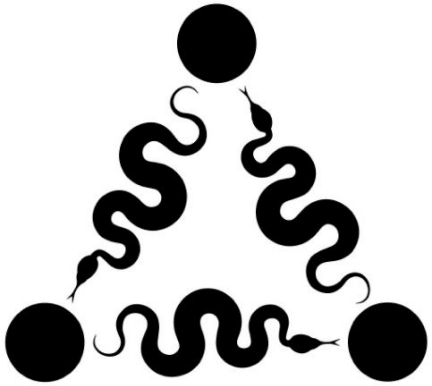
Knowledge Graph Embedding with PyKEEN in 2022

Dr. Charles Tapley Hoyt
@cthoyt @cthoyt
https://cthoyt.com

Postdoctoral Research Fellow
Laboratory of Systems Pharmacology
Harvard Medical School

Presented May 6th, 2022
Knowledge Graph Conference (KGC) 2022
https://bit.ly/pykeen-kgc2022 /
https://doi.org/10.5281/zenodo.6461152



PyKEEN v1.8

Predictions for the People







33 Datasets **42** Models **13** Losses **5** Regularizers **2** Trainers **3** Evaluators **42** Metrics

 @keenuniverse  https://github.com/pykeen  https://pykeen.github.io

```
$ pip install pykeen
```

Mehdi Ali
Ph.D. Candidate at the
University of Bonn and Fraunhofer IAIS

Max Berrendorf, Ph.D.
Researcher at
LMU Munich

Michael Galkin, Ph.D.
Postdoctoral Fellow at
Mila and McGill University

Charles Tapley Hoyt, Ph.D.
Postdoctoral Research Fellow at
Harvard Medical School

Sahand Sharifzadeh
Ph.D. candidate at
LMU Munich

Laurent Vermue, Ph.D.
Researcher at the
Technical University of Denmark

*alphabetic order by last name

Advisors



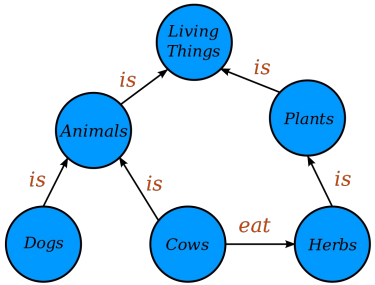
Contributors

Funding

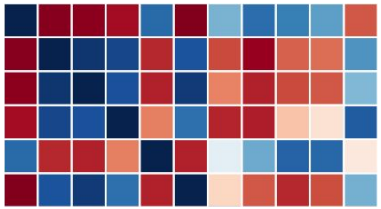
Funding Body	Program	Grant
DARPA	Young Faculty Award (PI: Benjamin Gyori)	W911NF2010255
DARPA	Automating Scientific Knowledge Extraction (ASKE)	HR00111990009
German Federal Ministry of Education and Research (BMBF)	Maschinelles Lernen mit Wissensgraphen (MLWin)	01IS18050D
German Federal Ministry of Education and Research (BMBF)	Munich Center for Machine Learning (MCML)	01IS18036A
Innovation Fund Denmark (Innovationsfonden)	Danish Center for Big Data Analytics driven Innovation (DABAI)	Grand Solutions

See also: https://pykeen.github.io/organization

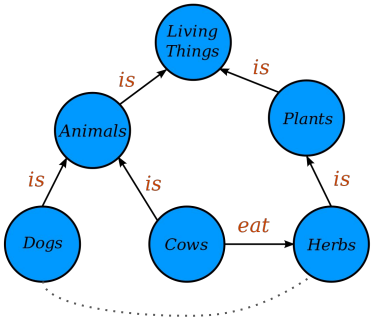
Background



A ***knowledge graph*** is a set of entities, relations, and triples



A ***representation*** is typically a vector representation for an entity or relation



Useful for ***downstream tasks*** like link prediction, clustering, entity alignment, question answering, dialogue

