2023-2024 学年
知识工程（双语）实验报告

任课教师：吴天星

院　系 __人工智能学院__

专　业 ____人工智能____

姓　名 ____蒋雨初_____

任　务 Knowledge Embedding

1 实验五

1.1 实验任务

1. 阅读 OpenKE README 文档及样例程序，了解数据集结构及模型参数意义，训练满足以下条件的 TransE、TransH 模型，并获取测试性能，要求:

   (a) Embedding size = 100;

   (b) 数据集为 WN18RR.

   其他参数可自行调整。

2. 将代码运行过程与实验结果 (链接预测) 截图形成实验报告。

1.2 实验背景

OpenKE

OpenKE 是一个由清华大学开发的开源知识嵌入框架，属于 THU-OpenSKL 项目的一部分，旨在通过表示学习结合结构化知识和非结构化语言。它包含多个子项目，如 KB2E、TensorFlow-Transx、Fast-TransX、OpenNRE 和 OpenKPM。特别地，OpenKE 专注于将大规模知识图谱中的结构化知识表示为嵌入。

OpenKE 提供了一个基于 PyTorch 的平台，用于知识图谱嵌入模型，同时也提供了优化和稳定的框架。它包含不同的版本，如 OpenKE-PyTorch、OpenKE-Tensorflow1.0、TensorFlow-TransX 和 Fast-TransX。特别值得注意的是，OpenKE-PyTorch 版本因其高效、可扩展和用户友好而受到推崇。

在 OpenKE 中包括的模型涵盖了多种知识嵌入方法。对于 TensorFlow 版本，这些包括 RESCAL、HolE、DistMult、ComplEx、Analogy、TransE、TransH、TransR 和 TransD。PyTorch 版本则在此基础上增加了 SimplE 和 RotatE。

OpenKE 的 README 文档还概述了用于评估模型的实验设置和度量标准。这包括诸如损坏测试三元组并测量正确实体的排名的程序，以及 Hits@10、Hits@3、Hits@1、平均排名和平均倒数排名等度量标准。

WN18RR

WN18RR 数据集是从 WN18 数据集衍生出来的，用于链接预测任务。WN18 数据集本身是从 WordNet 中抽取的，包含约 41,000 个同义词集（synsets）和 18 种关系，形成了 141,442 个三元组。WN18RR 在此基础上进行了修改，以解决原始 WN18 中存在的测试集泄漏问题。在 WN18 中，许多测试三元组可以在训练集中找到，但与另一种关系或逆关系相关联。因此，WN18RR 被创建来确保评估数据集不会出现逆关系测试泄漏。WN18RR 包含 93,003 个三元组，连接 40,943 个实体，并涉及 11 种关系类型

TransE

TransE 是一种处理关系作为翻译操作的嵌入模型。在 TransE 模型中，实体和关系被表示为同一维度的向量。模型的主要假设是头实体向量加上关系向量应该接近尾实体向量。因此，TransE 试图学习这样的嵌入，使得对于每个（头实体，关系，尾实体）三元组，头实体加上关系后的向量与尾实体的向量之间的差距最小。

TransE 模型参数如下：

- embedding_dim：维度大小，表示实体和关系的嵌入向量的维度。
- learning_rate：学习率，控制模型在学习过程中的更新速度。
- margin：间隔边界，在优化过程中用于区分正负样本的边界值。
- p_norm：用于计算距离时的范数类型，常见的有 L1 范数和 L2 范数。

其中 embedding_dim 已指定为 100。

TransH

TransH 模型是 TransE 的一个改进版本，它允许每个关系有其专属的超平面。在 TransH 中，每个关系不仅表示为一个向量，还与一个超平面相关联。这个超平面通过一个法向量来定义。实体在被用于关系的计算前，首先被映射到这个超平面上。TransH 允许不同的关系在不同的超平面上进行翻译操作，从而能够更好地处理具有不同性质的关系。

TransH 的可调节参数与 TransE 是一样的。

1.3 实验设置与结果

编写 TransE 和 TransH 的训练及测试代码，以 Hits@10 作为衡量指标，使用 optuna 进行超参数搜索，lr 和 margin 的变化范围分别是 [0.0001,1] 和 [1, 10]。nbatch 设置为 100，embedding dim 设置为 100。模型训练 100 轮，optuna 实验进行 50 次。TransE 的训练过程示例如图 1所示，TransH 的展示在图 2。对于 TransE，最佳的 lr 和 margin 分别是 0.148 和 7.753（分别保留三位小数），对应的 Hits@10 为 0.521。对于 TransE，最佳的 lr 和 margin 分别是 0.271 和 8.675，对应的 Hits@10 为 0.523。

