

本科毕业论文

论文题目：基于多模态知识图谱的课程学习系统设计与构建

|  |  |  |  |
| --- | --- | --- | --- |
| 姓 | 名：姚鸿韬 | 学 | 号：18302010017 |
| 院 | 系：软件学院 |  |  |
| 专 | 业：软件工程 |  |  |
| 指导教师：戴开宇 | | 职 | 称：高级讲师、硕导 |
| 单 | 位：软件学院 | | |

完成日期： 2022年 5 月 20 日

目 录

[目 录 I](#_Toc103950869)

[摘 要 1](#_Toc103950870)

[ABSTRACT 2](#_Toc103950871)

[第一章 绪 论 4](#_Toc103950872)

[1.1 研究背景 4](#_Toc103950873)

[1.2 相关研究和应用现状 4](#_Toc103950874)

[1.3 研究内容与意义 5](#_Toc103950875)

[1.3.1 研究内容 5](#_Toc103950876)

[1.3.2 研究意义 6](#_Toc103950877)

[1.4 论文组织结构 6](#_Toc103950878)

[第二章 系统相关技术与概念 8](#_Toc103950879)

[2.1 多模态知识图谱 8](#_Toc103950880)

[2.2 课程知识本体 9](#_Toc103950881)

[2.3 TF-IDF 9](#_Toc103950882)

[2.4 fastText 10](#_Toc103950883)

[2.5 开发平台与框架介绍 13](#_Toc103950884)

[第三章 课程学习系统的分析设计 15](#_Toc103950885)

[3.1 系统总体结构的分析设计 15](#_Toc103950886)

[3.1.1 系统的需求分析 15](#_Toc103950887)

[3.1.2 系统总体架构设计 16](#_Toc103950888)

[3.2 多模态教学资源链接知识点的分析设计 18](#_Toc103950889)

[第四章 课程学习系统的详细实现 20](#_Toc103950890)

[4.1 程序设计课程知识图谱的构建 20](#_Toc103950891)

[4.1.1 语料库的获取 20](#_Toc103950892)

[4.1.2 课程本体的详细设计 23](#_Toc103950893)

[4.1.3 课程概念的实例化 24](#_Toc103950894)

[4.1.4 课程知识点间关系的实例化 27](#_Toc103950895)

[4.1.5 课程知识图谱的存储 30](#_Toc103950896)

[4.2 多模态教学资源链接知识点的实现 31](#_Toc103950897)

[4.2.1 文本提取模块的实现 31](#_Toc103950898)

[4.2.2 fastText模型的训练 32](#_Toc103950899)

[4.2.3 资源分类与资源存储模块的实现 36](#_Toc103950900)

[4.3 系统总体功能的实现 37](#_Toc103950901)

[4.3.1 后端功能的配置与实现 37](#_Toc103950902)

[4.3.2 前端界面与交互的实现 39](#_Toc103950903)

[4.4 系统的部署 43](#_Toc103950904)

[4.4.1 数据库的部署 43](#_Toc103950905)

[4.4.2 后端应用的部署 44](#_Toc103950906)

[4.4.3 前端应用的部署 45](#_Toc103950907)

[第五章 课程学习系统的使用测试 46](#_Toc103950908)

[5.1 问卷设计 46](#_Toc103950909)

[5.2 结果分析 46](#_Toc103950910)

[第六章 总结与展望 49](#_Toc103950911)

[6.1 总结 49](#_Toc103950912)

[6.2 展望 50](#_Toc103950913)

[参考文献 51](#_Toc103950914)

[致 谢 53](#_Toc103950915)

# 摘 要

随着计算机科技的发达与各种MOOC平台的发展，在线教育成为了传统线下教育的一种有效备选方案。然而，在线教育在兼顾安全、高效等优点的同时，也可能带来“资源搜索费时”与“学习路径不清”两大问题，影响学生的学习效率。

为了顺应在线教育的大趋势，并行之有效地解决上述提到的两大问题，新的在线学习系统设计思路亟待被提出，以提升学生学习的积极性。本文以基于Java语言的“程序设计”课程为例，通过半自动化的方式抽取课程相关概念与概念间的关系，并使用机器学习的方式完成多模态课程教学资源到知识点概念的多对多匹配，完成了针对计算机课程领域的多模态知识图谱。基于多模态知识图谱，为课程的自学与初学者实现了用户友好的课程知识点学习系统，提升学生对知识点的获取效率，帮助学生构建对课程知识体系的完整理解。

在具体的实现细节中，本文使用基于Scrapy框架的爬虫应用获取Java课程的在线编程学习资源，使用TF-IDF算法提取其中的关键知识点概念；使用在线学习平台的语料集有监督训练fastText模型，试验并调优参数，然后使用训练完成的fastText模型完成多模态教学资源的文本信息多分类，将教学资源链接到知识点，并使用Neo4j存储多模态知识图谱；使用Spring Boot框架实现用户管理与知识图谱查询的后端系统；使用Vue.js框架实现前端交互界面与多模态教学资源的预览，并搭配D3.js框架完成知识图谱的可视化展示；最后使用Docker将系统部署到云服务器，为学生提供便利的访问途径。

最后，本文也对系统进行了使用测试与问卷调研，证明了系统的有效性，并对其不足与未来的改进方向做了总结与展望。

关键词 智慧教育，多模态，课程知识图谱，多分类，可视化

# ABSTRACT

With the development of computer technology and various MOOC platforms, online education has become an effective alternative to traditional offline education. However, while bringing the advantages of both safety and efficiency, online education may also cause two major problems: "time-consuming resource search" and "unclear learning path", which will affect students' learning efficiency.

In order to comply with the general trend of online education and effectively solve the above two problems in parallel, a new design idea of online learning system needs to be put forward to enhance students' initiative in learning. Taking the course of "Java basic programming" as an example, this paper extracts the relevant concepts of the course and the relationship between them in a semi-automatic way, completes the many to many matching between multi-modal course teaching resources and knowledge point concepts by means of machine learning, and completes the multi-modal knowledge graph in the field of computer education. Based on the multi-modal knowledge graph, a user-friendly course knowledge point learning system is structured for self-learners and beginners of the course, which improves the efficiency of students' acquisition of knowledge and helps students build a complete understanding of the course knowledge system.

When it comes to the specific implementation details, this paper uses the crawler application based on Scrapy framework to obtain the online programming learning resources of Java basic programming course, and uses TF-IDF algorithm to extract the key knowledge points and concepts; This paper uses the corpus set from the online learning platform to train the fastText model with supervised learning algorithm, tests and optimizes the parameters, and then uses the trained fastText model to complete the text information classification of multi-modal teaching resources, links the teaching resources to knowledge points, and uses Neo4j to store multi-modal knowledge graph; This paper uses Spring Boot framework to structure the back-end system of user management and knowledge graph query; This paper uses Vue.js framework to structure the front-end interactive interface and the preview of multi-modal teaching resources, and uses D3.js framework to complete the visual display of knowledge graph; Finally, this paper uses Docker to deploy the system to the cloud server to provide convenient access for students.

Finally, this paper carries out a use test and questionnaire survey of the system and proves the effectiveness of it. This paper also summarizes the shortcomings of the system and looks forward to the future improvement direction.