Image from https://en.wikipedia.org/wiki/Knowledge_graph

Where PyKEEN Started

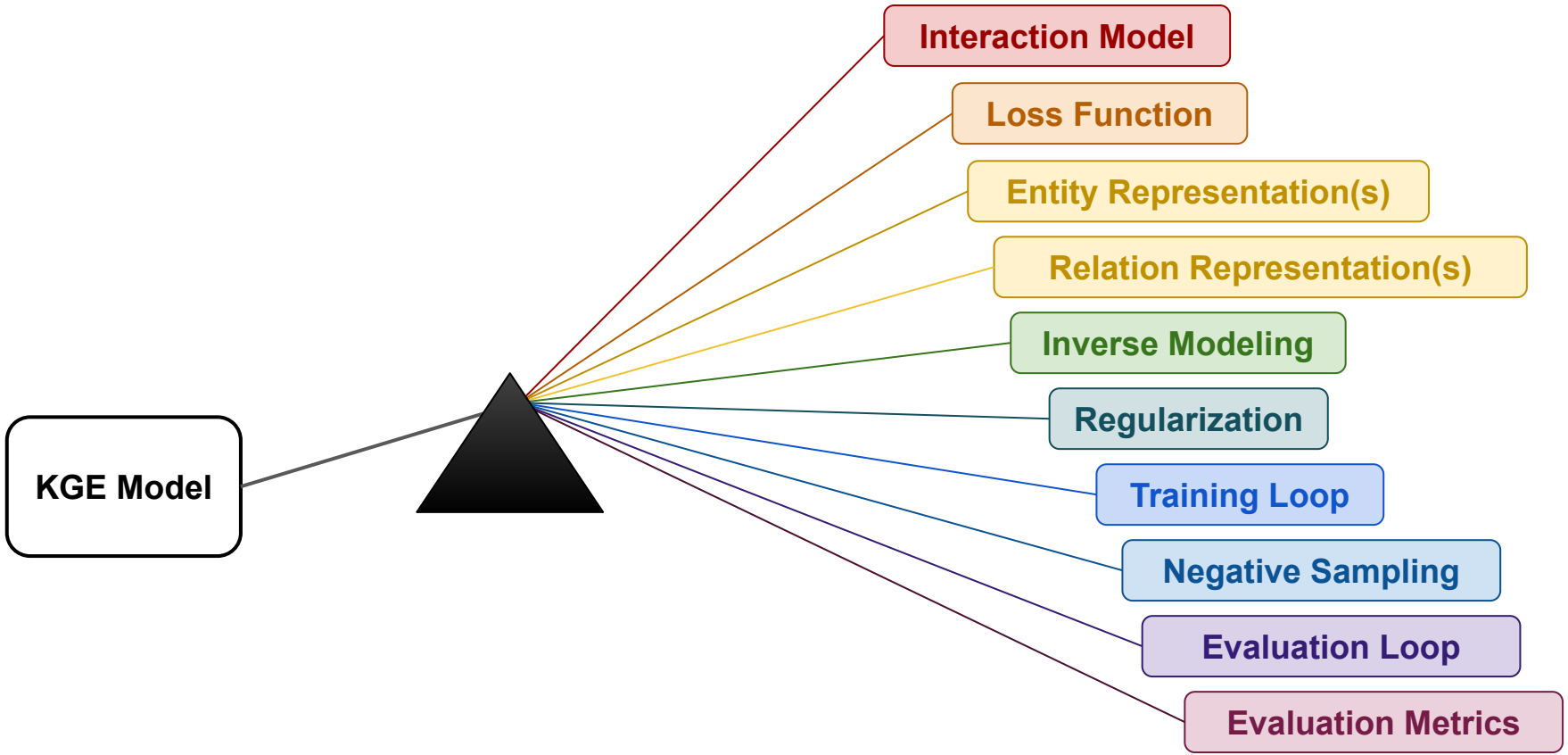
Back in 2018, there was a huge variety of:

- Decoders/interaction functions (e.g., DistMult, TransE, ComplEx)
- Benchmarking datasets (e.g., FB15k-237, YAGO-310, etc.)
- Preprocessing workflows (e.g., adding inverse triples)
- Training procedures (e.g., OWA, sLCWA, LCWA, CWA)
- Evaluation procedures and metrics

What was missing:

- Ability to reproduce previous results
- Meaningful way to compare different formulations of KGE models
- A modular architecture
- Legible, reusable, extendable, sustainable code
- Documentation for newcomers
- A moment of introspection and philosophizing

