# 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。

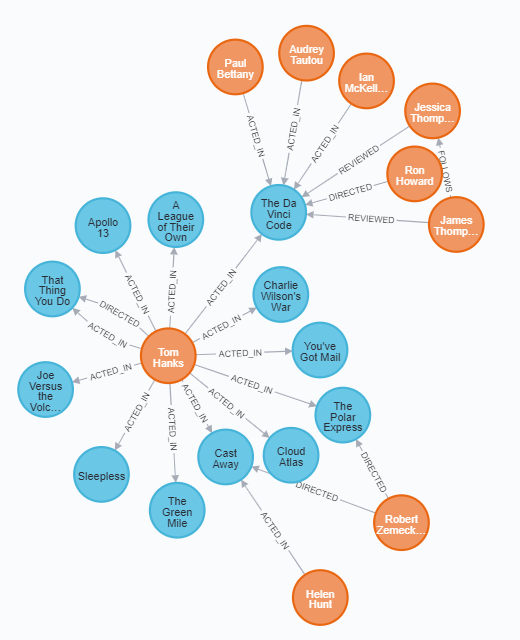


图5.1 电影知识图谱示例

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。

### 5.2.2 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。

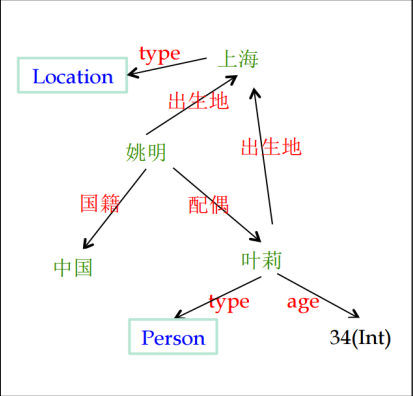


图5.2 知识示例

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。

### 5.2.3 知识图谱数据库比较

总体来讲， 基于关系的存储系统继承了关系数据库的优势， 成熟度较高， 在硬件性能和存储容量满足的前提下， 通常能够适应千万到十亿级三元组规模的管理。 官方测评显示， 关系数据库Oracle 12c配上空间和图数据扩展组件（ Spatial and Graph） 可以管理的三元组数量高达1.08万亿条！ 当然， 这样的性能效果是在 Oracle 专用硬件上获得的， 所需软硬件成本投入很大。 对于一般在百万到上亿级三元组的管理， 使用稍高配置的单机系统和主流 RDF 三元组数据库（ 如Jena、 RDF4J、 Virtuoso等） 完全可以胜任。 如果需要管理几亿到十几亿以上大规模的RDF 三元组， 则可尝试部署具备分布式存储与查询能力的数据库系统（ 如商业版的GraphDB和BlazeGraph、 开源的JanusGraph等） 。 近年来， 以Neo4j为代表的图数据库系统发展迅猛， 使用图数据库管理 RDF 三元组也是一

种很好的选择； 但目前大部分图数据库还不能直接支持RDF三元组存储， 对于这种情况， 可采用数据转换方式， 先将RDF预处理为图数据库支持的数据格式（ 如属性图模型） ， 再进行后续管理操作。

## 5.3 Neo4j的安装

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。

这一节我们将会直接进入动手部分安装Neo4j数据库，并将我们后面用到的电影相关知识问答的数据导入存储在Neo4j数据库中，并对Neo4j数据库进行一些简单的增删改查操作，方便我们后面项目中使用。

下面介绍Neo4j的安装配置：

### 5.3.1 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。

进入Neo4j的官网<https://neo4j.com/download/>，开发者注册信息后可免费下载Neo4j桌面打包安装版，（Neo4j Desktop），其中包括Neo4j企业版的全部功能，即Neo4j服务器，客户端即全部组件。

