基于电影知识图谱的问答系统项目文档

1. **项目调研**
   1. 项目背景

随着移动互联网的发展，万物互联成为了可能，这种互联所产生的数据也在爆发式地增长，而且这些数据恰好可以作为分析关系的有效原料。如果说以往的智能分析专注在每一个个体上，在移动互联网时代则除了个体，这种个体之间的关系也必然成为我们需要深入分析的很重要一部分。 在一项任务中，只要有关系分析的需求，知识图谱就“有可能”派的上用场。从一开始的Google搜索，到现在的聊天机器人、大数据风控、证券投资、智能医疗、自适应教育、推荐系统，无一不跟知识图谱相关。它在技术领域的热度也在逐年上升。

* 1. 知识图谱的概念

知识图谱是由Google公司在2012年提出来的一个新的概念。从学术的角度，我们可以对知识图谱给一个这样的定义：“知识图谱本质上是语义网络（Semantic Network）的知识库”。但这有点抽象，所以换个角度，从实际应用的角度出发其实可以简单地把知识图谱理解成多关系图（Multi-relational Graph）。

多关系图一般包含多种类型的节点和多种类型的边。比如左下图表示一个经典的图结构，右边的图则表示多关系图，因为图里包含了多种类型的节点和边。这些类型由不同的颜色来标记。在知识图谱里，我们通常用“实体（Entity）”来表达图里的节点、用“关系（Relation）”来表达图里的“边”。实体指的是现实世界中的事物比如人、地名、概念、药物、公司等，关系则用来表达不同实体之间的某种联系，比如人-“居住在”-北京、张三和李四是“朋友”、逻辑回归是深度学习的“先导知识”等等。

现实世界中的很多场景非常适合用知识图谱来表达。比如一个社交网络图谱里，我们既可以有“人”的实体，也可以包含“公司”实体。人和人之间的关系可以是“朋友”，也可以是“同事”关系。人和公司之间的关系可以是“现任职”或者“曾任职”的关系。 类似的，一个风控知识图谱可以包含“电话”、“公司”的实体，电话和电话之间的关系可以是“通话”关系，而且每个公司它也会有固定的电话。

* 1. 知识图谱的表示

知识图谱应用的前提是已经构建好了知识图谱，也可以把它认为是一个知识库。这也是为什么它可以用来回答一些搜索相关问题的原因，比如在Google搜索引擎里输入“Who is the wife of Bill Gates?”，我们直接可以得到答案-“Melinda Gates”。这是因为我们在系统层面上已经创建好了一个包含“Bill Gates”和“Melinda Gates”的实体以及他俩之间关系的知识库。所以，当我们执行搜索的时候，就可以通过关键词提取（"Bill Gates", "Melinda Gates", "wife"）以及知识库上的匹配可以直接获得最终的答案。这种搜索方式跟传统的搜索引擎是不一样的，一个传统的搜索引擎它返回的是网页、而不是最终的答案，所以就多了一层用户自己筛选并过滤信息的过程。

在现实世界中，实体和关系也会拥有各自的属性，比如人可以有“姓名”和“年龄”。当一个知识图谱拥有属性时，我们可以用属性图（Property Graph）来表示。下面的图表示一个简单的属性图。李明和李飞是父子关系，并且李明拥有一个138开头的电话号，这个电话号开通时间是2018年，其中2018年就可以作为关系的属性。类似的，李明本人也带有一些属性值比如年龄为25岁、职位是总经理等。

这种属性图的表达很贴近现实生活中的场景，也可以很好地描述业务中所包含的逻辑。除了属性图，知识图谱也可以用RDF来表示，它是由很多的三元组（Triples）来组成。RDF在设计上的主要特点是易于发布和分享数据，但不支持实体或关系拥有属性，如果非要加上属性，则在设计上需要做一些修改。目前来看，RDF主要还是用于学术的场景，在工业界我们更多的还是采用图数据库（比如用来存储属性图）的方式。

* 1. 知识抽取

知识图谱的构建是后续应用的基础，而且构建的前提是需要把数据从不同的数据源中抽取出来。对于垂直领域的知识图谱来说，它们的数据源主要来自两种渠道：一种是业务本身的数据，这部分数据通常包含在公司内的数据库表并以结构化的方式存储；另一种是网络上公开、抓取的数据，这些数据通常是以网页的形式存在所以是非结构化的数据。

前者一般只需要简单预处理即可以作为后续AI系统的输入，但后者一般需要借助于自然语言处理等技术来提取出结构化信息。比如在上面的搜索例子里，Bill Gates和Malinda Gate的关系就可以从非结构化数据中提炼出来，比如维基百科等数据源。

信息抽取的难点在于处理非结构化数据。在下面的图中，我们给出了一个实例。左边是一段非结构化的英文文本，右边是从这些文本中抽取出来的实体和关系。在构建类似的图谱过程当中，主要涉及以下几个方面的自然语言处理技术：

a. 实体命名识别（Name Entity Recognition）

b. 关系抽取（Relation Extraction）

c. 实体统一（Entity Resolution）

d. 指代消解（Coreference Resolution）

首先是实体命名识别，就是从文本里提取出实体并对每个实体做分类/打标签：比如从上述文本里，我们可以提取出实体-“NYC”，并标记实体类型为 “Location”；我们也可以从中提取出“Virgil’s BBQ”，并标记实体类型为“Restarant”。这种过程称之为实体命名识别，这是一项相对比较成熟的技术，有一些现成的工具可以用来做这件事情。其次，我们可以通过关系抽取技术，把实体间的关系从文本中提取出来，比如实体“hotel”和“Hilton property”之间的关系为“in”；“hotel”和“Time Square”的关系为“near”等等。另外，在实体命名识别和关系抽取过程中，有两个比较棘手的问题：一个是实体统一，也就是说有些实体写法上不一样，但其实是指向同一个实体。比如“NYC”和“New York”表面上是不同的字符串，但其实指的都是纽约这个城市，需要合并。实体统一不仅可以减少实体的种类，也可以降低图谱的稀疏性（Sparsity）；另一个问题是指代消解，也是文本中出现的“it”, “he”, “she”这些词到底指向哪个实体，比如在本文里两个被标记出来的“it”都指向“hotel”这个实体。