1.4 结果分析

分别绘制不同 margin 和 lr 下 Hits@10 的散点图，如图 3和图 4所示。TransE 和 TransH 对 margin 和 lr 的偏好基本一致，margin 最好设置在 8 附近，lr 则应设置大于 0.1。

```
Input Files Path : ./benchmarks/WN18RR/
The toolkit is importing datasets.
The total of relations is 11.
The total of entities is 40943.
The total of train triples is 86835.
Input Files Path : ./benchmarks/WN18RR/
The total of test triples is 3134.
The total of valid triples is 3034.
Finish initializing...
Epoch 99 | loss: 24.910914: 100%|██████████| 100/100 [00:54<00:00,  1.83it/s]
100%|██████████| 3134/3134 [00:05<00:00, 550.97it/s]
[I 2024-01-02 11:23:28,692] Trial 38 finished with value: 0.5215379595756531 and parameters: {'alpha': 0.25694637564710643, 'margin': 8.31989099224178}.
no type constraint results:
metric:                 MRR             MR              hit@10          hit@3           hit@1
l(raw):                 0.150575        2904.394043     0.444161        0.234525        0.010849
r(raw):                 0.208422        2056.943115     0.531270        0.294193        0.055201
averaged(raw):          0.179498        2480.668457     0.487715        0.264359        0.033025

l(filter):              0.210549        2882.520508     0.489789        0.360243        0.025526
r(filter):              0.253019        2051.764893     0.553287        0.404276        0.063816
averaged(filter):       0.231784        2467.142578     0.521538        0.382259        0.044671
```

图 1: 获得了最佳 Hits@10 的 TransE 的训练过程

```
Input Files Path : ./benchmarks/WN18RR/
The total of test triples is 3134.
The total of valid triples is 3034.
Finish initializing...
Epoch 99 | loss: 29.583403: 100%|██████████| 100/100 [01:28<00:00,  1.13it/s]
100%|██████████| 3134/3134 [00:09<00:00, 343.68it/s]
[I 2024-01-02 12:57:08,202] Trial 30 finished with value: 0.5226547718048096 and parameters: {'alpha': 0.2714440463091006, 'margin': 8.6749641
no type constraint results:
metric:                 MRR             MR              hit@10          hit@3           hit@1
l(raw):                 0.154607        3090.651611     0.441927        0.243778        0.011806
r(raw):                 0.221368        2277.415039     0.530313        0.314295        0.068922
averaged(raw):          0.187987        2684.033203     0.486120        0.279036        0.040364

l(filter):              0.217290        3068.520752     0.492661        0.372048        0.032865
r(filter):              0.263251        2272.272461     0.552648        0.412253        0.080408
averaged(filter):       0.240270        2670.396484     0.522655        0.392151        0.056637
0.5226547718048096
```

