软件课程设计报告

（ 面向豆瓣电影的知识图谱的设计与实现）

目 录

一、实验目的

为学生提供一个动手实践平台，帮助其加深对行业内相关技术的了解和掌握，提高其编程能力与解决具体问题的能力。

软件技术课程设计目的在于培养学生将所学的专业技能转化为实践的能力，因此在选题时充分考虑到了技术发展需要与实际的应用价值。学会快速获取和处理海量的数据并从中得到有价值的信息是信息时代的一项重要技能。通过完成本课程设计，将加深学生对网络爬虫、数据挖掘及软件编程技术的理解，同时锻炼其软件编程与解决实际问题的能力。

二、项目要求

1 配置开发环境

学习配置开发环境，实现网络爬虫爬取页面、从而形成结构化数据的基本功能，并给出软件运行及测试结果。

软件环境：

- 操作系统：Windows10
- IDE：PyCharm，neo4j
- Python 版本：python 3.7

2 实现网络爬虫

（1）从豆瓣电影主页面进入，以广度优先的爬取方式爬取到每个电影的详细地址，进行存储。

（2）爬取相关联的网页显示的相关信息，包括电影名称、导演、演员、电影类型等感兴趣的内容，最后以.txt 或.csv 格式进行存储。

3 构建知识图谱

学习 Neo4j 图数据库的使用，对使用爬虫爬取的电影数据进行分析，提取整合出知识图谱的结点和关系，并通过 Neo4j 构建电影知识图谱。

4 附加功能

（1）GUI 界面：

为用户提供友好的可视化操作界面

（2）代理池数据库：

搭建代理池主要分为存储、获取、检测、接口四个模块。

（3）多线程：

同步完成多项任务，采用多个线程同步爬取多个网页。

（4）词频统计：

将文本进行分词和词性标注，实现关键词提取。

（5）情感分析：

通过对电影评论的文本进行分词及词性标注，将电影的评论情感分为积极的，消极的或者中性。

三、知识概述

网络爬虫

网络爬虫是从 web 中发现、下载以及存储内容，是大数据处理分析的基础核心部分。从功能上来讲，爬虫一般分为数据采集，处理，储存三个部分。传统爬虫从一个或若干初始网页的 URL 开始，获得初始网页上的 URL，载抓取网页的过程中，不断从当前页面上抽取新的 URL 放入队列，直到满足系统的一条停止条件。

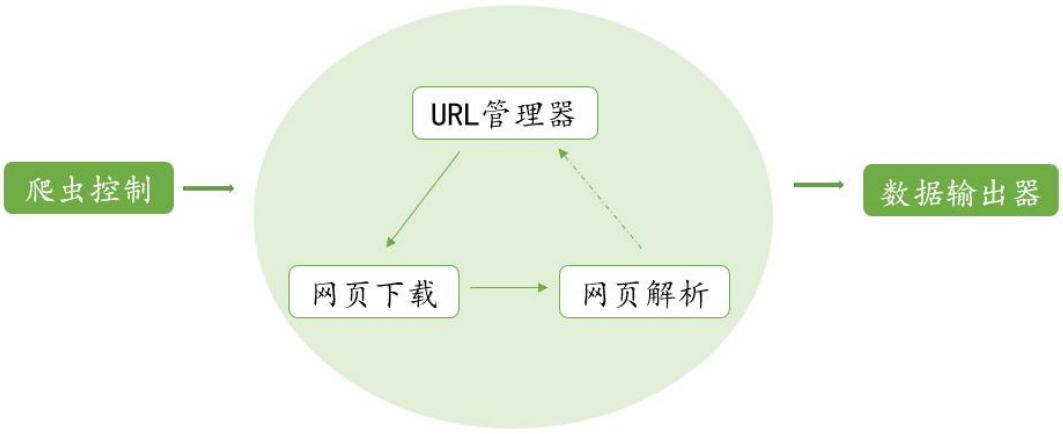
爬虫程序主要分为三个模块：爬虫控制端、爬虫运行模块、数据输出模块。爬虫控制端主要是用来启动、停止和监视爬虫；爬虫运行模块主要是获取 URL、下载网页和网页解析；数据输出模块主要是将爬取到的数据进行存储。

a. IP 被封:

使用代理 IP 进行爬取

使用 cookies 模拟登录

b. 动态加载



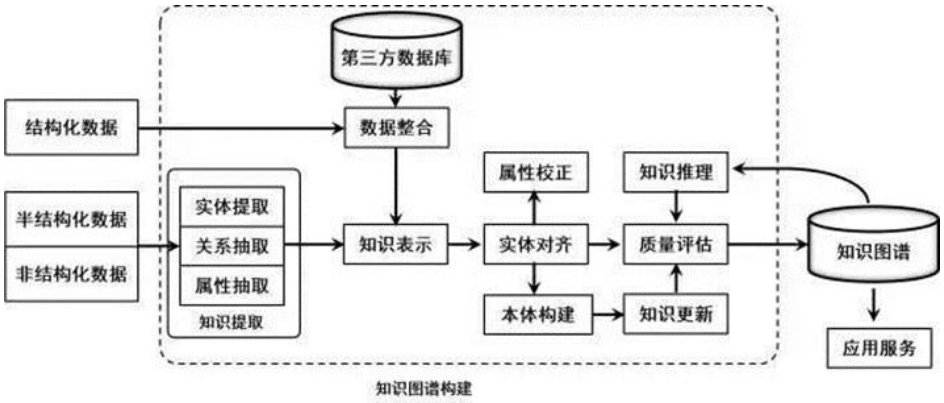
Neo4j 图数据库

Neo4j 是一个嵌入式，基于磁盘的，支持完整事务的 Java 持久化引擎。作为“面向网络的数据库”，它在图(网络)中而不是表中存储数据。Neo4j 提供了大规模可扩展性，在一台机器上可以处理数十亿节点/关系/属性的图，可以扩展到多台机器并行运行。

相对于关系数据库来说，图数据库善于处理大量复杂、互连接、低结构化的数据，这些数据变化迅速，需要频繁的查询——在关系数据库中，这些查询会导致大量的表连接，因此会产生性能上的问题。Neo4j 重点解决了拥有大量连接的传统 RDBMS 在查询时出现的性能衰退问题。通过围绕图进行数据建模，Neo4j 会以相同的速度遍历节点与边，其遍历速度与构成图的数据量没有任何关系。

知识图谱

一种基于图的数据结构，由节点(Point)和边(Edge)组成。大规模知识图谱的构建与应用需要多种智能信息处理技术的支持。知识图谱的构建主要依靠爬虫爬取到的数据，以及各种电影信息的对应关系进行知识图谱的构建。知识图谱提供了从“关系”的角度去分析问题的能力。



知识图谱构建

GUI 界面

图形用户界面是指采用图形方式显示的计算机操作用户界面。

图形用户界面是一种人与计算机通信的界面显示格式，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。

代理池数据库

搭建代理池主要分为存储、获取、检测、接口四个模块。

多线程

多线程是指从软件或者硬件上实现多个线程并发执行的技术。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。具有这种能力的系统包括对称多处理机、多核心处理器以及芯片级多处理或同时多线程处理器。在一个程序中，这些独立运行的程序片段叫作“线程”（Thread），利用它编程的概念就叫作“多线程处理”。

