PyKEEN

PyKEEN is a Python package for reproducible, facile knowledge graph embeddings.

The fastest way to get up and running is to use the `pykeen.pipeline.pipeline()` function.

It provides a high-level entry into the extensible functionality of this package. The following example shows how to train and evaluate the TransE model (`pykeen.models.TransE`) on the Nations dataset (`pykeen.datasets.Nations`) by referring to them by name. By default, the training loop uses the stochastic closed world assumption training approach (`pykeen.training.SLCWATrainingLoop`) and evaluates with rank-based evaluation (`pykeen.evaluation.RankBasedEvaluator`).

```
>>> from pykeen.pipeline import pipeline
>>> result = pipeline(
...     model='TransE',
...     dataset='Nations',
... )
```

The results are returned in a `pykeen.pipeline.PipelineResult` instance, which has attributes for the trained model, the training loop, and the evaluation.

PyKEEN has a function `pykeen.env()` that magically prints relevant version information about PyTorch, CUDA, and your operating system that can be used for debugging. If you're in a Jupyter notebook, it will be pretty printed as an HTML table.

```
>>> import pykeen
>>> pykeen.env()
```

Getting Started

- Installation
  - Linux and Mac Users
  - Google Colab and Kaggle Users
  - Windows Users
  - Development
  - Extras
- First Steps

  - Training a Model
  - Deeper Configuration
  - Loading a pre-trained Model
  - Mapping Entity and Relation Identifiers to their Names
  - Using Learned Embeddings
  - Beyond the Pipeline
  - Preview: Evaluation Loops
  - Training Callbacks
  - Next Steps
- Knowledge Graph Embedding Models
  - Representation
  - Interaction Function
- Tracking Results during Training
  - Using MLflow
  - Using Neptune.ai
  - Using Weights and Biases
  - Using Tensorboard
  - Using File-Based Tracking
- Saving Checkpoints during Training
  - Regular Checkpoints
  - Checkpoints on Failure
  - Checkpoints When Bringing Your Own Data
  - Word of Caution and Possible Errors
  - Checkpoints beyond the Pipeline and Technicalities
- A Toy Example with Translational Distance Models
- Understanding the Evaluation
  - Rank-Based Metrics
  - Ranking Types
  - Ranking Sidedness
  - Ranking Aggregation Scope
  - Filtering
  - Entity and Relation Restriction
- Optimizing a Model’s Hyper-parameters
  - Defaults
  - Strategies
  - Custom Strategies
  - Early Stopping
  - Configuring Optuna
  - Full Examples
- Running an Ablation Study

- Define Your Own HPO Ranges
- Run an Ablation Study With Your Own Data
- Run an Ablation Study From The Command Line Interface

- Performance Tricks
  - Entity and Relation IDs
  - Tuple Broadcasting
  - Filtering with Index-based Masking
  - Sub-batching & Slicing
  - Automated Memory Optimization
  - Evaluation Fallback
- Representations
  - Message Passing
  - Decomposition
  - Text-based
- Getting Started with NodePiece
  - Basic Usage
  - Anchor Selection and Searching
  - How many total anchors *num_anchors* and anchors & relations *num_tokens* do I need for my graph?
  - Using NodePiece with `pykeen.pipeline.pipeline()`
  - Pre-Computed Vocabularies
  - Configuring the Interaction Function
  - Configuring the Aggregation Function
  - NodePiece + GNN
  - Tokenizing Large Graphs with METIS
- Inductive Link Prediction
  - Organizing the Dataset
  - Featurizing Unseen Entities
  - Training & Evaluation
  - Full Inductive LP Example
- PyTorch Lightning Integration
  - Classes
- Using Resolvers
  - Determining Allowed Inputs
- Troubleshooting
  - Loading a Model from an Old Version of PyKEEN

Bring Your Own

- Bring Your Own Data
  - Pre-stratified Dataset
  - Unstratified Dataset
  - Bring Your Own Data with Checkpoints
- Bring Your Own Interaction
  - Implementing your first Interaction Module
  - Interactions with Hyper-Parameters
  - Interactions with Trainable Parameters
  - Interactions with Different Shaped Vectors
  - Interactions with Multiple Representations
  - Interactions with Different Dimension Vectors
  - Differences between `pykeen.nn.modules.Interaction` and `pykeen.models.Model`
  - *Ad hoc* Models from Interactions
  - Interaction Pipeline

Extending PyKEEN

- Extending the Datasets
  - Pre-split Datasets
  - Unsplit Datasets
  - Updating the `setup.cfg`
- Extending the Models
  - Implementing a model by subclassing `pykeen.models.ERModel`
  - Implementing a model by instantiating `pykeen.models.ERModel`
  - Using a Custom Model with the Pipeline

Reference

- Pipeline
  - Functions
  - Classes
- Models
  - Functions
  - Classes
  - Class Inheritance Diagram
- Datasets
  - pykeen.datasets Package

  - pykeen.datasets.base Module
  - pykeen.datasets.analysis Module
- Inductive Datasets
  - pykeen.datasets.inductive Package
- Entity Alignment
  - pykeen.datasets.ea.combination Module
- Triples
  - `CoreTriplesFactory`
  - `Instances`
  - `KGInfo`
  - `LCWAInstances`
  - `SLCWAInstances`
  - `TriplesFactory`
  - `TriplesNumericLiteralsFactory`
  - `get_mapped_triples()`
  - `compute_compressed_adjacency_list()`
  - `get_entities()`
  - `get_relations()`
  - `load_triples()`
  - `tensor_to_df()`
  - `remix()`
  - `deteriorate()`
- Training
  - Assumptions
  - Classes
  - Callbacks
  - Learning Rate Schedulers
- Stoppers
  - `NopStopper`
  - `EarlyStopper`
  - Base Classes
- Loss Functions
  - Pointwise Loss Functions
  - Pairwise Loss Functions
  - Setwise Loss Functions
  - Classes
  - Class Inheritance Diagram
- Regularizers

