# 第5章 基于知识库的问答系统

通过上一章节的学习，我们基于常用问答数据构建了一个简单的问答系统，在实际生成实践中，这种基于信息检索以及数据查询的方式有很大的不足。首先在构建整理问答数据的过程就需要耗费庞大的人力物力，其次这种方式仅仅适用于问题规模小且集中，存在少量语义异构信息的场景，而且对于精确率和召回率的要求较高。

对于用户特殊且复杂的信息需求，如何从当前多样化且非结构化的信息中获取用户所需要的答案，同时可以对用户的问题进行自动化的语义理解？本章我们将介绍基于知识库的问答系统，这种问答系统不仅可以实现对复杂问题的语义理解，而且可以对不同知识库的知识进行融合，并针对复杂问题进行相应的知识推理。

本章主要涉及到的知识点有：

* 基于知识库的问答系统介绍。
* 知识图谱介绍。
* Neo4j数据库安装配置。
* 导入数据到图数据库。
* Neo4j数据的简单操作增删改查。

注意：本章代码地址：xxx

## 5.1 基于知识库的问答系统

本节首先介绍什么是基于知识库的问答系统以及主流的基于知识库的问答系统的实现方法。

### 5.1.1 基于知识库的问答系统简介

基于知识库的问答系统（Knowledge Based Question Answering，KBQA，下文统称为KBQA）是目前应用最广泛的问答系统之一，适用于人们生活的方方面面，再医疗、金融、保险、零售等行业建立相应专业知识的问答系统，可以给用户提供更好的服务。

知识库（Knowledge Base，KB）是用于知识管理的一种特殊的数据库，用于相关领域知识的采集、整理和提取。知识库中的知识来源于各个领域的专家，是求解问题所需领域知识的集合，包括一些基本事实、规则和其他相关信息。知识库的表示形式通常是一个对象模型（object model），通常称为本体，包含一些类、子类和实体。不同于传统的数据库，知识库中存放的知识蕴含特殊的知识表示，其结构比传统数据库更加复杂，可以存放更多复杂语义表示的数据。

KBQA即给定自然语言问题，通过对问题进行语义理解和解析，并利用知识库中的专业知识进行查询、推理而得到答案的问答系统，也是当前最主流的问答系统。常见的知识库如Freebase、DBpedia等，知识库一般采用RDF格式对其中的知识进行表示。此外还有一些非结构化信息的知识库如维基百科、百度百科等。

KBQA一般包含问句理解模块、答案信息抽取模块、答案排序和生成模块等核心模块，其基本架构图如图5.1所示。



图5.1 KBQA基本架构图

KBQA系统中的问句理解模块主要完成提取问题中的实体信息的任务，答案信息抽取模块通过在知识库中查询以该实体节点为中心的知识库子图，并依据某些规则或模板从提取到的子图中抽取相应的信息，得到表征问题和候选答案特征的特征向量，最后将候选答案的特征向量作为分类模型的输入，通过模型输出的分类结果来筛选答案。KBQA系统的各模块间的关系如图5.2所示，其中主要包括问句分析（Question Analysis）、短语映射（Phrase Mapping）、语义消歧（Semantic Disambiguation）和构建查询（Query Construction）。



图5.2 KBQA各模块间关系

KBQA中的问句分析与自然语言理解（NLU）不同，主要是指识别问题中的信息词，如问题中的问题词（谁、什么、何时、何地、如何、怎么了、何事等），焦点词（名称、时间、地点），主题词，中心动词等词语。

短语映射主要负责将问句分析提取到的信息词与知识库或知识图谱中的资源对于的标签映射连接起来。常用的短语映射方法包括本体映射、同义词映射等，在这个过程中主要通过语义相似度计算来完成。这里我们可以根据在上一章节中掌握的语义相似度计算方法来完成。

语义消歧主要负责解决短语映射中出现的歧义问题，确保问句信息词和知识库的实体进行无歧义映射。常用的方法主要有两种，一种是基于相似度计算的方法，计算知识库实体标签与问句信息词之间的相似度；另一种是基于属性和参数等元信息的方法，通过判断属性和参数这些元信息是否一致来比较。

构建查询是指对前面几个模块生成的结果进行融合，得到最终的查询条件，将查询结果返回给用户。构建查询的方法一般为基于模板和基于问题分析的方法，其中基于模板构建形式化查询需要预先建立好查询模板，其中包含一些空槽位，将相关信息填入模板槽位后即可形成一个完成的查询条件；基于问题分析的方法则是通过语法树分析、依存树分析或语法槽位等方法，对自然语言进行解析来构成查询条件。

