Knowledge Alignment

需要手动运行的命令均已用[]标识

一、OpenEA配置

项目地址：https://github.com/nju-websoft/OpenEA

环境要求

- 使用上节课的Ubuntu虚拟机
- 安装好anaconda
- 设置好了.condarc文件

获取源码和数据集

- **源码获取：**

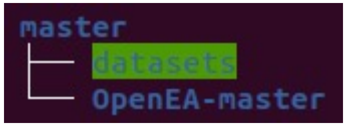
使用wget命令：[wget https://github.com/nju-websoft/OpenEA/archive/refs/heads/master.zip]

获取源码压缩包master.zip后解压得到master文件夹

- **数据集获取：**

将数据集datasets解压后放到master文件夹内

目录结构：



```
master
├─ datasets
└─ OpenEA-master
```

环境配置

- 创建环境

```
[conda create -n openea python=3.6]
```

*:python版本务必保证一致，影响tensorflow的安装

- 激活环境

```
[conda activate openea]
```

- 安装依赖包

```
[conda install tensorflow==1.12]
```

*:tensorflow务必保证版本一致

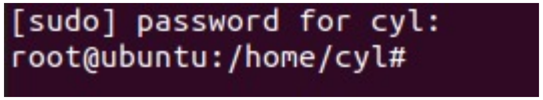
环境配置

- 安装依赖包

用[gcc -v]命令测试下gcc是否已安装，若显示未找到命令使用[sudo apt-get install gcc]安装

*：提示输入sudo密码，直接敲密码后按回车，终端不会显示密码

示例：



```
[sudo] password for cyl:
root@ubuntu:/home/cyl#
```

使用[cd master/OpenEA-master]进入master/OpenEA-master目录

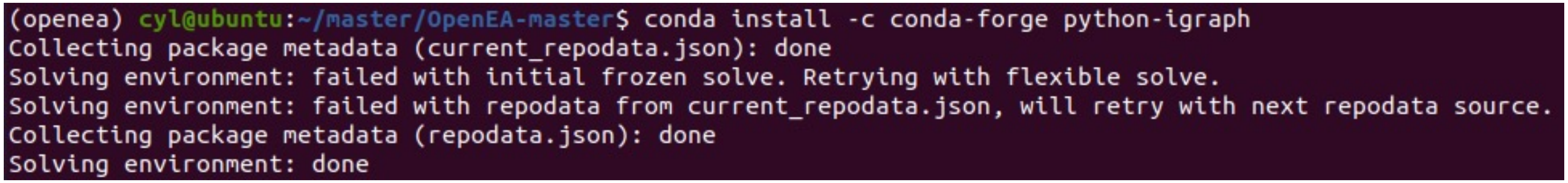
运行[pip install -e .]安装requirement.txt中的包

环境配置

- 安装依赖包

运行[conda install -c conda-forge python-igraph]

*：会提示几次failed，等待一会就好



```
(openea) cyl@ubuntu:~/master/OpenEA-master$ conda install -c conda-forge python-igraph
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done
```

运行测试

首先[cd run]进入python文件目录，接下来运行命令即可

运行的命令格式如下：

```
python main_from_args.py" "predefined_arguments"
"dataset_name" "split"
```

"predefined_arguments"表示预先定义的训练参数

"dataset_name"表示使用的数据集

"split"表示使用数据集的第几块来训练（模型使用K折交叉验证）

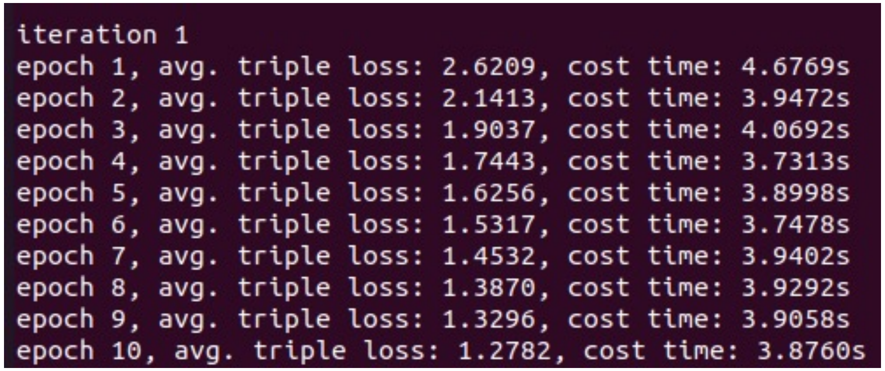
运行测试

输入

[python main_from_args.py ./args/bootea_args_15K.json D_W_15K_V1 721_5fold/1/]

