Uncertain Knowledge Embedding-unKR

- Conda安装及配置
- unKR介绍及使用

前置实验环境(选做，unKR也支持Windows系统)

- Linux操作系统
  - 推荐稳定版本的Ubuntu
    - 如：Ubuntu 20.04.3 LTS
    - 下载链接： https://ubuntu.com/download/desktop
  - 可以在虚拟机中配置
    - VMWare:下载地址：

      https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

      创建虚拟机时选择Minimal install
    - VMWare激活码(选择任意一个使用即可)：

      4A4RR-813DK-M81A9-4U35H-06KND

      NZ4RR-FTK5H-H81C1-Q30QH-1V2LA

      4C21U-2KK9Q-M8130-4V2QH-CF810

      4Y09U-AJK97-089Z0-A3054-83KLA

Conda简介

Conda是包及其依赖项和环境的管理工具

- 适用语言：Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN
- 适用平台：Windows, macOS, Linux

用途:

- 快速安装、运行和升级包及其依赖项。
- 在计算机中便捷地创建、保存、加载和切换环境。

Conda安装(Linux 环境下)

Anaconda下载安装

1.下载：$ wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2022.10-Linux-x86_64.sh

或 $ wget https://repo.anaconda.com/archive/Anaconda3-2022.10-Linux-x86_64.sh

2.修改安装包权限：chmod 777 Anaconda3-2022.10-Linux-x86_64.sh

3.安装 : $ ./Anaconda3-2022.10-Linux-x86_64.sh

Conda安装(Linux 环境下)

conda环境变量

安装结束会提示以下内容：

Do you wish the installer to initialize Anaconda3 by running

conda init?[yes|no]

1.输入yes，自动配置环境变量

Conda添加源

1. 添加源、删除源

conda config –add/remove channels xxxxx

2. 显示源

conda config ––show channels

Conda安装(Linux 环境下)

添加国内镜像源

1. 清华源

#添加镜像源

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/pro

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2

#显示检索路径

conda config --set show_channel_urls yes

#显示镜像通道

conda config --show channels

Conda安装(Windows 环境下)

- 从官网或国内镜像网站(https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/)选择适合的版本下载

- 找到Anaconda安装路径，Anaconda安装路径\Scripts以及Anaconda安装路径\Library\bin添加到环境变量
- 在命令行中输入conda –version后能看到版本号说明安装并配置成功

```
C:\Users\21140>conda -V
conda 23.3.1
```

Conda安装（通用）

利用conda创建python环境

1. 创建python3.8环境：　conda create -n unKR python=3.8

2. 激活环境：　conda activate unKR

不确定性知识图谱简介

- 不确定性知识图谱 (**U**ncertain **K**nowledge **G**raph, UKG) 可以被定义为一系列四元组的集合:$G = \{(h, r, t, s)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$，其中 $\mathcal{E}$ 和 $\mathcal{R}$ 分别为实体和关系的集合，$s \in [0,1]$ 为三元组 (h, r, t) 的置信度，表示三元组成立的概率

- 例如 (adults, capableof, drink_beer, 0.89) 是 ConceptNet 中的一个四元组，它表示 (adults, capableof, drink_beer) 成立的概率是0.89

unKR简介

- unKR是东南大学认知智能研究所(seucoin)基于PyTorch_Lightning开发的用于将不确定性知识图谱表示学习框架。 unKR是第一个用于不确定知识图谱表示学习的开源库，它是高度模块化和可扩展的，便于复现和定制不确定性知识图谱表示学习模型，unKR在统一的工作流程下成功重新实现了近年来发布的九个不确定性知识图谱表示学习模型，包括UKGE，PASSLEAF等。