PyKEEN has a modular architecture for KGEMs



PyKEEN is for Users

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

Train your first model

```
>>> from pykeen.hpo import hpo_pipeline
>>> hpo_pipeline_result = hpo_pipeline(
...     n_trials=30,
...     dataset='Nations',
...     model='TransE',
... )
```

Optimize your first model

```
from pykeen.nn.modules import Interaction

class TransEInteraction(Interaction):
    def forward(self, h, r, t):
        return -(h + r - t).norm(p=2, dim=-1)
```

Implement your own model

```
>>> from pykeen.pipeline import pipeline
>>> # Run the pipeline
>>> pipeline_result = pipeline(dataset='Nations', model='RotatE')
>>> model = pipeline_result.model
>>> # Predict tails
>>> predicted_tails_df = model.get_tail_prediction_df('brazil', 'intergovorgs')
```

Predict new links

Interaction Functions (42)

- NodePiece
- InductiveNodePiece
- TuckER
- HolE
- QuatE
- FixedModel
- RotatE
- DistMultLiteral
- BoxE
- TransR
- ERMLP
- CrossE
- TransF
- ConvE
- NTN
- ComplExLiteral
- ERMLPE
- InductiveNodePieceGNN
- PairRE
- CompGCN
- CP
- KG2E
- UM
- TorusE
- ProjE
- TransE
- DistMult
- DistMA
- RESCAL
- SE
- ConvKB
- TransH
- MuRE
- AutoSF
- TransD
- DistMultLiteralGated
- SimplE
- RGCN
- ComplEx

Datasets (33)

- CSKG
- Countries
- ConceptNet
- Kinships
- OpenBioLinkLQ
- CoDExMedium
- Nations
- WK3l15k
- WD50KT
- CoDExSmall
- PharmKG
- Hetionet
- DB100K
- FB15k
- WN18RR
- OpenEA
- DBpedia50
- WK3l120k
- OGBBioKG
- PharmKG8k
- AristoV4
- CoDExLarge
- NationsLiteral
- YAGO310
- WN18
- UMLS
- BioKG
- OGBWikiKG2
- CKG
- Wikidata5M
- FB15k237
- DRKG
- CN3l
- OpenBioLink

PyKEEN Benchmark (2020)

Results

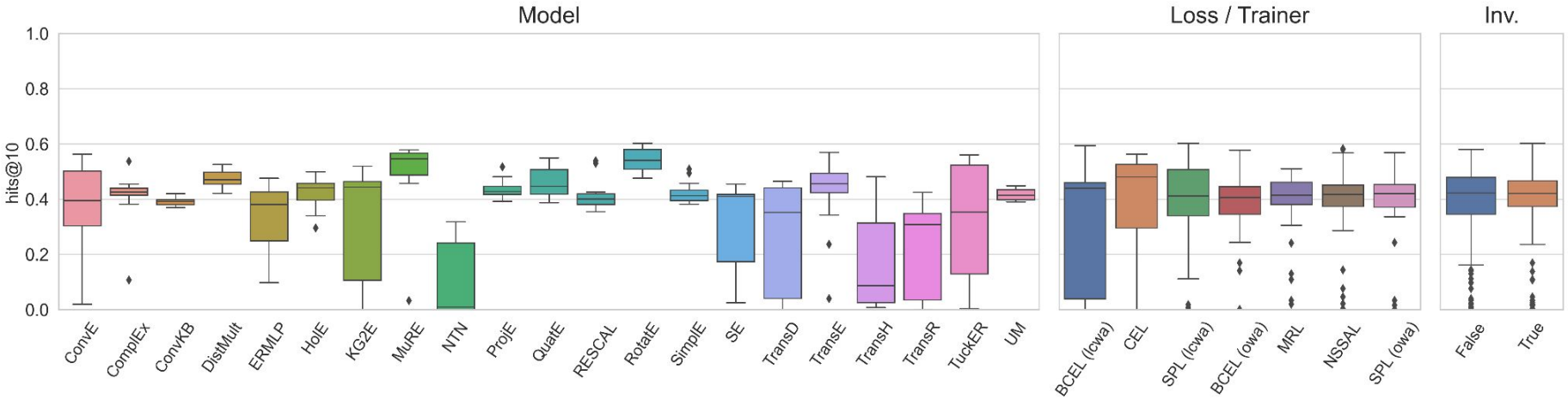
- SOTA can usually be achieved with arbitrary additional HPO
- RotatE was consistently the best performing model
- sLCWA + inverse relations is the best training paradigm
- CEL and MRL underperforming, NSSAL is usually the best
- Less complicated models can still perform well
- Old models can outcompete previous SOTA