词频统计

词频统计模块分为读取、分词统计、写入三个模块，是自然语言处理的基础。

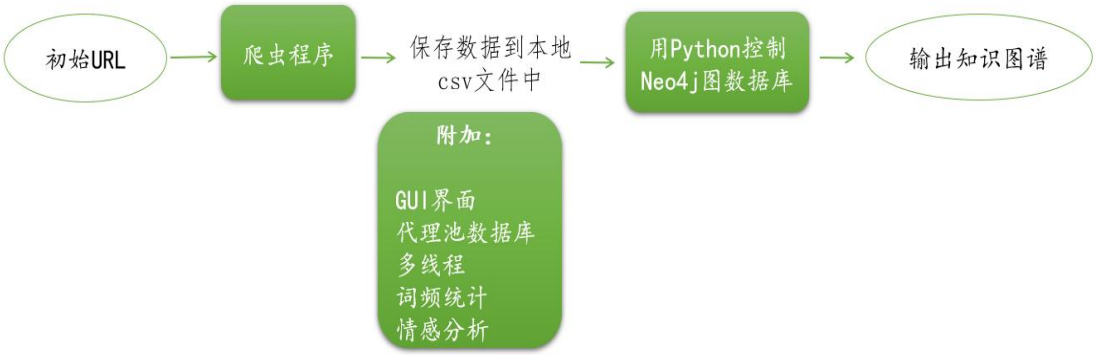
情感分析

情感分析在自然语言处理中，情感分析一般指判断一段文本所表达的情绪状态，属于文本分类问题。

四、整体设计

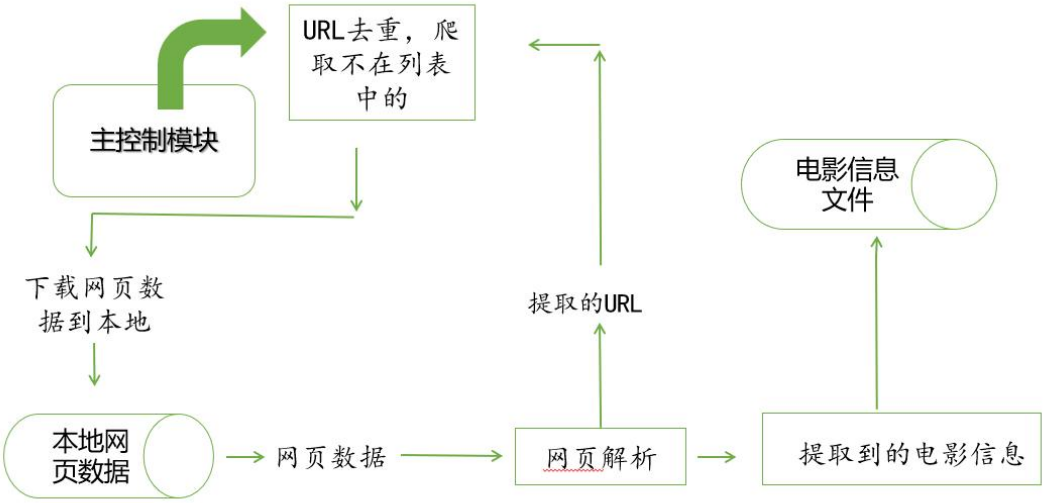
整体框架设计：

该项目是由爬虫获取网页上的数据，再调用 Neo4j 图数据库来对数据进行整合、关联，进而输出知识图谱。并且附加一系列功能使得项目更加完善，整体流程图如下：



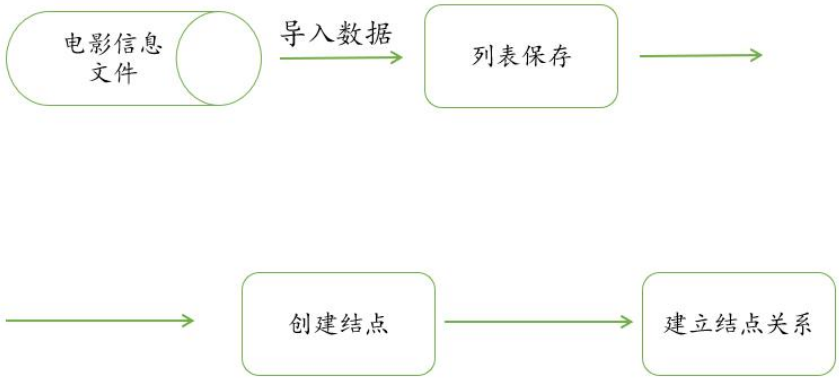
爬虫模块：

对于爬虫程序，我们设计了一个爬虫控制器作为主控制程序，以此启动程序。通过设计 proxy 获取代理 IP，headers 获取请求头。之后便是调用 url_manager 请求到网页数据。从网页数据中找到其中隐藏的 URL 和特定的数据；调用 web_loader 将 url 对应的网页数据保存在本地；接着调用 web_resolve 解析数据，对其进行分析，得到电影名称、导演、演员等信息）数据输出（数据输出器），将对应数据保存在对应的变量中，并返回一个新获取的 url 列表，然后再将列表中的元素传给 url_manager 保存到本地 csv 文件中以便创建知识图谱时调用数据。并且为了可视化设计，我们附加了爬虫加载过程中的背景设置。流程图如下：



知识图谱模块：

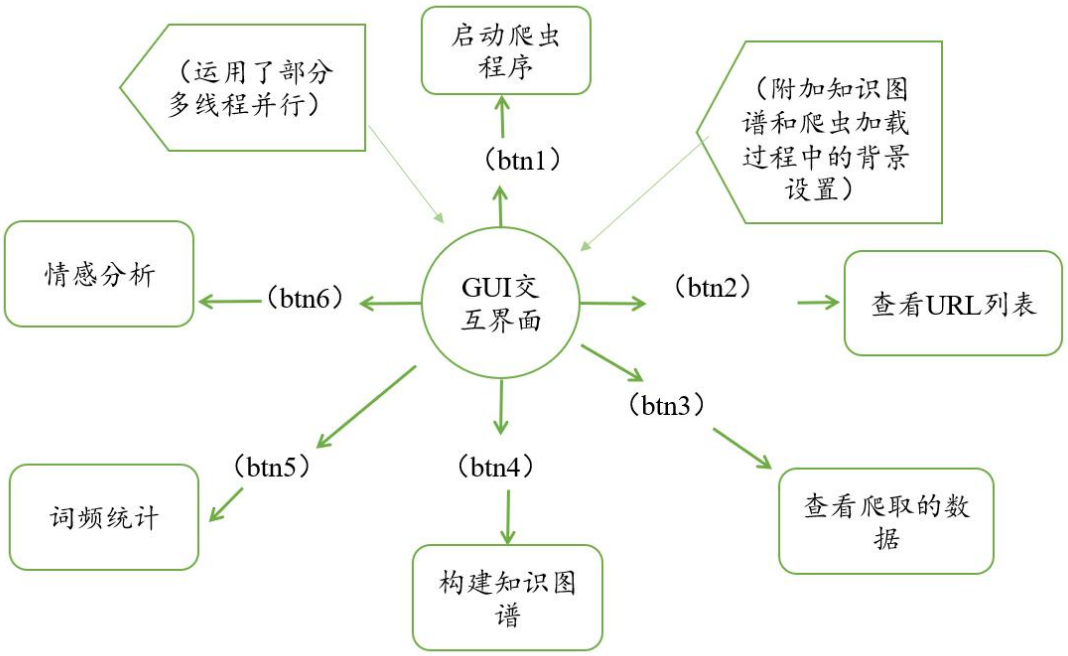
知识图谱主要依靠 py2neo 库来进行本地数据与 Neo4j 图数据库的联系。我们将电影信息 csv 文件从本地导入数据到 python 中，并将数据以列表形式存储，然后根据不同的属性 nodeSet 创建节点，并调用 relationshipSet 从而构建各个结点的关系。并且为了可视化设计，我们附加了线性增长的加载页面进度条，使得显示变得清晰。建立在知识图谱已经构建成功的情况下，我们可以使用 Cypher 语句进行查询从知识图谱中获取我们想要的信息。流程图如下：