unKR配置

下载地址：

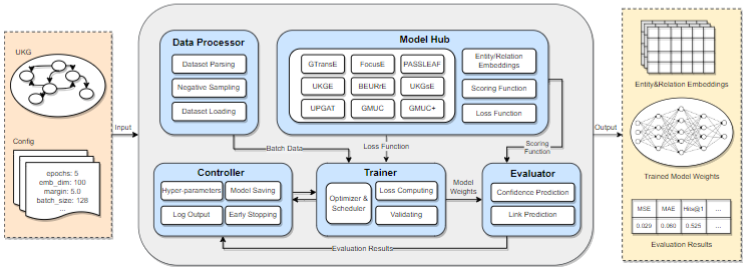
- 下载：https://github.com/seucoin/unKR
- 文档：https://seucoin.github.io/unKR/

unKR: A Python Library for Uncertain Knowledge Graph Reasoning by Representation Learning

pypi v1.0.1 | license Apache-2.0 | Doc online

unKR is an python library for uncertain Knowledge graph (UKG) Reasoning based on the PyTorch Lightning. It provides a unifying workflow to implement a variety of uncertain knowledge graph representation learning models to complete UKG reasoning. unKR consists of five modules: 1) Data Processor handles low-level dataset parsing and negative sampling, then generates mini-batches of data; 2) Model Hub implements the model algorithms, containing the scoring function and loss function; 3) Trainer conducts iterative training and validation; 4) Evaluator provides confidence prediction and link prediction tasks to evaluate models' performance; 5) Controller controls the training worklow, allowing for early stopping and model saving. These modules are decoupled and independent, making unKR highly modularized and extensible. Detailed documentation of the unKR is available at here.

unKR core development team will provide long-term technical support, and developers are welcome to discuss the work and initiate questions using `issue`.



unKR配置

配置流程

利用Anaconda创建虚拟环境并进入虚拟环境

conda create -n unKR python=3.8

conda activate unKR

下载unKR python库，我们提供了安装包，大家也可选择从github仓库下载：git clone https://github.com/seucoin/unKR.git

解压安装包后，从命令行进入解压后的unKR文件目录下，在conda环境下分别执行以下两行命令即可安装依赖环境：

pip install -r requirements.txt

python setup.py install

如果安装时出现pytorch lightning版本找不到的问题可能是pip版本过高导致的，可以尝试将pip版本降至24.0：
python -m pip install pip==24.0

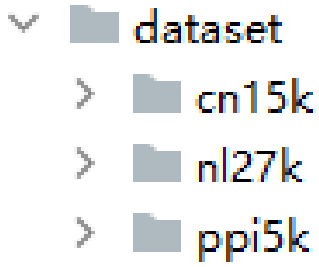
unKR简介

unKR框架结构

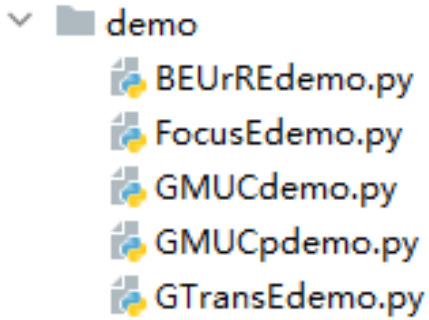
- unKR的核心代码都在src文件中，主要包含Data Model LitModel Evaluator四个模块，本实验重点在于使用工具而非模型详细原理，因此这部分只需了解即可
- **Data** 数据处理模块对输入的UKG数据集进行预处理，以便进行数据加载，主要包含UKGData、UKGSampler和UKGDataModule
- **Model** 模型模块实现了两种类型的UKG表示学习模型，即normal和few-shot模型，每个模型都被封装到一个类中，例如PASSLEAF模型的PASSLEAF类
- **LitModel** 模块用于指导模型的迭代训练和评估。每个模型对应于一个继承自BaseLitModel类的派生类，其中training_step函数用于根据Model Hub模块中定义的评分函数计算批量数据的损失；validation_step函数和test_step函数用于评估模型在验证集和测试集上的效果
- **Evaluator** unKR在eval_task中实现了置信度预测和链接预测这两个不确定性知识图谱补全的子任务