**Keywords** Intelligent Education, Multi-modal, Curriculum Knowledge Graph, Multi-classification; Visualization

1. 绪 论
   1. **研究背景**

近年来，“在线教育”这一概念被逐渐普及，并随着计算机科技的发达与各种MOOC（Massive Open Online Courses）平台和工具的普及而不断地快速发展着。尤其是在当下疫情常态化的大环境下，传统的线下面对面教育可能会大大提升老师与学生被病毒传染的风险，并因此大大受到地域与国家政策的多方面限制。在线教育作为一种备选方案，同时具备安全与免费两大优点，无疑是应对目前情况的最佳替代品。

在线教育除了能将传统课堂移动到云端，也为世界上更大范围的人群提供了高质量的自学平台，让全球各地的用户即便不在顶尖大学就读，也能体验这些大学的课程，收获高质量的知识与教育，摆脱自身条件的限制，真正实现教育的公平性。然而，这种自学平台快速发展的同时也带来了“信息过载”的问题。自学者必须不断在互联网上充斥着的大量各种类型的教程中自行搜索和选择，这种重复无疑会浪费大量时间，而且有时会导致某些重要知识的遗漏。此外，自学者可能会连续学习到多个不相关的知识点，或者跳过某些基础知识点而直接学习到更高阶的知识点，抑或者学习到重叠较大的课程等等。同时，自学者很难全面、深入地了解全网关于课程的信息，更不用说满足高效资源搜索和利用的需求。这正是当下在线教育面临的一大难题。

* 1. **相关研究和应用现状**

知识图谱（Knowledge Graph）的发展起源从语义网（Semantic Web）开始，它是在本世纪初由Tim Berners-Lee提出的。Tim作为万维网（World Wide Web）的发明者之一，在2001年提出了语义网的概念——一种让计算机能理解并处理复杂网络信息与逻辑推理的方式，试图将互联网从一种仅供人类使用的文档媒介提升为一种可自动处理数据和信息的媒介。语义网采用结构化语言，如RDF（Resource Description Framework，资源描述框架）三元组进行概念和信息的表示，并使用本体模型定义术语之间的分类与关系[[1]](#一)，而这些都成为了知识图谱的基础。2012年，Google收购了原本致力于构建一个世界级规模的开放Web语义知识库的Metaweb公司后，正式提出了知识图谱的概念，并将其应用于搜索引擎之中，以提供增强的语义搜索的功能。自此，知识图谱正式开始被业界广为研究与应用。

知识图谱在简化知识的表示和处理方面效果出众，因此被大量使用于语义搜索、智能问答系统、推荐系统等方面。近年来，知识图谱被越来越多应用于教育领域。2010年，M. Bucos提出了建立以本体为基础的电子教育平台的必要性，并提出了构建电子信息教育本体的一些方法，可看作后续使用知识图谱作为手段聚焦教育领域研究的基础[[2]](#二)；Yang Z.等人提出构建基于学科知识图谱的智能问答系统自动回答学生的问题，同时利用大数据算法分析学生的答疑行为，预测学生的未来学习情况，为教师形成反馈[[3]](#三)；Penghe Chen等人试图构建针对基础教育的知识图谱系统K12EduKG，借助命名实体识别技术识别各种来源中的学科知识点，并利用数据挖掘技术找出潜藏的知识点间的前提关系[[4]](#四)；在此基础上他们还更进一步，对比了几种知识抽取方法的性能，并尝试用系统构建了数学学科的知识图谱，与领域专家给出的结果进行对比，证明了系统的可行性[[5]](#五)。

此外，也有一些研究聚焦于本科计算机教育的知识图谱构建与应用，并将构建主体定位为一些教学资源。Chao Huang聚焦MOOC课堂，并以计算机网络课程为例，从课程大纲抽取课程知识图谱的基础框架，并从视频字幕中利用新提出的NNGP模型抽取知识点进行补充，这种新模型在对视频字幕进行知识点抽取的性能比现有模型更优[[6]](#六)；Thushari Atapattu等人提出了一种从课程幻灯片中提取概念图的方式，并对系统进行实验，认为将这种概念图作为扁平化PPT学习的补充是有利的[[7]](#七)。综上所述，知识图谱已成为实现智慧教育应用的有效方式，并能通过多种构建方式有效地从不同类型的信息来源获取知识，最终完成图谱的搭建。

* 1. **研究内容与意义**
     1. **研究内容**

本文主要设计并实现了多模态课程知识点学习系统，集合多种学习资源并明晰课程学习路线，以期提高学生在自学过程中的积极性。在研究了课程知识的领域本体构建方法与多模态知识图谱的相关技术后，开展了以下几项工作：

选择以基于Java语言的程序设计课程作为例子，从语料库抽取课程知识相关概念，设计了课程知识的领域本体，利用图数据库进行存储，以知识图谱的形式表达章节与知识点间的关系；

收集多种不同类型的课程相关学习资源，包括PPT（Microsoft Office PowerPoint）与教学视频等，通过自动化方式提取其中的文字信息；基于已收集好的知识点信息训练fastText模型，并将训练好的模型作为文本分类器对学习资源的文字版本进行多分类，最终将学习资源与知识点间的关系同样存储到图数据库中，形成基于Java语言的程序设计课程的多模态知识图谱；

在知识图谱的基础上，通过前端Vue.js框架与后端Spring Boot框架实现多模态课程知识点学习系统，包括知识点相关学习模块与个人相关管理模块，同时利用D3.js框架实现课程知识图谱的可视化功能，为学生提供用户友好的网站交互界面。

* + 1. **研究意义**

本文通过构建多模态课程知识点学习系统来试图解决在线自学过程中的“资源搜索费时”与“学习路径不清”这两个主要问题。系统能够提升学生在自学环节中对知识点相关信息获取的准确性，多模态的学习资源也能减少学生用于重复搜索知识点相关资料所浪费的时间，提高学生在自学过程中的积极性；同时，本文实现了课程概念间关系的可视化，帮助学生明确清晰的课程自学路线，避免知识点的遗漏或跳跃学习，有助于学生构建完整的课程知识体系；每个资源可能链接到多个知识点，也有利于学习者进行多个知识点间的融合运用，并进行查漏补缺。因此，通过本体论形式化地描述课程内容的语义关系，设计并实现的课程学习系统，解决学生在线自学过程中存在的问题，能够提高在线教育的质量并推进网络教育的发展，同时本文提出的系统构建方法也能被泛用到其它类型的课程及教学资料上，非常具备实际意义。

* 1. **论文组织结构**

本文共分为六章来组织和完成论文的整体内容结构。

第一章为绪论，介绍了本文的研究背景及相关研究的现状，并概括了本文的主要研究内容及意义。

第二章为系统相关的技术与概念的介绍，主要包括：多模态知识图谱介绍、课程知识本体的划分与表示、TF-IDF算法的详述、fastText技术的概述与要点详述和系统的开发平台与框架的简介。

第三章为系统的分析与设计介绍，主要包括对系统总体功能与架构的分析与设计以及对多模态课程教学资源多分类功能各模块的设计。

第四章为系统详细的实现步骤的介绍，主要包括课程知识图谱的构建、多模态课程资源链接知识点模块的实现及课程知识点学习系统的总体功能的实现与部署。课程知识图谱的构建包括语料库与课程相关概念的提取、课程知识点间关系的确定及图谱的储存；资源链接知识点模块包括fastText模型训练、文本提取模块、资源分类与存储模块的实现。

第五章是对系统的使用测试，主要邀请用户实际对系统进行使用，并设计调查问卷，分析数据结果，用以证明系统的有效性与可用性。

第六章为总结与展望，简要概述了本文的内容，并指出了不足之处和需要改进的部分，为以后的优化与研究点明了方向。

1. 系统相关技术与概念
2. **多模态知识图谱**

近年来，关于知识图谱的应用与研究不断兴起。知识图谱是描述现实世界中事物和它们之间关系的一种方式，用节点代表实例，用边表示实例包含的属性或它们之间的关系，这也类似于人类的认知方式。因此，知识图谱被选作收集、表示、存储和检索具有丰富信息和关系的数据的主流媒介。传统的知识图谱仅以纯文本的形式进行呈现与储存，而在当下这个大数据时代，数据会以更多形式存在，比如图片、音频、视频等，传统的知识图谱对此则无能为力。此外，考虑到本文的应用场景，对于学生而言，单纯枯燥的文字知识难以激发学生的学习兴趣，往往需要依靠图片、视频、幻灯片等多媒体资源来辅助学生对知识点进行理解与记忆[[9]](#九)。