- Classes
- Class Inheritance Diagram
- Base Classes

- Result Trackers
  - Functions
  - Classes
  - Class Inheritance Diagram
- Negative Sampling
  - Construction of Knowledge Graphs
  - Corruption
  - Classes
  - Class Inheritance Diagram
- Filtering
  - Identifying False Negatives During Training
  - Identifying False Negatives During Evaluation
  - Classes
  - Class Inheritance Diagram
- Evaluation
  - Functions
  - Classes
  - Class Inheritance Diagram
- Metrics
  - Classes
  - Class Inheritance Diagram
  - Functions
  - Classes
  - Class Inheritance Diagram
- Hyper-parameter Optimization
  - `HpoPipelineResult`
  - `hpo_pipeline_from_path()`
  - `hpo_pipeline_from_config()`
  - `hpo_pipeline()`
- Ablation
  - `ablation_pipeline()`
  - `prepare_ablation_from_config()`
  - `prepare_ablation()`
- Lookup

  - `model_resolver`
  - `loss_resolver`
  - `optimizer_resolver`
  - `regularizer_resolver`
  - `stopper_resolver`
  - `negative_sampler_resolver`
  - `dataset_resolver`
  - `training_loop_resolver`
  - `evaluator_resolver`
  - `metric_resolver`
- Prediction
  - High-Level
  - Low-Level
  - Potential Caveats
  - Migration Guide
  - Functions
  - Classes
  - Class Inheritance Diagram
- Uncertainty
  - Functions
  - Classes
- Sealant
  - `Sealant`
  - `reindex()`
  - `unleak()`
- Constants
  - `PYKEEN_BENCHMARKS`
  - `PYKEEN_CHECKPOINTS`
  - `PYKEEN_DATASETS`
  - `PYKEEN_EXPERIMENTS`
  - `PYKEEN_HOME`
  - `PYKEEN_LOGS`
  - `Constrainer`
  - `DeviceHint`
  - `GaussianDistribution`
  - `HeadRepresentation`
  - `InductiveMode`
  - `Initializer`
  - `LabeledTriples`
  - `Mutation`

    - `Normalizer`
    - `RelationRepresentation`
    - `TailRepresentation`
    - `Target`
    - `TargetColumn`
    - `TorchRandomHint`
    - `cast_constrainer()`
- `pykeen.nn`
    - Functional
    - Stateful Interaction Modules
    - Similarity
    - Representation
    - Initialization
    - Message Passing
    - PyG Message Passing
    - Weighting
    - Combinations
    - Perceptron
    - Utilities
    - NodePiece
- Utilities
    - `Bias`
    - `ExtraReprMixin`
    - `NoRandomSeedNecessary`
    - `Result`
    - `all_in_bounds()`
    - `at_least_eps()`
    - `broadcast_upgrade_to_sequences()`
    - `calculate_broadcasted_elementwise_result_shape()`
    - `check_shapes()`
    - `clamp_norm()`
    - `combine_complex()`
    - `compact_mapping()`
    - `complex_normalize()`
    - `compose`
    - `compute_box()`
    - `convert_to_canonical_shape()`
    - `create_relation_to_entity_set_mapping()`
    - `einsum()`
    - `ensure_complex()`
    - `ensure_ftp_directory()`
    - `ensure_torch_random_state()`

  - `ensure_tuple()`
  - `estimate_cost_of_sequence()`
  - `extend_batch()`
  - `fix_dataclass_init_docs()`
  - `flatten_dictionary()`
  - `format_relative_comparison()`
  - `get_batchnorm_modules()`
  - `get_benchmark()`
  - `get_connected_components()`
  - `get_devices()`
  - `get_df_io()`
  - `get_dropout_modules()`
  - `get_edge_index()`
  - `get_expected_norm()`
  - `get_json_bytes_io()`
  - `get_model_io()`
  - `get_optimal_sequence()`
  - `get_preferred_device()`
  - `get_until_first_blank()`
  - `invert_mapping()`
  - `is_cuda_oom_error()`
  - `is_cudnn_error()`
  - `is_triple_tensor_subset()`
  - `isin_many_dim()`
  - `logcumsumexp()`
  - `lp_norm()`
  - `negative_norm()`
  - `negative_norm_of_sum()`
  - `nested_get()`
  - `normalize_path()`
  - `normalize_string()`
  - `point_to_box_distance()`
  - `powersum_norm()`
  - `prepare_filter_triples()`
  - `product_normalize()`
  - `project_entity()`
  - `random_non_negative_int()`
  - `rate_limited()`
  - `resolve_device()`
  - `set_random_seed()`
  - `split_complex()`
  - `split_list_in_batches_iter()`
  - `tensor_product()`

  - `tensor_sum()`
  - `triple_tensor_to_set()`
  - `unpack_singletons()`
  - `upgrade_to_sequence()`
  - `view_complex()`
  - `env()`
  - `get_git_branch()`
  - `get_git_hash()`
  - `get_version()`

Appendix

- Analysis
  - Dataset Degree Distributions
- References

Indices and Tables

- Index
- Module Index
- Search Page