《软件实践》课程实验报告

暑期学校实验项目：高考志愿填报助手

小组名称	知识图谱 A 组						
姓　　名	朱佳涛	专业	人工智能	班级	091181	学号	09118108
实验时间	2020.8.31-2020.9.23		指导教师	孔祥龙		成绩	

一、 实验背景和目的

随着中国教育的不断发展，高考作为大学选拔人才的依据，其重要性与日俱增。无数考生及其亲属为高考志愿的选择而烦恼不已。为此设计并实现高考志愿填报助手，能给予考生及其亲属关于大学、排名、分数、专业等方面的信息，以及依据考生的高考成绩和位次做出合理的志愿推荐。

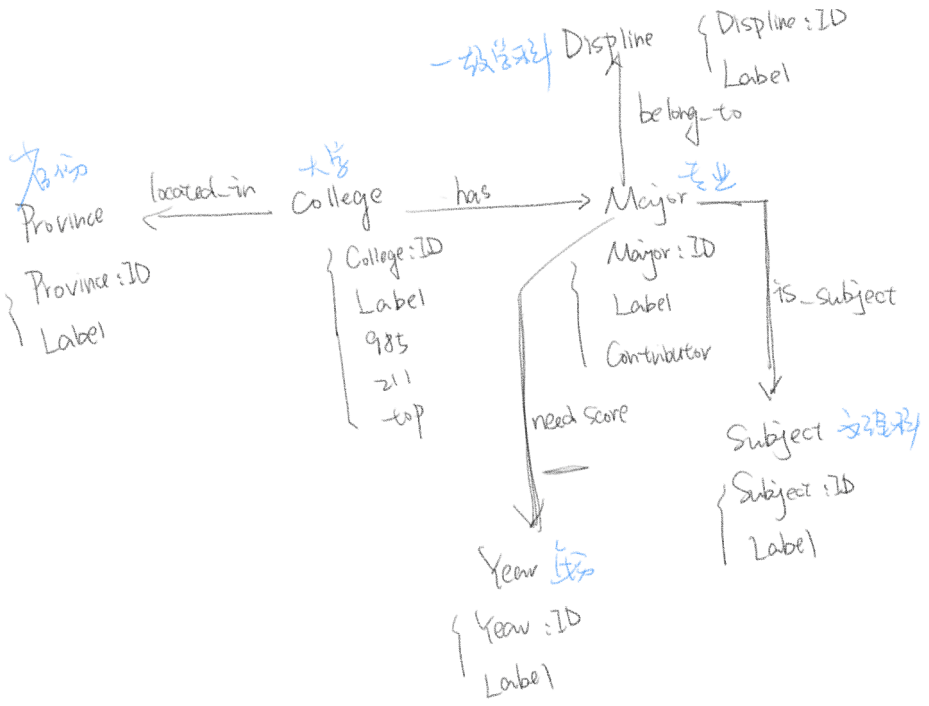
二、 小组任务和个人任务

小组任务：利用已有的数据构建一个小型的知识图谱，通过调用该知识图谱可获得直观的大学专业等可视化信息，以及对此做出的相关推荐。具体任务如下：获取数据源；对数据进行消歧处理；创建可导入 Neo4j 的 csv 文件；利用这些文件生成知识图谱；基于知识图谱构建显示网页。

个人任务：创建可导入 Neo4j 的 csv 文件，根据已有 entity 文件生成各 entity 之间的 relation 文件。

三、 个人任务需求分析

1. 首先要明确能输入 Neo4j 的文件格式具体应是如何。Relation 的设计即数据库中关系的设计，关系有三个必填字段：START_ID，END_ID 和:TYPE。故要将获取的数据源里的数据重新组织，使其符合该格式。
2. 其次设计该知识图谱的 ER 图，依据一开始的需求分析以及已有数据，可建立以大学为中心的 ER 图，且考虑将“省份”，“大学”，“专业”，“一级学科”，“年份”，“文理科”作为实体，创建以他们之间关系的 ER 图。下图为初步设计时手画的 ER 图。



图一：知识图谱 ER 图

3. 再次依据知识图谱的 ER 图，分析在该任务中所需要的 relation 关系有哪些。具体如下：大学与省份的位置关系；大学与专业的开设关系；专业与一级学科的从属关系；专业与科类的招生关系；专业与年份的录取分数线关系。
4. 然后从数据源里提取有用信息生成满足上述格式的 csv 文件。故提取出的 relation 如下所示：collegeID located_in provinceID；collegeID has majorID；majorID belong_to desiplineID；majorID is_subject subjectID；majorID score yearID。
5. 最后利用 python 代码将其实现。具体可运用 python 中 csv 文件读写相关功能。