unKR文件组织结构

- **数据集** 数据集在dataset文件夹下，具体包括 cn15k, nl27k, ppi5k 三个数据集

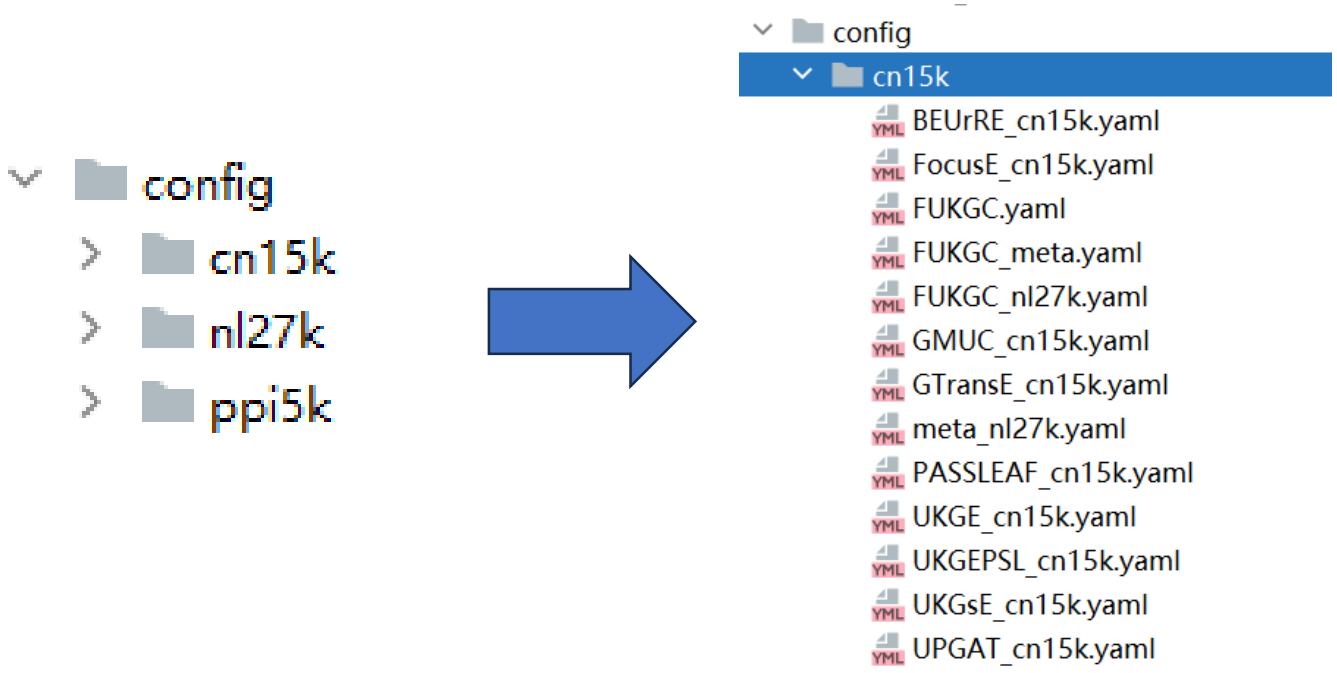


- **示例脚本** 示例脚本在demo文件夹下，提供了每个模型的训练流程编写示例



unKR文件组织结构

- **参数文件** 参数文件使用yaml，在config文件夹下，按数据集组织不同模型的对应参数设置



示例

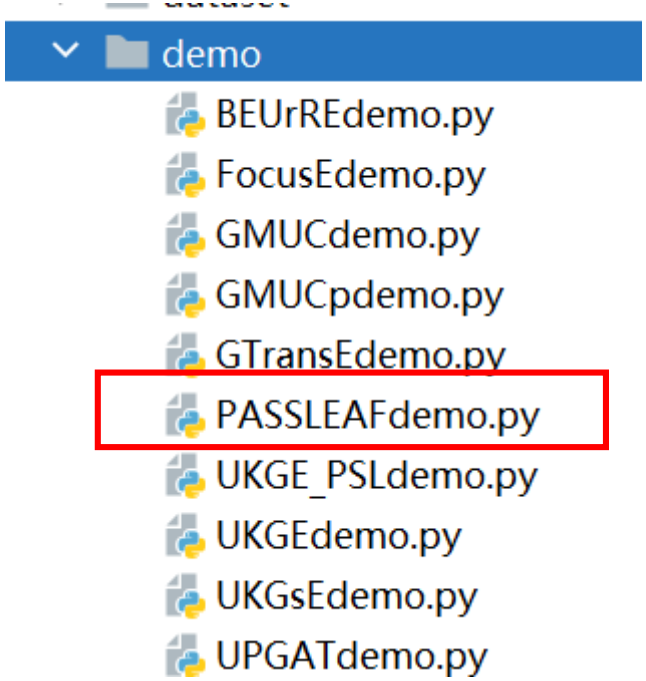
demo 模型参数示例

```yaml
check_val_every_n_epoch: 1
checkpoint_callback: null
checkpoint_dir: ''
cog_gamma: 0.5
cog_num: 0
cognitive: false
config_path: ''
confidence_filter: 0
data_class: KGDataModule
data_path: dataset/nl27k
dataset_name: nl27k
decoder_model: null
default_root_dir: null
detect_anomaly: false
deterministic: false
devices: null
dis_order: 2
dropout: 0
early_stop_patience: 10
emb_dim: 128
emb_shape: 20
enable_checkpointing: true
```

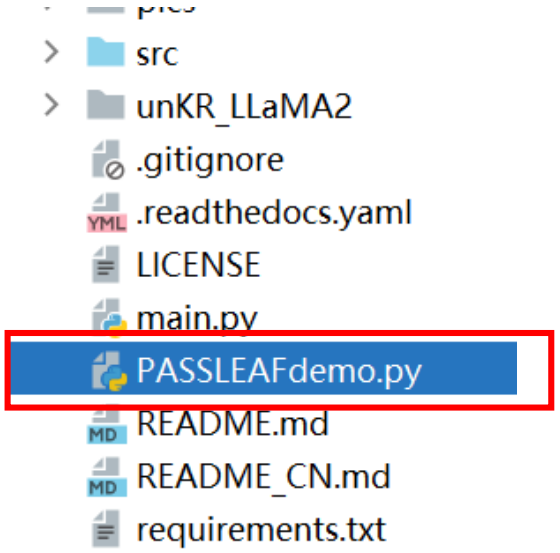
所有的参数都在yaml文件下设置
部分重要参数含义如下，在实验中需要根据实验的具体要求来调整:

data_path: 数据集的相对路径
dataset_name: 数据集的名称
model_name: 使用的模型
litmodel_name: 使用的模型的训练器
loss_name: 使用的损失函数
check_val_every_n_epoch: 每多少个epoch在验证集上测试并保存一次模型，注意这个值需要小于max_epochs
lr: 模型的学习率
max_epochs: 模型训练的最大epoch，最好是check_val_every_n_epoch的倍数
Train_bs: 训练时的batch size
Eval_bs: 评估时的batch size
emb_dim: 嵌入向量的维度
num_neg: 负采样的数据
gpu: 训练使用的gpu编号，如果不使用gpu则可以将该参数设置为cpu

示例-使用UKG表示学习模型的demo测试文件



1. 在demo文件夹下选择对应模型的示例脚本，将其放到与main.py平级的目录下（以PASSLEAF为例）



示例-使用UKG表示学习模型的demo测试文件

```
        path = model_checkpoint.best_model_path
    else:
        path = "./output/confidence_prediction/ppi5k/PASSLEAF
    lit_model.load_state_dict(torch.load(path)["state_dict"])
    lit_model.eval()
    trainer.test(lit_model, datamodule=kgdata)


if __name__ == "__main__":
    main(arg_path='config/ppi5k/PASSLEAFdemo.yaml')
```