下载最新版Neo4j社区版并解压，这里安装的是neo4j-community-3.5.5-windows，安装中可以自定义设置安装位置的路径。

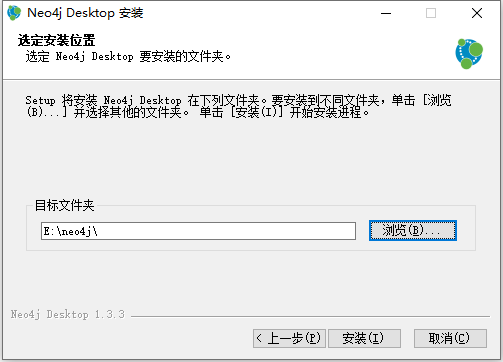


图5.3设置Neo4j安装位置

安装成功后的Neo4j Desktop数据库管理界面如图5.4所示。

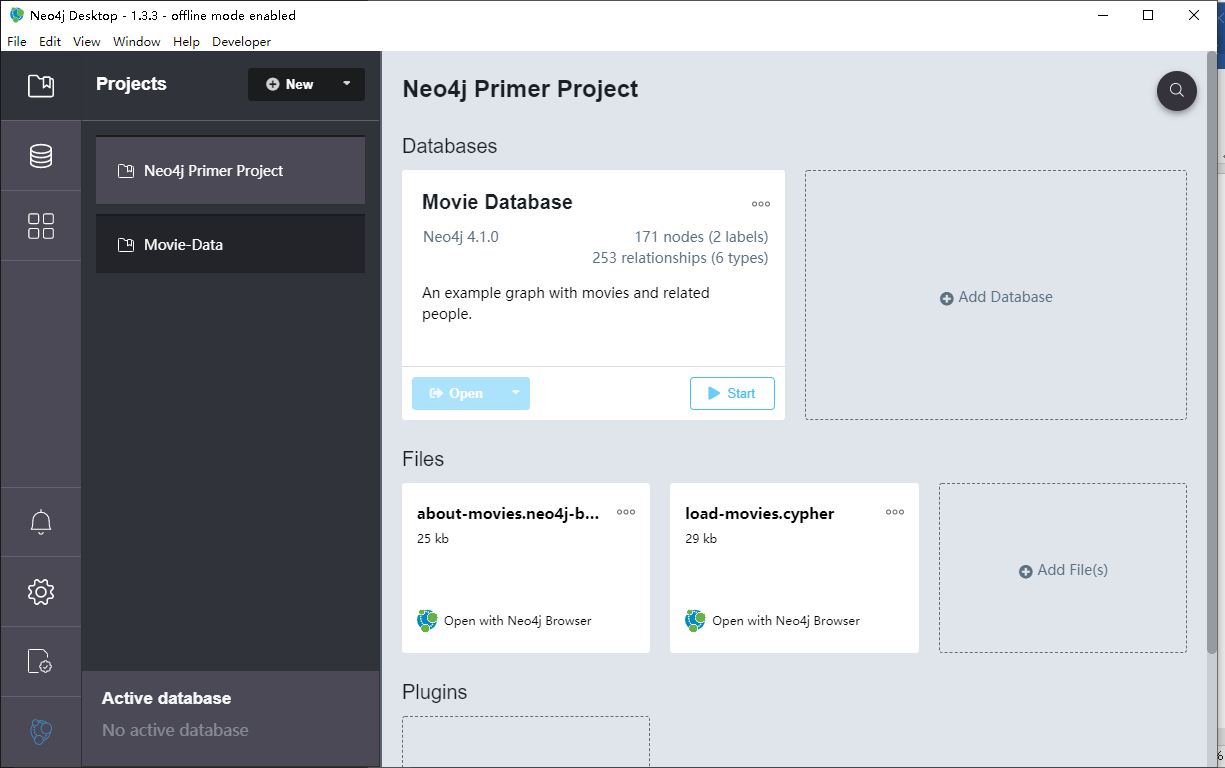


图5.4 Neo4j Desktop数据库管理界面

### 5.3.2 创建Neo4j Project

安装好Neo4j后，创建一个图数据库Project，命名为 Movie-Data。

点击图5.5中的“New”，点击右侧Databases中“Create a Local Database”，创建一个本地图数据库。

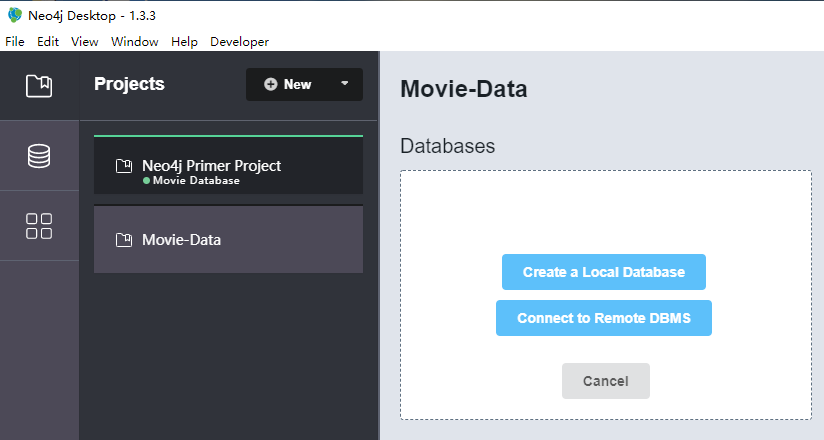


图5.5 创建数据库项目

创建本地数据库时，我们设置数据库项目名为“Movie-Data”，同时设置数据库名词也为“Movie-Data”，并给当前数据库设置密码，创建数据库时，可以选择当前使用的Neo4j版本，这里使用默认版本4.1.0即可，设置完成后点击“Create”完成创建。如图5.6所示。