* 1. 知识图谱的存储

知识图谱主要有两种存储方式：一种是基于RDF的存储；另一种是基于图数据库的存储。它们之间的区别如下图所示。RDF一个重要的设计原则是数据的易发布以及共享，图数据库则把重点放在了高效的图查询和搜索上。其次，RDF以三元组的方式来存储数据而且不包含属性信息，但图数据库一般以属性图为基本的表示形式，所以实体和关系可以包含属性，这就意味着更容易表达现实的业务场景。



根据最新的统计（2018年上半年），图数据库仍然是增长最快的存储系统。相反，关系型数据库的增长基本保持在一个稳定的水平。同时，我们也列出了常用的图数据库系统以及他们最新使用情况的排名。 其中Neo4j系统目前仍是使用率最高的图数据库，它拥有活跃的社区，而且系统本身的查询效率高，但唯一的不足就是不支持准分布式。相反，OrientDB和JanusGraph（原Titan）支持分布式，但这些系统相对较新，社区不如Neo4j活跃，这也就意味着使用过程当中不可避免地会遇到一些刺手的问题。如果选择使用RDF的存储系统，Jena或许一个比较不错的选择。

* 1. 知识图谱的架构

知识图谱的架构涉及知识表示、知识获取、知识处理和知识利用等多个方面。

一般情况下，知识图谱构建流程如下：首先确定知识表示模型，然后根据不同的数据来源选择不同的知识获取手段并导入相关的知识，接着利用知识推理、知识融合、知识挖掘等技术构建相应的知识图谱，最后根据不同应用场景设计知识图谱的表现方式，比如：语义搜索、智能推荐、智能问答等。

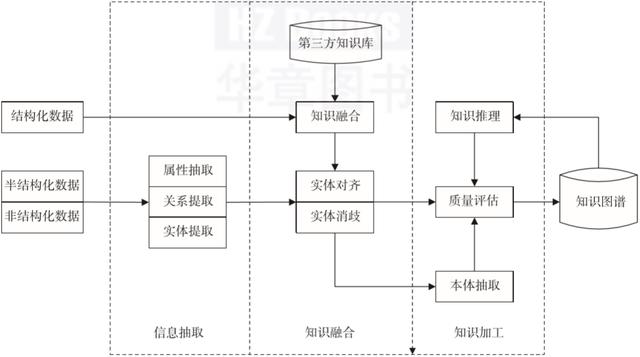
从逻辑上，我们可以将知识图谱划分为两个层次：数据层和模式层。数据层可以是以事实为单位存储的数据库，可以选用的图数据库有RDF4j、Virtuoso、Neo4j等三元组。

<实体，关系，实体>或者<实体，属性，属性值>可以作为基本的表达方式，存储在图数据库中。模式层建立在数据层之上，是知识图谱的核心。通常，通过本体库来管理数据层，本体库的概念相当于对象中“类”的概念。借助本体库，我们可以管理公理、规则和约束条件，规范实体、关系、属性这些具体对象间的关系。

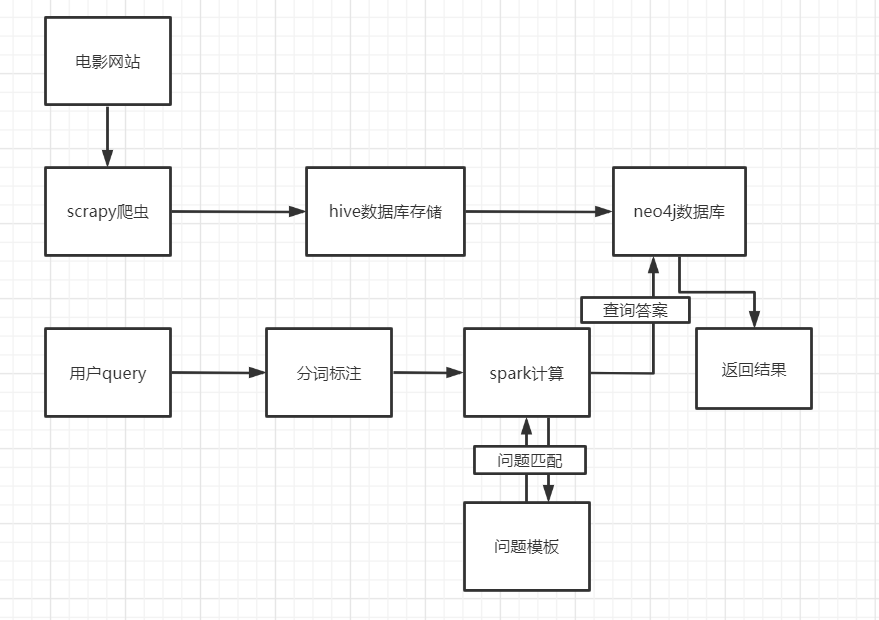
知识图谱有自顶向下和自底向上两种构建方式。自顶向下构建是指借助百科类数据源，提取本体和模式信息，并加入知识库中。自底向上构建是指借助一定的技术手段，从公开的数据中提取资源，选择其中置信度较高的信息，经人工审核后，加入知识库中。

在知识图谱发展初期，多数企业和机构采用自顶向下的方式构建知识图谱，目前大多企业采用自底向上的方式构建知识图谱。

知识图谱的架构如图所示：



1. **架构设计**
   1. 架构设计



项目目标是实现一个基于电影知识图谱的问答系统。项目架构设计如上图。首先是使用爬虫爬取数据，然后是对数据进行存储与转存，使用neo4j图数据库构建知识图谱。问答方面，首先使用中文分词框架对用户输入的问题进行分词与词性标注，接着使用spark的朴素贝叶斯分类器与问题模板进行匹配，问题匹配成系统可以识别的格式之后向neo4j数据库查询结果，数据库返回查询结果，展示给用户。