来测试下是否可以正常运行，该命令表示在D-W-15K数据集的第一块上训练BootEA

出现下述内容即可，报错警告不用理



```
iteration 1
epoch 1, avg. triple loss: 2.6209, cost time: 4.6769s
epoch 2, avg. triple loss: 2.1413, cost time: 3.9472s
epoch 3, avg. triple loss: 1.9037, cost time: 4.0692s
epoch 4, avg. triple loss: 1.7443, cost time: 3.7313s
epoch 5, avg. triple loss: 1.6256, cost time: 3.8998s
epoch 6, avg. triple loss: 1.5317, cost time: 3.7478s
epoch 7, avg. triple loss: 1.4532, cost time: 3.9402s
epoch 8, avg. triple loss: 1.3870, cost time: 3.9292s
epoch 9, avg. triple loss: 1.3296, cost time: 3.9058s
epoch 10, avg. triple loss: 1.2782, cost time: 3.8760s
```

关于命令参数的进一步解释

```
python main_from_args.py ./args/bootea_args_15K.json
D_W_15K_V1 721_5fold/1/
```

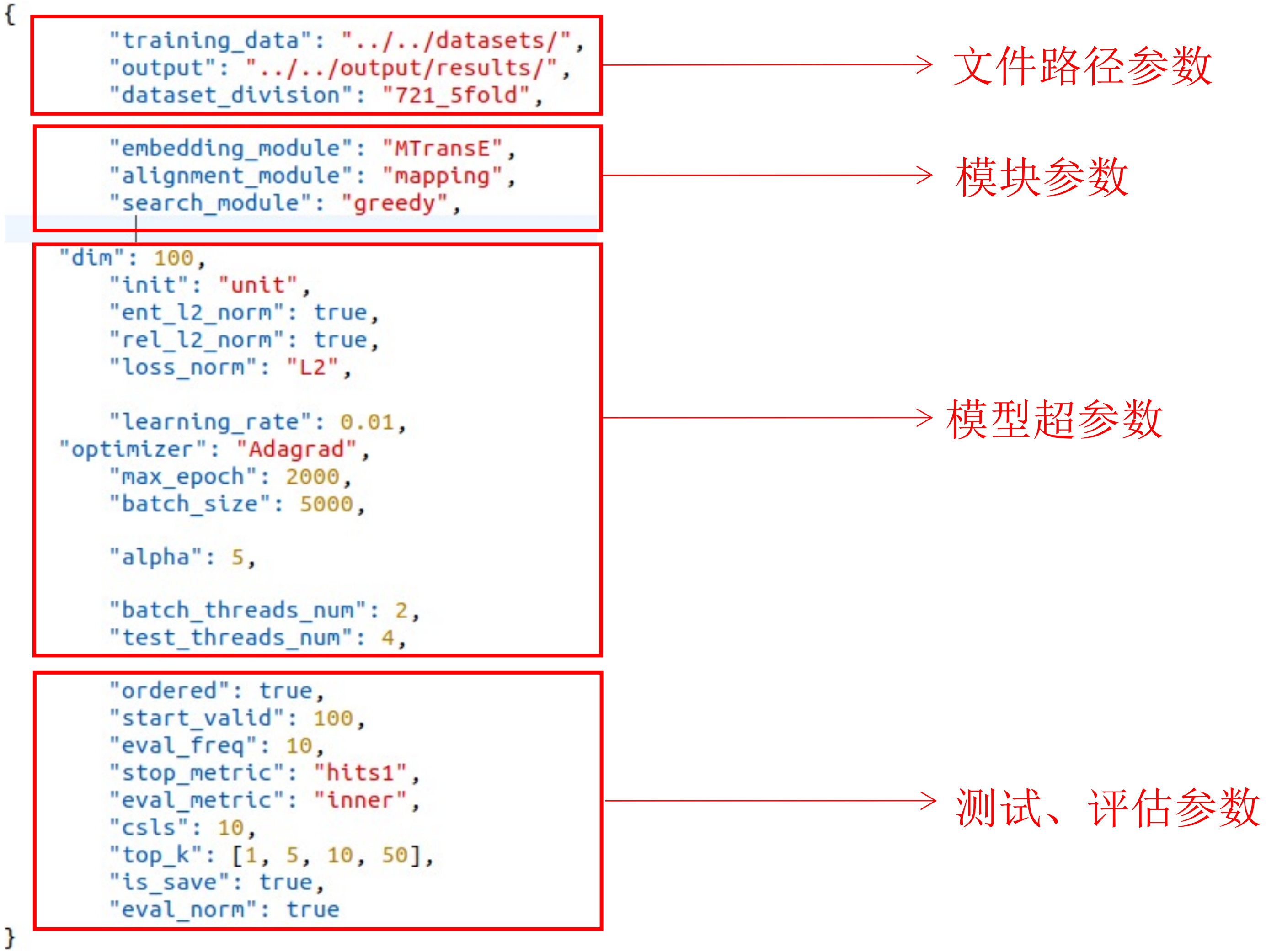
main_from_args.py表示要运行的python文件，也就是训练和测试的主要代码