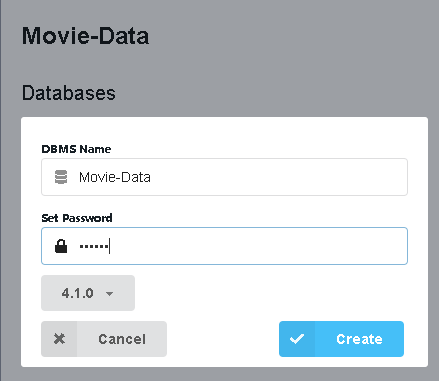


图5.6 创建数据库

创建数据库完成后，如图5.7所示，点击“Start”按钮启动数据库。

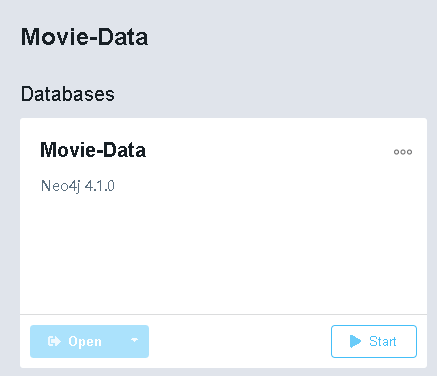


图5.7 启动数据库

启动数据库后，如图5.8所示，我们可以发现当前数据库为启动状态“Active”，可以通过点击“Stop”，停止数据库服务。数据库显示的状态中“0 nodes（0 labels）”及“0 relationships （0 types）”，由于我们当前还没有往数据库中存储数据，因此数据库中节点（nodes）与关系信息（relationships）均为0。