1. 环境搭建

系统环境：windows10 python3.7

Scrapy包安装：<https://blog.csdn.net/qq_42543250/article/details/81347368>

Hive安装：<https://blog.csdn.net/weixin_43861175/article/details/90372513>

Neo4j安装：<https://blog.csdn.net/vivian_ll/article/details/89312526>

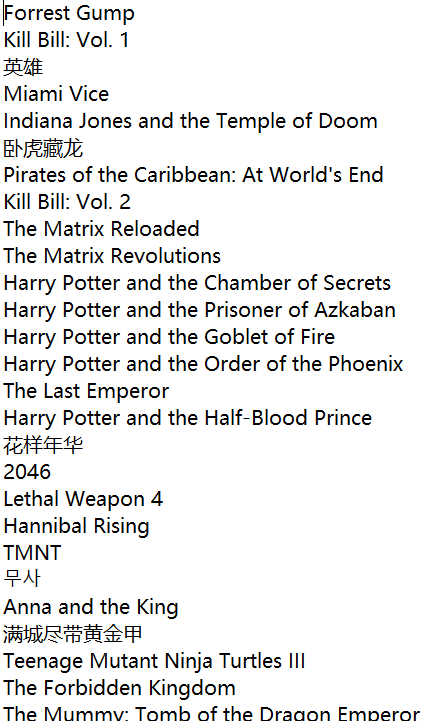
Py2neo安装：<https://blog.csdn.net/yangfengling1023/article/details/82049715>

Spark安装：https://blog.csdn.net/proplume/article/details/79798289

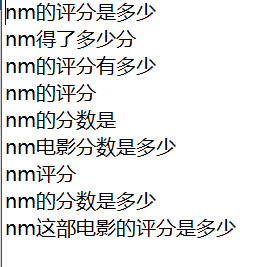
1. 数据采集

使用scrapy爬虫框架编写爬虫程序，从豆瓣电影网爬取电影数据，数据格式存储按照，电影名称，电影类型，主演，简介来存储，爬取的数据存储在hive数据库当中。

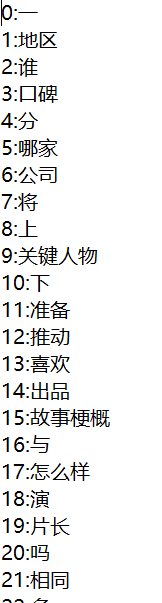
电影名称：



问题模板



自定义字典：



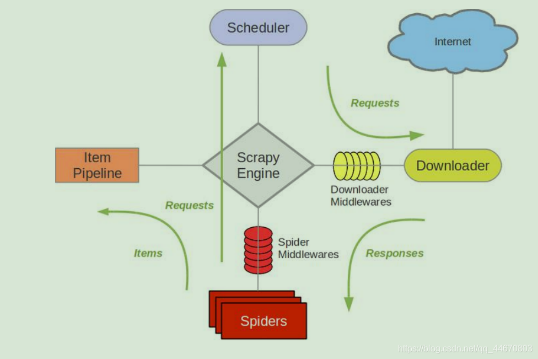
1. 数据处理

将爬取的数据从hive导出到csv文件中，然后将csv文件导入neo4j数据库中构建可视化的知识图谱。

1. 算法开发

爬虫设计：

Scrapy 是用纯 Python 实现一个为了爬取网站数据、提取结构性数据而编写的应用框架，用途非常广泛。框架的力量，用户只需要定制开发几个模块就可以轻松的实现一个爬虫，用来抓取网页内容以及各种图片，非常之方便。Scrapy 使用了 Twisted（其主要对手是 Tornado）异步网络框架来处理网络通讯，可以加快我们的下载速度，不用自己去实现异步框架，并且包含了各种中间件接口，可以灵活的完成各种需求。流程图：



算法步骤：

（1）新建项目 （Scrapy startproject xxx）：新建一个新的爬虫项目

（2）明确目标 （编写 items.py）：明确你想要抓取的目标

（3）制作爬虫 （spiders/xxspider.py）：制作爬虫开始爬取网页

（4）存储内容 （pipelines.py）：设计管道存储爬取内容

jieba分词：

中文分词是中文文本处理的一个基础步骤，也是中文人机自然语言交互的基础模块，在进行中文自然语言处理时，通常需要先进行分词。用户的查询语句要先进行中文分词与标注然后才能与问题模板进行匹配构建真正的查询语句。jieba分词算法使用了基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能生成词情况所构成的有向无环图(DAG), 再采用了动态规划查找最大概率路径，找出基于词频的最大切分组合，对于未登录词，采用了基于汉字成词能力的HMM模型，使用了Viterbi算法。

jieba分词支持三种分词模式：

1. 精确模式, 试图将句子最精确地切开，适合文本分析：

2. 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；

3. 搜索引擎模式，在精确模式的基础上，对长词再词切分，提高召回率，适合用于搜索引擎分词。

jiaba分词还支持繁体分词和支持自定义分词。

参考：<https://www.jianshu.com/p/2cccb07d9a4e>

分词算法步骤：

1、拿到原始句子（问题）

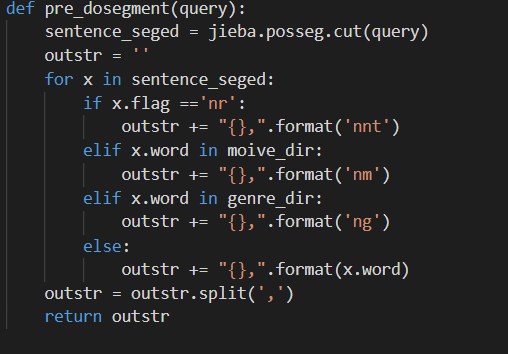
2、对原句子进行抽象，例如将人名用nr替换并抽象句子，比如张学友的生日是多少替换成nr的生日是多少