2. 打开PASSLEAFdemo.py文件，在文件的末尾设置参数文件路径

◈示例-使用UKG表示学习模型的demo测试文件

3. 完成上文的步骤后，在unKR主目录下输入python PASSLEAFdemo.py运行对应的python文件即可（使用pycharm或者vscode配置好conda环境的同学直接点击运行按钮也可以），出现如下的输出说明运行正常

```
(unKR) D:\change\unKR>python PASSLEAFdemo.py
0         Non-trainable params
2.6 M     Total params
10.252    Total estimated model params size (MB)
D:\Anaconda\envs\unKR\lib\site-packages\pytorch_lightning\callbacks\model_checkpoint.py:631: UserWarning: Checkpoint directory output/co
  rank_zero_warn(f"Checkpoint directory {dirpath} exists and is not empty.")
Validation sanity check: 0it [00:00, ?it/s]D:\Anaconda\envs\unKR\lib\site-packages\pytorch_lightning\trainer\data_loading.py:132: UserWa
onsider increasing the value of the `num_workers` argument` (try 20 which is the number of cpus on this machine) in the `DataLoader` ini
  rank_zero_warn(
Global seed set to 321
D:\Anaconda\envs\unKR\lib\site-packages\pytorch_lightning\trainer\data_loading.py:132: UserWarning: The dataloader, train_dataloader, do
rkers` argument` (try 20 which is the number of cpus on this machine) in the `DataLoader` init to improve performance.
  rank_zero_warn(
Epoch 0:   0%|
:\Anaconda\envs\unKR\lib\site-packages\pytorch_lightning\utilities\data.py:59: UserWarning: Trying to infer the `batch_size` from an amb
., batch_size=batch_size)`.
  warning_cache.warn(
Epoch 1:  12%|███                       | 199/1640 [00:19<02:21, 10.20it/s, loss=146, v_num=2, Eval_MAE=0.192, Eval_MSE=0.0507, E
```

请注意：以上流程只是直接使用unKR里给定的示例demo和yaml参数文件运行，但是实际在使用时需要根据具体的需求调整选择对应的demo文件与yaml文件，并调整yaml文件里的参数

示例-使用UKG表示学习模型的demo测试文件

示例运行完成后，实验结果应当按照类似如下的结构组织


```
{'Test_MAE': 0.18434199690818787,
 'Test_MSE': 0.04802199825644493,
 'Test_hits@1': 0.3183360099792480 5,
 'Test_hits@10': 0.5357279777526855,
 'Test_hits@3': 0.4529300034046173,
 'Test_mr': 509.4578552246094,
 'Test_mrr': 0.400411993265152,
 'Test_raw_hits@1': 0.01625699922442436 2,
 'Test_raw_hits@10': 0.1258980035781860 4,
 'Test_raw_hits@3': 0.03894099965691566 5,
 'Test_raw_mr': 527.4779052734375,
 'Test_raw_mrr': 0.05987099930644035,
 'Test_raw_wmr': 524.3674926757812,
 'Test_raw_wmrr': 0.06014100089669227 6,
 'Test_wmr': 506.4567565917969,
 'Test_wmrr': 0.4064109921455383 3}
```

示例

输出测试结果

置信度预测：置信度预测是不确定性知识图谱补全的重要任务之一，置信度预测任务旨在预测三元组的置信度，使其尽量与真实值接近。我们通常使用均方误差Mean Squared Error (MSE)和平均绝对误差Mean Absolute Error (MAE) 来评估置信度预测的结果。我们将假设共有 N 个样本，其中第 i 个样本的真实置信度定义为 s_i，模型的预测置信度为 $\hat{s}_i$，MAE与MSE的公式为：

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|s_i - \hat{s}_i|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(s_i - \hat{s}_i)^2$$


```
{'Test_MAE': 0.18434199690818787,
 'Test_MSE': 0.04802199825644493,
 'Test_hits@1': 0.31833600997924805,
 'Test_hits@10': 0.5357279777526855,
 'Test_hits@3': 0.4529300034046173,
 'Test_mr': 509.4578552246094,
 'Test_mrr': 0.400411993265152,
```