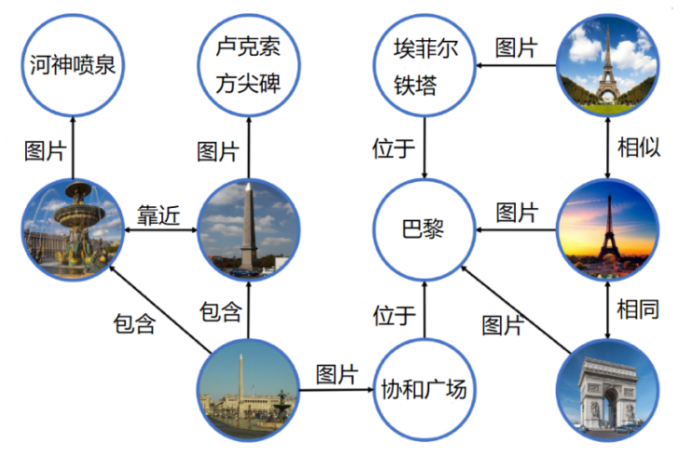
为此，多模态知识图谱（Multi-Modal Knowledge Graph）的概念被提出。所谓模态，就是信息的特定展现形式。通过将实体接地到相应的图片、音频和视频数据上，并将实体映射到它们在物理世界中富有意义的对应参照物中，计算机在面对一个特定实体时能产生类似真人的“体验”[[10]](#十)。多模态知识图谱不再将实体信息局限在文本形式，而是试图在其中加入其他模态的知识，以丰富其形式与内容。例如，在搜索引擎搜索“复旦大学”时，展示出来的结果不仅包含复旦大学的简介、专业等文字形式信息，也包括类似校徽、校内照片等的图片信息，甚至还会包括其所在位置与周边环境的地图等，以帮助人们对这个实体有更全面的认识。对于学生而言，多模态形式的知识点学习不仅可以长时间地吸引他们的注意力，确保他们学习的积极与主动性，多种模态的教学资源也能相互提供补充，完善学生对知识点的整体理解，因此多模态知识图谱在教育领域的应用研究也在被重视。

图2-1 多模态知识图谱样例[[10]](#十)

1. **课程知识本体**

本文所构建的课程知识本体是一种领域本体（Domain-specific Ontology）。区别于通用本体（General Ontology），领域本体只关注特定领域或者小范围内的的概念，具有很强的针对性。课程知识本体提供与课程概念相关的基本范围，为了能够形成对课程内部概念的一致理解，一般包含用形式化语言描述的课程中的定义及定义之间的关系。从分层角度而言，课程知识本体可以被划分为三层[[11]](#十一)，顶层为课程概念，中间层为章节概念，属于课程概念，底层为知识点概念，属于章节概念。这也符合实际的课程大纲与课程教学结构。除了概念层次间的属于及包含关系，知识点概念间还可以定义前驱与后继关系，因为知识点的学习有时存在依赖性，需确保明确的学习顺序。另外，本文将多模态的教学资源概念也纳入课程知识本体范围之内，与知识点概念层同级，并定义与知识点概念间存在关联关系。

知识图谱的表示形式与本体的结构相匹配，因此本文选择使用知识图谱来存储并展示课程知识本体，而多模态的知识图谱则可以将教学资源也纳入存储范围。在本文的多模态知识图谱中，使用节点表示课程、章节、知识点与教学资源概念，使用节点间的边表示包含、前驱等关系，使用节点的属性表示概念相关的信息，如教学资源对应的URL（Uniform Resource Locator，统一资源定位器）与知识点对应的优先级等。多模态知识图谱的展现形式使概念之间的关系更为清晰，能更好地支持课程知识点学习系统的知识图谱可视化展示、教学资源在线预览及其它功能的展示。

1. **TF-IDF**

TF-IDF（Term Frequency-Inverse Document Frequency），是一种用统计学思想从多篇长文本中分别提取关键字的方式。一个单词在单一文本中多次出现，说明这个单词对这篇文章一定非常重要，但如果这个单词在每篇文章中都出现十分频繁，就说明这个单词非常常见，不具有很好的代表性，不能体现这篇文章的特点。TF-IDF的算法思想就是基于上述描述。

词频（TF）表示在长文本出现某个词语的频率，其计算如下：

其中，代表在下标为j的长文本中出现下标为i的词汇的次数。因此，上式等于长文本中某个词汇的数量除以长文本中总的词语数量。

逆文本频率（IDF）则用来衡量一个词汇在整个语料集中的普遍程度，其计算公式为：

其中，对数内为长文本的总个数除以包含该词汇的长文本个数，分母加1则为了防止除以0的情况发生。

最后，将词频与逆文本频率相乘，即可得到某一特定词语针对某一特定语料库内文件的词频-逆文本频率指数TF-IDF：

TF-IDF常被搜索引擎作为网页与用户查询之间相关程度的度量或评级指标之一应用。主流搜索引擎网站，如Google，曾公开承认在其排序算法中使用TF-IDF[[16]](#十六)，因此本文选择使用TF-IDF进行课程语料库中知识点概念的抽取算法。

1. **fastText**

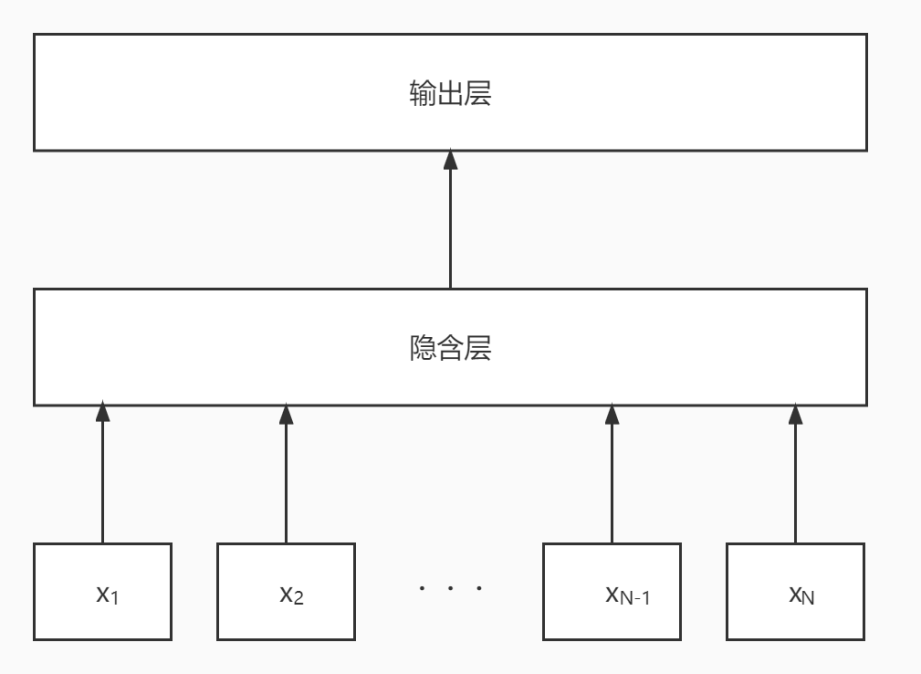
****fastText是Facebook AIResearch于2016年推出的一款词向量计算工具，也能用来完成文本的分类，具有免费、轻量级等多重优点。fastText的主要作者之一Tomas Mikolov同样也是另一种优秀的词向量生成模型word2vec的作者，而fastText的模型框架与word2vec的CBOW（Continues Bag Of Words，连续词袋模型）架构非常相似，其主要的文本分类框架如下图所示。

图2-2 fastText文本分类模型架构[[8]](#八)