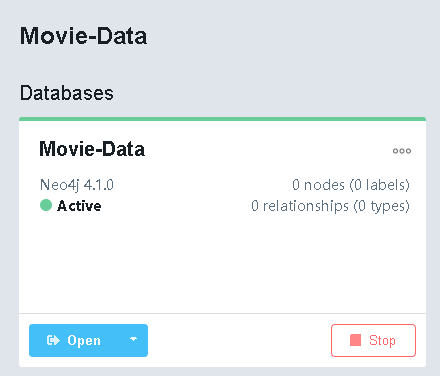


图5.8 查看Neo4j数据库状态

创建数据库完成后，下一步我们将简单学习一下Neo4j数据库的操作，便于我们在后面的项目中使用Neo4j数据库。

点击“Open”按钮，使用Neo4j Browser方式打开，如图5.9所示：

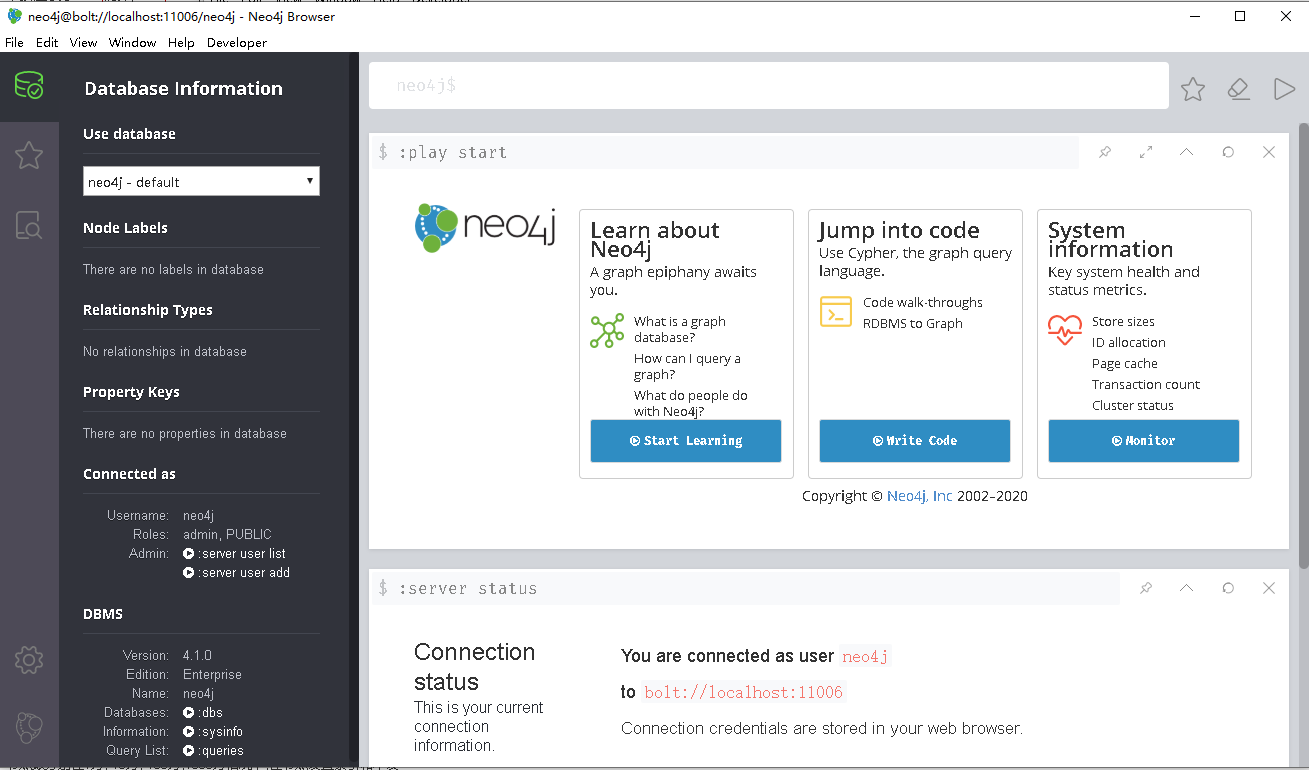


图5.9 打开Neo4j数据库

以上查看数据库的方式相当于访问bolt://localhost:11006该路径，除了Bolt方式，Neo4j还支持Http和Https方式访问，即通过浏览器访问地址<http://localhost:11007>，同样可以访问Neo4j数据库，后面项目中实际使用Neo4j也是通过这种方式来操作。

## 5.4 Neo4j数据库的简单操作

通过Neo4j Browser方式打开Neo4j数据库后，我们接下来介绍一下如何操作Neo4j数据库，包括添加数据，删除数据，修改数据及查找数据。

这里需要简单学习一种Neo4j的图形查询语言：Cypher，这是一种类似SQL的使用模式来描述图数据，这种声明式的语言只需要我们描述我们想要查找的数据而不需要描述如何去查找，因此可以帮助我们方便快捷地对图数据库进行存储和检索数据。

### 5.4.1 添加数据

首先我们通过下面的命令创建一条数据，并将其存储到数据库中。

CREATE (ee:Person { name: "Emil", from: "Sweden", klout: 99 })

将命令在Neo4j Browser中的编辑框进行执行，结果显示如下：

Added 1 label, created 1 node, set 3 properties, completed after 153 ms.

通过“CREATE”命令，我们创建了一个“ee”的节点，它的标签信息为“Person”，该节点有三个属性分别为“name”、“from”和“klout”。

创建成功后，通过Neo4j的图形化界面，我们可以发现，当前数据库中出现一个“Person”的标签，点击该标签，会出现一个名称为“Emil”的节点，该节点即为我们刚刚创建的节点。下面我们通过命令行的方式来查看该节点的信息。

添加节点后，我们同样通过“CREATE”创建节点之间的关系，如下面命令中表示，在“Emil”节点于“Johan”之间存在关系“KNOWS”，且这个关系的属性“Since”为“2001”.

MATCH (ee:Person) WHERE ee.name = "Emil"

CREATE (js:Person { name: "Johan", from: "Sweden", learn: "surfing" }),

(ir:Person { name: "Ian", from: "England", title: "author" }),

(rvb:Person { name: "Rik", from: "Belgium", pet: "Orval" }),

(ally:Person { name: "Allison", from: "California", hobby: "surfing" }),

(ee)-[:KNOWS {since: 2001}]->(js),(ee)-[:KNOWS {rating: 5}]->(ir),

(js)-[:KNOWS]->(ir),(js)-[:KNOWS]->(rvb),

(ir)-[:KNOWS]->(js),(ir)-[:KNOWS]->(ally),

(rvb)-[:KNOWS]->(ally)

添加节点以及节点之间关系完成后，数据库中各节点之间关系如图5.10所示：

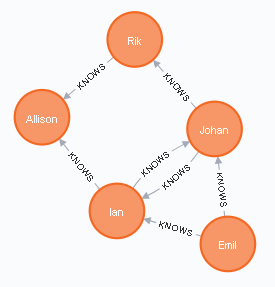


图5.10 添加节点及节点间关系后数据库中数据情况

### 5.4.2 查找数据

我们可以通过下面的命令来查找我们刚刚创建的数据。

MATCH (ee:Person) WHERE ee.name = "Emil" RETURN ee

通过“MATCH”命令，来查找Neo4j中的数据，Cypher不需要我们描述如何去查找目标数据，仅需要描述所查找数据满足的条件即可。这里我们要查找的数据是这样的，该数据的标签信息为“Person”，该数据的“name”属性为“Emil”，最后返回满足以上条件的数据。

经过执行以上命令，返回如下数据信息：

结果展示：

{

"identity": 0,

"labels": [

"Person"

],

"properties": {

"name": "Emil",

"from": "Sweden",

"klout": 99

}

}

除了查找节点信息，Neo4j还可以根据节点间的关系进行查找，如下面的命令展示了查找标签为“Person”的节点，该节点与“Emil”节点关系为“KNOWS”，最终返回这些节点的全部信息。

MATCH (ee:Person)-[:KNOWS]-(friends) WHERE ee.name = "Emil" RETURN ee, friends

通过执行以上命令，得到的结果如图5.11所示：

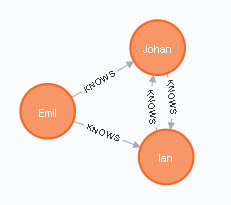


图5.11 根据节点关系查找数据信息

### 5.4.1 修改数据

要对Neo4j中的数据进行修改，可以使用“SET”命令，如下面的命令，将“Emil”节点的“from”属性修改为“China”。

MATCH (n:Person {name:"Emil"}) SET n.from ="China" return n

经过查询发现，修改后该节点信息如下，可以发现已经修改成功。

结果展示：

{

"identity": 0,

"labels": [

"Person"