3、抽象句子匹配问题模板（一堆问题数据集合由Spark进行训练并计算），比如 nr 生日

4、问题模板还原成最终的问题，比如 nr 生日，替换其中的nr=张学友，最后效果就是 张学友 生日

5、拿到问题后，去图形数据库neo4j中查找问题的答案。

关键代码：



Spark朴素贝叶斯分类：

使用spark的朴素贝叶斯分类器对问题进行匹配，得到相应的问题模板进行查询。Byesian算法是统计学的分类方法，它是一种利用概率统计知识进行分类的算法。在许多场合，朴素贝叶斯分类算法可以与决策树和神经网络分类算法想媲美，该算法能运用到大型数据库中，且方法简单，分类准确率高，速度快，这个算法是从贝叶斯定理的基础上发展而来的，贝叶斯定理假设不同属性值之间是不相关联的。但是现实说中的很多时候，这种假设是不成立的，从而导致该算法的准确性会有所下降。

参考：<https://blog.csdn.net/lsshlsw/article/details/45174391>

贝叶斯分类算法步骤：

1、生成（或外部文件加载）训练集样本【样本：LabelPoint类型，再细一点就是double数组构造的稠密/稀疏向量】

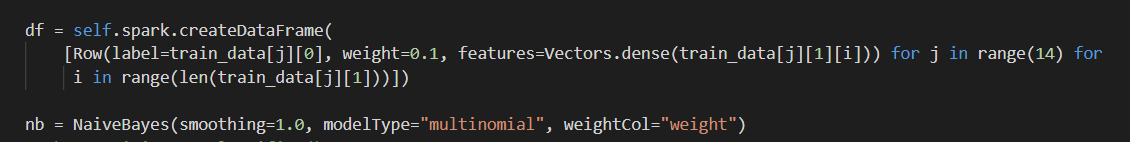
2、生成（或外部参数传进）测试数据样本【样本：LabelPoint类型，再细一点就是double数组构造的稠密/稀疏向量】

3、根据训练样本集合由SparkContext实例创建出一个可以被并行操作的分布式数据集RDD

4、贝叶斯分类器训练RDD

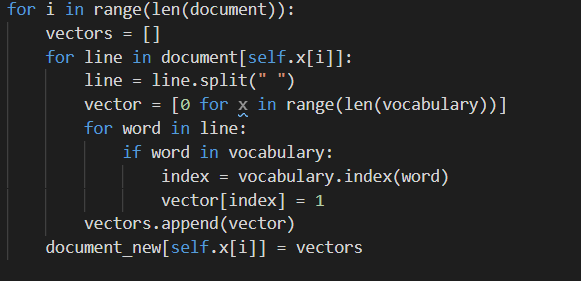
5、贝叶斯分类器拿着测试数据样本跟训练的数据进行概率预测（predict），最后返回我们定义的类别标签号