./args/bootea_args_15K.json则是main_from_args.py运行时使用到的模型参数，这里是bootea在15K规模数据集上的参数

D_W_15K_V1 721_5fold/1/则表示具体的训练集

模型参数

以mtranse_args_15K.json为例



```
{
        "training_data": "../../datasets/",
        "output": "../../output/results/",
        "dataset_division": "721_5fold",

        "embedding_module": "MTransE",
        "alignment_module": "mapping",
        "search_module": "greedy",

    "dim": 100,
        "init": "unit",
        "ent_l2_norm": true,
        "rel_l2_norm": true,
        "loss_norm": "L2",

        "learning_rate": 0.01,
    "optimizer": "Adagrad",
        "max_epoch": 2000,
        "batch_size": 5000,

        "alpha": 5,

        "batch_threads_num": 2,
        "test_threads_num": 4,

        "ordered": true,
        "start_valid": 100,
        "eval_freq": 10,
        "stop_metric": "hits1",
        "eval_metric": "inner",
        "csls": 10,
        "top_k": [1, 5, 10, 50],
        "is_save": true,
        "eval_norm": true
}
```

文件路径参数

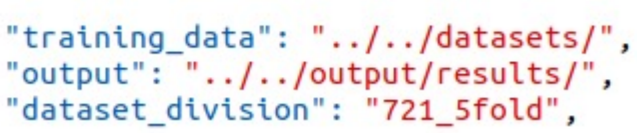
模块参数

模型超参数

测试、评估参数

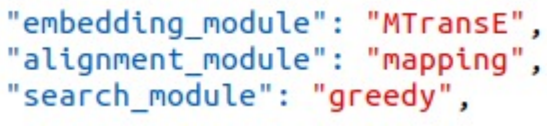
模型参数

对一些关键参数的解释



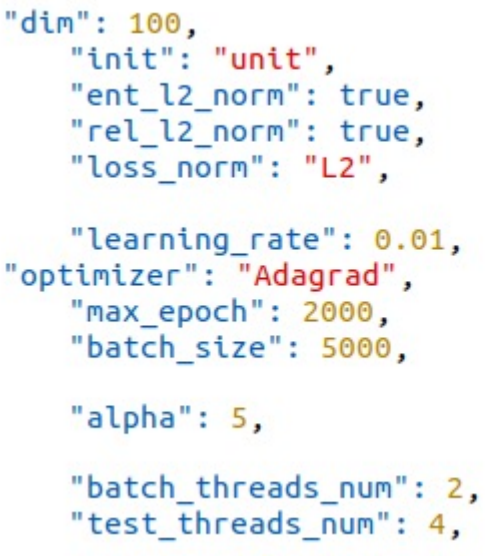
```
"training_data": "../../datasets/",
"output": "../../output/results/",
"dataset_division": "721_5fold",
```

- 数据集路径
- 输出结果路径
- 数据集划分方式



```
"embedding_module": "MTransE",
"alignment_module": "mapping",
"search_module": "greedy",
```

- embedding模式
- 对齐模式
- 搜索（不同语言中相同实体或关系）模式



```
"dim": 100,
    "init": "unit",
    "ent_l2_norm": true,
    "rel_l2_norm": true,
    "loss_norm": "L2",

    "learning_rate": 0.01,
"optimizer": "Adagrad",
    "max_epoch": 2000,
    "batch_size": 5000,

    "alpha": 5,

    "batch_threads_num": 2,
    "test_threads_num": 4,
```