GUI 模块：

GUI 界面是基于 python 自带的 Tkinter 库来编写的。我们具体实现了利用 GUI 控制程序的起止、利用 GUI 显示爬取用到的 URL、利用 GUI 显示电影信息 csv 文件、利用 GUI 控制知识图谱的构建以及利用 GUI 查找 Neo4j 图数据库中知识图

谱的相关信息。流程图如下：



词频统计模块：

词频统计模块我们一开始尝试了马尔可夫建模过程，利用其作词频统计并作为情感分析的基础，但该建模过于简单，考虑的状态也不够多。因此我们改用 THULAC（THU Lexical Analyzer for Chinese）。这是由清华大学自然语言处理与社会人文计算实验室研制推出的一套中文词法分析工具包，具有中文分词和词性标注功能。通过实验我们发现其分词的 F1 值可达 97.3%，词性标注的 F1 值可达到 92.9，同时进行分词和词性标注速度为 300KB/s，每秒可处理约 15 万字。只进行分词速度可达到 1.3MB/s。通过对训练集进行中文分词，再将分词后的词符作为输入，进行向量的转换。最后用文本卷积神经网络对向量进行特征提取最后分类。

情感分析模块：

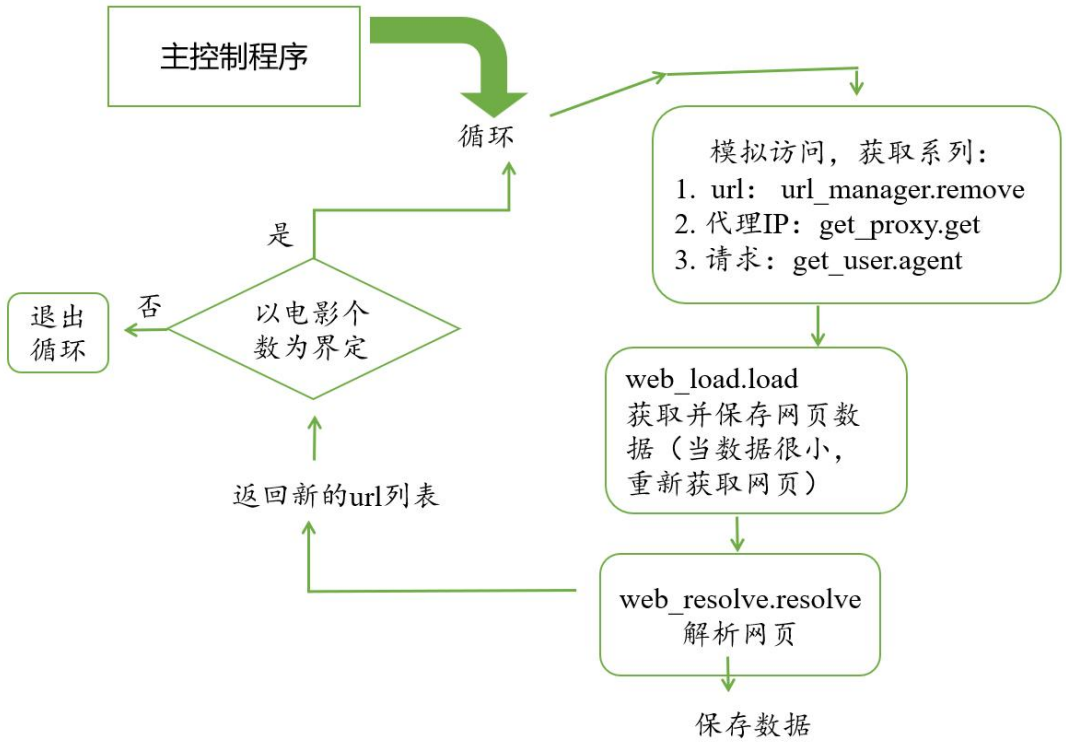
鉴于情感分析 top 250 的电影，正向评论过多（我们爬取了最热评论并做了保存），我们从 kaggle 上下载了的一份数据集作为检验的依据。我们主要调用了 BERT，其是一个预训练的语言表征模型。选取它的原因是它不再像以往一样采用传统的单向语言模型或者把两个单向语言模型进行浅层拼接的方法进行预训练，而是采用新的 masked language model（MLM），以致能生成深度的双向语言表征。情感分析具体实现如下：在 bert 输出层后加上一个全连接层作为 BertForClassification，并利用下载的 kaggle 数据集对其进行微调，将 bert 的

语言表证能力迁移到下游任务。

五、实现方法

爬虫程序：

整个函数主要是对各个子函数的组装与调用来完成整个爬虫程序。整个函数的流程图如下：



首先，主函数 spider()是为爬虫运行器提供初始 URL、代理 IP 以及控制爬虫运行器开始、停止的条件（以电影个数为界定）。在主函数中我们设计了全局变量传入初始 URL。在该函数中我们设计了 url 变量来进行 url 队列的管理。其中，URL 管理被放置在 url_manager.py 文件当中，该文件中有两个函数 load()与 remove()。load 函数用于将解析出来的 URL 加入队列。URL 被存放在队列中，这是由于我们要求按广度优先的方式进行爬取。remove 函数用于取出一个未爬取的 URL。并临时保存 top250 的榜单数据和每个电影的详细信息。

设计 proxy 获取代理 IP，headers 获取请求头。之后便是调用 url_manager.remove()来获取要爬取的 url；调用 web_loader.load()将 url 对应的网页数据保存在本地；网页下载被放置在 web_load.py 文件当中，该文件中有一个函数 load()。该函数用于请求网页数据，并将其下载到本地的网页数据库。接着调用 web_resolve.resolver()解析数据，网页解析被放置在 web_resolve.py 文件当中，该文件有一个函数 resolver()函数。该函数主要是接受本地网页数据库中传进来的数据，对其进行分析，得到我们所需要的电影名称、导演、演员等信息，