关键代码：



训练集划分关键代码：

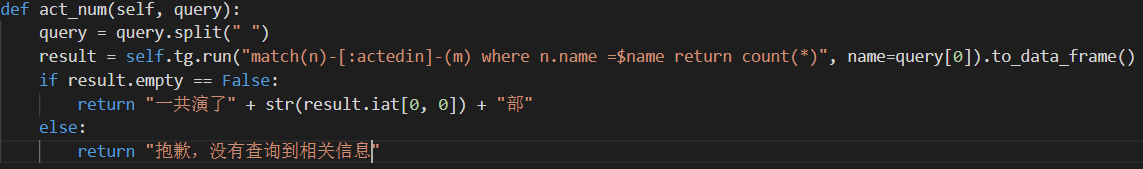
将分词后的每一行数据转换成行向量与问题模板进行匹配。



Neo4j问题查询：

使用neo4j的cypher查询语句进行查询匹配；

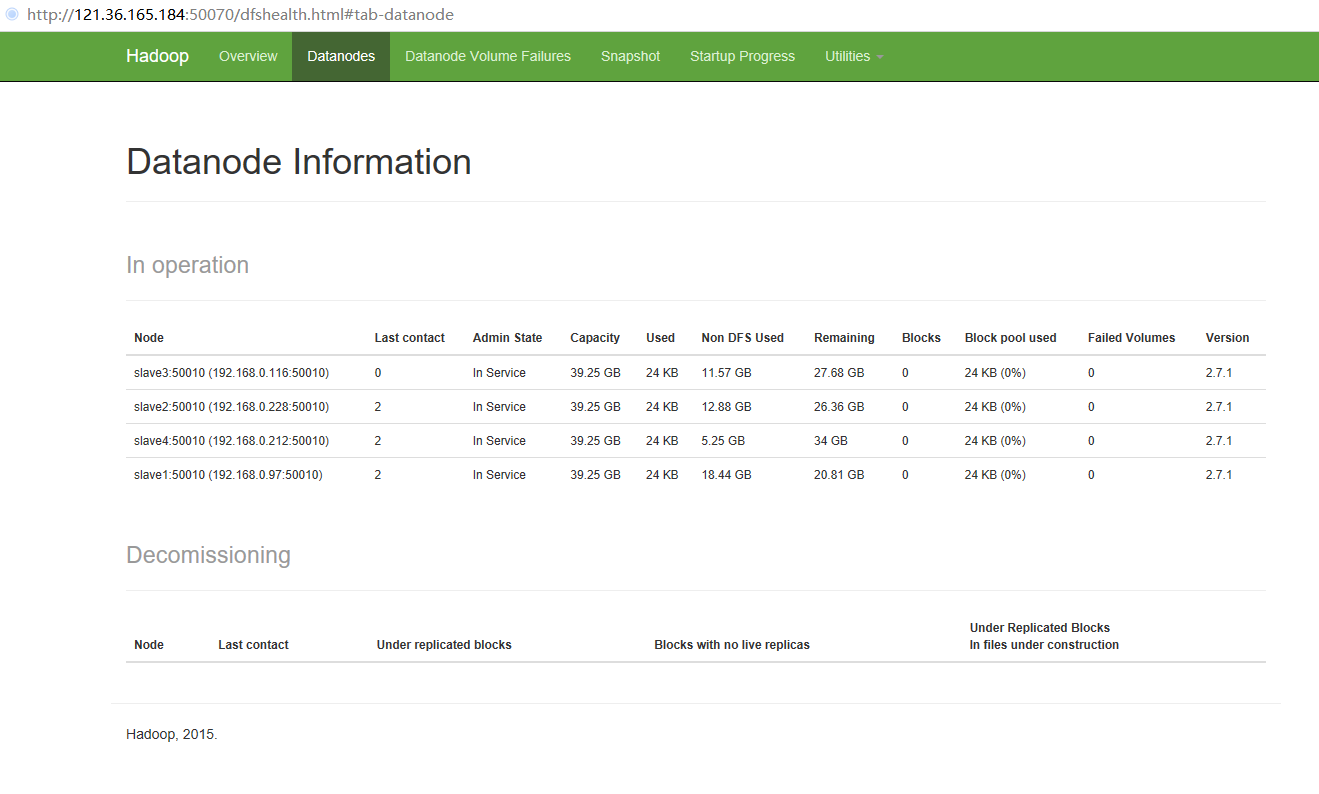
示例代码：



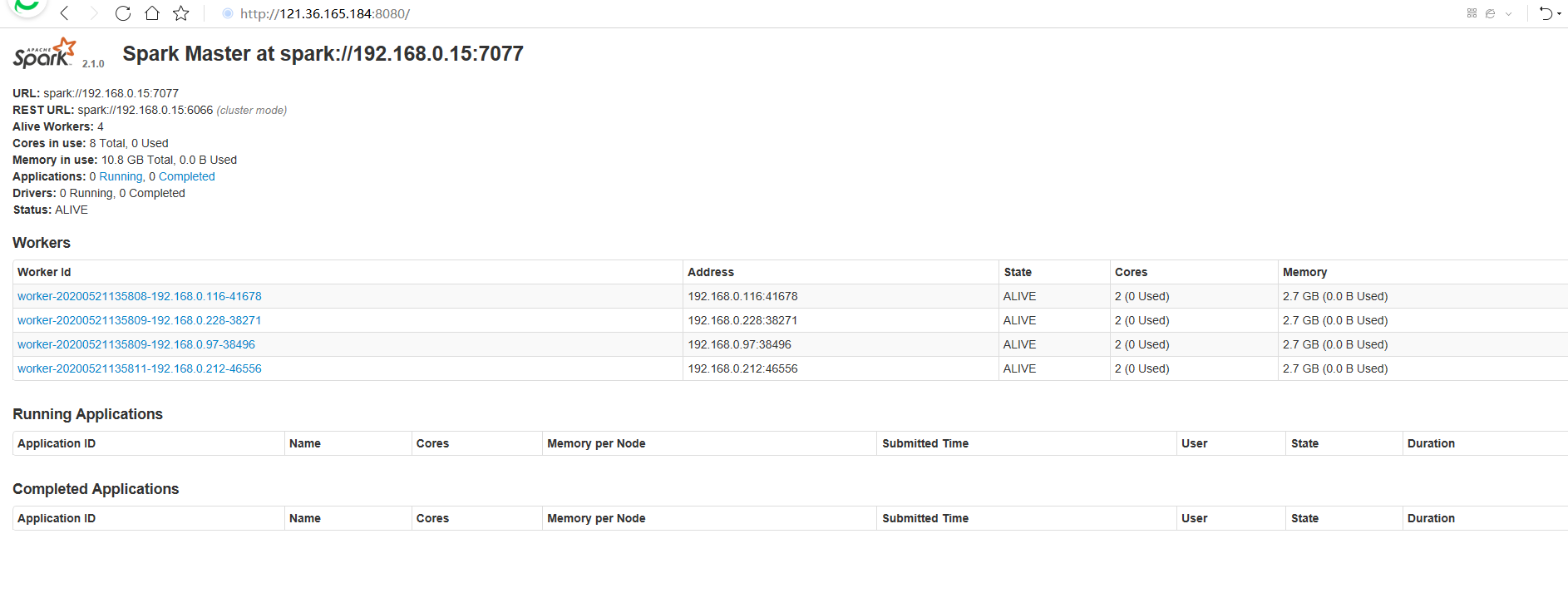
1. **项目实现及效果展示**
2. 分布式环境搭建：

使用5台华为云主机搭建了一个分布式的hadoop和spark平台：

Hadoop master节点状态展示：



Spark主节点状态展示：

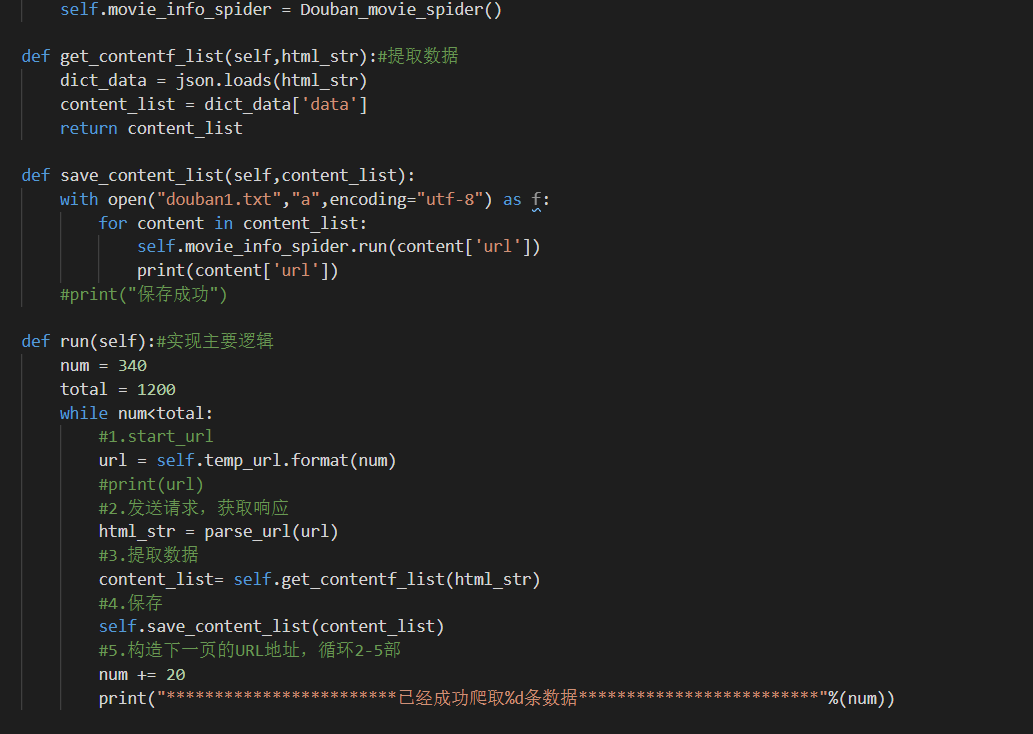


1. **数据爬取**

**爬虫逻辑：**

**爬取url-list：**

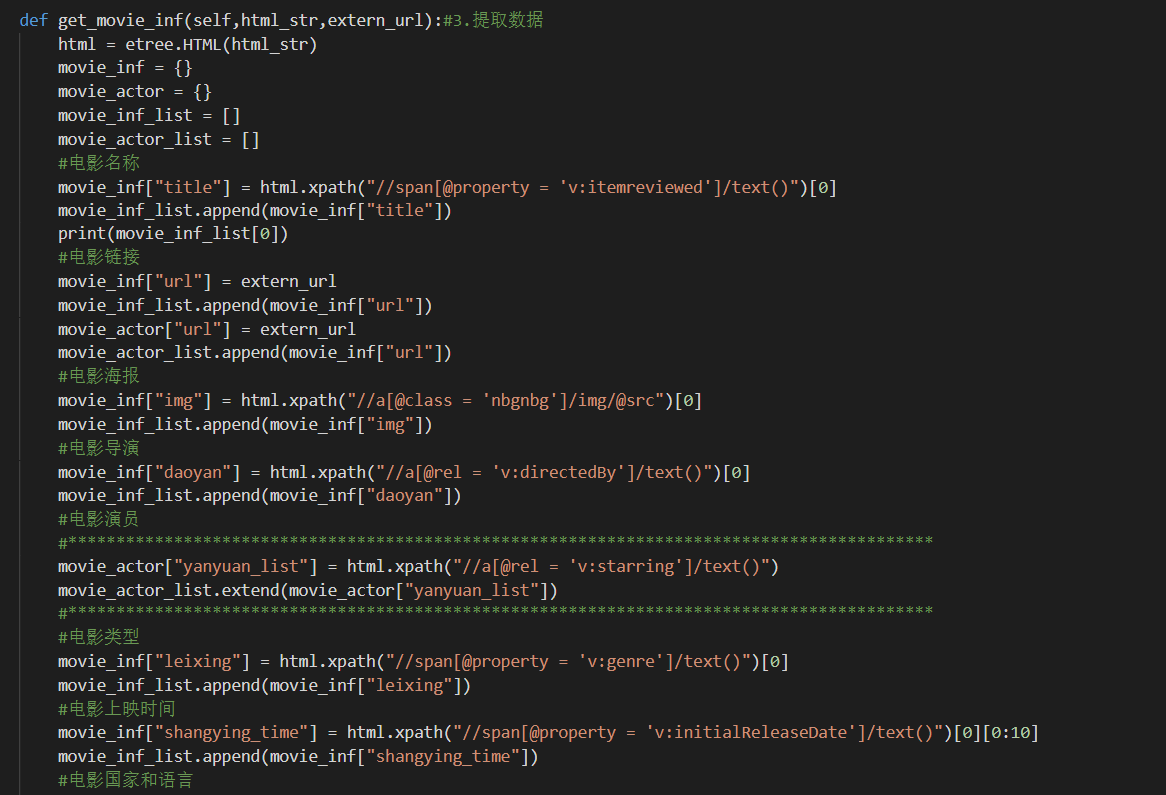
**这一步是为了爬取url构建一个urllist，为下一步的解析做准备。**

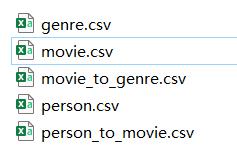
