三元组分析工具triplets_analysis_tool

triplets_analysis_tool是一个三元组<h, r, t>类型的知识图谱数据集分析工具，实现了对**三元组的统计**、**数据集分割**、**实体类型恢复并导入Neo4j数据库**以及**从Neo4j数据库中导出三元组数据集文件**。

内容列表

背景

目前知识图谱的大部分开源数据集都是以**三元组<h, r, t>形式**提供的，常见的例如FB15k、wordnet、YAGO3-10等，诚然他们方便导入模型训练embedding，但难以直观估计其规模和稀疏程度。而将三元组导入图数据库进行可视化又缺乏实体的类型标注。另外，在使用存放于图数据库的私有知识图谱构建三元组类型的数据集时，首先需要将图数据库转化为三元组数据集，在embedding任务上，分割训练集、验证集和测试集时往往要保证所有实体和关系在训练集中至少出现一次。

基于存在的上述问题，triplets_analysis_tool比较方便的实现了下面的功能：

1. 对一个或多个三元组<h, r, t>数据集文件**进行统计**，以了解数据集规模，**统计结果输出至csv文件保存**；
2. 对一个或多个三元组<h, r, t>数据集文件中的三元组按比例**重新分割成训练集、验证集和测试集**，同时可选择保证分割完成的**训练集中出现了数据集中所有实体和关系**；
3. 对一个或多个三元组<h, r, t>数据集文件中的**三元组的实体类型进行预测**，**导入Neo4j数据库**，以方便schema视图的查看；
4. 将Neo4j数据库中的**知识图谱导出为三元组**<h, r, t>格式的文本数据集文件。

使用说明

所有功能均在`triplets_analysis.py`中实现，可以在该文件中调用相关类和函数或者在自己的程序中import该文件，完成相关工作。

```
import triplets_analysis
```

所需依赖

当操作涉及Neo4j数据库的时候，请安装`py2neo`库。

一种安装方法是在Terminal中，键入以下代码：

```
pip install py2neo
```

实例化

对**三元组数据集**进行操作，需要**实例化**`triplets_analysis`，其参数为数据集的路径(list类型)。

```
import triplets_analysis

addr = ['train.txt', 'valid.txt', 'test.txt']
triplets = triplets_analysis(addr)
```

对**Neo4j数据库**的操作，需要**实例化**`Neo4j_analysis`，参数为数据库的ip地址、用户名、密码。

```
import triplets_analysis

neo4j = Neo4j_analysis("http://127.0.0.1//:7474", 'testGraph', '123456')
```

三元组统计

```
triplets.save_result_to_CSV()
# triplets.save_result_to_CSV(savepath='./')
```

参数`savepath`是保存路径，默认为调用文件的根目录。结果将输出至`analysis_result.csv`中。

本方法将统计各个文件和汇总文件中的头实体数量`head_number`、关系类型数量`relation_type_number`、尾实体数量`tail_number`、实体种类数`entity_number`、三元组条数`triplet_number`，并生成csv结果。

例如对FB15k的分析为：

name	head_number	relation_type_number	tail_number	entity_number	triplet_number
train.txt	13781	237	13379	14505	272115
valid.txt	7652	223	5804	9809	17535
test.txt	8171	224	6376	10348	20466
sum	13891	237	13504	14541	310116

数据集分割

```
triplets.split_dataset(ratio='10:1:1', savepath='./',
conditional_random=True)
# triplets.split_dataset(ratio='-:3000:3000')
```

参数`ratio`是分割比例字符串，两个英文":"分隔，依次为**训练集、验证集、测试集**。如果不需要验证集或测试集，可使**对应的数据为0或空**。如果某一数据为"-"，表示非"-"的数据为对应数据集的条目数，**"-"所对应的数据集会根据总数据数量减去已设置的数据计算得到**。例如在总三元组数量为20000的数据集中，`ratio='-:3000:3000'`表示分割后训练集数量为14000条，验证集3000条，测试集3000条。

参数`savepath`是保存路径，默认为调用文件的根目录。结果将输出至`train_new.csv`、`valid_new.csv`和`test_new.csv`中。

参数`conditional_random`决定随机分割**是否考虑保证全部实体和关系在训练集中都出现**，以方便embedding工作的使用，默认为`True`。**注意**，当`conditional_random=True`时，该算法随着数据集数量的增加，花费时间将增加。而且该算法可能出现求不出最好结果的情况，**如遇到分割失败，可以尝试重新分割，或修改**`ratio`。

三元组导入Neo4j

```
triplets.triplets_to_Neo4j(ip="http://127.0.0.1//:7474",
username='testGraph', password='123456', deleteAll=False)
# triplets.triplets_to_Neo4j(ip="http://127.0.0.1//:7474",
username='testGraph', password='123456')
```

参数`ip`是Neo4j的访问地址，`port`为`HTTP port`。

参数`username`、`password`分别是Neo4j的用户名和密码。

参数`deleteAll`决定在写入Neo4j前是否将Neo4j中已存在的数据清除，默认为False。

该方法将三元组根据关系进行了实体分类，主要依据是同一类关系两端连接的是特定类型的实体。该方法通过一种特别设计的`union-find`算法实现，**在数据量特别大时，求解时间可能较长**，请耐心等待。

导入Neo4j的过程根据知识图谱规模用时可能较长。**三元组<h, r, t>中**`头实体h`**和**`尾实体t`**作为实体导入，其label为上一步求解得到的实体类型，**`关系r`**作为关系导入。**

Neo4j导出三元组

```
neo4j.neo4j_to_triplets(addr='./triplets.csv', delimiter='\t')
# neo4j.neo4j_to_triplets(addr='./triplets.csv')
```

参数`addr`为保存路径。

参数`delimiter`为分隔符，默认是Tab制表符"\t"。

示例

请参考工具包。

```
addr = ['train.txt', 'valid.txt', 'test.txt']
# 实例化
triplets = triplets_analysis(addr)
# 三元组统计
triplets.save_result_to_CSV(savepath='./')
# 数据集分割
triplets.split_dataset(ratio='10:1:1', savepath='./',
conditional_random=True)
# 三元组导入Neo4j
```

```
triplets.triplets_to_Neo4j("http://127.0.0.1//:7474", 'testGraph', '123456', deleteAll=True)

# 实例化
neo4j = Neo4j_analysis("http://127.0.0.1//:7474", 'testGraph', '123456')
# Neo4j导出三元组
neo4j.neo4j_to_triplets('./triplets.csv')
```

维护者

@sguangxuan。