示例

输出测试结果

链接预测：链接预测是不确定性知识图谱补全的另一个重要任务，它旨在基于给定的头实体和关系，预测其对应的尾实体。链接预测的衡量指标主要包含 Hit@K, MR 与 MRR。WMR和WMRR分别是MR与MRR的置信度加权形式，能更好地在评估时反映不确定性知识图谱的置信度信息。

Hit@ K： Hit@K 将知识图谱数据层事实性知识补全和知识图谱本体补全视为一个排序任务，即给定知识图谱三元组的任意两部分预测剩余的部分，例如给定三元组的主语和谓语预测三元组的宾语。

当预测出所有缺失部分时，按照相关性对预测结果进行排序并取前N个结果。假定所有需要预测的三元组个数为 K，而所有预测结果中正确结果在前N个结果的个数为N_j个，则计算Hit@N 如公式下所示：

$$Hits@K = \frac{N_j}{N}$$

```
{'Test_MAE': 0.184341996908187 87,
 'Test_MSE': 0.048021998256444 93,
 'Test_hits@1': 0.318336009979248 05,
 'Test_hits@10': 0.535727977752685 5,
 'Test_hits@3': 0.452930003404617 3,
 'Test_mr': 509.457855224609 4,
 'Test_mrr': 0.400411993265152,
```

示例

输出测试结果

MRR（Mean Reciprocal Rank）：首先计算每一个预测三元组中预测结果的位置排名得分 rankj如公式下所示：

$$rank_j = \frac{1}{R_j}$$

其中 R_j是预测正确结果的排名，然后针对所有N个三元组计算 MRR及其加权形式WMRR 如公式如下：

$$MRR = \frac{1}{N}\sum_{j=1}^{N} rank_j$$

$$WMRR = \frac{1}{N}\sum_{j=1}^{N} s_j rank_j$$

其中s_j是第j个三元组的置信度

```
{'Test_MAE': 0.18434199690818787,
 'Test_MSE': 0.048021998256444493,
 'Test_hits@1': 0.31833600997924805,
 'Test_hits@10': 0.5357279777526855,
 'Test_hits@3': 0.4529300034046173,
 'Test_mr': 509.4578552246094,
 'Test_mrr': 0.400411993265152,
```

```
'Test_wmrr': 0.4064109921455383}
```

示例

输出测试结果

- MR（Mean Rank）：与 MRR 类似，MR 也是评估整体排序质量的指标，MR 的计算公式如下所示：

其中 R_j 是预测三元组正确结果的排名，针对所有N个三元组计算 MR 及其加权形式 WMR 如公式如下：

$$MR = \frac{1}{N}\sum_{j=1}^{N} R_j$$

$$WMR = \frac{1}{N}\sum_{j=1}^{N} s_j R_j$$

其中s_j是第j个三元组的置信度


```
{'Test_MAE': 0.18434199690818787,
 'Test_MSE': 0.04802199825644493,
 'Test_hits@1': 0.31833600997924805,
 'Test_hits@10': 0.5357279777526855,
 'Test_hits@3': 0.4529300034046173,
 'Test_mr': 509.4578552246094,
 'Test_mrr': 0.400411993265152,
```

```
'Test_wmr': 506.4567565917969
```

课堂作业

- 阅读unKR 样例程序，了解数据集结构及模型参数意义，分别在ppi5k和nl27k两个数据集上训练满足以下条件的 UKGE 模型，并获取测试性能，要求：
  - 嵌入向量的维度为256
  - 最大epoch为10
- 其他参数可自行调整（注：所有的参数都可以通过对应的yaml文件设置）
- 将yaml文件里所进行的参数设置操作，以及最终的实验结果截图，形成实验报告。操作步骤需要体现出使用yaml文件具体对哪些参数进行了设置，最终的实验结果需要包含MAE，MSE，MRR，MR，WMRR，WMR以及Hits@N的各项指标。