],

"properties": {

"name": "Emil",

"from": "China",

"klout": 99

}

}

### 5.4.1 删除数据

如果要删除节点或者关系的属性，可以使用“REMOVE”命令，或者也可以用通过“SET”命令，将其置为null。如下面的命令。

MATCH(n:Person {name:"Allison"}) remove n.hobby return n

以上命令删除了“Allison”节点的“hobby”属性，执行成功后，该节点信息如下：

结果展示：

{

"identity": 4,

"labels": [

"Person"

],

"properties": {

"name": "Allison",

"from": "California"

}

}

采用“SET”命令删除属性“from”的命令如下，查看返回结果可以发现已经删除成功。

MATCH(n:Person {name:"Allison"}) SET n.from=null return n

结果展示：

{

"identity": 4,

"labels": [

"Person"

],

"properties": {

"name": "Allison"

}

}

## 5.5 导入电影数据到Neo4j

除了使用“CREATE”的方式在Neo4j数据库中创建数据，Neo4j也支持其他方式导入数据，如通过“LOAD CSV”的方式直接导入CSV格式的数据，此外Neo4j还支持通过API的方式导入JSON格式的数据。本节我们将接下来项目中要使用的数据批量导入Neo4j数据库。

该数据集来源于IMDB数据库，这是一个关于电影演员、电影、电视节目、电视明星和电影制作的在线数据库。我们项目中需要查询某部电影的信息，比如电影的评分，上映时间、电影简介信息等，也可能查询某位演员的个人信息，出演了那些电影等信息。因此需要关注两个实体，分别是电影和演员，然后演员和电影之间有着直接的关系：**[act]**，即某人出演了某部电影，于是可以用这个关系连链接演员和电影，此外实体还有自己的属性，如电影的上映时间，内容简介，演员的个人信息等。数据的格式如下：

首先在movie.csv中存储电影相关信息，其数据格式字段及示例数据如下：

表5.1 电影数据格式字段介绍

|  |  |  |
| --- | --- | --- |
| 字段 | 说明 | 备注 |
| mid | 电影id | 非空 |
| title | 电影名称 |  |
| introduction | 电影简介信息 |  |
| rating | 电影评分 |  |
| releasedate | 电影上映日期 |  |

"mid","title","introduction","rating","releasedate"

"13","Forrest Gump","阿甘（汤姆·汉克斯 Tom Hanks 饰）于二战结束后不久出生在美国南方阿拉巴马州一个闭塞的小镇，他先天弱智，智商只有75，然而他的妈妈是一个性格坚强的女性，她常常鼓励阿甘“傻人有傻福”，要他自强不息。阿甘像普通孩子一样上学，并且认识了一生的朋友和至爱珍妮（罗宾·莱特·潘 Robin Wright Penn 饰），在珍妮和妈妈的爱护下，阿甘凭着上帝赐予的“飞毛腿”开始了一生不停的奔跑。阿甘成为橄榄球巨星、越战英雄、乒乓球外交使者、亿万富翁，但是，他始终忘不了珍妮，几次匆匆的相聚和离别，更是加深了阿甘的思念。有一天，阿甘收到珍妮的信，他们终于又要见面…","8.300000190734863","1994-07-06"

"24","Kill Bill: Vol. 1","新娘（乌玛·瑟曼饰）曾经是致命毒蛇暗杀小组（D．I．V．A．S）的一员，企图通过结婚脱离血腥的生活。但是她的前同僚（汉纳、刘玉玲、薇薇卡·A·福克斯、迈克尔·马德森等人扮演）以及所有人的老板比尔（大卫·卡拉丁饰）的到来破坏了这一切。“比尔，”新娘请求说，“我怀孕了，是你的孩子。”但是回答她的是“砰”的一声枪响！4年后她在一家医院醒来，就立刻开始着手一次从得克萨斯到冲绳、东京以及墨西哥的复仇之旅，为了一个目标她要大开杀戒。“当我到达目的地之后，我将杀死比尔。”","7.800000190734863","2003-10-10"

"79","英雄","战国末期，赵国有三个名震天下的侠客，他们是：“长空”、“残剑”、“飞雪”。因为他们，秦王十年里没睡过一个安稳觉。可是他们却被一个默默无闻，名叫无名的秦国剑客所杀。消息传来，秦王振奋，急召无名上殿相见。在秦王的大殿里，神秘的烛火燃烧着，秦王与无名只有十步的距离，无名将击杀长空、残剑、飞雪的故事娓娓道来：他利用三人之间爱恨交织的关系，瓦解了他们的力量，各个击破，因此取胜。可秦王机智过人，听出了无名故事中的破绽，说出了另一个故事的\*\*：残剑等三人是主动求败，献出生命，用苦肉计帮助无名上殿，无名才是真正最危险的刺客，而无名告诉秦王：他看错了一个人，那就是残剑。于是真正的故事从头叙起……最后，无名拿起了剑，此时他离秦王只有十步，他的绝技是“十步一杀”。","7.300000190734863","2002-12-19"