### 5.1.2 主流的KBQA实现方法

实现KBQA的方法可以分为基于模板匹配的方法、基于语义分析的分析方法、基于图遍历的方法、基于深度学习的方法和其他优化方法。

注意：xxx

1. 基于模板匹配的方法是指将用户输入的问题转换为预先定义的问题模板格式，然后将自然语言处理问句与知识库种的本体概念进行映射匹配，在实际系统中，一个问句通常会匹配到多个模板，因此需要对每个模板进行评估排序，将得分最高的模板作为最终模板。这种方法的优点在于查询响应速度快，一旦匹配到正确模板，则准确率较高，缺点是需要建立庞大的模板库来满足用户的各种问题，需要耗费较大的人力物力。
2. 传统的问答系统大多采用基于语义芬妮下的方法来完成问句理解的任务，通过对自然语言进行语义上的分析，将其转换为一种知识库可以理解的语义，进而获取知识库中的知识，并进行推理查询所需要的答案。常用的技术为构建语法树，构建出的语法树即为语义分析的结果。这种方法的优点在于准确率高，而且随着对问句语义分析的深入，可以回答处理相对复杂的问题，但是语义分析需要一定的专业知识，且需要编写大量规则，实现难度较大，且难以跨域使用。
3. 基于图遍历方法主要用于解决语义词汇映射和歧义的问题，将关系抽取转换为图搜索和图遍历的过程，弱化语义分析方法中关系抽取和映射的难度。在执行过程中主要包括三个模块，分别是问句理解（Question Understanding）、图遍历（Graph Traversal）和焦点约束排序（Focus Constraint Ranking），其中问句理解负责提取问题中的实体，使用实体链接的方法检测候选实体，并通过拓扑关系发现实体的内在联系，然后在知识库中查询该实体，得到以该实体为中心的知识库子图，最后抽取用于描述答案的问题核心词生成最终答案。
4. 基于深度学习的方法是一种基于匹配的方法，传统的方法往往需要人工编写模板，人工涉及语义分析规则，需要耗费较大的人力。借助深度学习一方面可以将原有的语义分析、实体识别通过深度神经网络来完成，降低人工成本，另一方面直接采用端到端的策略，在系统中输入问句和知识库，系统直接返回答案，将其作为一个“黑盒”来使用。

## 5.2 知识图谱

### 5.2.1 知识图谱

知识图谱是一种用图模型来描述知识和建模世界万物之间的管理关系的技术方法。知识图谱由节点和边组成，节点可以是实体，如一个人，一部电影等，或者一个抽象的概念，如人工智能，动物等，边可以是实体的属性，如姓名，电影名称等，或者是实体之间的关系如朋友等。如图5.1所示为一个典型的电影知识图谱示例。

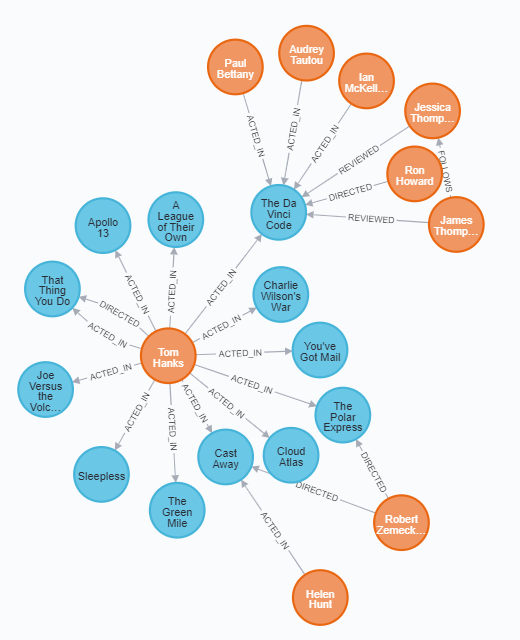


图5.1 电影知识图谱示例

在该知识图谱中，红色节点表示电影演员，蓝色节点表示电影，我们可以看到汤姆汉克斯（Tom Hanks）参演了电影《达芬奇密码》（The Da Vinci Code）、《荒岛余生》（Case Away）、《阿波罗13号》（Apollo 13）等，他们之间的关系用“ACTED\_IN”来表示，同样电影《达芬奇密码》中的电影演员也包括Paul Bettany、Ron Howard、Audrey Tautou等，而且可以看到Jessica Thompson与James Thompson之间也存在关系“FOLLOWS”等信息。