图 2: 获得了最佳 Hits@10 的 TransH 的训练过程

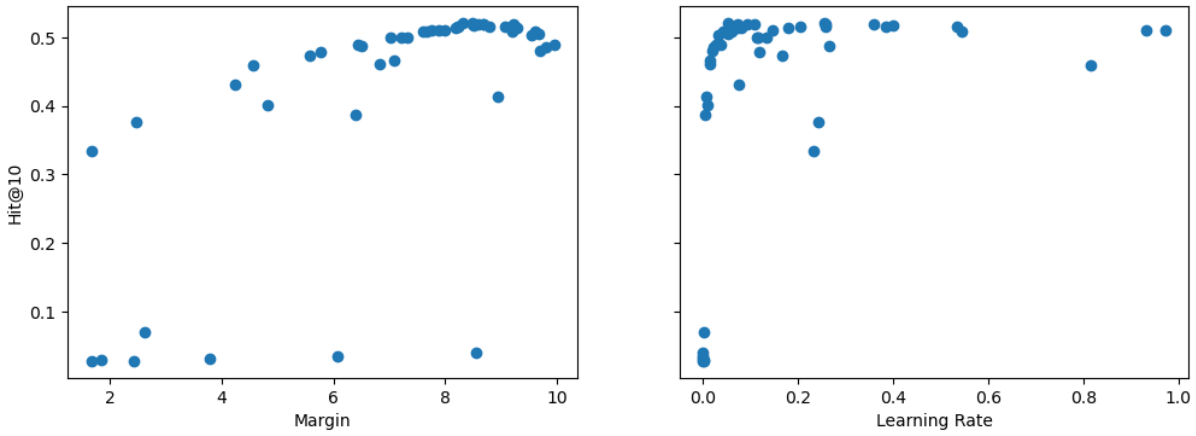

图 3: TransE：不同 margin 和 lr 对 Hits@10 的影响

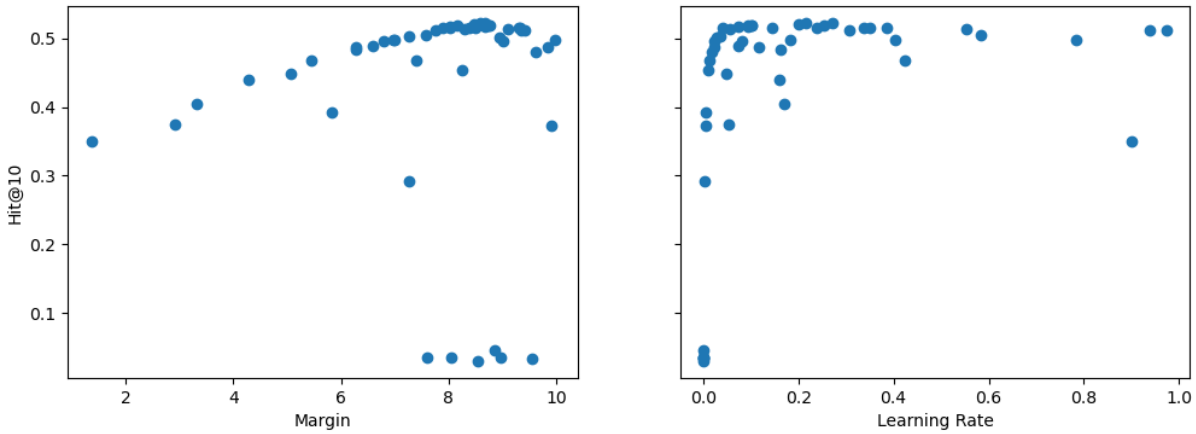

图 4: TransH：不同 margin 和 lr 对 Hits@10 的影响

2 实验六

2.1 实验任务

完成实验要求并回答相关问题。

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?

2.2 实验设置与结果

分别使用 EN_FR_15K_V2 的 split1 和 EN_DE_15K_V2 的 split2 来运行 MTransE，运行结果展示在图 5和图 6。

2.3 讨论

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?

mtranse_args_15K.json 和 mtranse_args_100K.json 大体相同，只有批大小和线程数不同。这反映了针对不同数据集大小或计算资源的优化。

1. **Batch Size:** mtranse_args_15K.json 中的 batch_size 为 5000，而 mtranse_args_100K.json 中为 20000。这表明后者处理的数据批次是前者的四倍