在person.csv中存储演员相关信息，其数据格式字段及示例数据如下：

表5.2 演员数据格式字段介绍

|  |  |  |
| --- | --- | --- |
| 字段 | 说明 | 备注 |
| pid | 演员id |  |
| birth | 演员出生日期 |  |
| death | 演员逝世日期 | \N表示该演员未逝世 |
| name | 演员名 |  |
| biography | 个人档案 |  |
| birthplace | 出生地 |  |

"pid","birth","death","name","biography","birthplace"

"643","1965-12-31",\N,"巩俐","新加坡华裔女演员，祖籍中国山东，毕业于中央戏剧学院，联合国促进和平艺术家，联合国全球环境保护大使。1987年，因主演电影《红高粱》成名，该片获得柏林电影节金熊奖。1992年，凭借主演的电影《秋菊打官司》获得威尼斯国际电影节最佳女演员奖，该片亦获得金狮奖。1993年，主演的电影《霸王别姬》获得金棕榈奖，因而巩俐成为世界影史第二位主演影片包揽欧洲三大国际电影节影片奖的演员。1993年，获得柏林国际电影节金摄影机奖并上榜美国《人物》全球最美50人。1996年，登上美国《时代周刊》封面并上榜全球十大人物。1997年，担任戛纳国际电影节评委会成员。2000年，担任柏林国际电影节评委会主席；同年，获得蒙特利尔国际电影节最佳女演员奖及艺术成就大奖。2002年，担任威尼斯国际电影节评委会主席。2003年，担任东京国际电影节评委会主席。2004年，获得戛纳国际电影节纪念大奖；同年，上榜美国《首映》影史百大伟大表演。2005年，入选中国电影百年50位有突出贡献艺术家。2006年，上榜美国《时代周刊》60年亚洲英雄；同年，上榜美国《华盛顿邮报》全球年度5位伟大演员。2010年，法国文化部授予其“艺术与文学勋章”司令勋位。2014年，担任上海国际电影节评委会主席；同年，二度上榜日本《电影旬报》百大外国女星。2015年，入选联合国16位影响人类文化艺术家。","Shenyang, Liaoning Province, China"

"695","1937-03-16","1999-04-14","乔宏","","Shanghai, China"

"1336","1963-04-26",\N,"李连杰","李连杰（Jet Li），1963年4月26日生于北京市，华语影视男演员、导演、制作人 、武术运动员、商人。1971年进入北京什刹海体校，从而开始武术运动员的生涯。1975年参加全运会武术套路比赛，获得全能冠军。1979年因伤退出武术界。1982年主演个人首部电影《少林寺》，该片打破华语电影在中国内地的票房纪录。1986年自导自演动作片《中华英雄》。1991年主演的动作片《黄飞鸿之壮志凌云》奠定其在影坛的地位。1992年凭借武侠片《笑傲江湖Ⅱ：东方不败》获得更多关注。1995年获得第32届台湾电影金马奖大陆人士特别奖 。1998年赴美国好莱坞发展，同年出演其在好莱坞的首部电影《致命武器4》。2001年担任动作片《变种元素》的制作人。2002年因主演武侠片《英雄》而成为美国《时代周刊》的封面人物 。2003年凭借动作片《宇宙追缉令》获得第12届MTV电影奖最佳打斗场面提名。2008年凭借动作片《投名状》获得第27届香港电影金像奖最佳男主角 。2009年成为中国企业家俱乐部会员。2010年入选美国《时代周刊》“年度最具影响力人物百人榜” 。2011年参与创办太极禅国际文化发展有限公司。2013年在《中华武术》三十年颁奖盛典中，获得“中华武术30年最具武术影响力人物奖”。2016年主演古装片《封神传奇》 。李连杰热心公益慈善。2007年创建公益组织壹基金。2009年担任世界卫生组织亲善大使。2010年担任国际红十字会亲善大使。2014年在《中国慈善家》“中国慈善名人榜”中排名第二位。","Beijing, China"

除了上面两个主要的实体数据，我们还需要导入二者之间的关系数据。其中每一条数据用于表示该数据中pid表示的该演员出演了mid表示的该电影，该数据存储在person\_to\_movie.csv文件中。

表5.3 演员与电影之间关系数据格式字段介绍

|  |  |  |
| --- | --- | --- |
| 字段 | 说明 | 备注 |
| pid | 演员id |  |
| mid | 电影id |  |

"pid","mid"

"163441","13"

"240171","24"

"1336","79"

"1337","79"

"1338","79"

"1339","79"

"1340","79"