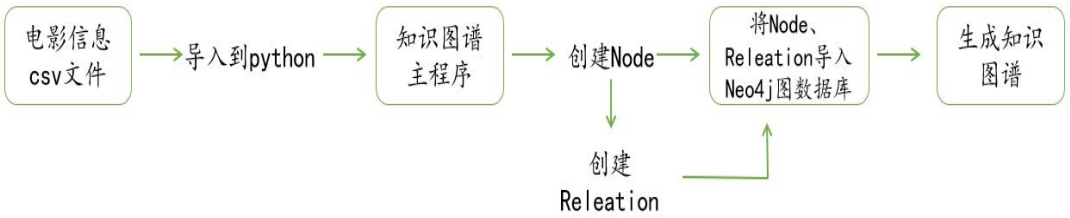
并将其保存在列表中。将对应数据保存在对应的变量中，并返回一个新获取的 url 列表，将新的 url 列表传给 url_manager.load()进行保存。整个函数主要是对各个子函数的组装与调用来完成整个爬虫程序。并且设计了参数用来显示 GUI 界面中的背景设置。

最后，date_out()完成数据输出，这个函数被放置在 data_ouput.py 文件中。该函数主要是通过调用 python 的 csv 库文件，对列表的字典类别的数据进行写文件的操作来完成电影信息 csv 文件的输出。

知识图谱程序：

知识图谱我们使用的是 neo4j。知识图谱创建主要依靠 py2neo 库来进行本地数据与 Neo4j 图数据库的联系。

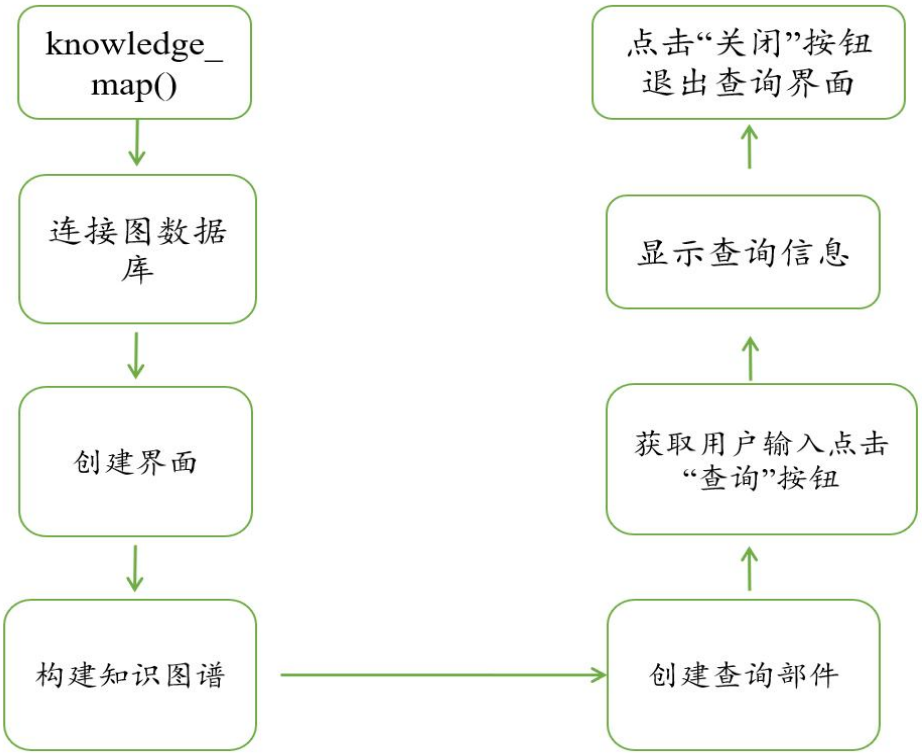
与上述模块相同，主函数也是起到整合子函数的作用。主要是对导入的数据进行处理，使数据更加方便的被用于创建结点、创建关系。



主函数是要完成将本地文件导入到 Neo4j 数据库并完成知识图谱的创建。首先，利用 python 的 csv 库将爬虫程序保存到本地的 csv 文件导入到 python 中（通过 python 中的 csv 库对文件进行转换，最后转换成字典的列表，避免 cypher 语言使用 neo4j 读取 csv 文件这种较复杂的节点关系的实现方式）。然后则是创建各个结点以及各个节点的关系。创建结点和节点之间的关系主要调用 py2neo 的库函数，使用了 node_set()和 relationship_set()函数。

node_set()函数主要功能是结点的创建（Node 方法）以及去重（如果已经有相同属性的结点则不需要再次创建）。relationship_set()该函数主要是用于结点之间关系的建立，将不同电影信息中共同的信息进行关联。

查找我们用的是 cypher 语句，对知识图谱中的节点与关系进行提取，然后对于输入字符进行匹配。实现查找的过程中，思路比较清晰。该模块首先将知识图谱中的结点和关系都找到，然后在结点和关系中进行匹配字符串，找到相应的结点或者关系，再根据结点或关系进行输出显示。Knowledge_map()函数既承担了调用知识图谱模块的功能，也要构建知识图谱搜索的界面。第一步是要将其与 Neo4j 图数据库进行连接。而后根据自己的设计创建出要显示的界面，之后调用知识图谱模块。在知识图谱模块完成后，我们建立起自己的查询模块。当文本框中的内容改变，获取用户的查询内容。之后便是调用 show_result()函数，该函数是用来完成对知识图谱的搜索的，函数显示我们查询到的内容。其实现流程如下：



GUI 程序：

GUI 模块实现我们主要采用 python 中自带的 Tkinter 进行编写。同时，GUI 用到了 text（来展示出查找到的节点数据以及与其他节点的关系。text 组件的控制也与其他 GUI 组件不太一样）。gui 函数是用来显示 GUI 主界面的函数，该函数设计的传递参数为函数名，也就是我们该 GUI 实现的六个功能。通过创建一个窗口，在窗口内创建一个框架（Frame 进行创建）来放置我们的一个标题（用 Text 创建）和按钮（通过 Button 进行创建）。f_url_list()与 f_data()函数这两个函数也是实现的功能之一，二者都是从外部获取要显示的数据，利用 ListBox 和 Treeview 部件提供的方法将其显示在创建的窗口之中。

