BenchEmbedd: A FAIR Benchmarking tool for knowledge graph Embeddings

Afshin Sadeghi[1,2] * **, Xhulia Shahini[1], Martin Schmitz[1], and Jens Lehmann[1,2]

[1] Department of Computer Science, University of Bonn, Germany
[2] Fraunhofer IAIS, Germany
sadeghi@cs.uni-bonn.com, shahinixhulja@gmail.com,
schmitz.kessenich@gmail.com, jens.lehmann@cs.uni-bonn.de

Abstract. Knowledge graph embedding models have been studied comprehensively recently. However, these studies lack an evaluation system that compares their efficiency in a reproducible manner that follows the FAIR principles. In this study, we extend the general HOBBIT benchmarking platform to evaluate the efficiency of embedding models with such criteria. The demo benchmark, source code of this study, and installation and usage guide are openly available in https://github.com/mlwin-de/BenchEmbed. In this paper, we explain the structure of this Benchmarking tool and demonstrate the usage of the benchmarking system for the knowledge graph embedding models.

Keywords: Knowledge graph embedding · Benchmarking · Link prediction.

1 Introduction

A knowledge graph is a heterogeneous multi-relational graph composed of knowledge about the world presented in a structured form i.e. facts are represented by entities that are connected using relations. Knowledge graphs embedding (KGE) models learn a mathematical approximation of knowledge graphs and produce representations for their entities and relations. These methods have been comprehensively studied recently [4, 3, 6] and are applied in many downstream Machine Learning and Natural Language Processing (NLP) tasks. A gap in current KGE studies is a standard independent evaluation environment that evaluates the efficiency of models in a fair setting (e.g. with same vector sizes). Furthermore, these studies suffer from the lack of a systematic reproducible evaluation. To target these issues, we extended the HOBBIT [2] platform is a Holistic benchmarking approach for Big Linked Data. with a new set of benchmarks with the aim to evaluate the efficiency of knowledge graph embedding models with the aforementioned criteria. We released this **Benchmarking** tool with the name

* The two preceding authors are co-first.

BenchEmbedd. A demo benchmark, source code, installation, and usage guide of this project are openly available[3].

We chose HOBBIT as the base, because it is developed under FAIR principles [5]. We follow the same concepts in making this BenchEmbedd.Another advantage of the platform is generating dockerized benchmarking, i.e., that once a system (image) is generated, it be executed locally on a personal system or a local cluster or be deployed on computing services such as Amazon Web Services (AWS).

The produced benchmarks are accessible, transferable, and easily reusable. This setting promotes reliable scientific publications, because it allows researchers to repeat the evaluations of an study without concerns about standardized evaluation hardware. We ensure the reproducibility of the evaluations by generating benchmark systems which are executable (docker) images of the exact environment of an original evaluation made by a researcher. The method is easily extensible by making a new copy and adding more models to it. In the following section, we explain the structure of our benchmarking platform. We then in explain the functionalities in Section 3 and the Demonstration of BenchEmbedd in Section 4.

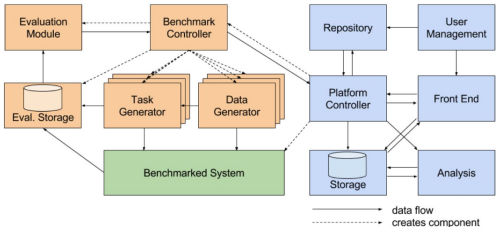

Fig. 1. HOBBIT platform structure. BenchEmbedd extends it with evaluations and metrics for Link prediction on knowledge graphs.

2 Structure:

Figure[4] 1 illustrates the components in the HOBBIT platform structure. To make a HOBBIT-based benchmark, we created the green and orange components in this figure. These parts consist of the Benchmark Components (in orange) and Benchmark System (in green).

The **Benchmark Components** provide the tasks and data for the system. All benchmark components together work as an infrastructure for benchmarking a system on the task of link prediction. This section consists of Evaluation

[3] https://github.com/mlwin-de/BenchEmbedd
[4] The diagram is from [2].

Module, Evaluation storage, Benchmark Controller, Task Generator, and Data Generator.

The **Benchmark System** contains a complete ready-to-run Benchmarking workflow within a controlled dockerized[5] running environment. A Benchmark System can contain configurations for running multiple tests on different models and different test datasets. To Extend BenchEmbedd to other datasets it is enough to duplicate and extend a new Benchmark System configuration of the benchmarking platform. Section 4 explains a demo System and explains the steps to make a new System.

