BioKEEN Documentation

Release 0.0.15-dev

Mehdi Ali, Charles Tapley Hoyt, and Daniel Domingo-Fernández

Dec 11, 2019

Getting Started

BioKEEN (Biological KnowlEdge EmbeddiNgs) is a package for training and evaluating **biological** knowledge graph embeddings built on PyKEEN. Within BioKEEN several biomedical databases are directly accessible for training and evaluating biological knowledge graph embeddings (see *Biological Databases*).

Because we use PyKEEN as the core underlying framework, currently, implementations of 10 knowledge graph emebddings models are avaialble for BioKEEN. Furthermore, it can be run in training mode in which users provide their own set of hyper-parameter values, or in hyper-parameter optimization mode to find suitable hyper-parameter values from set of user defined values. BioKEEN can also be run without having experience in programing by using its interactive command line interface that can be started with the command "biokeen" from a terminal.

Installation is as easy as getting the code from PyPI with `python3 -m pip install biokeen`.

CHAPTER 1

Citation

If you use BioKEEN in your work, please cite[1]:

1.1 Installation

There are several ways to download and install BioKEEN.

Warning: BioKEEN requires Python 3.6+

1.1.1 Easiest

Download the latest stable code from PyPI with:

```
$ pip install biokeen
```

1.1.2 Get the Latest

Download the most recent code from GitHub with:

```
$ pip install git+https://github.com/SmartDataAnalytics/BioKEEN.git
```

1.2 Train and Evaluate

1.2.1 Tutorial

[1] Ali, M., *et al.* (2018). BioKEEN: A library for learning and evaluating biological knowledge graph embeddings.

Step 1: Start CLI

```
biokeen start
```

Step 2: Select data source

```
##################################################
#              Welcome to BioKEEN                #
##################################################
--------------------------------------------------------------------------------------------------------------
This interface will assist you to configure your experiment.

BioKEEN can be run in two modes:
1.) Training mode: BioKEEN trains a model based on a set of user-defined hyper-parameters.
2.) Hyper-parameter optimization mode: Apply Random Search to determine the most appropriate set of hyper-parameter values
--------------------------------------------------------------------------------------------------------------
Do you want to use one of the databases provided by BioKEEN? [Y/n]:
```

Step 3: Select database

```
Current Step: Please select the database you want to train on:
1: compath
2: hippie
3: kegg
4: mirtarbase
5: msig
6: reactome
7: sider
8: wikipathways
9: drugbank
10: adeptus
11: hsdn
12: interpro
> Please select one of the options: :
```

Step 4: Specify execution mode

```
Current Step: Please choose the execution mode: (1) training mode or (2) hyper-parameter search mode.
Do you have hyper-parameters? If not, PyKEEN will be configured for hyper-parameter search. [y/N]:
```

Step 5: Select KGE model

```
Current Step: Please choose one of the provided models.
Depending on which model you select, PyKEEN will assist you to configure the required hyper-parameters.

Please select the embedding model you want to train:
TransE: 1
TransH: 2
TransR: 3
TransD: 4
Structure Embedding (SE): 5
Unstructured Model (UM): 6
DistMult: 7
ERMLP: 8
RESCAL: 9
ConvE: 10
> Please select one of the options:
```

Step 6: Specify model dependent hyper-parameters

Step 7: Specify the batch-size

```
Current Step: Please specify the batch size.
Typical batch sizes are 32,64 and 128

Please type the batch size comma:
> Batch size:
```

Step 8: Specify the number of training epochs

```
Current Step: Please specify the number of epochs
The number of epochs defines how often to iterte over the whole training set during the training the model.

Please type the number of epochs:
> Epochs:
```

Step 9: Specify whether to evaluate the model

```
Current Step: Please specify the number of epochs
The number of epochs defines how often to iterte over the whole training set during the training the model.

Please type the number of epochs:
> Epochs:
```

Step 10: Specify whether to evaluate the model

```
Current Step: Please specify the number of epochs
The number of epochs defines how often to iterte over the whole training set during the training the model.

Please type the number of epochs:
> Epochs:
```

Step 11: Provide a random seed

```
Current Step: Please specify a random seed (positive integer).
The random seed is used for negative sampling, and for the random train and test split if requested.

Please specify the random seed.
> Random seed:
```

Step 12: Specify preferred device

```
Current Step: Please specify the preferred device (GPU or CPU).
> Please type 'GPU' or 'CPU':
```

Step 13: Specify the path to the output directory

```
Please provide the path to your output directory.


> Path to output directory:
```

1.2.2 Reference

1.3 Perform Inference

1.3.1 Starting the Prediction Pipeline

```
biokeen predict -m /path/to/model/directory -d /path/to/data/directory
```

where the value for the argument **-m** is the directory containing the model, in more detail following files must be contained in the directory:

- configuration.json
- entities_to_embeddings.json
- relations_to_embeddings.json
- trained_model.pkl

These files are created automatically created when an experiment is configured through the CLI.