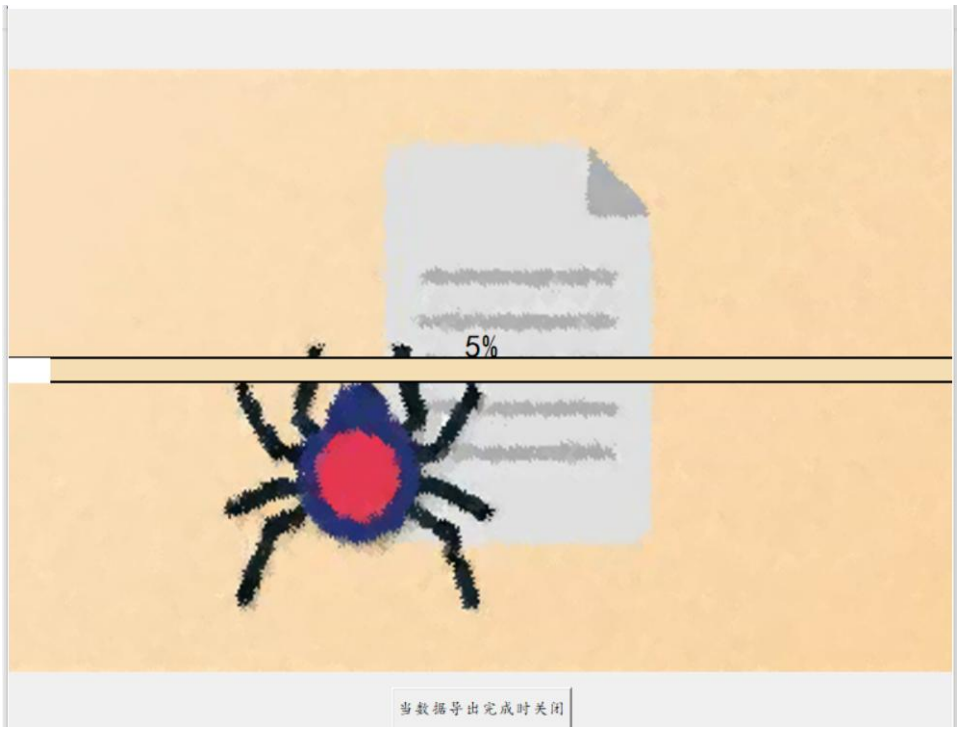
语义分析程序：

词频统计和情感分析模块我们主要依赖于 thulac，中文 bert,transformer（我们 Huggingface 上下载了中文 BERT 预训练模型）。通过读取电影详细信息.csv 文件，我们调用 token_stats.py 统计分词出现次数，并保存到 token_freq.csv。在爬取了最热评论并保存下来后，我们发现对于 top 250 的电影，基本是正向评论过多，我们从 kaggle 上下载了的一份数据集作为检验的依据。我们主要调用了 f_senti_analysis 函数（基于 bert），其是一个预训练的语言表征模型。在 bert 输出层后加上一个全连接层作为 bertForClassification，并利用下载的 kaggle 数据集对其进行微调，，并在输出层（第 12 层）之后加上了一个分类头，实现情感分析。

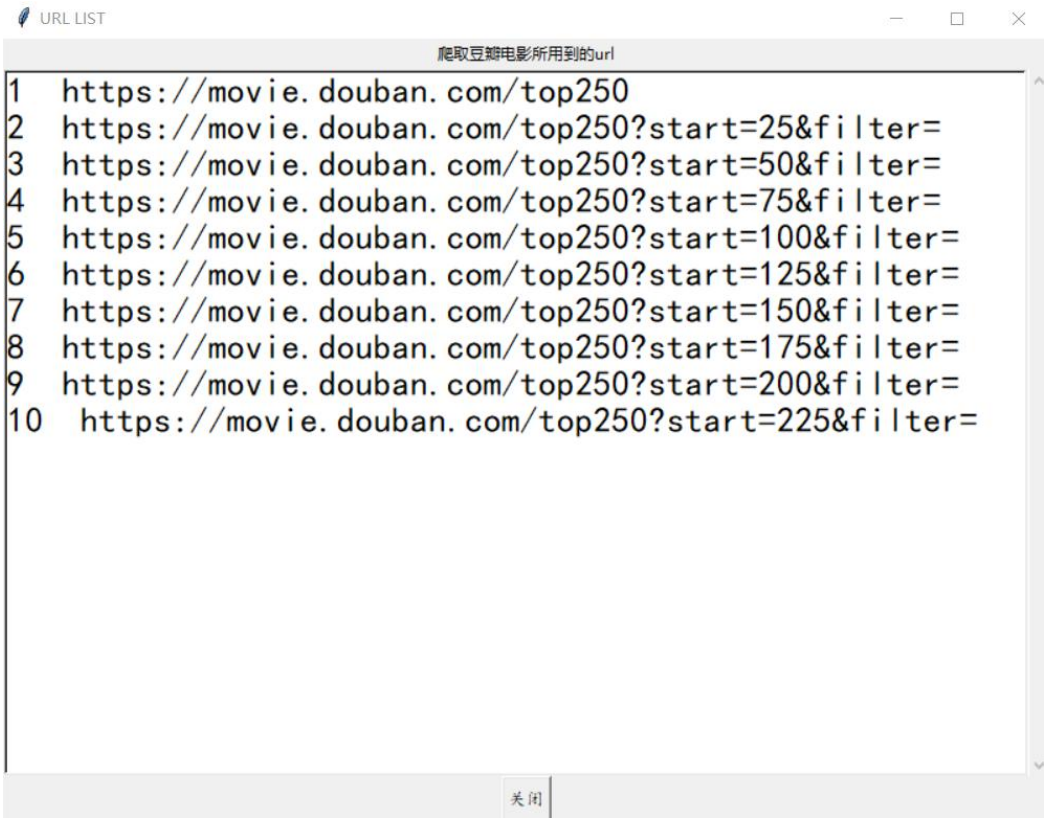
六、效果展示

GUI 界面

点击 **Btn1** 按钮


5%
当数据导出完成时关闭

点击 **Btn2** 按钮

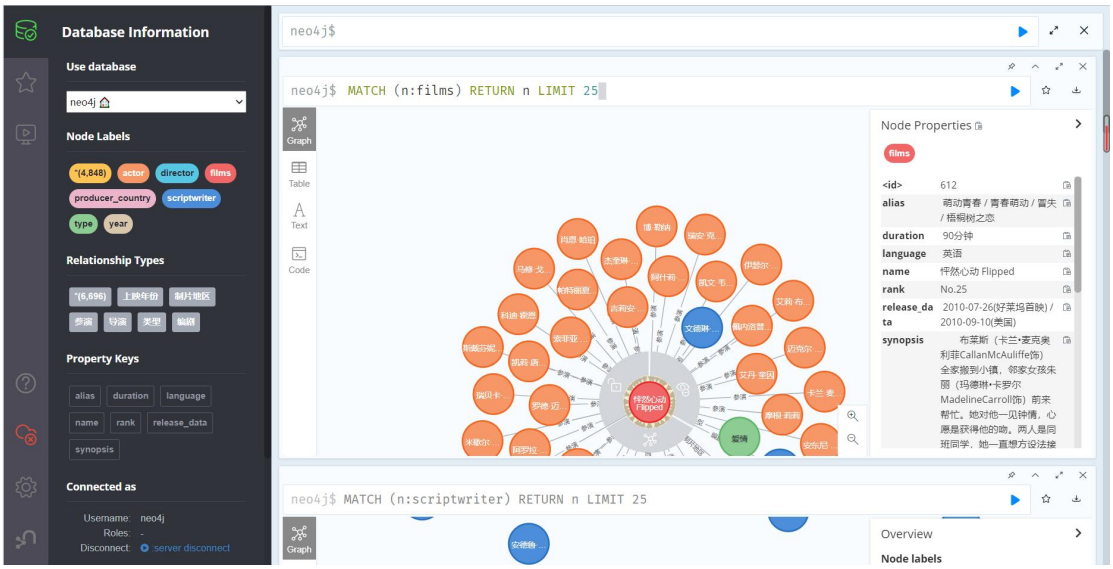

URL LIST
爬取豆瓣电影所用到的url
1 https://movie.douban.com/top250
2 https://movie.douban.com/top250?start=25&filter=
3 https://movie.douban.com/top250?start=50&filter=
4 https://movie.douban.com/top250?start=75&filter=
5 https://movie.douban.com/top250?start=100&filter=
6 https://movie.douban.com/top250?start=125&filter=
7 https://movie.douban.com/top250?start=150&filter=
8 https://movie.douban.com/top250?start=175&filter=
9 https://movie.douban.com/top250?start=200&filter=
10 https://movie.douban.com/top250?start=225&filter=
关闭