此外，我们还对电影定义了不同类型，并使用不同gid表示，该数据存储在genre.csv中。

表5.4 演员与电影之间关系数据格式字段介绍

|  |  |  |
| --- | --- | --- |
| 字段 | 说明 | 备注 |
| gid | 电影类型id |  |
| gname | 电影类型名称 |  |

gid,gname

12,冒险

14,奇幻

16,动画

18,剧情

27,恐怖

28,动作

35,喜剧

36,历史

37,西部

53,惊悚

80,犯罪

99,纪录

878,科幻

9648,悬疑

10402,音乐

10749,爱情

10751,家庭

10752,战争

10770,电视电影

最后还存储了电影类别信息与电影id之间的关系，该数据存储在movie\_to\_genre.csv中。

表5.5 演员与电影之间关系数据格式字段介绍

|  |  |  |
| --- | --- | --- |
| 字段 | 说明 | 备注 |
| mid | 电影id |  |
| gid | 电影类型id |  |

"mid","gid"

"79","12"

"82","12"

"87","12"

"146","12"

"285","12"

"604","12"

将以上数据准备好后，通过下面的代码将其导入Neo4j数据库中。

**代码5.1 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)

导入成功后，数据库中给数据如图5.12所示。

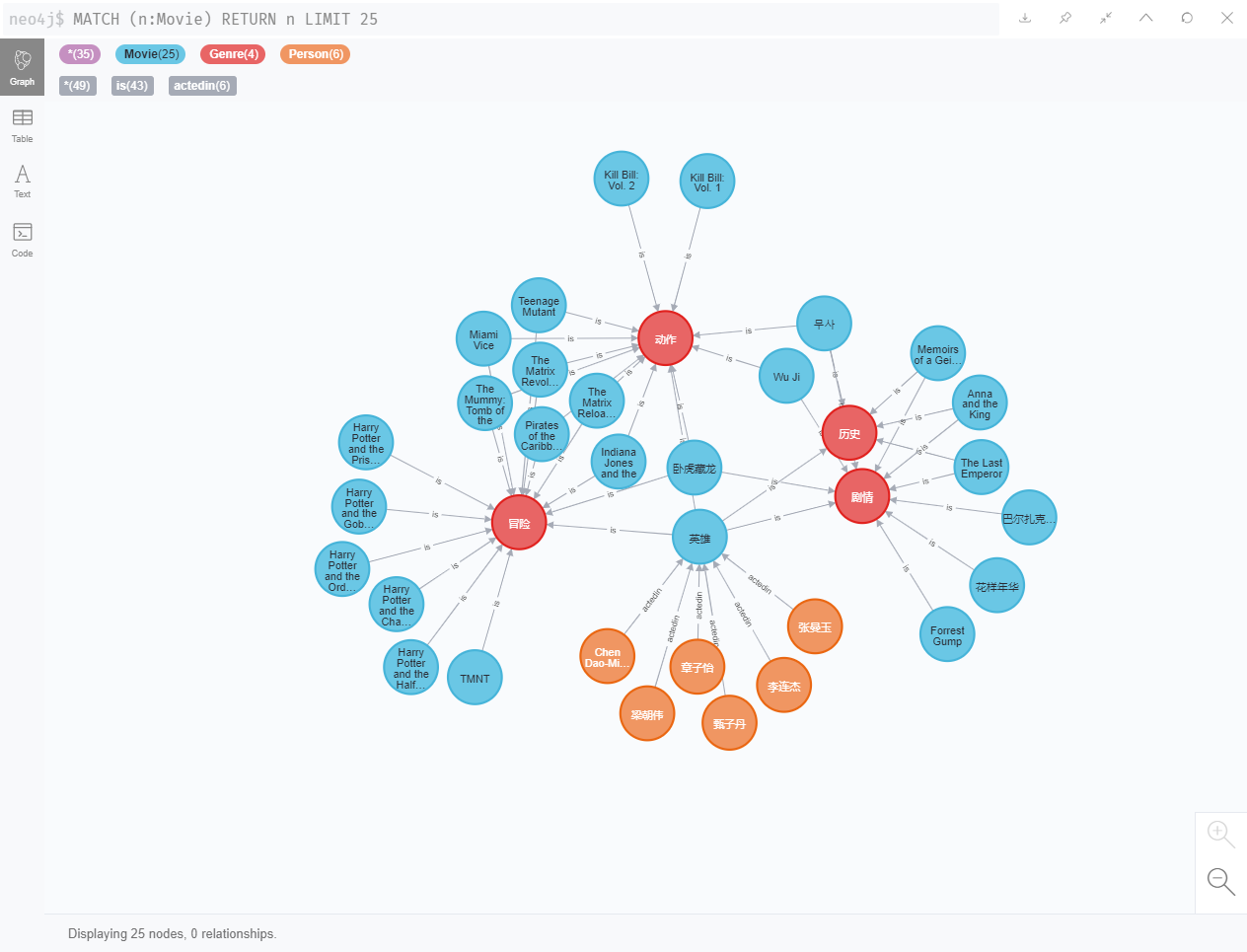


图5.12 导入Neo4j后数据展示情况

数据导入成功后，我们写一个简单的测试脚本进行查询，查询结果显示可以获取到相应的数据。

**代码5.2 查询Neo4j数据库**

#!/usr/bin/env python

# \_\*\_ coding:utf-8 \_\*\_

from py2neo import Graph, Node, Relationship, NodeMatcher

def query(query\_SQL):