- embedding维度

- 实体L2范数
- 关系L2范数
- 损失函数正则化方式

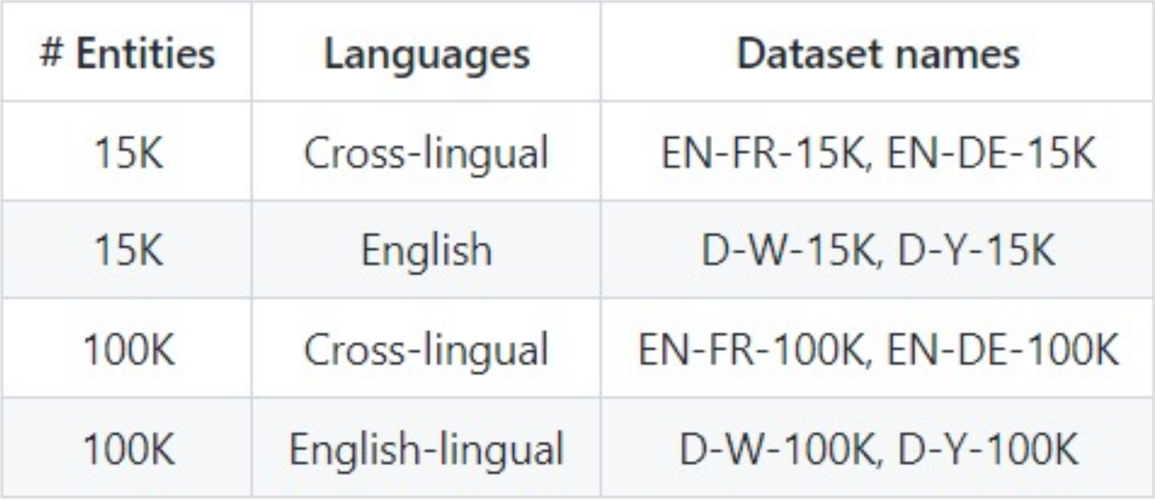
- 学习率
- 优化器
- 最大训练代数
- 单批训练数据量

- 一些线程数

数据集说明

数据来自于DBpedia，Wikidata，YAGO3

数据集名称及对应实体数和语言：



# Entities	Languages	Dataset names
15K	Cross-lingual	EN-FR-15K, EN-DE-15K
15K	English	D-W-15K, D-Y-15K
100K	Cross-lingual	EN-FR-100K, EN-DE-100K
100K	English-lingual	D-W-100K, D-Y-100K

数据集说明

目录结构(EN_FR_15K_V1)：

```
EN_FR_15K_V1/
├── attr_triples_1: attribute triples in KG1
├── attr_triples_2: attribute triples in KG2
├── rel_triples_1: relation triples in KG1
├── rel_triples_2: relation triples in KG2
├── ent_links: entity alignment between KG1 and KG2
├── 721_5fold/: entity alignment with test/train/valid (7:2:1) splits
│   ├── 1/: the first fold
│   │   ├── test_links
│   │   ├── train_links
│   │   └── valid_links
│   ├── 2/
│   ├── 3/
│   ├── 4/
│   ├── 5/
```

论文中的实验数据

Var_n表示模型的一些变种，主要是损失函数的改变，结合上课教案或论文Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment理解其含义