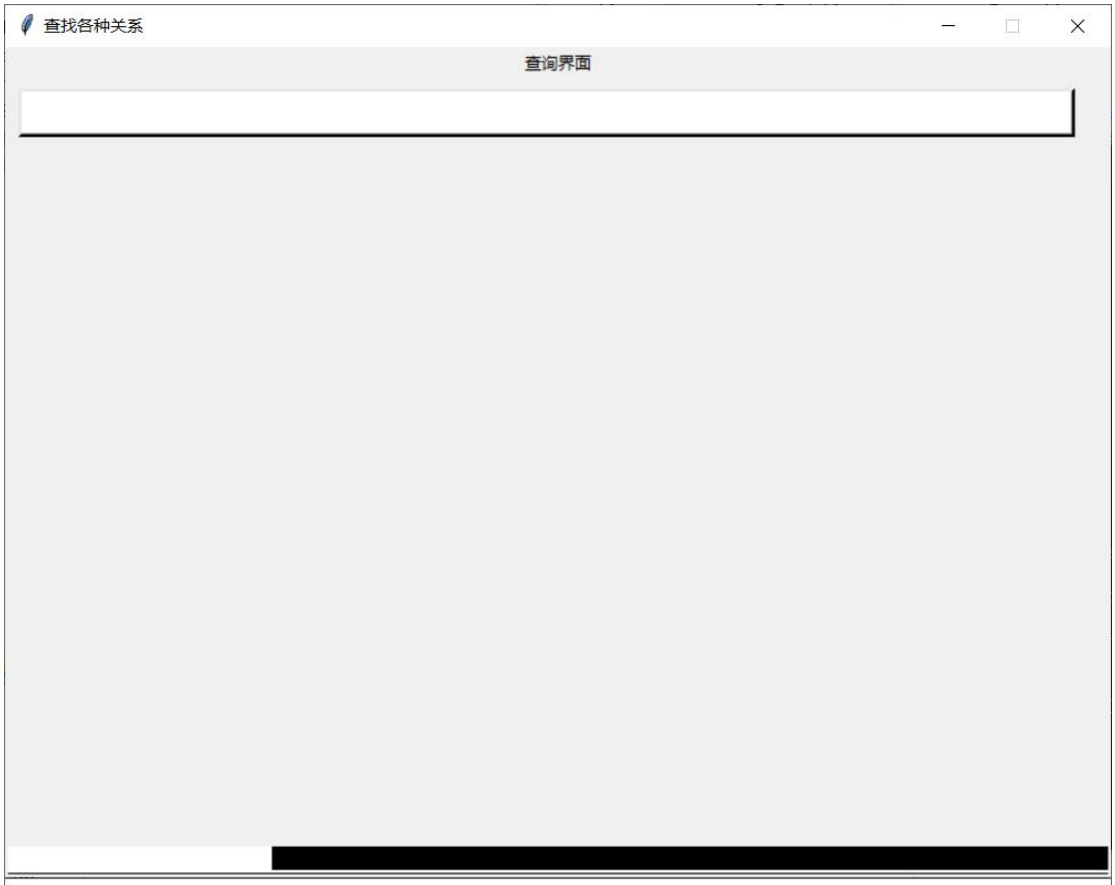
点击 Btn3 按钮

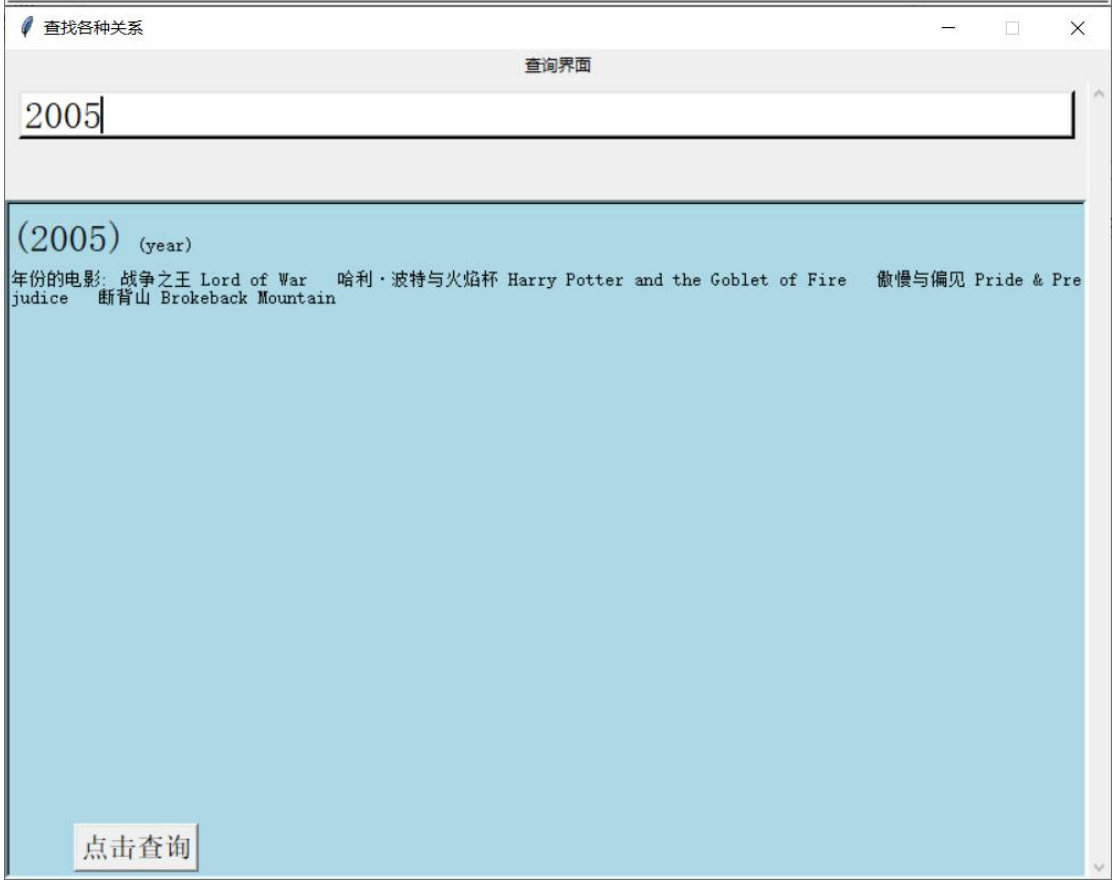
豆瓣Top250电影详细信息

排名	电影名称	年份	导演	编剧	主演
No.1	肖申克的救赎 The Shawshank Red	(1994)	弗兰克·德拉邦特	弗兰克·德拉邦特	蒂姆·罗宾斯 / 摩根·弗里曼 / 鲍勃·冈顿 / 威廉姆·
No.2	霸王别姬	(1993)	陈凯歌	芦苇 / 李碧华	张国荣 / 张丰毅 / 巩俐 / 葛优 / 英达 / 蒋雯丽 /
No.3	阿甘正传 Forrest Gump	(1994)	罗伯特·泽米吉斯	艾瑞克·罗斯 / 温	汤姆·汉克斯 / 罗宾·怀特 / 加里·西尼斯 / 麦凯尔·
No.4	泰坦尼克号 Titanic	(1997)	詹姆斯·卡梅隆	詹姆斯·卡梅隆	莱昂纳多·迪卡普里奥 / 凯特·温丝莱特 / 比利·赞
No.5	这个杀手不太冷 Léon	(1994)	吕克·贝松	吕克·贝松	让·雷诺 / 娜塔莉·波特曼 / 加里·奥德曼 / 丹尼·爱
No.6	美丽人生 La vita è bella	(1997)	罗伯托·贝尼尼	温琴佐·切拉米 /	罗伯托·贝尼尼 / 尼可莱塔·布拉斯基 / 乔治·坎塔
No.7	千与千寻 千と千尋の神隠し	(2001)	宫崎骏	宫崎骏	柊瑠美 / 入野自由 / 夏木真理 / 菅原文太 / 中村
No.8	辛德勒的名单 Schindler's List	(1993)	史蒂文·斯皮尔伯	托马斯·肯尼利 /	连姆·尼森 / 本·金斯利 / 拉尔夫·费因斯 / 卡罗琳·
No.9	盗梦空间 Inception	(2010)	克里斯托弗·诺兰	克里斯托弗·诺兰	莱昂纳多·迪卡普里奥 / 约瑟夫·高登-莱维特 / 艾
No.10	忠犬八公的故事 Hachi: A Dog's Ta	(2009)	拉斯·霍尔斯道姆	斯蒂芬·P·林赛 /	理查·基尔 / 萨拉·罗默尔 / 琼·艾伦 / 罗比·萨布莱
No.11	星际穿越 Interstellar	(2014)	克里斯托弗·诺兰	乔纳森·诺兰 / 克	马修·麦康纳 / 安妮·海瑟薇 / 杰西卡·查斯坦 / 麦