### 5.1.2 知识图谱的表示与存储

为了方便计算机的处理和理解，一般使用RDF来描述知识信息，通过三元组的形式保存在知识库中。在RDF中，知识以三元组的形式出现，每条知识可以被分解为如下形式：

（subject，predicate，object）。例如，“姚明出生在上海，姚明的妻子是叶莉”可以写成以下三元组：（姚明，出生地，上海），（姚明，配偶，叶莉）。RDF中的主语是一个个体（Individual），个体是类的实例。RDF中的谓语是一个属性，属性可以连接两个个体，或者连接一个个体和一个数据类型的实例。如图5.2展示。

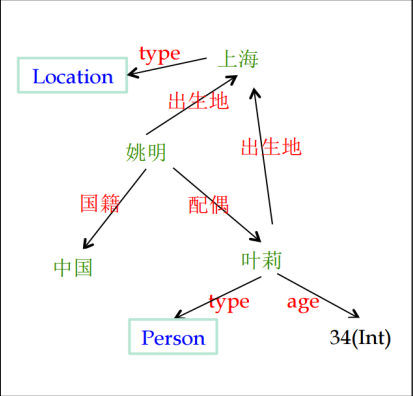


图5.2 知识示例

随着知识图谱规模的日益增长，无法满足用户的查询、检索、推理分析等需求，另一方面传统的关系型数据库也无法有效管理大规模知识图谱数据。

当前可用于知识图谱存储的数据库主要有三类，分别是基于传统的关系型数据库、面向RDF的三元组数据库和原生图数据库。

传统的关系型数据库是当前数据管理的主流数据库产品，商业数据库包括Oracle、DB2、SQL Server等，开源数据库包括MySQL、PostgreSQL等。面向RDF的三元组数据库是专门为存储大规模RDF数据而开发的知识图谱数据库，指出RDF的标注查询语言。常用的RDF三元组数据库包括Apache旗下的Jena、Eclipse旗下的RDF4j及源于学术界的RDF-3X

和gStore，以及用于商业的GraphDB和BlazeGraph等。

除了上面的这两种，还有一种原生图数据库，其中典型代表如当前最流行的图数据库Neo4j，本书中的实战项目均使用Neo4j。此外还有JanusGraph、OrientDB、Cayle等图数据库。

### 5.1.3 常用的图数据库及优缺点

本项目中我们采用从百度知道人工爬取的保险相关问答数据，该数据集中共包含8000余条保险相关问题及答案，经过数据清理，去除了原数据集中的问题id、url 、qid、reply\_t、user字段，对question、r

## 5.2 Neo4j数据库的安装配置

Neo4j 是目前最流行的图形数据库，支持完整的事务，在属性图中，图是由顶点（Vertex），边（Edge）和属性（Property）组成的，顶点和边都可以设置属性，顶点也称作节点，边也称作关系，每个节点和关系都可以由一个或多个属性。Neo4j创建的图是用顶点和边构建一个有向图，通过查询语言cypher来遍历获取图上的节点信息。

关系型数据库只对单个Join操作进行优化查询，而多重Join操作查询的性能显著下降。图形数据库适合查询关系数据，由于图形遍历的局部性，不管图形中由多少节点和关系，根据遍历规则，Neo4j只访问与遍历相关的节点，不受到总数据集大小的影响，从而保持期待的性能；相应地，遍历的节点越多，遍历速度越慢，但是变慢是线性的，这使得图形数据库不适合做海量数据统计分析。对于存在大量丰富关系的数据，遍历的性能不受图形数据量大小的影响，这使得Neo4j成为解决图形问题的理想数据库。