New Questions

- When should we stop optimizing?
- Can we trust metrics?
- Are we really making fair evaluations?
- Are bigger, slower models really better?
- How much ablation is necessary given insight as to some obvious best pairs?

Code: https://github.com/pykeen/benchmarking, Paper: https://arxiv.org/abs/2006.13365

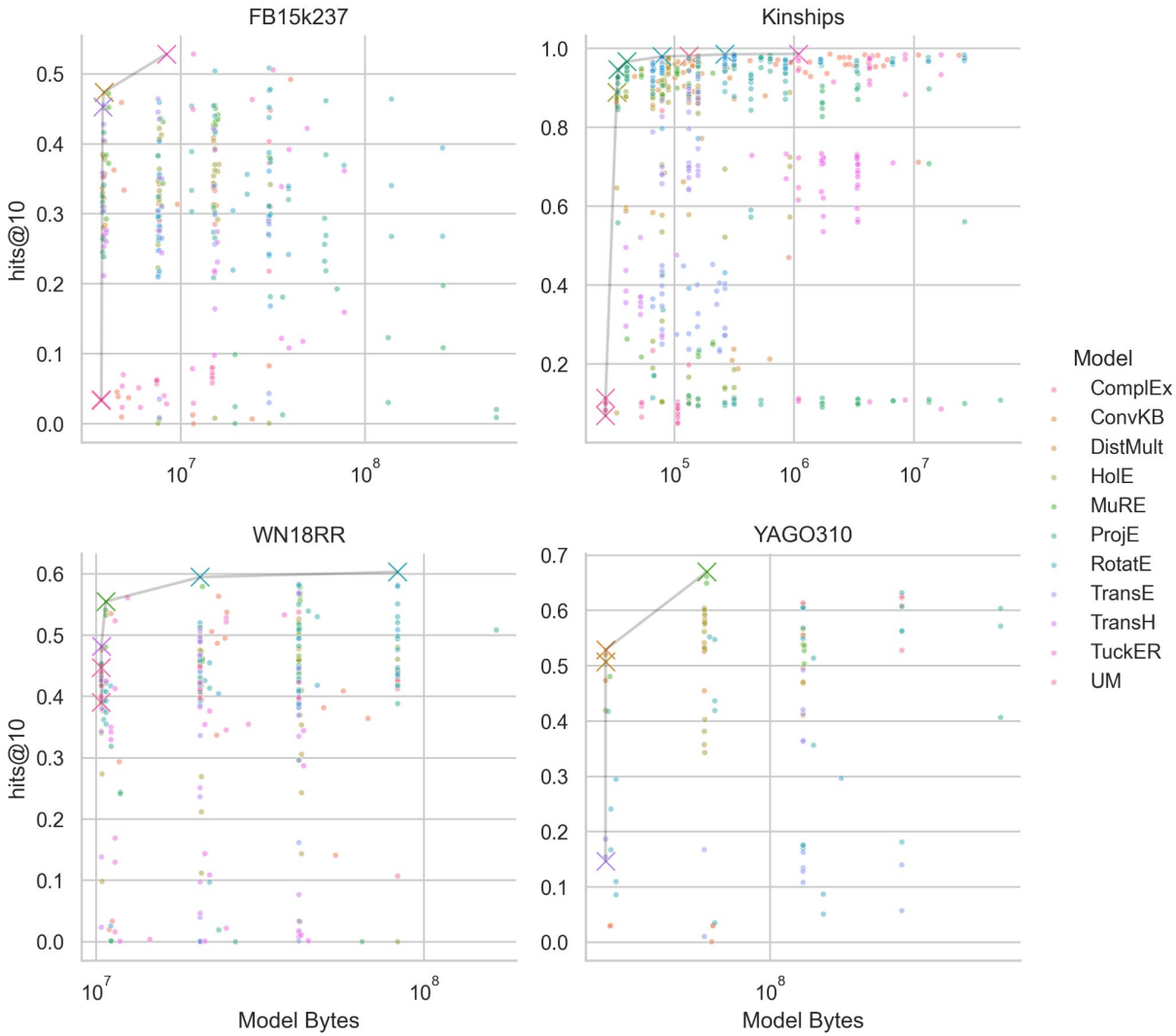
FB15k-237 Results



- New models proposed frequently, squeezing minimal gains on standard metrics
- New tools like batch normalization and improved optimizers lead to SOTA results.
- E.g., WN18-RR, TransE, 56.98% Hits@10

Reported in: Ali, M., *et al.* (2020). *Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework*. http://arxiv.org/abs/2006.13365

Bigger, more complex models aren't even necessarily better.

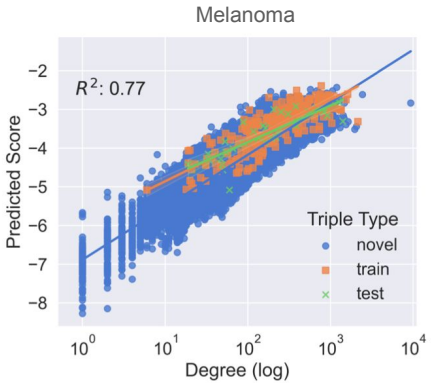


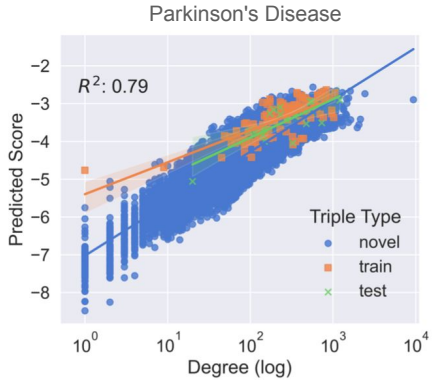
Reported in: Ali, M., *et al.* (2020). *Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework.* http://arxiv.org/abs/2006.13365

Adoption

Pharmaceutical Industry - AstraZeneca

- Core business of pharmaceutical is to identify safe, active modulators of disease
- Bonner *et al.* applied link prediction to biomedical knowledge graphs
  - *Chemogenomics* - between chemical and protein
  - *Target identification* - between protein and disease
  - *Drug repositioning* - between chemical and disease
- Reference: http://arxiv.org/abs/2112.06567





Academia (Highlights)

- Ontologies
  - https://repository.kaust.edu.sa/handle/10754/663430
- Biomedicine
  - https://doi.org/10.1093/bioinformatics/btaa274
  - https://openreview.net/forum?id=qI-IS8DPq_N
- Precision and Personalized Medicine
  - https://doi.org/10.1093/bioinformatics/btab340
- Bibliometrics and Meta-research
  - https://arxiv.org/abs/1904.12211
  - https://recnlp2019.github.io/papers/RecNLP2019_paper_20.pdf
- Bias Detection
  - https://arxiv.org/abs/2109.10697
- Entity Typing
  - https://dl.acm.org/doi/abs/10.1145/3460210.3493563
- Industry 4.0
  - https://link.springer.com/chapter/10.1007/978-3-030-59051-2_12