上图中，总体模型框架共分三层。底层为输入层，图示的输入为N个单词的n-gram特征，这些单词被用来表示整个文档信息；中间为隐含层，在此模型会将输入层的n-gram特征求和取平均，得到文档向量作为本层输出；顶层为输出层，隐含层得到的文档向量在此通过Softmax回归，完成多类别分类任务。

与一般的深度神经网络模型以及word2vec相比，fastText做出的突破主要体现在两点：字符级别的n-gram特征及分层Softmax回归。这两大特点造就了fastText的高精度与高性能。

* **字符级别的n-gram特征**

n-gram特征多被用于词向量的计算或文本特征的提取，核心思想是将长文本内容排序后取一个长度为n的窗口，窗口每次向后移动一位，每移动一次就保存一次内容，最终整体文本的特征内容就由这多个长为n的文本片段表示。比如，“我在复旦大学读书”相应的bigram（2-gram）特征可为“<我/在>，<在/复旦大学>,<复旦大学/读书>”。

类似上面的例子，大部分以往的模型都是以单词作为基本单元进行切割，即词粒度的n-gram特征。虽然这种想法非常自然，但是存在一些问题：首先，词粒度模型忽略了单词内部的信息，比如“复旦”和“复旦大学”两个单词包含较多的公共字符，但这种单词内部特征由于被替换为不同的标识符而丢失了；此外，一些低频词由于在语料集中非常罕见，无法得到足够的训练，因此最后得到的词向量效果不佳；如果额外出现了一些词表本身不包含，或者输入错误的词语，词粒度模型就更加无能为力了。

为此，fastText选择使用字符级别的n-gram特征，即将每个单词打散到字符粒度，试图通过分析字符间的顺序关系得到单词内部更细致的语义。将字符级别的n-gram求和就能计算得到原始单词的词向量。从最后得出的训练效果来看，字符粒度的n-gram特征基本解决了上述问题。由于词内n-gram也可以应用到其它词语，低频词能得到比词粒度模型效果更好的词向量，形式上相似的单词，其词向量也会更相似；语料集以外单词的词向量也仍然可以将通过语料集内单词训练生成的对应字符级别n-gram特征相加来计算。这一特点保证了fastText的高精度。

* **分层Softmax回归**

Softmax回归（Softmax Regression），是多分类版本的逻辑回归（Logistic Regression），一般被作为多分类模型的最后一层。Softmax将输入经过指数变换以及归一化后，保证值域在范围内，得到最终输出的类别概率值，以体现此模型对每种分类的置信度。一般的Softmax计算时间复杂度为，其中h为词向量维度，k为分类类别个数。在类别数较大时，该时间复杂度是较高的。

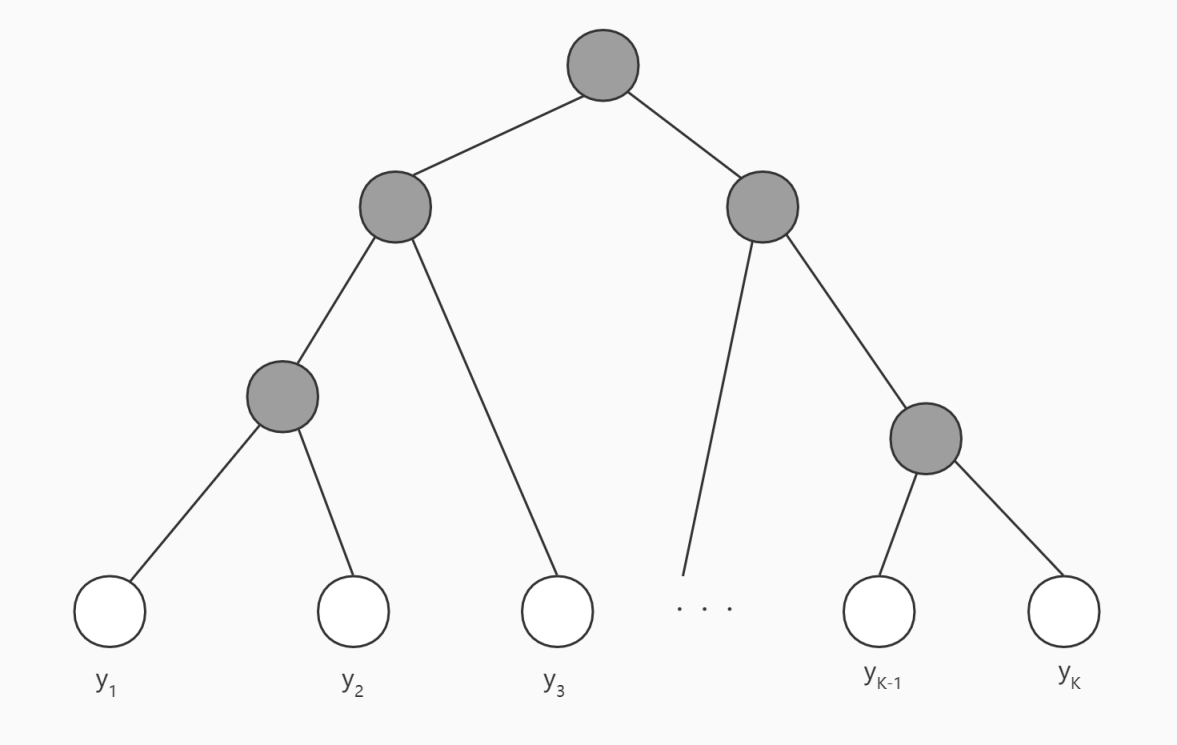
分层Softmax回归的基本思想是在一般Softmax的基础上将一个全局的多分类问题多次分解，每次分解都得到一个二分类。具体如下图所示：

图2-3 分层Softmax示例

分层Softmax二叉树的结构是根据每种分类出现频次所构造的哈夫曼树（Huffman Tree）。上图中，K个不同的分类构成K个白色叶子节点，K-1个灰色中间节点都作为普通的逻辑回归单元，从根节点开始根据节点的逻辑回归输出决定接下来的路径，而最终得到的叶子节点输出就等于路径上所有逻辑回归单元输出的乘积。通过树的层级结构，模型不需要将每种分类情况的概率都计算得出，其计算时间复杂度降为，大大提升了高类别数情况下的时间复杂度，也保证了fastText模型的高性能。

正是源于上述两大特性，fastText能够做到取得不逊色于主流深度神经网络模型分类精度的同时将模型训练与分类计算速度降低几个数量级。综合其模型结构简单、训练效率高及分类效果好的特点，本文选择fastText作为文本分类模型，用来对教学资源提取出的文本进行多分类，并链接到相应知识点上。

1. **开发平台与框架介绍**

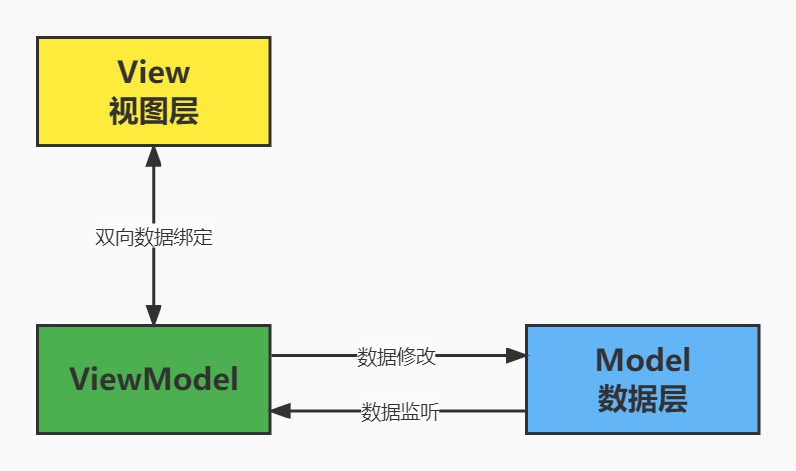
本文选择使用Vue.js框架构建前端。Vue.js是目前最热门的前端JavaScript框架之一，具备简单轻巧、易上手等优点，并可实现组件化的复用设计。Vue.js采用MVVM（Model-View-ViewModel）设计模式，以双向数据绑定作为核心思想，通过ViewModel层隔离视图层与数据层间的联系，由ViewModel层对数据内容进行监听，并实时更新视图的对应内容；同时用户的操作导致的视图数据变化也会被ViewModel感知，实时同步数据层内容，降低视图层与数据层的耦合程度，利于开发。同时，Vue.js还可以与多种扩展库，如Vuex、Vue Router、Element UI等，完成组件间数据通信与页面路由等功能，便于用户快速构建复杂的前端应用。