'''

查询Neo4j数据库

:param query\_SQL: 查询命令

:return:

'''

# 连接数据库

graph = Graph("http://localhost:11010", username="neo4j", password="123456")

# 存储查询结果

result = []

# 执行查询语句

query\_result = graph.run(query\_SQL)

for i in query\_result:

result.append(i.items()[0][1])

return result

if \_\_name\_\_ == '\_\_main\_\_':

# 查询电影《英雄》的电影简介信息

query\_SQL = "match (m:Movie)-[]->() where m.title='英雄' return m.introduction"

result = query(query\_SQL)

print('query result:', result)

查询结果显示如下：

query result: ['战国末期，赵国有三个名震天下的侠客，他们是：“长空”、“残剑”、“飞雪”。因为他们，秦王十年里没睡过一个安稳觉。可是他们却被一个默默无闻，名叫无名的秦国剑客所杀。消息传来，秦王振奋，急召无名上殿相见。在秦王的大殿里，神秘的烛火燃烧着，秦王与无名只有十步的距离，无名将击杀长空、残剑、飞雪的故事娓娓道来：他利用三人之间爱恨交织的关系，瓦解了他们的力量，各个击破，因此取胜。可秦王机智过人，听出了无名故事中的破绽，说出了另一个故事的\*\*：残剑等三人是主动求败，献出生命，用苦肉计帮助无名上殿，无名才是真正最危险的刺客，而无名告诉秦王：他看错了一个人，那就是残剑。于是真正的故事从头叙起……最后，无名拿起了剑，此时他离秦王只有十步，他的绝技是“十步一杀”。', '战国末期，赵国有三个名震天下的侠客，他们是：“长空”、“残剑”、“飞雪”。因为他们，秦王十年里没睡过一个安稳觉。可是他们却被一个默默无闻，名叫无名的秦国剑客所杀。消息传来，秦王振奋，急召无名上殿相见。在秦王的大殿里，神秘的烛火燃烧着，秦王与无名只有十步的距离，无名将击杀长空、残剑、飞雪的故事娓娓道来：他利用三人之间爱恨交织的关系，瓦解了他们的力量，各个击破，因此取胜。可秦王机智过人，听出了无名故事中的破绽，说出了另一个故事的\*\*：残剑等三人是主动求败，献出生命，用苦肉计帮助无名上殿，无名才是真正最危险的刺客，而无名告诉秦王：他看错了一个人，那就是残剑。于是真正的故事从头叙起……最后，无名拿起了剑，此时他离秦王只有十步，他的绝技是“十步一杀”。', '战国末期，赵国有三个名震天下的侠客，他们是：“长空”、“残剑”、“飞雪”。因为他们，秦王十年里没睡过一个安稳觉。可是他们却被一个默默无闻，名叫无名的秦国剑客所杀。消息传来，秦王振奋，急召无名上殿相见。在秦王的大殿里，神秘的烛火燃烧着，秦王与无名只有十步的距离，无名将击杀长空、残剑、飞雪的故事娓娓道来：他利用三人之间爱恨交织的关系，瓦解了他们的力量，各个击破，因此取胜。可秦王机智过人，听出了无名故事中的破绽，说出了另一个故事的\*\*：残剑等三人是主动求败，献出生命，用苦肉计帮助无名上殿，无名才是真正最危险的刺客，而无名告诉秦王：他看错了一个人，那就是残剑。于是真正的故事从头叙起……最后，无名拿起了剑，此时他离秦王只有十步，他的绝技是“十步一杀”。', '战国末期，赵国有三个名震天下的侠客，他们是：“长空”、“残剑”、“飞雪”。因为他们，秦王十年里没睡过一个安稳觉。可是他们却被一个默默无闻，名叫无名的秦国剑客所杀。消息传来，秦王振奋，急召无名上殿相见。在秦王的大殿里，神秘的烛火燃烧着，秦王与无名只有十步的距离，无名将击杀长空、残剑、飞雪的故事娓娓道来：他利用三人之间爱恨交织的关系，瓦解了他们的力量，各个击破，因此取胜。可秦王机智过人，听出了无名故事中的破绽，说出了另一个故事的\*\*：残剑等三人是主动求败，献出生命，用苦肉计帮助无名上殿，无名才是真正最危险的刺客，而无名告诉秦王：他看错了一个人，那就是残剑。于是真正的故事从头叙起……最后，无名拿起了剑，此时他离秦王只有十步，他的绝技是“十步一杀”。']