* 补充说明文档

重要：如果你已完成实验和作业且没啥问题，可以直接忽略此文档！（不用看了！）

这是一个对实验二、实验三、部分作业做补充说明的文档，包含某些错误修正、部分同学的疑问。如果依然有问题，可联系助教。

作业

1. **hw2 1.4 基于用户的协同过滤计算，需要排除掉相关性小于0的用户吗**?

答：在我们课程里**需要排除相似度<0的邻居用户。**详见第八讲ppt37页左右（user-based cf后面的文字部分）

不过，实际情况下，其实负相关邻居也是有用的，因为其代表一种反向偏好，对推荐系统的建模有潜在价值。

但负相关邻居往往不稳定、噪声大（尤其在稀疏数据里），可能造成模型不稳定或效果下降，（比如在相似用户很少时最终预测分数可能被负相关用户拉成负分），所以一般实际的协同过滤中也不考虑负相关用户。

实验二

> 如果你已经完成实验二，可以忽略此部分！

1. **实验二文档表述修正**

修正1：

经典的 TransE 算法采用 Hinge loss 作为损失函数。它是一类典型的基于负采样（negative sampling）的 margin-based 损失，**其核心思想是迫使模型将正三元组的得分提升，同时将负三元组的得分压低，并保持二者至少相差一个给定的 margin**。知识图谱训练集中

$$\mathcal{D} = \{(h_i, r_i, t_i)\}_{i=1}^{N}$$

说明：在很多其他的排序或推荐任务中，我们使用的打分函数满足“分数越大表示越相关”，故hinge loss通常约束“正样本得分要高于负样本得分一个 margin”；但是，在本实验的TransE 中，打分函数是距离形式，得分越小越好，因此对应的 hinge 损失目标会变成“负样本的距离要大于正样本的距离一个 margin”。

修正2：

在本实验中，我们改用了同为 pair-wise 损失的 **BRP loss 函数**，它的优化目标是让正样本的得分显著高于负样本，从而提升模型对真实三元组与虚假三元组的区分能力。其形式如下：

替换两个f的位置，即改成：
f(负样本)-f(正样本)

$$\mathcal{L}_{BPR} = -\sum_{i=1}^{N} log\ \sigma(f(h_i, r_i, t_i) - f(h_i, r_i, t_i^-))$$

同理，需要对讲义里的BPR loss形式做修正：BPR loss 的核心思想是最大化“正样本相对于负样本的偏好概率”。在 TransE 中，由于打分函数是距离形式（距离越小越好），因此 BPR 实际上在推动模型满足：正样本的距离小于负样本的距离。

总的来说，在本实验的 TransE 设定中，无论是 hinge loss 还是 BPR loss，其约束目标是一致的，即**正样本的距离应小于负样本的距离**。理清打分函数语义和损失函数的关系即可。

2. **实验代码部分笔误**

说明：如果你已经做完了实验，就不用理会这一条！修改这些只是为了防止同学们补全代码后无法一次跑通，卡在奇怪的bug上，浪费了时间。且我们非常鼓励大家按自己的理解和需求随意改代码，实验的验收标准是跑通即可，可以随意修改我们的代码。

①parser_Embedding_based.py

```
save_dir = 'trained_model/{}/link_prediction/dime{}dimr{}_lr{}_l2{}_{}/'.format(
    args.data_name, args.embed_dim, args.relation_dim, args.lr, args.cf_l2loss_lambda, args.KG_embedding_type)
args.save_dir = save_dir
```

改成：args.kg_l2loss_lambda

②loader_kg.py

根据自己的文件名修改下

```
# 直接使用已经划分好的 3 个文件
self.kg_train_file = os.path.join(self.data_dir, "kg_train.txt")
self.kg_valid_file = os.path.join(self.data_dir, "kg_valid.txt")
self.kg_test_file  = os.path.join(self.data_dir, "kg_test.txt")
```

实验三

> 如果你已经完成实验三，可以忽略此部分！

1. **embedding模型下载好了，怎么用？我需要放到哪个文件夹里吗**

答：放在哪个文件夹都可以的。使用时，建立embedding模型时，把model_name字段设置成模型路径就可以（即：“model_name” = "/xxxx(路径)/bge-base-en-v1.5"。建议直接用模型的绝对路径）

具体可参考langchain在semantic_search任务的文档，embedding使用部分（选用huggingface接口，其支持加载本地模型）。文档链接：https://docs.langchain.com/oss/python/langchain/knowledge-base#huggingface

2. **对比“RAG”和“不基于检索让大模型直接回答”这两种实验设置时，prompt是不是不一样?**

答：是的。

我们此处不做过于严格的控制变量的比较，二者prompt可以不一样。主要体会RAG和纯大模型的区别即可。

且事实上，由于RAG的prompt里引导了大模型根据检索到的片段来回答问题，所以当我们去除检索、只依赖大模型作答时，prompt里应该把这部分删去。

3. **关于embedding模型的选择：**

法律数据集是中文的，故选用的编码模型最好选用中文的。

我们在文档里给的两个模型为m3e-base和bge-base-en-v1.5，其中前者可编码中文&英文，而后者是针对英文文本的编码模型（其实在本实验中不太适用），故如果效果不好可改换为bge-base-zh-v1.5（bge系列中一个编码中文的embedding模型）。

4. **对于实验三中部分链接的更新：**

> 由于langchain版本更新，部分文档链接路由出错，这里给出最新版本的参考。
>
> 如果大家在查阅文档时遇到问题，可以在官方文档的搜索栏里查找。如果你依然卡住，可尽快联系助教（不希望卡在查阅文档/配置环境上太久而浪费了大家的时间）

Build a semantic search engine with LangChain - Docs by LangChain

（中文版：使用 LangChain 构建语义搜索引擎 - LangChain 文档 - LangChain 教程）

这是一个semantic search/语义检索的完整代码流程示例。你可以在这个页面找到**文档向量化、向量入库、数据检索**的相关操作，即如何使用embedding模型将文档向量化、如何构建向量数据库（比如FAISS与 Chromadb）、如何基于数据库进行检索（db.similarity_search）。对应**实验"任务说明"[1][2]部分。**

5. **用RAG的时候好多问题都回答"未找到相关答案"，这是正常的现象嘛?**

答：正常。

因为我们给大模型的prompt明确写了："你需要使用以下检索到的上下文片段来回答问题，禁止根据常识和已知信息回答问题。如果你不知道答案，直接回答'未找到相关答案'"。所以一旦根据检索到的内容无法作答时，大模型会尽量诚实地返回"未找到相关答案"。

（也说明这个大模型的指令遵循能力还蛮好的）

感兴趣的同学可以修改prompt，比如把限制性的提示词删去，然后再看看大模型返回答案的变化。