图2-4 MVVM模式

本文选择使用D3.js框架完成知识图谱的显示。D3（Data-Driven Documents）是一个JavaScript函数库，帮助用户使用HTML、CSS、SVG（Scalable Vector Graphics，可缩放矢量图形）及Canvas（画布）完成数据可视化的任务。D3.js完全遵循现有的Web标准，因此可以独立运行于现代浏览器中，使用强大的预封装的可视化组件驱动DOM组件来操作。D3.js本质上是JavaScript，因此其实所有功能都可以由原生JavaScript实现，D3.js的优势在于它能大大减少用户的工作量，将数据可视化的复杂步骤简化到几个函数调用，用户只需要负责输入对应的数据配置即可。基于上述原因，本文选择使用Vue.js框架构建前端形成课程知识点学习系统的单页应用，并结合D3.js框架完成课程知识点图谱的可视化。

本文选择使用Spring Boot框架实现后端应用。Spring Boot框架基于现有的Spring框架，能大大减少程序员开发时需要进行的项目配置与环境搭建过程。Spring Boot框架也集合了Spring框架已有的优点，通过依赖注入帮助程序员实现控制翻转，通过面向切面保证程序的灵活性，同时使用更为简单、轻便。本文选择使用Maven进行后端的依赖管理。Maven是由Apache开源的一个项目管理工具，帮助程序员完成统一的第三方库，并能够通过指定类库的版本解决冲突问题。

本文选择使用Neo4j图数据库作为后端数据库之一，存储课程知识点图谱信息。Neo4j是一个流行的NoSQL（Not Only SQL）图数据库。图数据库的存储主要由顶点、边与属性构成，这与知识图谱的结构非常吻合，因此非常适合用于存储图谱信息。Neo4j为用户提供了一种新的查询语言Cypher，类似常见的SQL语言，可以对数据库进行高效的操作。本文选择使用MySQL作为另一个后端数据库。Java为MySQL和Neo4j都提供了JDBC（Java Database Connectivity，Java数据库连接）封装与驱动，而Spring Boot则为JDBC提供了进一步封装，使用自动注入将数据库相关的配置信息直接注入到对应的JDBC Template中，并允许用户使用定义好的数据访问接口简便地操作数据库。结合以上理由，本文使用Spring Boot同时连接MySQL与Neo4j数据库完成后端系统的构建。

1. 课程学习系统的分析设计
2. **系统总体结构的分析设计**
3. **系统的需求分析**

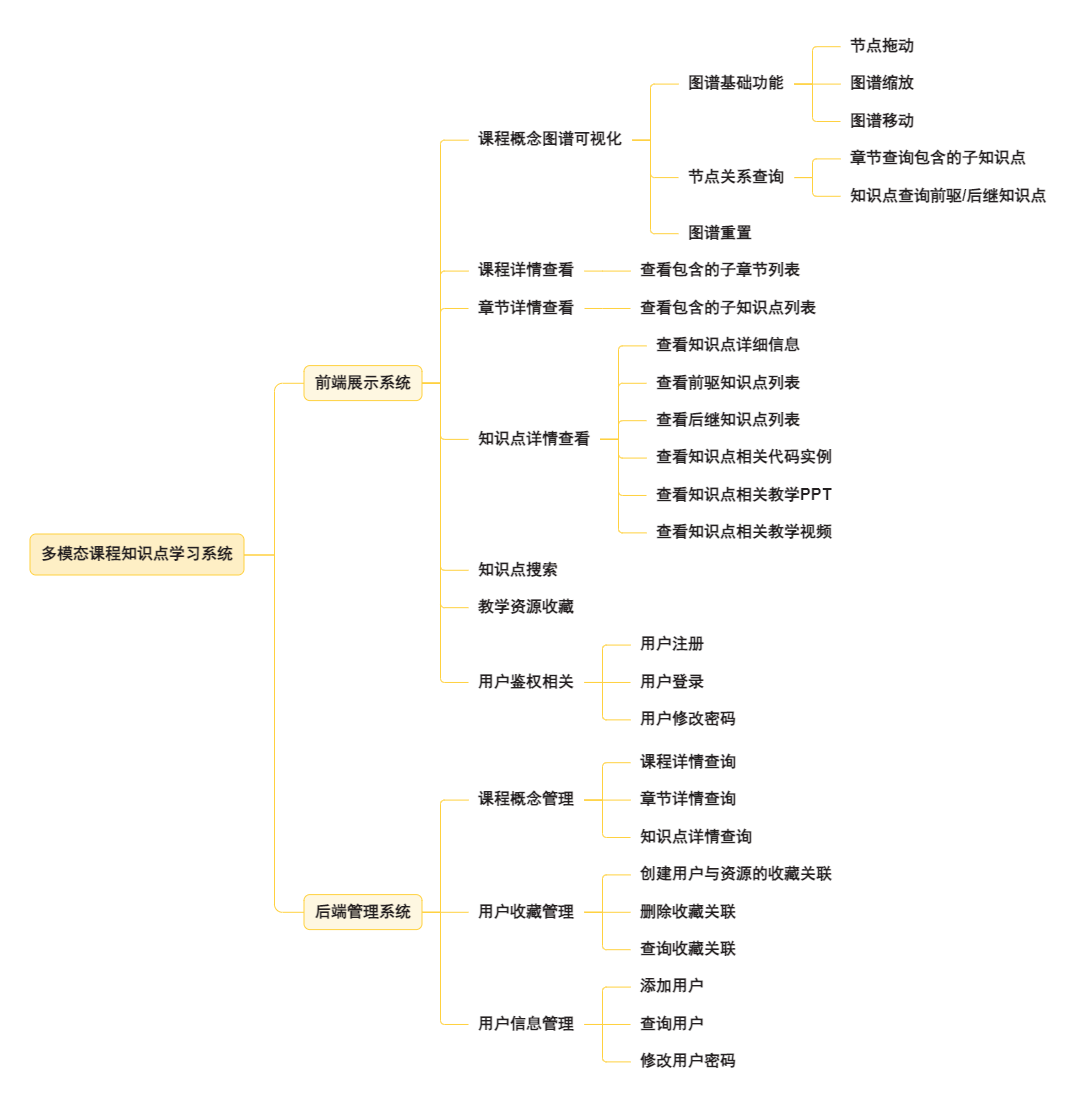
本文旨在通过多模态的课程知识点学习系统完善学生对知识点理解与认识，保证学生学习的积极性，同时帮助学生明晰知识点间的关联与学习路径，因此系统的功能重点在于多模态教学资源的展示、知识点图谱的展示以及知识点详情的学习。系统的总体功能需求可分解为若干功能单元，需求结构图如下所示：

