|  |  |
| --- | --- |
| 文件名 | 说明 |
| triples\_1 | 70414个由ids编码的中文关系三元组  中文实体的名称 关系 中文实体的名称，例：   1. 819 8738   根据ent\_ids\_1：  0→http://zh.dbpedia.org/resource/阿卜杜拉·居尔  8738→http://zh.dbpedia.org/resource/雷杰普·塔伊普·埃尔多安  根据 |
| triples\_2 | 95142个由ids编码的英文关系三元组  英文实体的名称 关系 英文实体的名称，例：  34460 2133 11719 |
| id\_features\_1 | features\_zh经过谷歌翻译后的实体名称  1 Kim Dae Jung |
| id\_features\_2 | 10500 Abdullah Gül |
| features\_zh | 1 Jin Dazhong |
| features\_en | 10500 Abdullah Gül |
| entity2id\_zh | 中文实体→id，例：  http://zh.dbpedia.org/resource/美国 8407 |
| entity2id\_en | 英文实体→id，例：  http://dbpedia.org/resource/United\_States 2858 |
| id2entity\_zh | id→中文实体，例：  8407 美国 |
| id2entity\_en | id→英文实体，例：  2858 United\_States |
| ent\_ids\_1 | 中文实体的名称→id，例：  0 http://zh.dbpedia.org/resource/阿卜杜拉·居尔 |
| ent\_ids\_2 | 英文实体的名称→id，例：  10500 http://dbpedia.org/resource/Abdullah\_Gül |
| ent\_name2id | 由ent\_ids\_1与ent\_ids\_2生成的，是一个字典{实体名称：id} |
| ref\_ent\_ids | 实体 |
| train.ref | 训练用的 |
| test.ref | 测试用的 |
| entity\_seeds | 15000对种子实体，例：   1. 2858   根据id2entity\_zh：8407→美国  根据id2entity\_en：2858→United\_States |
| train\_entity\_seeds | 训练用的9000对种子实体 |
| test\_entity\_seeds | 测试用的4500对种子实体 |
| valid\_entity\_seeds | 验证用的1500对种子实体 |
|  |  |
|  |  |

EASY中的DBP15K数据集来自于GMNN和AttrGNN

1. 图神经网络库PyTorch geometric
2. 构建图

图包括节点和边

torch\_geometric.data.Data用于构建图，它包括5个属性，每一个属性都不是必须的，可以为空。

·x：用于存储每个节点的特征，形状是[num\_nodes, num\_node\_features]

·edge\_index：用于存储节点之间的边，形状是 [2, num\_edges]

·pos：存储节点的坐标，形状是[num\_nodes, num\_dimensions]

·y：存储样本标签。如果是每个节点都有标签，那么形状是[num\_nodes, \*]；如果是整张图只有一个标签，那么形状是[1, \*]

·edge\_attr：存储边的特征。形状是[num\_edges, num\_edge\_features]

在EASY中，process\_graph的参数如下：

二、EASY开源代码

位置：dbp15k.py

CLASS DBP15k(InMemoryDataset)：

def process\_graph(self, triple\_path, feature\_path):

#读取关系三元组，生成张量

g1 = read\_txt\_array(triple\_path, sep='\t', dtype=torch.long)

#切分为三部分：实体subj 关系 实体obj

subj, rel, obj = g1.t()

#建立字典{实体名称id：实体名称}

name\_dict = {}

with open(feature\_path, 'r', encoding = 'utf-8') as f:

for line in f:

info = line.strip().split('\t')

info = info if len(info) == 2 else info + ['']

#去除标点符号

seq\_str = remove\_punc(info[1]).strip()

if seq\_str == "":

seq\_str = '<unk>'

name\_dict[int(info[0])] = seq\_str

idx = torch.tensor(list(name\_dict.keys()))

assoc = torch.full((idx.max().item() + 1,), -1, dtype=torch.long)

assoc[idx] = torch.arange(idx.size(0))

subj, obj = assoc[subj], assoc[obj]

edge\_index = torch.stack([subj, obj], dim=0)

edge\_index, rel = sort\_edge\_index(edge\_index, rel)

# xs = [None for \_ in range(idx.size(0))]

names = [None for \_ in range(idx.size(0))]

for i in name\_dict.keys():

names[assoc[i]] = name\_dict[i]

# x = torch.nn.utils.rnn.pad\_sequence(xs, batch\_first=True)

return edge\_index, rel, assoc, names

参数：

·triple\_path：即triples\_1、triples\_2文件路径

·feature\_path：即id\_features\_1、id\_features\_2文件路径

返回值：

·edge\_index：排序后的边的id

·rel：排序后的关系的id

·assoc：实体名称张量，索引为实体名称id

·names：实体名称的id

位置：eval.py

#评估NEAP结果

@torch.no\_grad()

def evaluate\_sim\_matrix(link, sim\_x2y, sim\_y2x=None, ignore=(None, None), start="\t", no\_csls=True):

"""

参数：

link(Tensor)

sim\_x2y

sim\_y2x

Ignore

start

no\_csls

"""

print("start evaluate\_sim\_matrix")

start\_outer = start

start = start + start

device = link.device

sim\_x2y = sim\_x2y.to(device)

if sim\_x2y.is\_sparse:

sim\_x2y = sim\_x2y.to\_dense()

MRR = 'MRR'

match\_sim0, match\_id0, sim\_matrix0 = get\_topk\_sim(sim\_x2y)

result = get\_hit\_k(match\_id0, link, 0, ignore=ignore[0], start=start)

result[MRR] = get\_mrr(link, sim\_matrix0, 0, start=start)

if sim\_y2x is not None:

sim\_y2x = sim\_y2x.to(device)

if sim\_y2x.is\_sparse:

sim\_y2x = sim\_y2x.to\_dense()

match\_sim1, match\_id1, sim\_matrix1 = get\_topk\_sim(sim\_y2x)

result\_rev = get\_hit\_k(match\_id1, link, 1, ignore=ignore[1], start=start)

result\_rev[MRR] = get\_mrr(link, sim\_matrix1, 1, start=start)

if no\_csls:

return result, result\_rev

print(start\_outer + '------csls')

match\_sim0, match\_id0, sim\_matrix0 = get\_csls\_sim(sim\_matrix0, match\_sim0, match\_sim1)

match\_sim1, match\_id1, sim\_matrix1 = get\_csls\_sim(sim\_matrix1, match\_sim1, match\_sim0)

result\_csls\_0 = get\_hit\_k(match\_id0, link, 0, ignore=ignore[0], start=start)

result\_csls\_0[MRR] = get\_mrr(link, sim\_matrix0, 0, start=start)

result\_csls\_1 = get\_hit\_k(match\_id1, link, 1, ignore=ignore[1], start=start)

result\_csls\_1[MRR] = get\_mrr(link, sim\_matrix1, 1, start=start)

return result, result\_rev, result\_csls\_0, result\_csls\_1

else:

return result