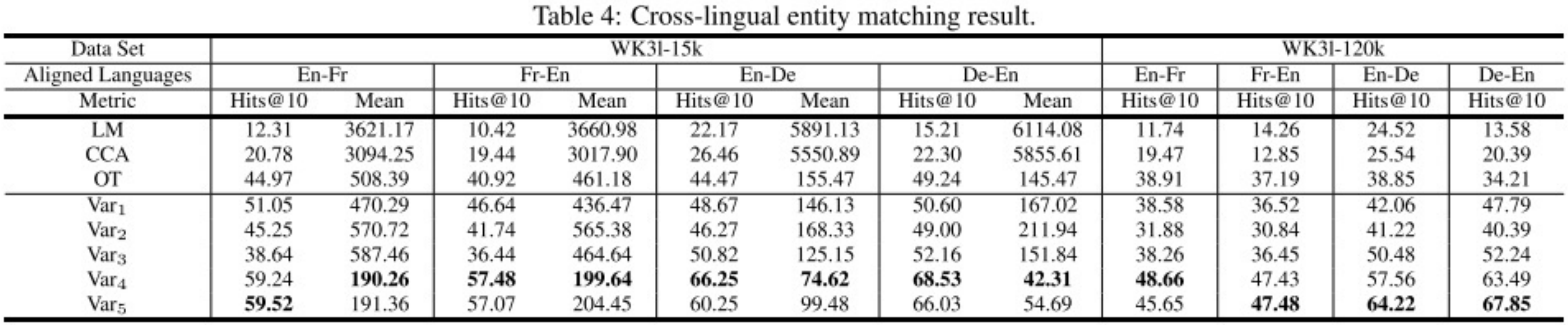


Table 4: Cross-lingual entity matching result.

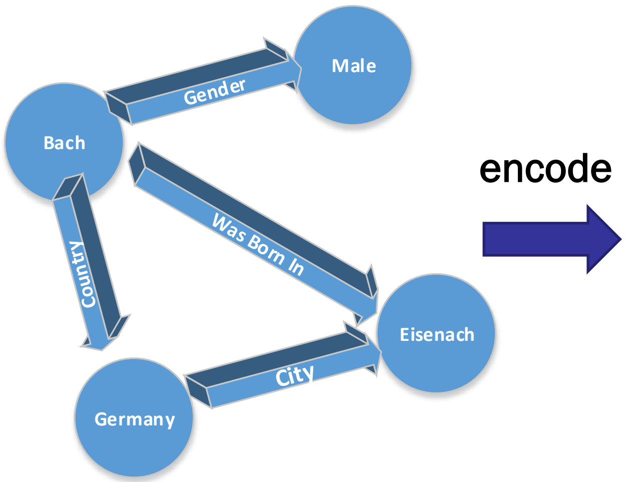
Data Set	WK3l-15k								WK3l-120k			
Aligned Languages	En-Fr		Fr-En		En-De		De-En		En-Fr	Fr-En	En-De	De-En
Metric	Hits@10	Mean	Hits@10	Mean	Hits@10	Mean	Hits@10	Mean	Hits@10	Hits@10	Hits@10	Hits@10
LM	12.31	3621.17	10.42	3660.98	22.17	5891.13	15.21	6114.08	11.74	14.26	24.52	13.58
CCA	20.78	3094.25	19.44	3017.90	26.46	5550.89	22.30	5855.61	19.47	12.85	25.54	20.39
OT	44.97	508.39	40.92	461.18	44.47	155.47	49.24	145.47	38.91	37.19	38.85	34.21
Var_1	51.05	470.29	46.64	436.47	48.67	146.13	50.60	167.02	38.58	36.52	42.06	47.79
Var_2	45.25	570.72	41.74	565.38	46.27	168.33	49.00	211.94	31.88	30.84	41.22	40.39
Var_3	38.64	587.46	36.44	464.64	50.82	125.15	52.16	151.84	38.26	36.45	50.48	52.24
Var_4	59.24	**190.26**	**57.48**	**199.64**	**66.25**	**74.62**	**68.53**	**42.31**	**48.66**	47.43	57.56	63.49
Var_5	**59.52**	191.36	57.07	204.45	60.25	99.48	66.03	54.69	45.65	**47.48**	**64.22**	**67.85**

二、MTransE介绍&实例

MTransE介绍

传统KG Embedding：用于单语言场景

以TransE为例：

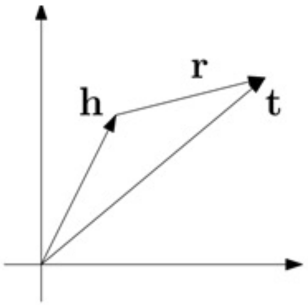


Bach	(0.036, -0.12, ..., 0.323)
City	(0.102, 0.671, ..., -0.101)
Germany	(0.138, 0.551, ..., 0.222)
...	