****

**爬取的urllist：**

****

**解析url，通过xpath匹配获取电影信息并保存到csv文件中：**

****

****

genre.csv存储的是电影类别：



Movie.csv存储的是电影名称，电影简介，电影类型，电影评分，电影上映日期：



Movietogenre.csv存储的是电影id到类别id的映射

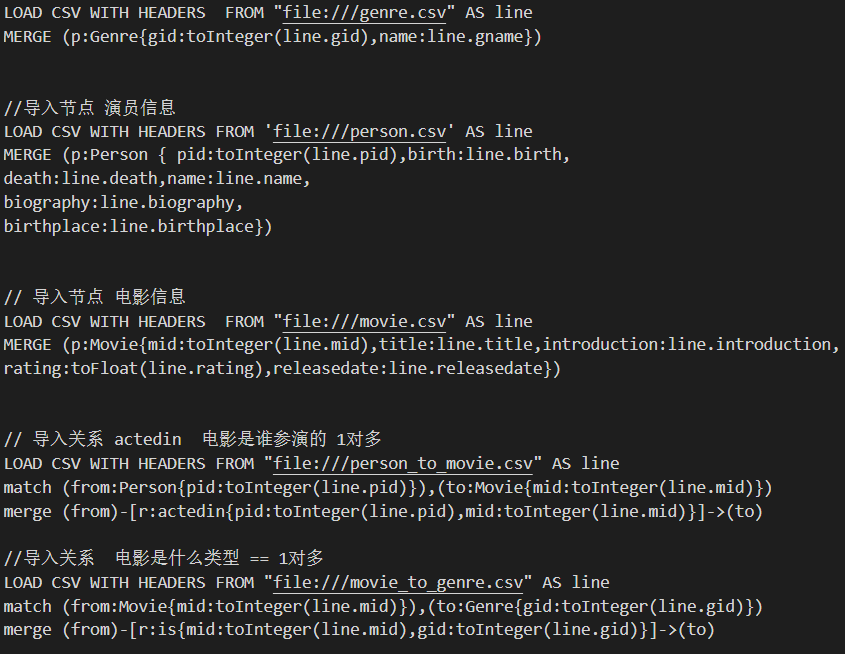
Person.csv存储的是演员信息：



Persontomovie.csv存储的是演员id和电影id的映射关系。

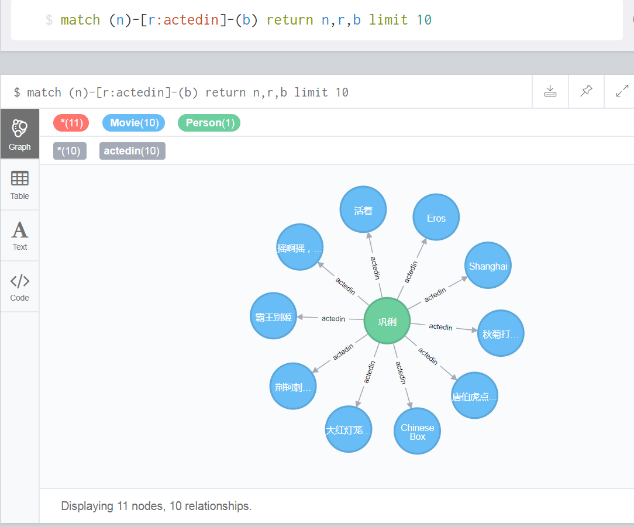
1. 知识图谱搭建

使用cypher语句将csv文件导入到neo4j图数据库中。



注意要做一些类型转换，比如将电影id转化成int类型，电影评分转换成float类型。

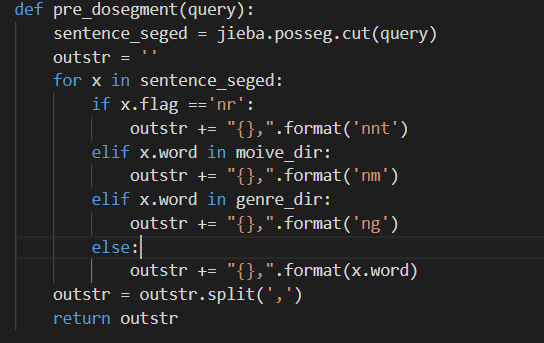
**导入完成后启动neo4j，查询效果展示：**



1. **Query匹配：**

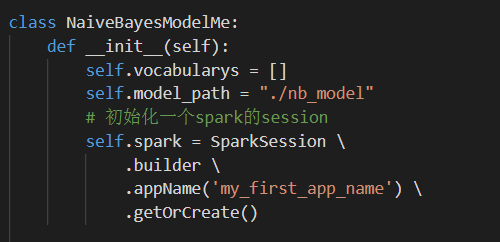