3 Functionalities

In BenchEmbedd we perform a Link Prediction evaluation task. KGE models learn knowledge graphs in the form of triples (head, relation, tail), and the link prediction task tests KGE models in how efficiently they predict missing links (triples) in a knowledge graph. Figure 2 shows a knowledge graph with 4

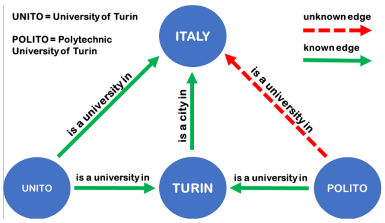

Fig. 2. An example of a knowledge graph with a missing link.

entities, where the green relations are known. In this example, the link prediction task tests how well the missing triple ("Polito", "is a university in", "Italy") is estimated by a knowledge graph learning model. A KGE model is efficient if it generates a high score for the missing link indicating the existence of this relation. The current implementation computes the following metrics: HIT@1, HIT@3, HIT@10, and Mean Reciprocal Rank. The current implementation includes the test for TransE [1] model, while the benchmark is open to be extended to other models. We configured a benchmark to test over the WN18rr benchmarking dataset for the demo.

4 Demonstration

The benchmark is a java Maven project. After the setup [6] of BenchEmbedd, to execute a sample Benchmark system online one needs to follow these steps:

[5] https://www.docker.com
[6] Setup guide is in https://github.com/mlwin-de/BenchEmbedd#installation

- Login to the website https://master.project-hobbit.eu/.
- Select “Benchmarks”.
- Select “MLwin Benchmark” in the drop-down list of “Benchmarks”.
- Select a desired System to Benchmark in the drop-down list “System”.
- Press the “Submit” Button.

At this stage, a pop-up window will appear. There the Experiment Status shows the progress of the running experiment and clicking the link in the popup window shows the experiment results once the experiment is finished. Figure 3 illustrates an example of the result table after running the demo benchmark system.

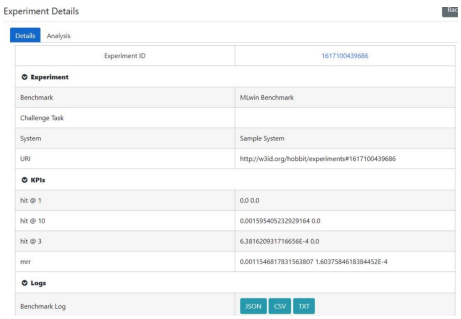

Fig. 3. An example of demonstrated evaluation results.

Adding new models: To include more metrics and datasets in the context of the knowledge graph link prediction task, it is possible to make a new Benchmark test environment with a new configuration. Then a new independent Benchmark dockerized system (colored green in Figure 1 entitled as “Benchmarked System”) is created on this configuration. The steps to make a new Benchmark environment by extending the current demo Benchmark configuration is:

- Writing a Benchmark System file.
- Providing a set of pre-trained embedding vectors.
- Creating a system docker image.
- Writing a system meta-data file.
- Creating a HOBBIT GitLab account to load up the files.

The steps to write a Benchmark System file are:

- Extend the TransEtest.java file for a new benchmark system file. It contains the method “test_triple” that is the base for the link prediction tests.
- Provide trained embeddings with names “entity2vec.txt” and “relation2vec.txt”.

Our Sample System is trained using the TransE model and the output files of the training process of this repository are converted from “.npy” to “.txt”

files using our script at "src/kge_output_to_data.py". To test the System on the Benchmark we setup the docker image that contains both the implemented system and the trained embedding vector files. [7].

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gerbil: <http://w3id.org/gerbil/vocab#> .
@prefix hobbit: <http://w3id.org/hobbit/vocab#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<http://project-hobbit.eu/sample-system/system> a       hobbit:SystemInstance;
    rdfs:label  "Sample System"@en;
        rdfs:comment "Example system for SDK Example Benchmark"@en;
        hobbit:implementsAPI <http://project-hobbit.eu/mlwin/API>;
        hobbit:imageName "git.project-hobbit.eu:4567/sadeghi.afshin/sample-system/system-adapter" .
```

Fig. 4. An example of system meta-data file.

To declare the user name and system name to HOBBIT a new system required to adopt system the meta-data file "system.ttl". Figure 4 shows an example of a system meta-data file whose label is adopted to "sample-system" and includes the GitLab username. To upload the benchmark system a HOBBIT GitLab account is required that can be created in git.project-hobbit.eu. Afterwards, the created system (docker image) can be pushed to HOBBIT GitLab.

5 Acknowledgement

This study is partially supported by the MLwin project[8] (Maschinelles Lernen mit Wissensgraphen, grant 01IS18050F of the Federal Ministry of Education and Research of Germany). MLwin Project aims to promote and study the application of Machine learning methods in knowledge graphs.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NeurIPS. pp. 1–9 (2013)
2. Röder, M., Kuchelev, D., Ngonga Ngomo, A.C.: Hobbit: A platform for benchmarking big linked data. Data Science **3**(1), 15–35 (2020)
3. Sadeghi, A., Graux, D., Yazdi, H.S., Lehmann, J.: MDE: multiple distance embeddings for link prediction in knowledge graphs. In: ECAI (2020)
4. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. TKDE (2017)
5. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The fair guiding principles for scientific data management and stewardship. Nature Scientific data (2016)

[7] mvn commands: https://github.com/mlwin-de/BenchEmbedd#benchmark-the-system-online

[8] https://mlwin.de/

6. Zhang, S., Tay, Y., Yao, L., Liu, Q.: Quaternion knowledge graph embeddings. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) NeurIPS (2019)