Embeddings

TransE

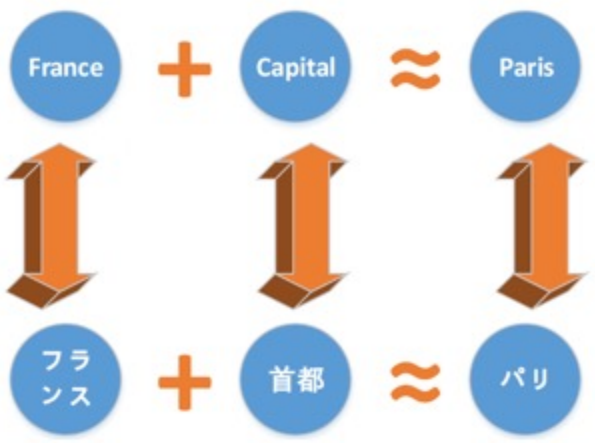
🟥 $h + r \approx t$



MTransE介绍

MTransE将TransE的应用扩展到了多语言的场景

- 训练语料：不同语言，且已经部分对齐的KG Embedding
- 下游任务：知识对齐、跨语言Q&A、多语言聊天机器人



MTransE介绍

- MTransE由knowledge model和alignment model组成

knowledg model：知识编码，TransE

alignment model：编码后的知识的对齐

- 用于对齐的不同损失函数：

distance-based axis calibration

$$S_{a_1} = \|\mathbf{h} - \mathbf{h}'\| + \|\mathbf{t} - \mathbf{t}'\|$$

$$S_{a_2} = \|\mathbf{h} - \mathbf{h}'\| + \|\mathbf{r} - \mathbf{r}'\| + \|\mathbf{t} - \mathbf{t}'\|$$

translation vectors

$$S_{a_3} = \left\|\mathbf{h} + \mathbf{v}^e_{ij} - \mathbf{h}'\right\| + \left\|\mathbf{r} + \mathbf{v}^r_{ij} - \mathbf{r}'\right\| + \left\|\mathbf{t} + \mathbf{v}^e_{ij} - \mathbf{t}'\right\|$$

linear transformations

$$S_{a_4} = \left\|\mathbf{M}^e_{ij}\mathbf{h} - \mathbf{h}'\right\| + \left\|\mathbf{M}^e_{ij}\mathbf{t} - \mathbf{t}'\right\|$$

$$S_{a_5} = \left\|\mathbf{M}^e_{ij}\mathbf{h} - \mathbf{h}'\right\| + \left\|\mathbf{M}^r_{ij}\mathbf{r} - \mathbf{r}'\right\| + \left\|\mathbf{M}^e_{ij}\mathbf{t} - \mathbf{t}'\right\|$$

MTransE介绍

MTransE如何得到A语言中的实体E_A在B语言中对应的实体E_B？

- 对E_A编码得到A的embedding，假设是Em_A，在语言A的空间中

- 用模型学习到的翻译向量V或M对Em_A进行翻译，将其映射到语言B的空间中，得到Em_A,

- 在Em_A,周围寻找离它最近的Em_B，作为E_B对应的embedding

- 将Em_B解码为E_B

实例

使用命令

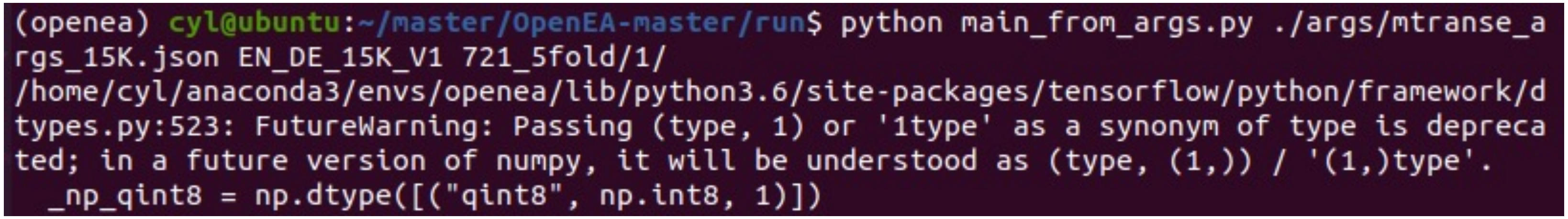
```
[python main_from_args.py ./args/mtranse_args_15K.json
EN_DE_15K_V1 721_5fold/1/]
```