**1.jieba分词：**

**对数据进行分词处理，对于电影，类型采用自定义词典，这样就可以对query语句进行分词目的是为下面的分类做准备。**

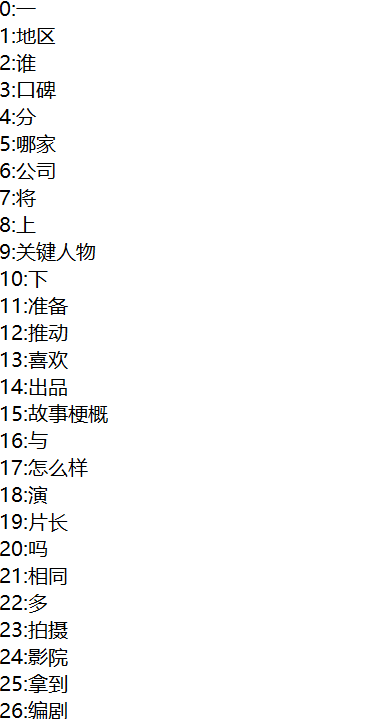
****

**2.spark的朴素贝叶斯分类器：**

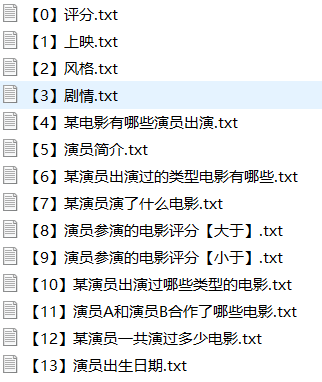
**初始化一个spark session：**

****

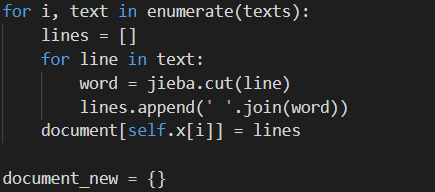
**Vocabulary文件存储高频词库，作为问题匹配模板。**

****

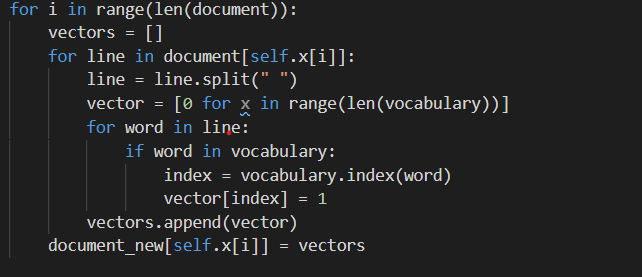
**14种问题模板：**

****

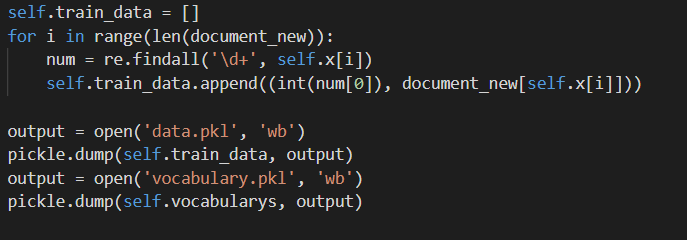
**对问题文件中的词进行分词操作并将文件名作为字典索引的key**

****

**对分词后的数据文件的每一行进行和高频词库的匹配，将数据文件的每一行转为一个行向量：**

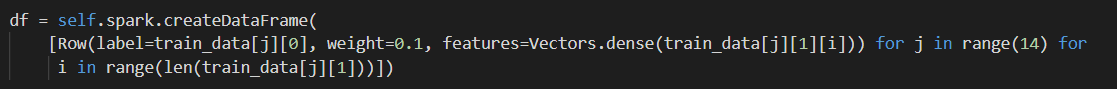