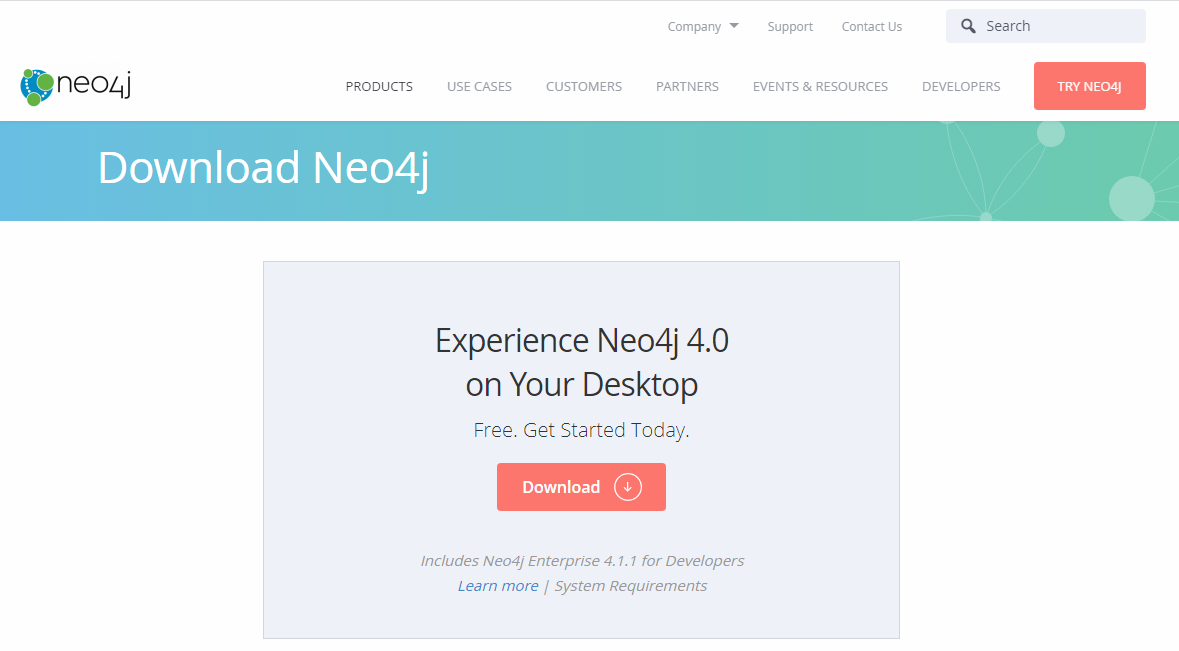
下面介绍Neo4j的安装配置：

### 5.2.1 下载安装Neo4j

Neo4j是基于Java的图形数据库，运行Neo4j需要启动JVM进程，因此需要预先安装JDK，且版本需要在1.8以上。

JDK的安装这里暂不介绍

点击下载页面



激活密钥

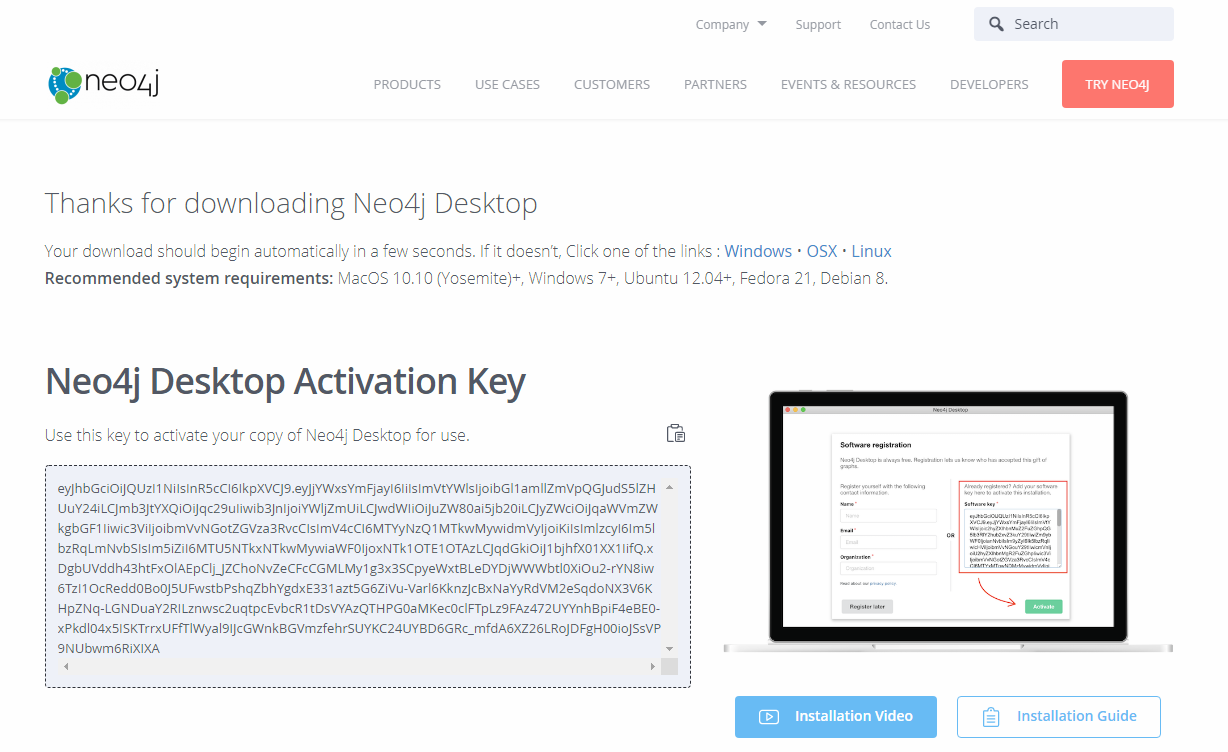
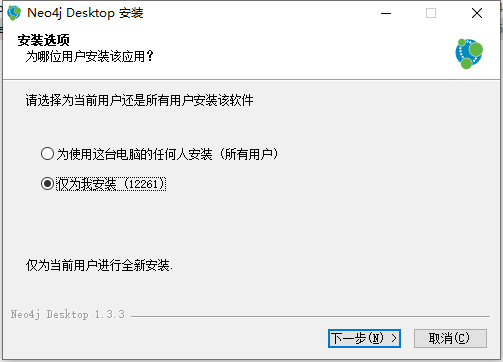
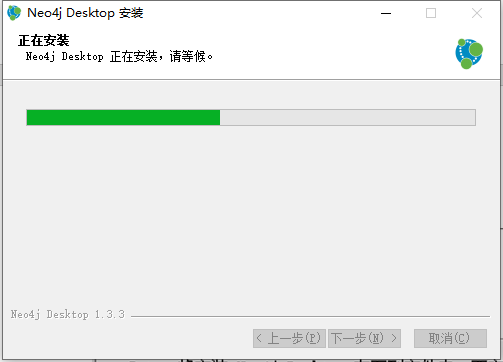
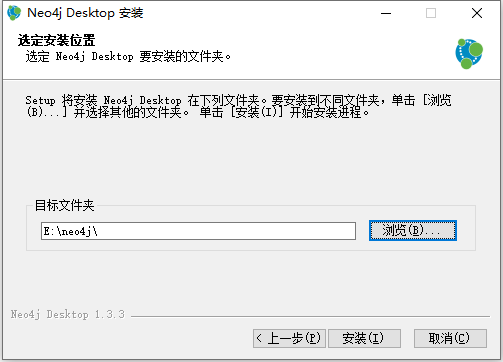