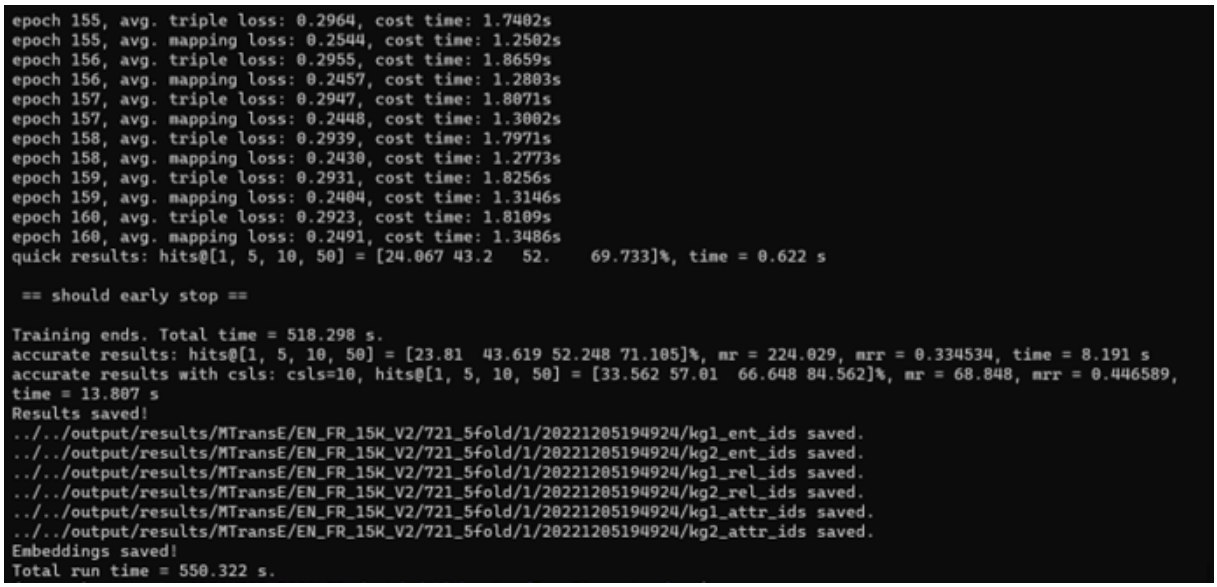

图 5: EN_FR_15K_V2 的 split1 运行 MTransE

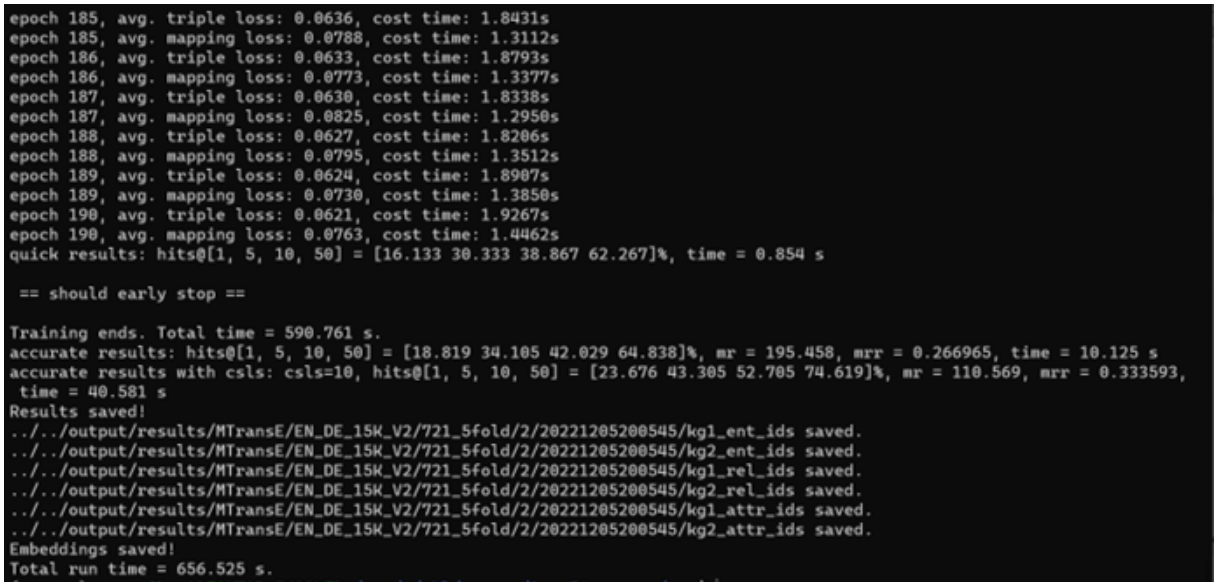

图 6: EN_DE_15K_V2 的 split2 运行 MTransE

大。更大的批量大小可以提高数据处理的效率，特别是在有足够内存和计算资源的情况下。但它也可能需要更多的内存和计算能力。

2. **Batch Threads Number 和 Test Threads Number:** 在 mtranse_args_15K.json 中，batch_threads_num 和 test_threads_num 分别为 2 和 4；而在 mtranse_args_100K.json 中，这些值分别为 3 和 12。这意味着后者在进行批处理和测试时使用的并行线程数更多，这可以加速处理过程，特别是在多核处理器环境下。

需要不同的配置文件而不是单一的 mtranse_args.json 的原因如下：

- 数据集大小和复杂度: 不同大小的数据集可能需要不同的处理策略。较小的数据集可能不需要太多的并行处理能力，而较大的数据集则可能从更多的并行处理中受益。
- 资源限制: 在资源较少的环境下（如内存限制），较小的批量大小和较少的并行线程可能更为合适。
- 性能优化: 不同的配置可以根据特定的硬件或应用场景进行优化，以提高效率和性能。
- 灵活性和可定制性: 提供不同的配置文件使用户可以根据自己的需求和资源情况选择最合适的配置，而不是被迫适应一个“一刀切”的配置。

什么是 earlystop？这个实例中为什么需要 earlystop?

Early Stopping 是一种用于避免过拟合的技术，在训练机器学习模型时经常使用。它的基本原理是在训练过程中定期监测模型在验证集上的表现。如果发现模型的性能不再提升或开始恶化（例如，验证集上的错误率开始增加），则提前终止训练。这种方法的关键在于，它试图在获得最优泛化性能和避免过度学习（或过拟合）之间找到平衡。

我们使用 EN_FR_15K_V2 的 split1 来运行 MTransE 的结果来进行分析。从 epoch100 以后，每次隔 10 个都会显示一个 quick results，如表 1所示。观察这些结果可以发现，到了最后阶段，Hits 的性能已经没有明显的提升，并且 Hits@1,Hits@5 还有了下降的趋势，所以应该停止迭代，避免过拟合。

Epoch	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	Hits@50 (%)	Time (s)
100	22.2	38.3	46.3	64.9	0.734
110	22.7	40.2	48.5	66.5	0.732
120	23.3	40.9	50.1	67.4	0.734
130	23.9	42.3	51.2	68.3	0.740
140	24.3	43.5	51.2	69.2	0.722
150	24.1	43.3	51.6	69.5	0.662
160	24.1	43.2	52.0	69.7	0.622

表 1: 从 epoch=100 到 epoch=160 的 Quick Results