在EN_DE_15K_V1的721_5fold/1/上运行mtranse

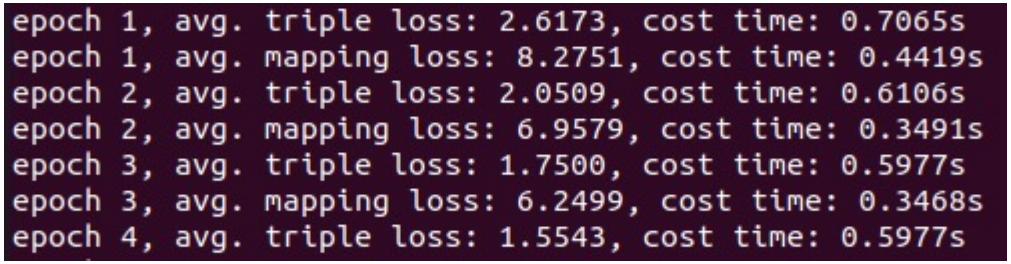
参数意义可以查看之前的解释

实例

主要运行过程：

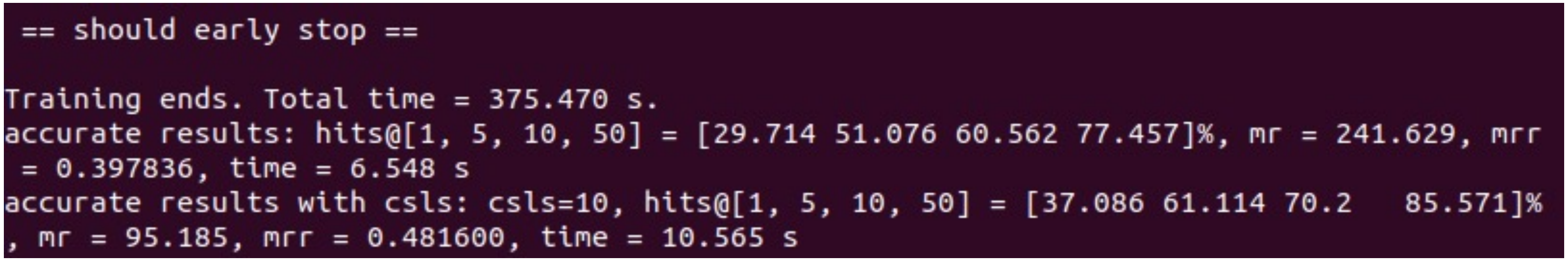


```
(openea) cyl@ubuntu:~/master/OpenEA-master/run$ python main_from_args.py ./args/mtranse_a
rgs_15K.json EN_DE_15K_V1 721_5fold/1/
/home/cyl/anaconda3/envs/openea/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:523: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depreca
ted; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
```



```
epoch 1, avg. triple loss: 2.6173, cost time: 0.7065s
epoch 1, avg. mapping loss: 8.2751, cost time: 0.4419s
epoch 2, avg. triple loss: 2.0509, cost time: 0.6106s
epoch 2, avg. mapping loss: 6.9579, cost time: 0.3491s
epoch 3, avg. triple loss: 1.7500, cost time: 0.5977s
epoch 3, avg. mapping loss: 6.2499, cost time: 0.3468s
epoch 4, avg. triple loss: 1.5543, cost time: 0.5977s
```

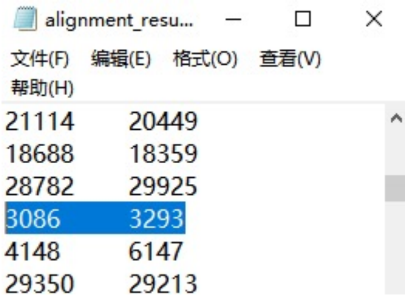
运行结果：



```
 == should early stop ==

Training ends. Total time = 375.470 s.
accurate results: hits@[1, 5, 10, 50] = [29.714 51.076 60.562 77.457]%, mr = 241.629, mrr
 = 0.397836, time = 6.548 s
accurate results with csls: csls=10, hits@[1, 5, 10, 50] = [37.086 61.114 70.2   85.571]%
, mr = 95.185, mrr = 0.481600, time = 10.565 s
```

实例



```
21114	20449
18688	18359
28782	29925
3086	3293
4148	6147
29350	29213
```