图3-1 系统需求结构图

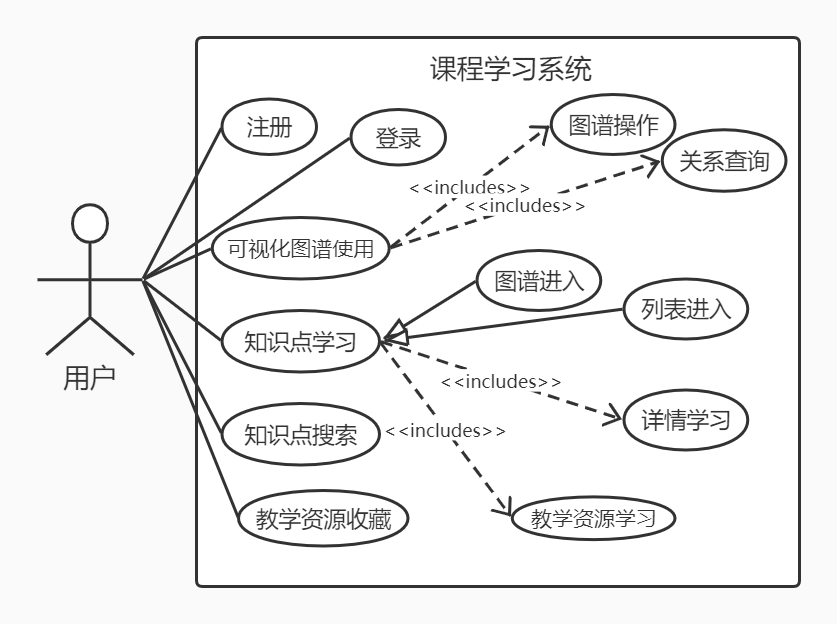
用户经过注册与登录后，进入课程知识的图谱展示界面。用户可以任意缩放、拖动图谱及图谱中的节点；点击节点会展示节点的概述，并可以根据节点的类型查询相应的关系节点：章节节点可以查询属于它的子知识点（包含关系）；知识点节点可以分别查询其前驱知识点或后继知识点；也可以从此界面跳转到节点的详情页面。在课程/章节概念的详情页面，展示对应包含的子章节/知识点列表，可以通过点击跳转。知识点概念的详情页面是系统功能的核心，主要功能包括知识点详细信息学习、前驱/后继知识点列表展示、知识点相关代码实例展示、知识点相关教学PPT查看与下载、知识点相关教学视频观看等。前驱/后继知识点列表界面可以点击知识点跳转到对应页面；教学资源页面可以进行资源的收藏/取消收藏；教学资源可能同时关联多个知识点，也可以通过点击跳转到对应知识点。用户还可以通过在搜索栏输入关键字搜索包含关键字的所有知识点，并点击跳转。在用户的个人模块，用户可以查看自己收藏的教学PPT以及教学视频，并修改自己的密码。具体的系统用例图如下所示：

图3-2 系统用例图

综合以上功能，课程知识点学习系统可以满足自学者在学习过程中的各方面需求，通过多模态的教学资源丰富学生的学习过程，并通过可视化的图谱明晰概念间的关系与知识点的学习顺序，最大程度地帮助学生完成课程的学习。

1. **系统总体架构设计**

本文中的系统通过前后端分离的方式进行开发。前端选择使用Vue.js框架搭建。界面主要使用Element UI的组件库进行构建，完成页面的整体布局及导航栏、消息提醒、按钮等页面展示，以提供优美的交互界面。在知识图谱可视化的实现上，使用D3.js框架提供的封装函数生成SVG标签，并根据Vue.js中配置的Node与Relation数据动态绘制节点与关系图形。展示教学资源时，通过以节点属性形式存储的URL访问并获取存储于阿里云OSS（Object Storage Service，对象储存服务）上的真实资源，并使用前端对应组件标签显示。在组件间通信的实现上，使用Vuex记录全局数据完成各界面间数据与信息的传递。在路由跳转的实现上，为完成单页应用的设计，使用Vue Router的Hash模式，利用浏览器修改锚点（#）页面不发生刷新的特性，完成前端界面改变但不重新向后端发送请求的功能。同时，还利用Vue Router的路由守卫功能对进入特定页面的条件进行限制，防止恶意跳过鉴权阶段等现象发生。

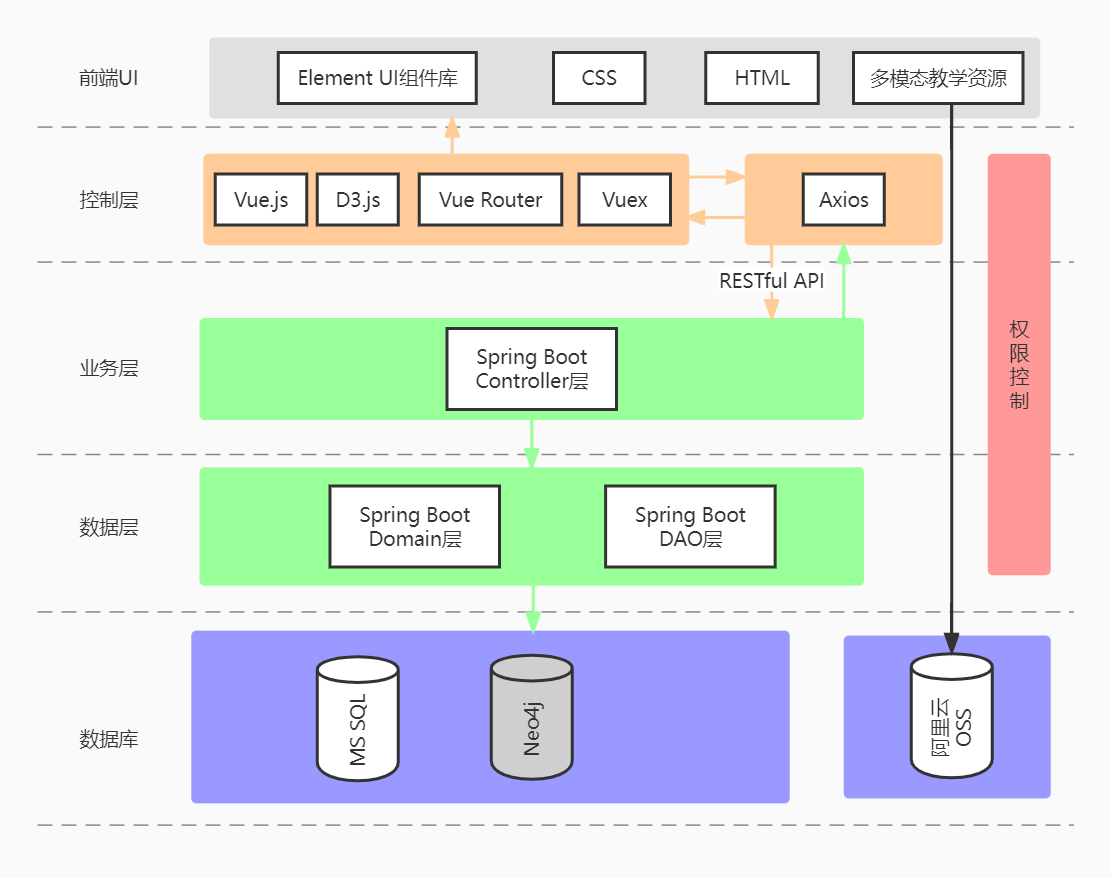
后端选择使用Spring Boot框架搭建，同时连接MySQL与Neo4j两个数据库，主要分为Domain层、DAO（Data Access Object）层及Controller层。Domain层即实体层，包含与两个数据库中数据表/实体一一对应的实体类，如User类对应MySQL中的用户表，Knowledge类对应Neo4j中的知识点实体等，将数据表中的字段与课程实体概念的属性映射为实体类的属性，并且每个实体类都是一个标准的JavaBean，包含一个无参的构造方法及每个属性的getter/setter方法。DAO层负责对数据库进行操作。Controller层负责响应并转发前端的请求，通过@RequestMapping注解指定转发的请求使用的方法及路径，并通过@RequestParam或@RequestBody指定请求的参数；同时，由于后端功能相对简单，将原本在Service层的业务逻辑实现移动到Controller层，通过对DAO层进行调用一并完成，为前端提供交互的接口。总体的系统架构图如下所示：