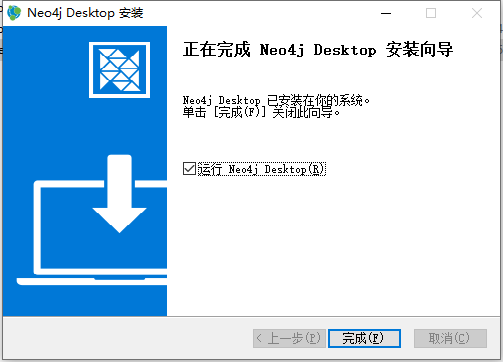
图4.4 智能客服系统架构

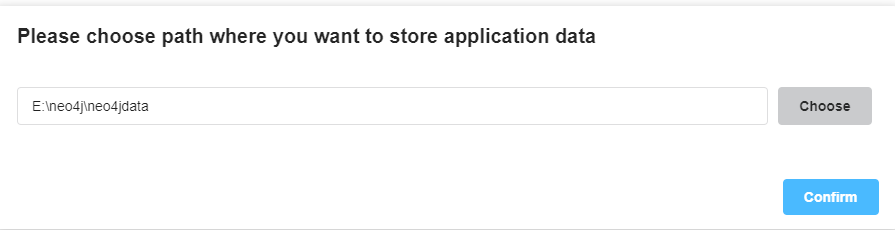
从官网下载最新版Neo4j社区版并解压，这里安装的是neo4j-community-3.5.5-windows