No.12	楚门的世界 The Truman Show	(1998)	彼得·威尔	安德鲁·尼科尔	金·凯瑞 / 劳拉·琳妮 / 艾德·哈里斯 / 诺亚·艾默
No.13	海上钢琴师 La leggenda del piani	(1998)	朱塞佩·托纳多雷	亚利桑德罗·巴里	蒂姆·罗斯 / 普路特·泰勒·文斯 / 比尔·努恩 / 克兰
No.14	三傻大闹宝莱坞 3 Idiots	(2009)	拉吉库马尔·希拉	维德胡·维诺德·乔	阿米尔·汗 / 卡琳娜·卡普尔 / 马达范 / 沙尔曼·乔
No.15	机器人总动员 WALL·E	(2008)	安德鲁·斯坦顿	安德鲁·斯坦顿 /	本·贝尔特 / 艾丽莎·奈特 / 杰夫·格尔林 / 佛莱德·
No.16	放牛班的春天 Les choristes	(2004)	克里斯托夫·巴拉	乔治·沙普罗 / 勒	热拉尔·朱尼奥 / 让-巴蒂斯特·莫尼耶 / 弗朗索瓦
No.17	无间道 無間道	(2002)	刘伟强 / 麦兆辉	麦兆辉 / 庄文强	刘德华 / 梁朝伟 / 黄秋生 / 曾志伟 / 郑秀文 / 陈
No.18	疯狂动物城 Zootopia	(2016)	拜伦·霍华德 / 瑞	拜伦·霍华德 / 瑞	金妮弗·古德温 / 杰森·贝特曼 / 伊德里斯·艾尔巴
No.19	大话西游之大圣娶亲 西遊記大結局之	(1995)	刘镇伟	刘镇伟	周星驰 / 吴孟达 / 朱茵 / 蔡少芬 / 蓝洁瑛 / 莫文
No.20	熔炉 도가니	(2011)	黄东赫	黄东赫 / 孔枝泳	孔刘 / 郑有美 / 金贤秀 / 郑仁絮 / 白承焕 / 张光
No.21	教父 The Godfather	(1972)	弗朗西斯·福特·科	马里奥·普佐 / 弗	马龙·白兰度 / 阿尔·帕西诺 / 詹姆斯·肯恩 / 理查·
No.22	当幸福来敲门 The Pursuit of Happ	(2006)	加布里埃莱·穆奇	斯蒂夫·康拉德	威尔·史密斯 / 贾登·史密斯 / 坦迪·牛顿 / 布莱恩·
No.23	控方证人 Witness for the Prosecut	(1957)	比利·怀尔德	阿加莎·克里斯蒂	泰隆·鲍华 / 玛琳·黛德丽 / 查尔斯·劳顿 / 爱尔莎·
No.24	龙猫 となりのトトロ	(1988)	宫崎骏	宫崎骏	日高法子 / 坂本千夏 / 糸井重里 / 岛本须美 / 北