图3-3 前后端分离的系统总体架构

系统通过Docker部署到阿里云平台的服务器，为所有用户提供随时随地可访问的课程知识点学习系统服务。Docker是一款应用容器引擎。一次Docker部署主要涉及一个Docker镜像（Image）与Docker容器（Container）。容器基于镜像创建，并且多个容器可以基于同一个镜像。容器允许用户打包一个应用及其所有依赖包并进行发布。容器与容器间采用沙盒机制，即使发布于同一个服务器上，相互之间也不会有任何影响，完全独立。

根据上述架构分析，多模态课程知识点学习系统可以分为四个部分分别部署，包括：前端Vue.js应用、后端Spring Boot应用、MySQL数据库以及Neo4j图数据库。部署系统时，分别为每一个应用编写对应的Dockerfile文件，以及可能的需要的配置文件，然后基于对应的镜像（Node.js、Java、MySQL及Neo4j）构建容器并部署。容器间的交互完全采用HTTP调用，即IP地址（Internet Protocol Address）+端口号的方式完成。

1. **多模态教学资源链接知识点的分析设计**

多模态教学资源需要经过分类算法后能链接到对应的知识点，而包括多模态教学资源的图谱才能完成本文课程知识点学习系统的功能需要。在构建了课程知识概念图谱后，本文对多模态教学资源经过分类并链接到相关知识点节点的过程进行了分析与设计，包括以下主要步骤：

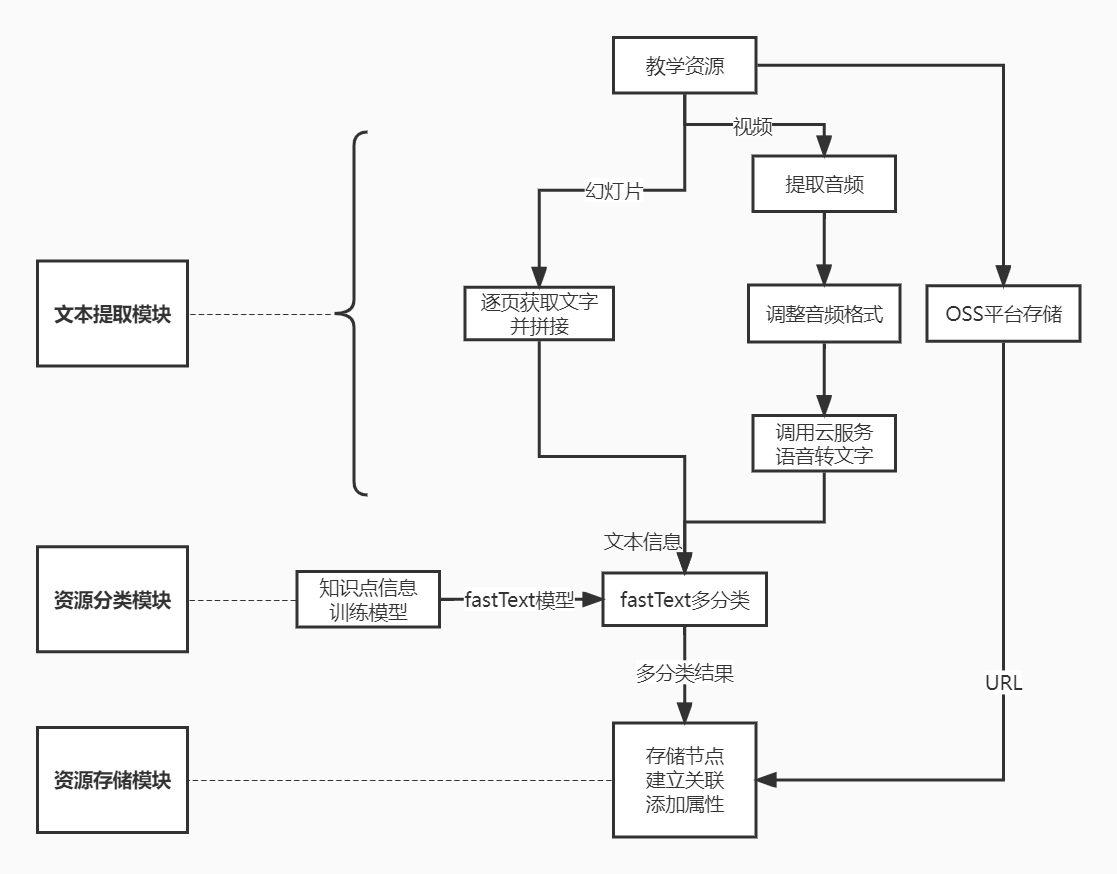
1. 根据教学资源类型，使用不同方法提取其中的文字信息；
2. 对文字信息进行中文分词；
3. 使用知识点详细信息训练fastText模型；
4. 使用训练好的fastText模型对文字信息进行多分类；
5. 根据多分类结果，将教学资源存储为图谱节点，并与对于知识点相连；
6. 将教学资源上传到阿里云OSS平台托管；
7. 将OSS平台提供的URL设置为教学资源节点的属性。

根据以上的教学资源链接知识点流程，功能主要可分为三个子模块，即文本提取模块、资源分类模块以及资源存储模块。功能的模块及流程图如图3-4所示。

fastText的分类模型使用的是有监督训练，即通过使用已有正确分类标注的数据训练模型，学习训练集中的相关特征。本文使用的训练资料来源于知识点对应的学习信息，经过中文分句、分词与停用词去除后，每句句子用作一行训练输入。训练的预期标注输出则包括两个部分：

1. 包含本句训练输入句子的知识点，这一部分是确定存在的；
2. 本句句子包含的关键字所属的其它知识点。

在得到了训练资料并设置合适的训练参数后，就可以对fastText模型进行训练。训练模型时需要随机将训练资料分为训练集与验证集，防止过拟合现象发生。通过上述方式得到并保存训练好的fastText模型后，就可以使用到后续的资源多分类步骤中。

图3-4 资源链接知识点模块及流程图

1. 课程学习系统的详细实现
2. **程序设计课程知识图谱的构建**

Java具备简单、可移植、多线程等多方面优点，并可以构建Web应用程序、分布式系统、数据库、Android应用程序等各种类型应用，是当下最流行的开发语言之一。对于学生而言，Java具有强类型、面向对象等特点，利于编程初学者理清计算机底层数据的原理，并且为后续的面向对象程序设计课程打好扎实的基础，因此Java程序设计成为了很多大学计算机相关专业程序设计入门课程的首选。当然，Java程序设计知识点内容繁多，各知识点间相互穿插，对于初学者，尤其是自学者而言难以入手，因此本文选择基于Java语言的程序设计课程作为研究案例进行课程知识图谱与课程知识点学习系统的构建。

1. **语料库的获取**

在确定选择基于Java语言的程序设计课程作为系统及研究的主要对象后，本文主要参照Java课程的常用教材《Java语言程序设计（基础篇）》[[12]](#十二)，并结合在线编程学习平台菜鸟教程提供的Java教程[[13]](#十三)及在线视频网站bilibili上发布的黑马程序员Java教程[[14]](#十四)作为辅助的信息来源，用于获取程序设计的知识信息及多模态教学资源。

其中，《Java语言程序设计（基础篇）》教材是PDF（Portable Document Format）格式的文件。本文选择使用基于Python的第三方库pdfplumber来将PDF文件转化为一般的TXT格式文件，便于后续的处理与概念提取。具体实现步骤如下：

