [wn18|wn18_rr]#use wn18 or wn18_rr
$ pykg2vec-train(.exe) -mn transe -exp True #apply paper's settings
```

Secondly, users can run the following command for discovering golden hyperparameters.

```
$ pykg2vec-tune(.exe) -mn transe -ds wn18_rr # Tune hyperparameters
Found Golden Setting: # after at max 100 trials.
{'L1_flag': True, 'batch_size': 2279, 'hidden_size': 80,
 'learning_rate': 0.05314, 'margin': 8.58, 'opt': 'adam'}
```

The results of running the mentioned scripts are shown in Table 2 and Table 3. Table 2 demonstrates the performance of KGE methods, while Table 3 shows the effect of utilizing the setting found by **Bayesian Optimizer** and the comparison with other KGE libraries.

(filtered)	TransE	TransH	ComplEx	DistMult	KG2E_KL	TransD
Mean Rank	69.52	77.60	111.75	123.76	64.76	57.73
Mean Reciprocal Rank	0.36	0.32	0.45	0.34	0.31	0.33
Hit-10 Ratio	0.61	0.62	0.73	0.57	0.61	0.60

Table 2: The results of KGE methods on FB15k using the settings in the original papers.

(filtered)	*pykg2vec* (arbitrary/found setting)	OpenKE	Ampligraph
Mean Rank	6467/2079	-	2692
Mean Reciprocal Rank	0.13/0.19	-	0.22
Hit-10 Ratio	0.37/0.46	0.512	0.54

Table 3: The effect of applying the found hyperparameter setting on TransE for WN18_RR.

5. Discussion & Conclusion

Pykg2vec is a Python library with extensive documentation that includes the implementations of a variety of state-of-the-art Knowledge Graph Embedding methods and modular building blocks of the embedding pipeline. In response to the growing machine learning *reproducibility crisis*, *pykg2vec* aims to help researchers and developers to quickly test algorithms against their custom knowledge based or utilize the modular blocks to adapt this library for their custom algorithms.

[‡]More programming examples and performance metrics are in `https://pykg2vec.readthedocs.io/`.

References

Mehdi Ali, Charles Tapley Hoyt, Daniel Domingo-Fernandez, Jens Lehmann, and Hajira Jabeen. Biokeen: A library for learning and evaluating biological knowledge graph embeddings. *bioRxiv*, page 475202, 2018.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24:2546–2554, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 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.

Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, and Nicholas McCarthy. Ampligraph: a library for representation learning on knowledge graphs, mar 2019.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Miao Fan, Qiang Zhou, Emily Chang, and Fang Zheng. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia conference on language, information and computing*, pages 328–337, 2014.

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 139–144, 2018.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. 2015.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. 2015.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. 2011.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. 2016.

Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. 2013.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. pages 2071–2080, 2016.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. 2014.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. Transg: A generative model for knowledge graph embedding. pages 2316–2325, 2016.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. 2014.