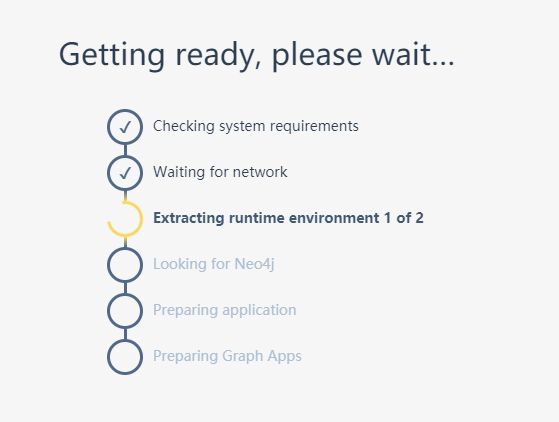
<https://neo4j.com/download/>





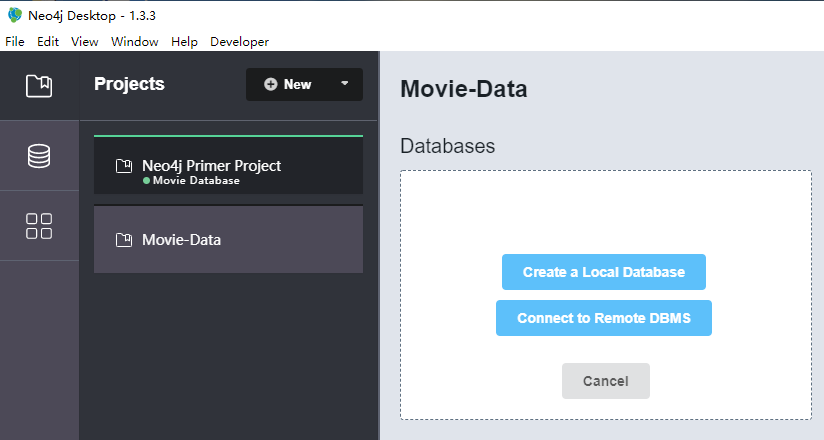




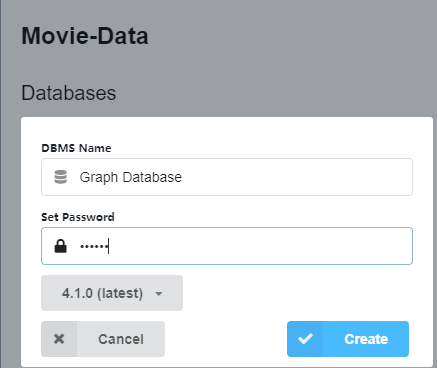


### 5.2.2 运行Neo4j

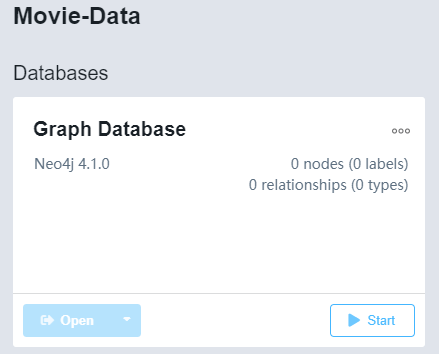
创建Project Movie-Data

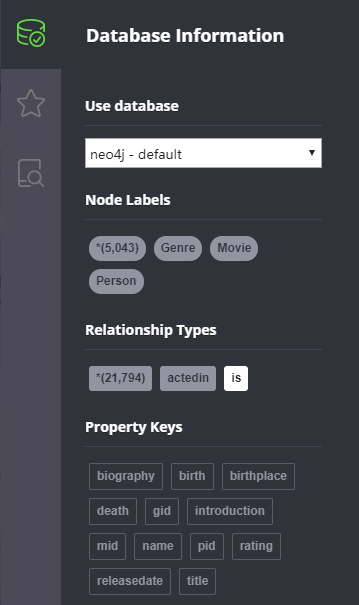


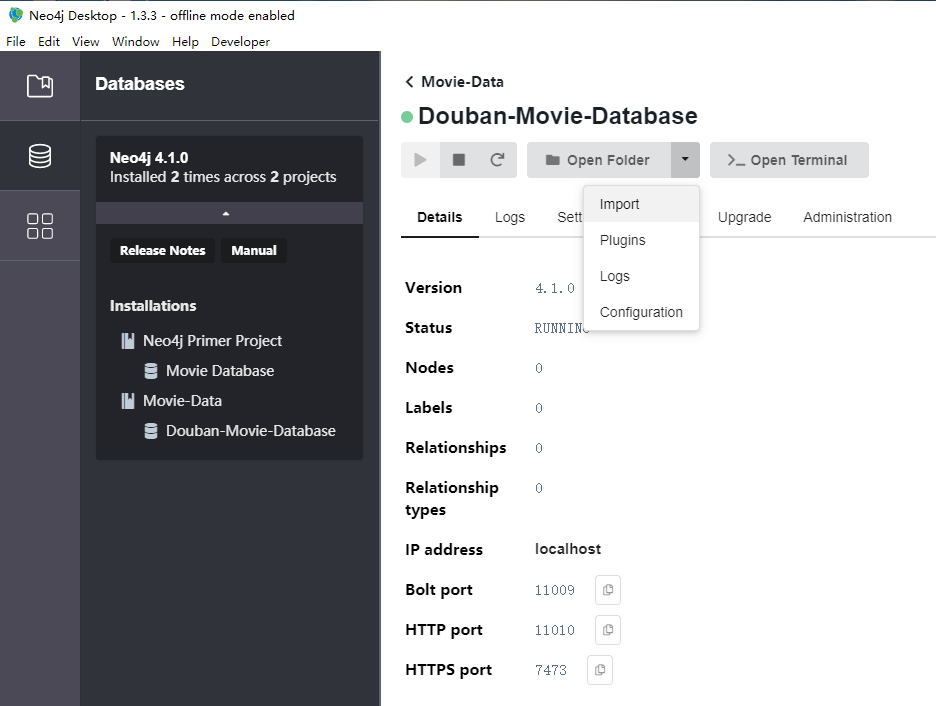
创建本地数据库 设置密码

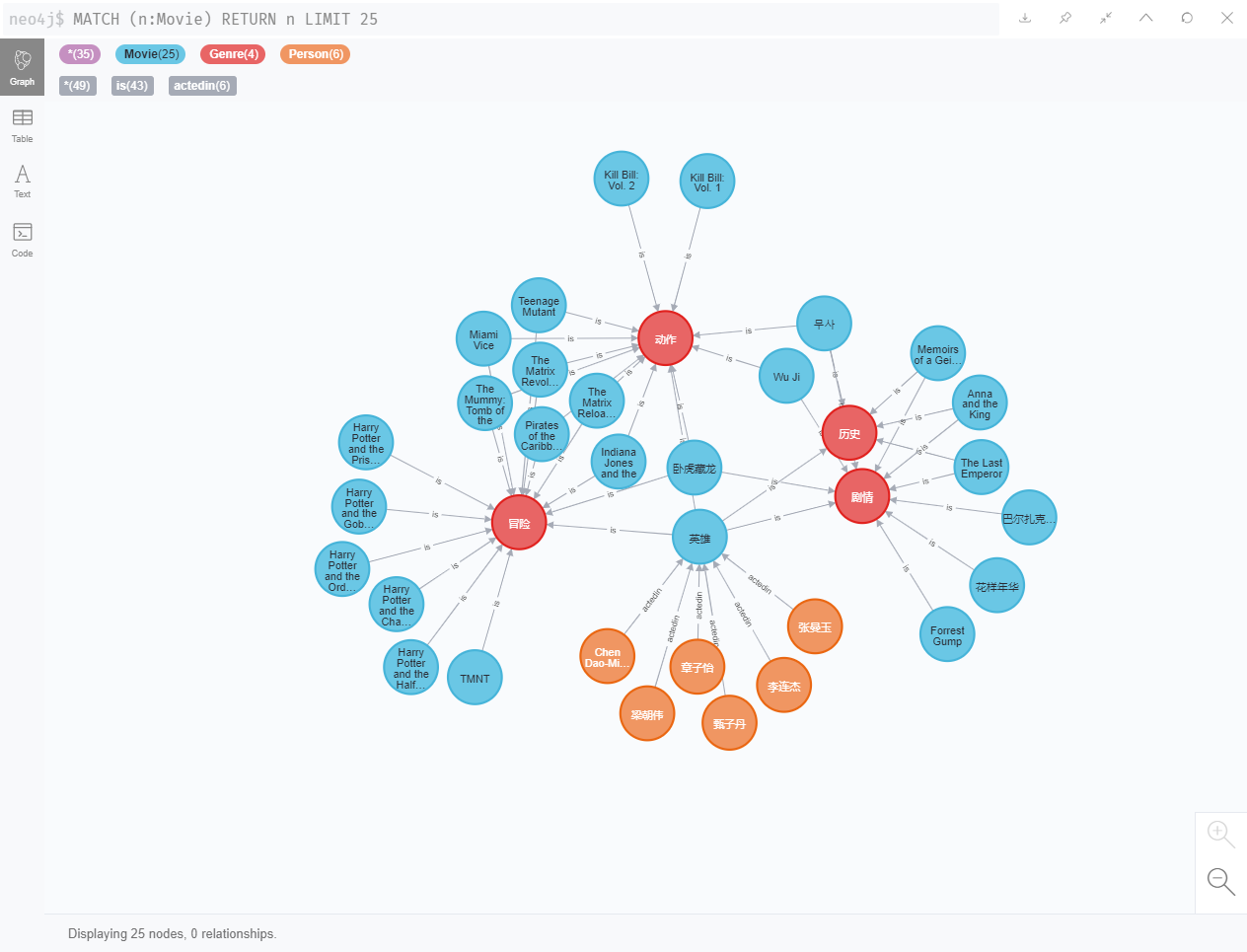


启动数据库









### 4.2.4 封装API并部署在服务器

为了便于访问我们的智能客服服务，我们将我们的代码封装成一个API，并部署在服务器上，是的通过http请求即可访问获取。这里采用aiohttp来实现服务部署。相关知识可以访问：<https://docs.aiohttp.org/en/stable/>

智能客服服务部署在我们本地，接口设置为9010，故访问地址为：<http://127.0.0.1:9010>

通过GET请求进行访问，参数为question

项目整体代码在如下地址提供：xxx

### 4.2.5 智能客服效果展示

服务部署以后，我们通过在浏览器输入想查询的问题，系统会给出相应回复。问题数据通过在访问地址中拼接?question=XXX的方式传递。

如：

案例1：

<http://127.0.0.1:9010/?question=>天热了，很想去潜水，除了装备、教练以及费用外还要什么保险？

{

"系统回复": "",

"相似问题": "",