Current and Future Directions

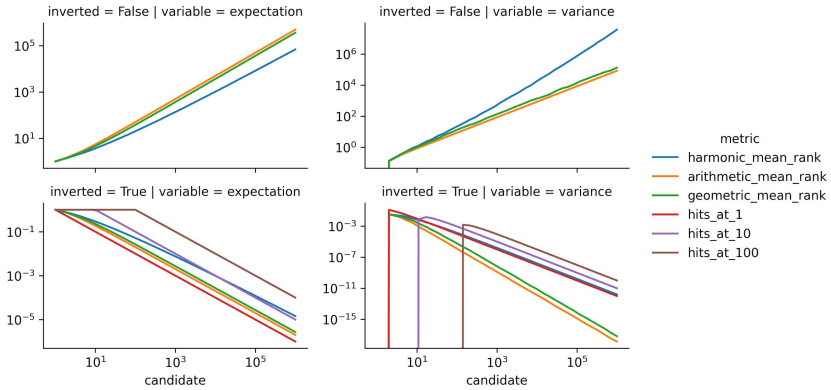
Graph Neural Networks

- PyKEEN implements two GNN-based KGE models: R-GCN and CompGCN
- **Goal:** more easily integrate components like RGCNConv in PyKEEN models
- **Goal:** better promote reusability of PyKEEN components

```
602 + class PygRGCNLayer(Layer):
603 +     """An alternate implementation of the R-GCN layer using PyG."""
604 +
605 +     def __init__(
606 +         self,
607 +         input_dim: int,
608 +         num_relations: int,
609 +         output_dim: Optional[int] = None,
610 +         use_bias: bool = True,
611 +     ):
612 +         """Initialize the PyG convolution."""
613 +         from torch_geometric.nn import RGCNConv
614 +
615 +         super().__init__()
616 +         self.conv = RGCNConv(
617 +             in_channels=input_dim,
618 +             out_channels=output_dim,
619 +             num_relations=num_relations,
620 +             bias=use_bias,
621 +         )
622 +
623 +     def reset_parameters(self):  # noqa: D102
624 +         self.conv.reset_parameters()
625 +
626 +     def forward(
627 +         self,
628 +         x: torch.FloatTensor,
629 +         edge_index: Tuple[torch.LongTensor, torch.LongTensor],
630 +         edge_type: torch.LongTensor,
631 +     ):  # noqa: D102
632 +         return self.conv(x, edge_index, edge_type)
633 +
634 +
```

Better Evaluation Metrics

Problem: Ranked-based metrics for link prediction on knowledge graphs are dataset-dependent



Solution: introduce additional affine statistical adjustments (like Bonferroni in statistics) inspired by Berrendorf, *et al.* (2020)

Adjustment by expectation and optimum

$$M^*(r_1, \ldots, r_n) = \frac{M(r_1, \ldots, r_n) - E[M]}{\max(M) - E[M]}$$

- adjusted mean rank index (AMRI)
- adjusted mean reciprocal rank index (AMRRI)
- adjusted geometric mean rank index (AGMRI)
- adjusted hits @ k index (AH@K)

Adjust by expectation and variance (z-score)

$$M^*(r_1, \ldots, r_n) = \frac{M(r_1, \ldots, r_n) - E[M]}{\sqrt{Var[M]}}$$

- *z*-mean rank (zMR)
- *z*-mean reciprocal rank (zMRR)
- *z*-geometric mean rank (zGMR)
- *z*-hits @ k (zH@K)

Code: https://github.com/pykeen/ranking-metrics-manuscript, Pre-print: https://arxiv.org/abs/2203.07544

Entity Alignment with KGE Models

Problem: we often need to integrate different KGs but not all entities have mappings

Solution: formalize as a link prediction task by concatenating KGs with a special ***same as*** relation between mapped entities (see arxiv:1906.02390 , arxiv:1911.08936)

* caveat: specialized models often perform superior

Future: PyKEEN already packages most common EA benchmarking datasets for evaluation