The value for the argument **-d** is the directory containing the data for which inference should be applied, and it needs to contain following files:

- entities.tsv
- relations.tsv

where *entities.tsv* contains all entities of interest, and relations.tsv all relations. PyKEEN will create all possible combinations of triples, and computes the predictions for them, and saves them in data directory in *predictions.tsv*.

Optionally, a set of triples can be provided that should be exluded from the prediction, e.g. all the triples contained in the training set:

```
pykeen-predict -m /path/to/model/directory -d /path/to/data/directory -t /path/to/
↪triples.tsv
```

Hence, it is easily possible to compute plausibility scores forr all triples that are not contained in the training set.

1.3.2 CLI Manual

1.4 Summarize all Experiments

Here, we describe how to summarize all experiments into a single csv-file. To get the summary, please provide the path to parent directory containing all the experiments as sub-directories, and the path to the output file:

```
biokeen summarize -d /path/to/experiments/directory -o /path/to/output/file.csv
```

1.5 Train and Evaluate

Here, we explain how to define and run experiments programmatically. This should be done using PyKEEN.

1.5.1 Configure your experiment

To programmatically train (and evaluate) a KGE model, a python dictionary must be created specifying the experiment:

```
config = dict(
    training_set_path           = 'data/corpora/fb15k/compath.tsv',
    test_set_ratio              = 0.1,
    execution_mode              = 'Training_mode',
    kg_embedding_model_name     = 'TransE',
    embedding_dim               = 50,
    normalization_of_entities   = 2,  # corresponds to L2
    scoring_function            = 1,  # corresponds to L1
    margin_loss                 = 1,
    learning_rate               = 0.01,
    batch_size                  = 32,
    num_epochs                  = 1000,
    filter_negative_triples     = True,
    random_seed                 = 2,
    preferred_device            = 'cpu',
)
```

1.5.2 Run your experiment

```
results = pykeen.run(
    config=config,
    output_directory=output_directory,
)
```

1.5.3 Access your results

Show all keys contained in `results`:

```
print('Keys:', *sorted(results.results.keys()), sep='\n  ')
```

1.5.4 Access trained KGE model

```
results.results['trained_model']
```

1.5.5 Access the losses

```
results.results['losses']
```

1.5.6 Access evaluation results

```
results.results['eval_summary']
```

1.6 Apply a Hyper-Parameter Optimization

Here, we describe how to define an experiment that should perform a hyper-parameter optimization mode.

1.6.1 Configure your experiment

To run experiments programmatically, the core software library PyKEEN should be used. To run PyKEEN in hyper-parameter optimization (HPO) mode, please set **execution_mode** to **HPO_mode**.In HPO mode several values can be provided for the hyper-parameters from which different settings will be tested based on the hyper-parameter optimization algorithm. The possible values for a single hyper-parameter need to be provided as a list. The **maximum_number_of_hpo_iters** defines how many HPO iterations should be performed.

```
config = dict(
    training_set_path           = 'data/corpora/compath.tsv',
    test_set_ratio              = 0.1,
    execution_mode              = 'HPO_mode',
    kg_embedding_model_name     = 'TransE',
    embedding_dim               = [50,100,150]
    normalization_of_entities   =  2,  # corresponds to L2
    scoring_function            = [1,2],  # corresponds to L1
    margin_loss                 = [1,1.5,2],
    learning_rate               = [0.1,0.01],
    batch_size                  = [32,128],
    num_epochs                  = 1000,
    maximum_number_of_hpo_iters = 5,
    filter_negative_triples     = True,
    random_seed                 = 2,
    preferred_device            = 'cpu',
)
```

1.6.2 Run your experiment

The experiment will be started with the *run* function, and in the output directory the exported results will be saved.

```
results = pykeen.run(
    config=config,
    output_directory=output_directory,
)
```

1.6.3 Access your results

Show all keys contained in `results`:

```
print('Keys:', *sorted(results.results.keys()), sep='\n  ')
```

1.6.4 Access trained KGE model

```
results.results['trained_model']
```

1.6.5 Access the losses

```
results.results['losses']
```

1.6.6 Access evaluation results

```
results.results['eval_summary']
```

1.7 Handling BEL

biokeen.convert.**to_pykeen_path**(*df*, *path*)

Write the relationships in the BEL graph to a KEEN TSV file.

If you have a BEL graph, first do:

```
>>> from biokeen.convert import to_pykeen_df, to_pykeen_path
>>> graph = ...  # Something from PyBEL
>>> df = to_pykeen_df(graph)
>>> to_pykeen_path(df, 'graph.keen.tsv')
```

Return type bool

biokeen.convert.**to_pykeen_df**(*graph*, *use_tqdm=True*)

Get a DataFrame representing the triples.

Return type DataFrame

1.8 Biological Databases

The following biological databases can be used for training and evaluating knowledge graph embeddings. This is done by using the Bio2BEL universe.

Source	DOI
ADEPTUS	
ComPath	
DrugBank	
ExPASy	
HIPPIE	
HSDN	
KEGG	
miRTarBase	
MSigDB	
Reactome	
SIDER	
InterPro	
WikiPathways	

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

T