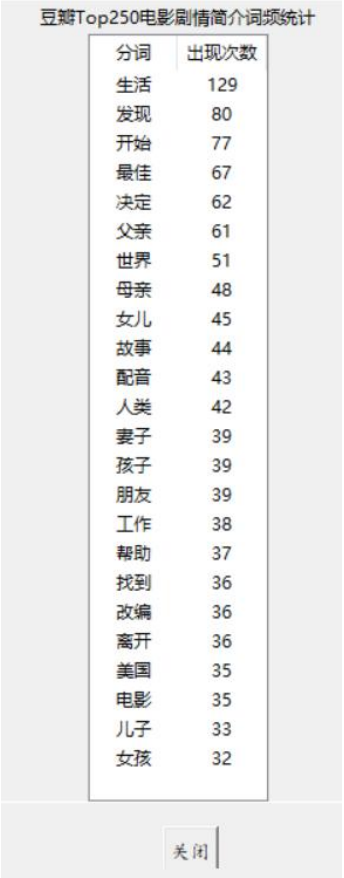
点击 Btn4 按钮



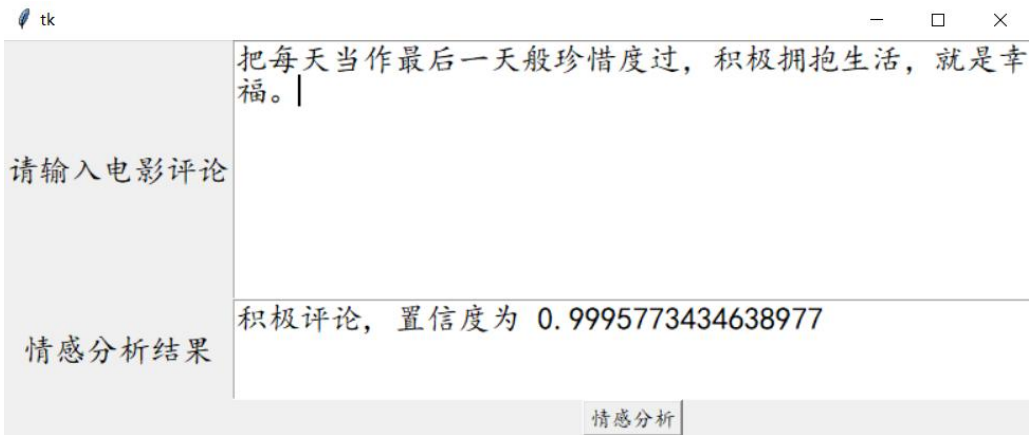

查找各种关系
查询界面


查找各种关系
查询界面
2005
(2005) (year)
年份的电影: 战争之王 Lord of War 哈利·波特与火焰杯 Harry Potter and the Goblet of Fire 傲慢与偏见 Pride & Prejudice 断背山 Brokeback Mountain
点击查询

点击 Btn5 按钮


豆瓣Top250电影剧情简介词频统计
分词 出现次数
生活 129
发现 80
开始 77
最佳 67
决定 62
父亲 61
世界 51
母亲 48
女儿 45
故事 44
配音 43
人类 42
妻子 39
孩子 39
朋友 39
工作 38
帮助 37
找到 36
改编 36
离开 36
美国 35
电影 35
儿子 33
女孩 32
关闭

点击 Btn6 按钮


tk
请输入电影评论
把每天当作最后一天般珍惜度过，积极拥抱生活，就是幸福。
情感分析结果
积极评论，置信度为 0.9995773434638977
情感分析

七、困难及方案

反爬障碍

由于爬虫爬取网页数据时由于频繁访问，会被锁定 IP，我们使用了软件技术课程设计指导书提供参考 github 上的 IP 代理池源码。该代理池利用数据库进行存储，代理 IP 不断更新，剔除不可用 IP 和添加新的 IP，因此能够有效地获取有效代理 IP，与爬虫程序的交互也很友好。

当然配置该代理池的过程也并非一帆风顺，其中遇到了 server 和 schedule 同时启动，端口 PORT 的更新，Redis 数据库的下载等一系列困难，中途还尝试了其他开源的高收藏的 IP 代理池源码。总的来说按照 github 说明文档配置，需要一步一步按部就班，小心翼翼。

除此以外，对于 user-agent 我们也是随机获取的，用来伪装。我们发现单一的浏览器访问频率过高也会导致被反爬机制检测，从而使爬虫不能正常工作，因此我们从网上找了很多不同浏览器的 User-Agent 添加到我们的 headers 中，每次爬取随机选取一个 User-Agent 添加到请求头中。纵使如此，我们发现使用校园网频繁后，IP 代理池 schedule 爬不了了，全被挡了，并且在教学楼的效果优于学生宿舍。对此，我们改用手机热点解决了这一问题。

知识图谱遇到的问题

知识图谱我们使用的是 neo4j，由于之前没有用到过，需要再网上搜集资料学习。遇到的第一个问题就是 neo4j 语句的选择。neo4j 是通过 cypher 语句来实现的，但我对这个语句不是很了解因此代码写起来很费劲。之后我找到了 neo4j 在 python 上的库 py2neo，可以直接按照 python 的语法进行图数据库的操作

（节点的构建、删除、更改，以及关系的构建、添加等等），直接调用函数，方便快捷。

第二个问题是 csv 文件的读取。在网上搜集资料，发现需要 cypher 语言才能使用 neo4j 读取 csv 文件，但这对于较复杂的节点关系的实现。因此我没有采用这种方式读取，而是通过 python 中的 csv 库对文件进行转换，最后转换成字典的列表。

语义分析困难

情感分析是 NLP 中常见的分类任务。起初我们利用 THULAC 分词库对训练集（从 Kaggle 上下载的电影评论）进行中文分词，再将分词后的词符作为输入，利用 word2vec 进行词符到 embedding 向量的转换。然后我们用文本卷积神经网络对 embedding 向量进行特征提取最后分类。但是实验表明我们的模型在训练集上表现很好，但是在测试集（即我们爬取到的电影评论）上泛化能力不佳。经过思考分析，我们认为，word2vec 将相同的词符映射到相同的 embedding 向量，而忽略了词符在语境中的含义。因此，我们经过文献查阅，发现 BERT 预训练模型很好地考虑了词符在语境中的意义（BERT 采用 MLM 和 NSP 进行预训练），因此我们从 Huggingface 上下载了中文 BERT 预训练模型，并在输出层（第 12 层）之后加上了一个分类头。我们抛弃了之前用 THULAC 分词得到的结果，使用 BERT 预训练后得到的 embedding 矩阵。实验结果表明，我们用训练集对 BERT 进行微调后，仍然能在测试集上取得良好的泛化性能。

GUI 设计问题

GUI 我们使用的是 TKinter。在 GUI 学习过程中，我们发现了 GUI 控制与之前玩过的一个触控显示屏的程序控制语句有异曲同工之妙，例如控件、框架、事件等等。在 GUI 实现过程中，也遇到了一些问题。例如图片的插入。一开始我们想直接在框架中插入图片，但是显示出来的图片严重影响了布局，改了几个参数也无法调好。之后我用了画布在进行图片的插入，对于画布的控制方便了布局。

同时，我们在爬虫界面加入一张背景图像是遇到了 bug。经过 debug，我们发现原因是我们代码中“from tkinter import *” 以及“from PIL import Image”， 忽略了 tkinter 中也有 Image 这个模块，就导致了模块名的冲突，因此报错。这提醒我们：“from xxx import *”并不是一个良好的代码习惯，在今后的代码比那些过程中需要避免。

环境配置

一个大工程代码的实现这一过程中难免会遇到各种问题，需要经常 debug，甚至是环境的配置也是极为繁琐的一个过程。例如一开始为了安装图形图像数据库，安装 java 的过程中我们遇到了版本冲突问题。因为之前都是用的 java8 版本，为了运行 neo4j，需要下载 java11。因此将以前的 java8 删除了，然后重新下载 java11。下载并安装了之后，检测到没有版本号。出现一系列错误，提示没有找到 java SE 对应版本。解决这一问题涉及到了环境变量 path 优先级配置，残留 java 文件删除等等。又如语义分析，词频统计部分需要用到 transformer，而其依赖 pytorch 库，pytorch 的配置是众所周知较为繁琐的。

总而言之，在完成本课程设计的时候遇到的或大或小的问题远远超乎想象，例如知识图谱的构建以及匹配查询的实现，极大的考查了我们网上查找资料来解决的能力。也极大普及我们的知识面，例如 IP 代理池部分 debug 时了解了部分端口知识，Redis 数据库的操作指令等等。