"推荐答案": "您好！除了您所说的准备、教练等专业人士陪同之外，购买一份合适的户外运动保险很有必要，毕竟潜水存在一定风险。目前国内关于潜水运动的保险并不多，一般都要求是下潜不超过18m。具体您可以了解下HUTS保险中的适合国外潜水等海岛相关水上运动的产品众行天下-水上运动保险，是海岛潜水专属的保障。主要是因为它同时包括潜水意外医疗补偿以及紧急医疗运送，是高风险水上运动的理想选择。"

}

案例2：

<http://127.0.0.1:9010/?question=>本周末公司组织一次大型户外拓展活动，100人什么HUTS保险合适？

{

"系统回复": "小助手没有这个问题的答案呢，给您推荐以下相似问题及答案以供参考哦~\n",

"相似问题": "相似问题：本周末公司会组织200人的户外拓展活动，什么保险能保障员工安全？\n",

"推荐答案": "推荐答案：您好，关于户外拓展活动，存在一定的风险，并且贵公司有200人参加，所以建议可以考虑下HUTS保险中的众行天下-拓展训练保险，是专门针对户外运动这块的保险。最高可赔付10万元，关键是价格比较实惠，适合集体投保。"

}

案例3：

<http://127.0.0.1:9010/?question=>太阳系有几个行星呢

{

"系统回复": "啊哦，小助手还没有掌握这方面的知识呢，我会将您的问题记录下来，并尽快找到专业的答案。",

"相似问题": "",

"推荐答案": ""

}

## 5.3 导入数据到Neo4j数据库

将已经整理好的电影数据导入Neo4j数据库。

**代码5.9 导入CSV文件**

//导入节点 电影类型 == 注意类型转换

LOAD CSV WITH HEADERS FROM "file:///genre.csv" AS line

MERGE (p:Genre{gid:toInteger(line.gid),name:line.gname})

//导入节点 演员信息

LOAD CSV WITH HEADERS FROM 'file:///person.csv' AS line

MERGE (p:Person { pid:toInteger(line.pid),birth:line.birth,

death:line.death,name:line.name,

biography:line.biography,

birthplace:line.birthplace})

// 导入节点 电影信息

LOAD CSV WITH HEADERS FROM "file:///movie.csv" AS line

MERGE (p:Movie{mid:toInteger(line.mid),title:line.title,introduction:line.introduction,

rating:toFloat(line.rating),releasedate:line.releasedate})

// 导入关系 actedin 电影是谁参演的 1对多

LOAD CSV WITH HEADERS FROM "file:///person\_to\_movie.csv" AS line

match (from:Person{pid:toInteger(line.pid)}),(to:Movie{mid:toInteger(line.mid)})

merge (from)-[r:actedin{pid:toInteger(line.pid),mid:toInteger(line.mid)}]->(to)

//导入关系 电影是什么类型 == 1对多

LOAD CSV WITH HEADERS FROM "file:///movie\_to\_genre.csv" AS line

match (from:Movie{mid:toInteger(line.mid)}),(to:Genre{gid:toInteger(line.gid)})

merge (from)-[r:is{mid:toInteger(line.mid),gid:toInteger(line.gid)}]->(to)

## 5.4 Neo4j数据的简单操作增删改查