****

**提取数据文件Title中的数字为类别标签,将其保存为训练集的形式：**

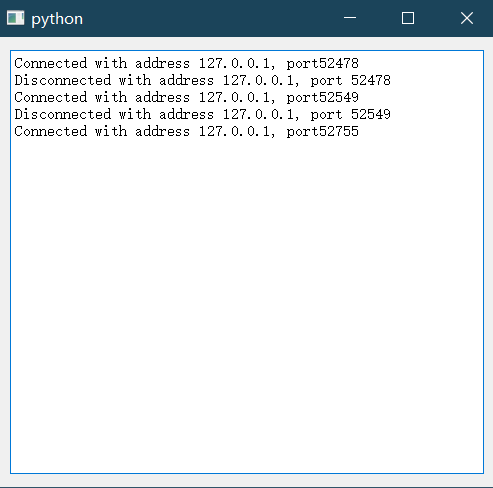
****

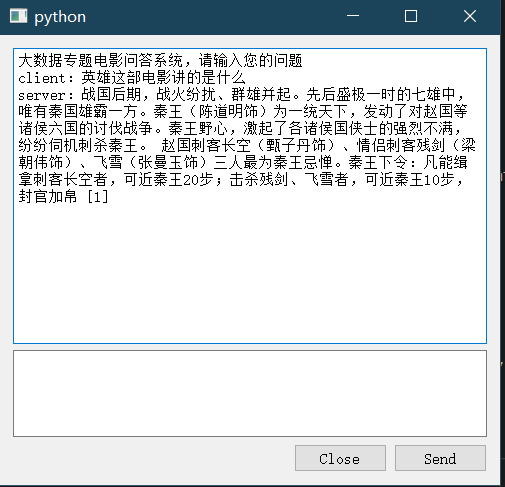
**模型训练：**

**构建一个spark类型的dataframe格式的训练集形式：**

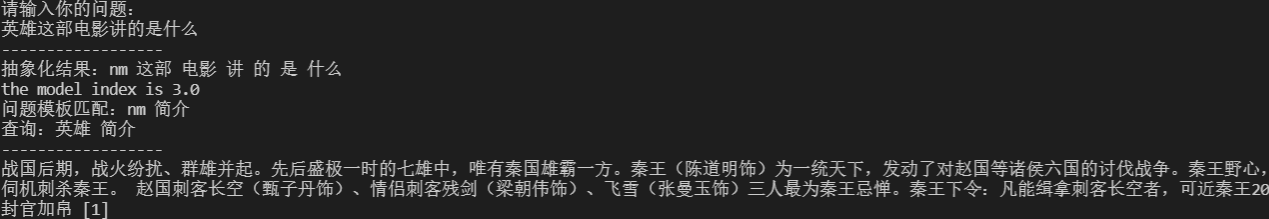
****

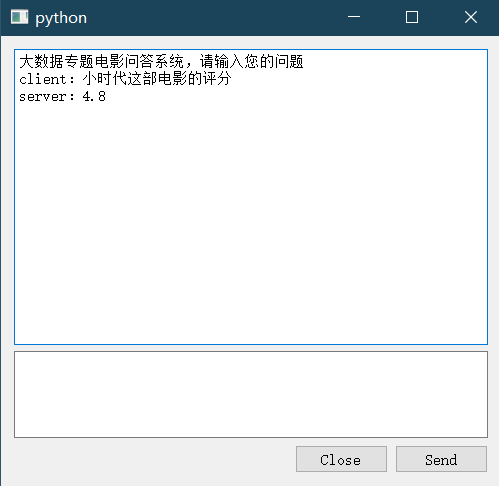
1. **效果展示：**

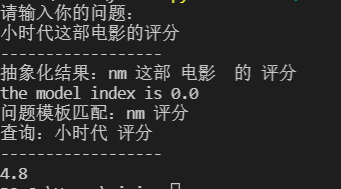
****

****

**后台过程显示：**

****

****



1. 项目总结

分布式计算节点

遇到的困难：

1 环境的搭建

2 爬虫程序

3 hive数据存储：数据存储结构

4 neo4j数据导入：从csv文件中构建知识图谱

5 用户问题分词与匹配：解析用户query

6 Spark分类器数据集划分：模型训练

不足：

Spark使用的还不是分布式的，是单节点的。

模型训练有的时候会出现分词错误和判断问题类型错误的情况

使用的数据量比较小，很多问题都无法回答。