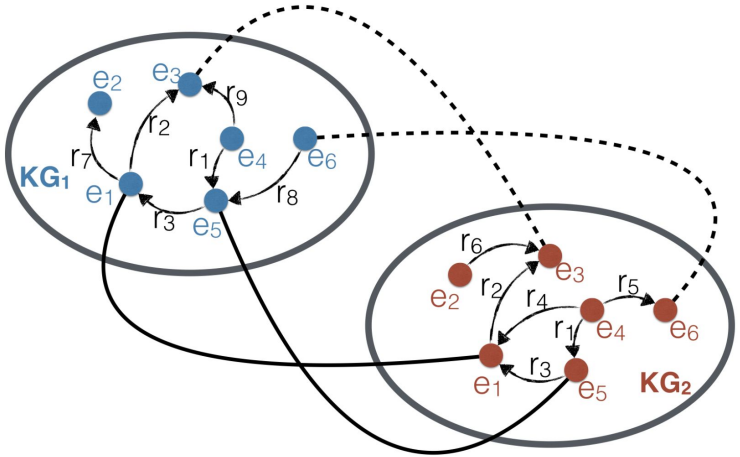
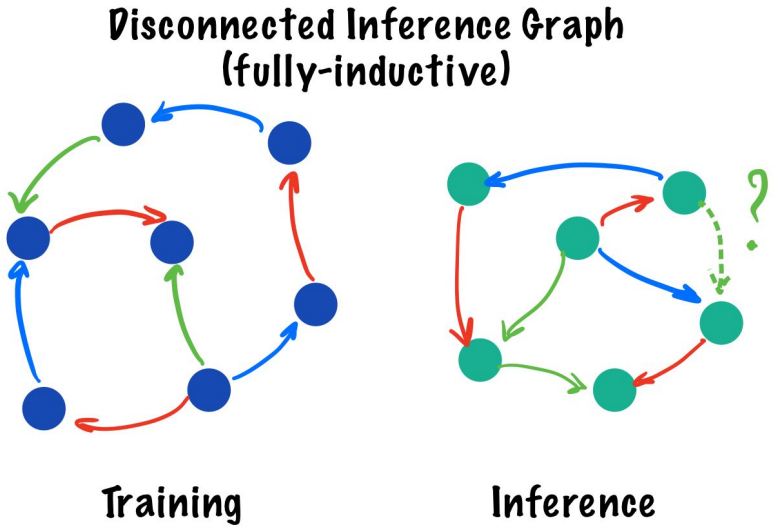


Image from Zhu et al., 2017
https://www.ijcai.org/proceedings/2017/0595.pdf

Inductive Link Prediction and NodePiece

Problem: Previous knowledge graph embedding models could only be trained and applied in the *transductive* setting

Motivation: Want to apply KGEMs to entities outside of training graph => we need to move from *transductive* to *inductive* models



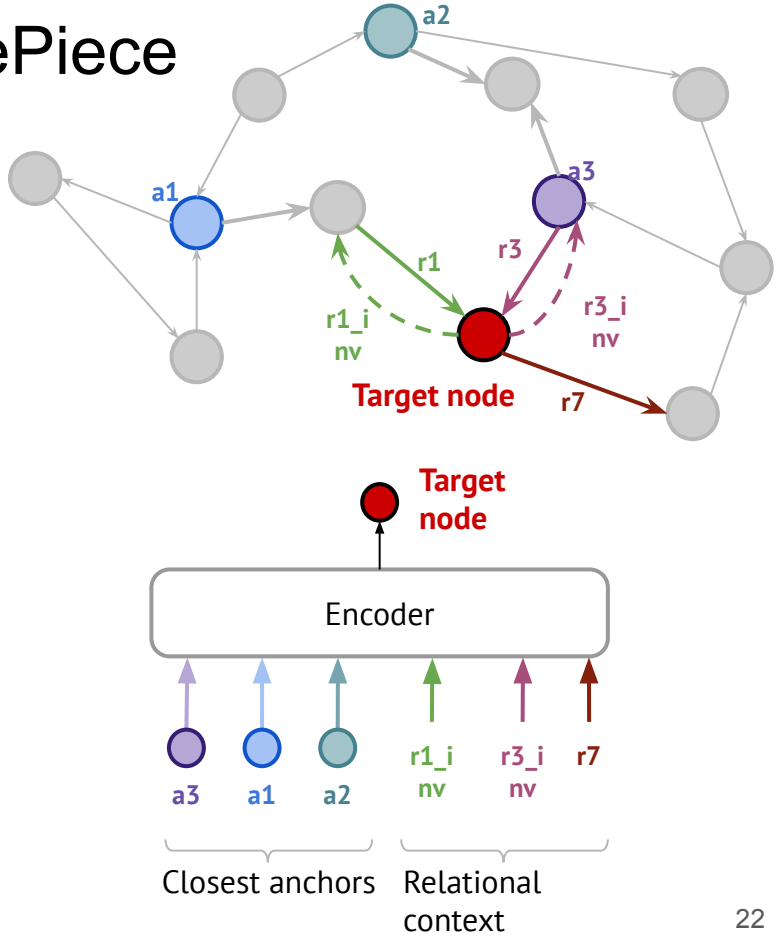
Inductive Link Prediction and NodePiece

NodePiece: moves beyond "shallow embeddings" and uses a combination of compositional strategies:

- Anchor or relation tokenization
- Degree, page rank, personal page rank searcher
- MLP, DeepSet, or arbitrary aggregator

- Inductive out-of-the-box - any new node can be tokenized with a fixed-size vocabulary

Implementation: NodePiece is now available in PyKEEN along with a generalized training and evaluation workflow for inductive datasets



Paper: https://arxiv.org/abs/2106.12144

Inductive Link Prediction Challenge

More Problems:

- New models can train under inductive link prediction scenario
- Meaningful benchmarks (especially for biomedicine) aren't available

Solution: We implemented an algorithm for generating inductive link prediction benchmarks and made two based on Wikidata

Future: apply to biomedical knowledge graphs (e.g., using chemical similarity for induction)

Model	MRR	H@100	H@10	H@5	H@3	H@1	AMRI
InductiveNodePieceGNN	0.0705	**0.374**	**0.1458**	**0.0990**	**0.0730**	0.0319	**0.682**
InductiveNodePiece	0.0651	0.287	0.1246	0.0809	0.0542	0.0373	0.646

Baseline Results for Comparison

Code: https://github.com/pykeen/ilpc2022, Dataset: https://zenodo.org/record/6321299, Paper: https://arxiv.org/abs/2203.01520

Language Models and KGE Models

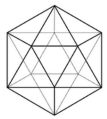
- PyKEEN wraps 🤗 transformers (e.g., BERT) as representations for two tasks:
  - Improve a language model with explicit structured knowledge from a KG
  - Improve a KGE model with implicit unstructured knowledge in a language model
- Simple method for enabling inductive link prediction using entity labels
- See more: https://pykeen.readthedocs.io/en/stable/tutorial/representations.html#label-based

```python
from pykeen.datasets import get_dataset
from pykeen.models import ERModel
from pykeen.nn.representation import (
    LabelBasedTransformerRepresentation,
)
from pykeen.pipeline import pipeline

dataset = get_dataset(dataset="Nations")
entity_representations = LabelBasedTransformerRepresentation.from_triples_factory(
    triples_factory=dataset.training,
)
result = pipeline(
    dataset=dataset,
    model=ERModel,
    model_kwargs=dict(
        interaction="ermlpe",
        interaction_kwargs=dict(
            embedding_dim=entity_representations.embedding_dim,
        ),
        entity_representations=entity_representations,
        relation_representations_kwargs=dict(
            shape=entity_representations.shape,
        ),
    ),
    training_kwargs=dict(
        num_epochs=1,
    ),
)
```

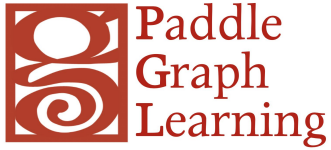
Alternate KGEM Software

Pykg2vec



STELLAR GRAPH

OpenKE

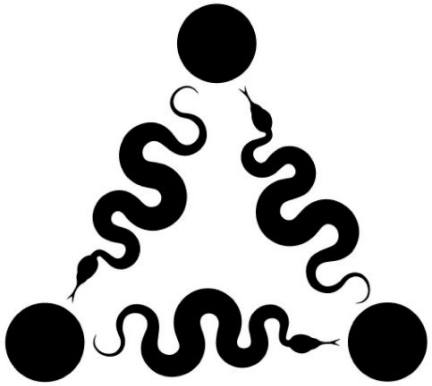


GraphVite

See our awesome list: https://github.com/pykeen/kgem-software-review

Wrapping Up

- Chat with us on the issue tracker for new features or support
- Join us! We can provide mentorship.
- Implement your next KGE model with PyKEEN
- Use PyKEEN in your downstream tasks



PyKEEN v1.8

Predictions for the People







33 Datasets **42** Models **13** Losses **5** Regularizers **2** Trainers **3** Evaluators **42** Metrics

 @keenuniverse  https://github.com/pykeen  https://pykeen.github.io

```
$ pip install pykeen
```