对齐结果举例：

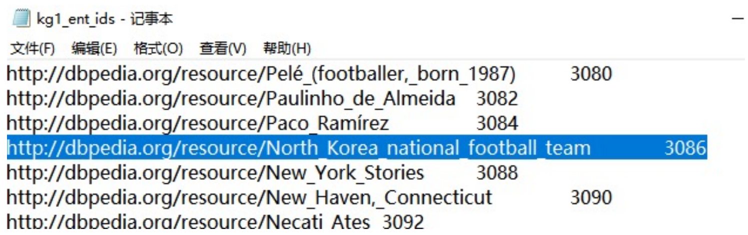
alignment result.txt中找出一对结果：3086　　3293

3086对应英语为：

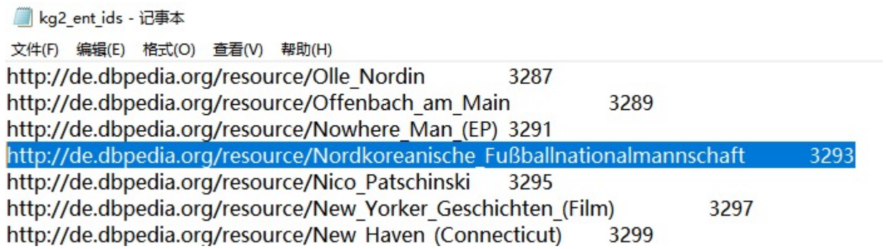
North_Korea_national_football_team

3293对应德语为：

Nordkoreanische_Fußballnationalmannschaft



```
http://dbpedia.org/resource/Pelé_(footballer,_born_1987)	3080
http://dbpedia.org/resource/Paulinho_de_Almeida	3082
http://dbpedia.org/resource/Paco_Ramírez	3084
http://dbpedia.org/resource/North_Korea_national_football_team	3086
http://dbpedia.org/resource/New_York_Stories	3088
http://dbpedia.org/resource/New_Haven,_Connecticut	3090
http://dbpedia.org/resource/Necati_Ates	3092
```



```
http://de.dbpedia.org/resource/Olle_Nordin	3287
http://de.dbpedia.org/resource/Offenbach_am_Main	3289
http://de.dbpedia.org/resource/Nowhere_Man_(EP)	3291
http://de.dbpedia.org/resource/Nordkoreanische_Fußballnationalmannschaft	3293
http://de.dbpedia.org/resource/Nico_Patschinski	3295
http://de.dbpedia.org/resource/New_Yorker_Geschichten_(Film)	3297
http://de.dbpedia.org/resource/New_Haven_(Connecticut)	3299
```

*需要使用v1.1数据集，【百度云】 password：9feb

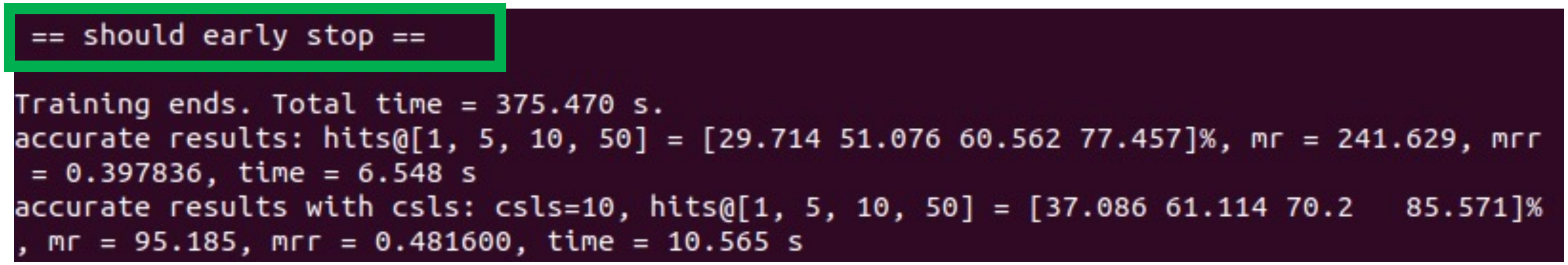
三、作业

作业

1.分别使用EN_FR_15K_V2的split1和EN_DE_15K_V2的split2来运行MTransE，记录使用的命令和结果

2.mtranse_args_15K.json和mtranse_args_100K.json有何区别，为什么要设置这种区别，而不是直接写一个mtranse_args.json?

3.什么是earlystop?这个实例中为什么需要earlystop?



```
 == should early stop ==

Training ends. Total time = 375.470 s.
accurate results: hits@[1, 5, 10, 50] = [29.714 51.076 60.562 77.457]%, mr = 241.629, mrr
 = 0.397836, time = 6.548 s
accurate results with csls: csls=10, hits@[1, 5, 10, 50] = [37.086 61.114 70.2   85.571]%
, mr = 95.185, mrr = 0.481600, time = 10.565 s
```