四、实验过程（需附上关键代码及相关说明）

依据原始数据源的数据以及本组其他组员对于 entity 数据的清洗以及更新后，获取到规范的写入关系中的数据较为容易。

首先要明确 relation 所需的数据所在哪一个具体的 entity 文件，以及在该 entity 文件中具体的位置。后可利用 csv 文件的读写功能，将所需数据从指定 entity 文件以及位置，通过列表传送到新建的 relation 文件中，随即保存至 csv 文件中。

以大学与省份的位置关系为例，relation 文件所需数据为“college.csv”中的“collegeID”列以及“provinceID”列，即可在读取每行数据并保存至 colleges[]中后，只获取每行中的这两个数据，传送所要生成的“located_in.csv”文件中。

下图为该例具体代码：

```python
def located_in():
    colleges=[]
    f = open('college_improved.csv','r',encoding='UTF-8')
    reader = csv.reader(f)
    for i in reader:
        colleges.append(i)
    f.close()
    w = open('located_in.csv','a',encoding='UTF-8')
    writer = csv.writer(f)
    writer.writerow([':START_ID','located_in',':END_ID',':TYPE'])
    for i in colleges:
        row = []
        row.append(i[0])
        row.append('located_in')
        row.append(i[1])
        row.append('LOCATED_IN')
        writer.writerow(row)
    w.close()
```

图二：relationCreate.py 函数代码示例

将原本 college.csv 中的省份 ID 信息与大学 ID 信息这两列获取到（即代码中的 i[0]与 i[1]），并加入关系的名称“LOCATED_IN”，即可创建“located_in.csv”。

同理，写入五个类似的函数即可实现该需求，最后调用这五个函数，即可获取五份可输入 Neo4j 的 csv 文件。

```python
if __name__ == '__main__':
    belong_to()
    has()
    located_in()
    need_score()
    is_subject()
```

图三：relationCreate.py 调用代码示例

五、实验结果与分析

调用该五个函数，可获得五个 csv 文件，分别反映了大学与省份的位置关系；大学与专业的开设关系；专业与一级学科的从属关系；专业与科类的招生关系；专业与年份的录取分数线关系。下为一部分结果示例：

	A	B	C	D
1	:START_ID	located_in	:END_ID	:TYPE
2	c10001	located_in	p25	LOCATED_IN
3	c10002	located_in	p25	LOCATED_IN
4	c10003	located_in	p25	LOCATED_IN
5	c10004	located_in	p25	LOCATED_IN
6	c10006	located_in	p25	LOCATED_IN
7	c10007	located_in	p25	LOCATED_IN
8	c10008	located_in	p25	LOCATED_IN
9	c10009	located_in	p25	LOCATED_IN
10	c10010	located_in	p25	LOCATED_IN
11	c10011	located_in	p25	LOCATED_IN
12	c10013	located_in	p25	LOCATED_IN

图四：result 示例

此为“located_in.csv”中的一部分，可知该表较好的表达了实体间的关系，且完全符合 Neo4j 的格式需求，即每行数据均满足:START_ID　located_in　:END_ID　:TYPE 的分布。

六、实验总结与心得体会

本次实验带给我收获颇丰，具体总结如下：

1. 实验整体十分浩大，但人数众多，故分配任务自然尤为重要。组与组之间相互联系，上一组的工作稍有瑕疵，便另我们组的进展缓慢，进而进行组与组之间的沟通交流，由于第一组的数据清洗工作进度较缓，导致我们组消歧成员的进度也较缓。这既让我对团队工作有了更深刻的认识，也理解了沟通的重要性。
2. 利用 Git 仓库进行团队开发的优势。课程设计初期，组内交流进度多以在线文件传输的方式，故有部分组员和部分数据并未在 Git 仓库中呈现。在课程设计后期，改掉该习惯，所有数据以及文件传输均通过 Git 仓库进行。这作为团队开发的良好习惯，应尽早养成。
3. 在组内工作时，仍会因组员间交流的欠缺，导致部分任务衔接处的重复劳动或者双双不作为。往往又要花费人力和时间才能解决这样的问题。这更让我知晓沟通的重要性。
4. 在编写我自己部分的代码时，时刻有着一种责任感，自己的代码部分犹如一大工艺品中的小小齿轮，若完成的不好，将会影响整个工艺品，故更会对自己提高要求，精益求精完成自己应做到的部分。
5. 仅仅只是看着手头高考分数线、排名、大学的数据，就会使我想起两年前望着志愿册发呆的自己。高中生涯的点滴心情一一浮现，落寞谨慎又不知所措。每年都会有千千万万个那时的自己。一个团队做出这样一个具有高考填报功能的产品，是因为有着庞大的需求，而这需求不是冰冷的利益数字，而是每一个参与其中的开发者都有着“帮助别人”的决心，倘若有我出了一份力的高考志愿系统真的能给需要帮助的人一点指引的话，更会坚定我成为一个优秀程序员的决心吧。理想万岁！

2020 年 9 月制