1. 调用pdfplumber.open()函数并传入教材的路径将本地PDF文件打开为一个pdfplumber.PDF实例，此实例即代表整个PDF文件，后续的所有处理都基于这个实例完成；
2. 遍历PDF.pages属性。此列表属性包含了PDF文件每一页对应的pdfplumber.Page实例；
3. 对每个pdfplumber.Page实例，通过Page.chars属性获取每一页上所有的字符。
4. 对每一字符做清洗，例如去除页眉页脚，清除多余空格及换行符等

经过以上步骤，可以获取教材PDF的目录以及对应章节的文字格式信息。

为了实现从自动化地从在线编程学习平台获取程序设计的课程概念详细信息以及对应的代码实例，本文选择使用Scrapy框架[[15]](#十五)制作有针对性的网络爬虫程序进行页面数据的采集。Scrapy是一个基于Python的第三方框架，用于自动化地获取互联网上的Web页面，并根据规则得到其中的网页结构，符合本文的应用需要。一个基于Scrapy框架的爬虫应用一般包含以下组件：

1. Scrapy Engine（引擎）：负责整体的控制调度、数据传递等；
2. Scheduler（调度器）：负责列队来自引擎的请求（Request），并在前一个请求处理完毕时按序返还后一个；
3. Downloader（下载器）：负责将请求发送至互联网，并将收到的Response响应返还；
4. Spider（爬虫）：爬虫应用的核心，负责处理响应，按用户制定的规则从响应中清洗出网页结构与数据，制作为Item，并交还下一步需要爬取的新请求；
5. Item Pipeline（管道）：负责处理来自爬虫的Item数据，并进行过滤、保存等操作；
6. Downloader/Spider Middlewares（下载/爬虫中间件）：可选的扩展组件，允许用户自定义一些辅助功能。

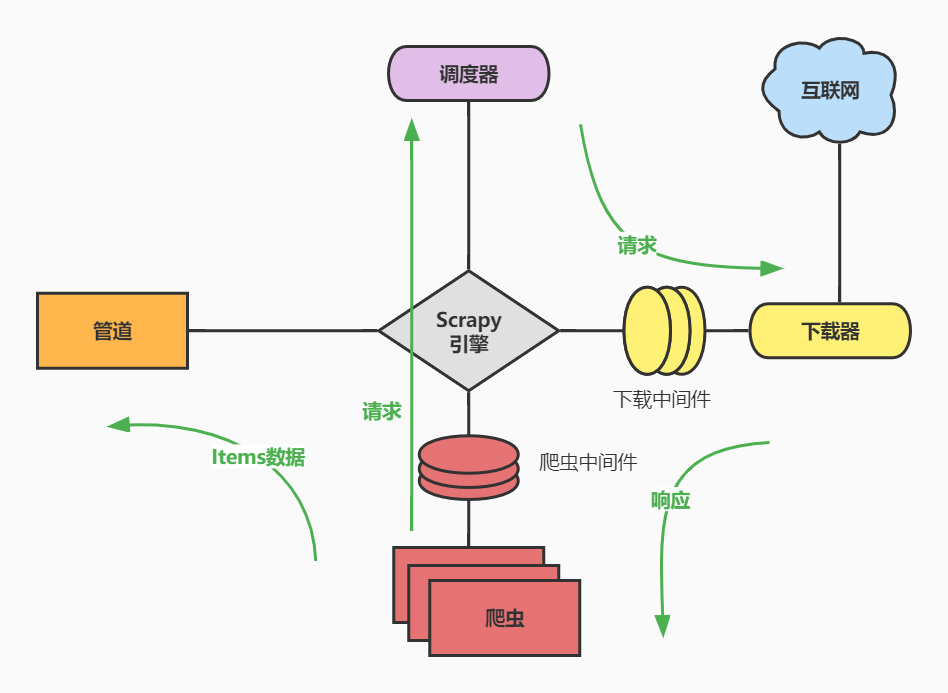
下图展示了Scrapy的基本架构，其中绿色箭头代表框架中的数据流向。

图4-1 Scrapy架构图[[15]](#十五)

以下是Scrapy应用的爬虫核心示例代码：

1. **class** RunoobSpider(scrapy.Spider):
2. name = 'runoob'
3. allowed\_domains = ['www.runoob.com']
4. base\_url = 'https://www.runoob.com'
5. start\_urls = [base\_url + "/java"]
7. **def** parse(self, response, \*args, \*\*kwargs):
8. name=response.xpath('//\*[@id="content"]/h1/text()').extract\_first()
9. content = response.xpath('//\*[@id="content"]').extract()
10. item = HTMLItem()
11. item['name'] = name
12. item['content'] = content
13. **yield** item
14. url\_list = response.xpath('//\*[@id="leftcolumn"]/a/@href').extract()
15. **for** url **in** url\_list:
16. **yield** scrapy.Request(self.base\_url + url, callback=self.parse)

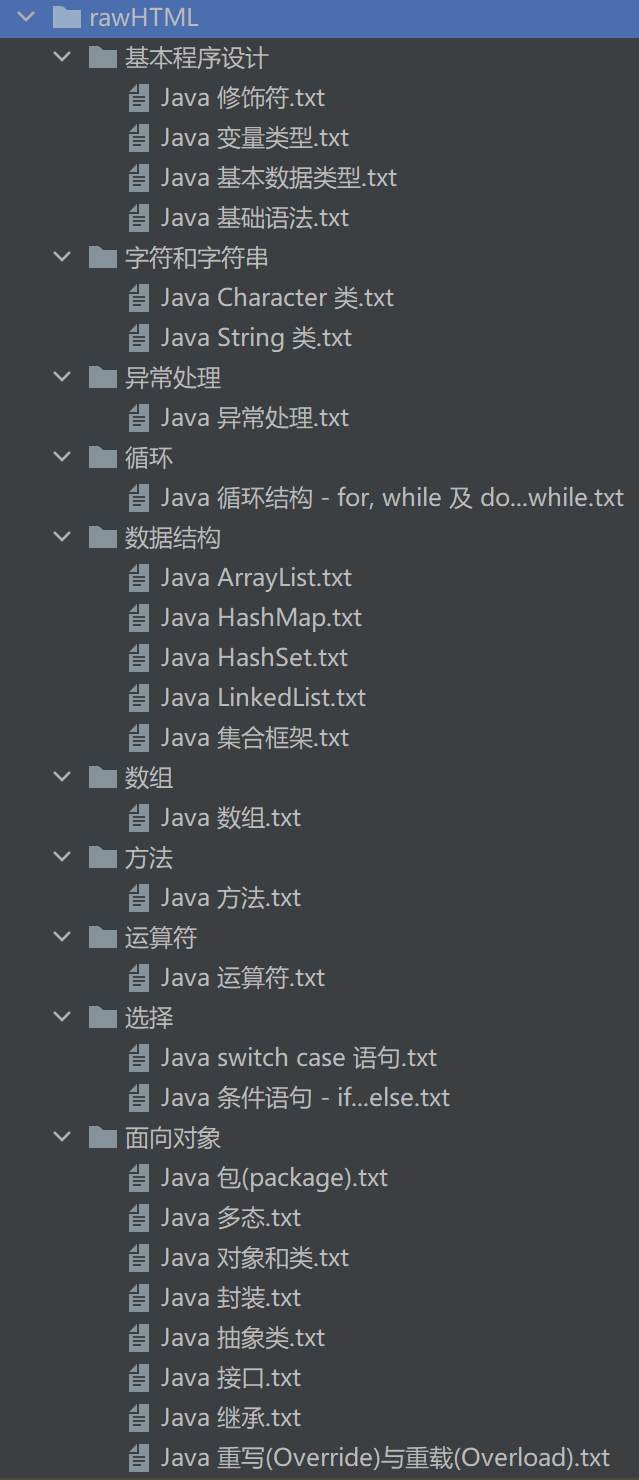
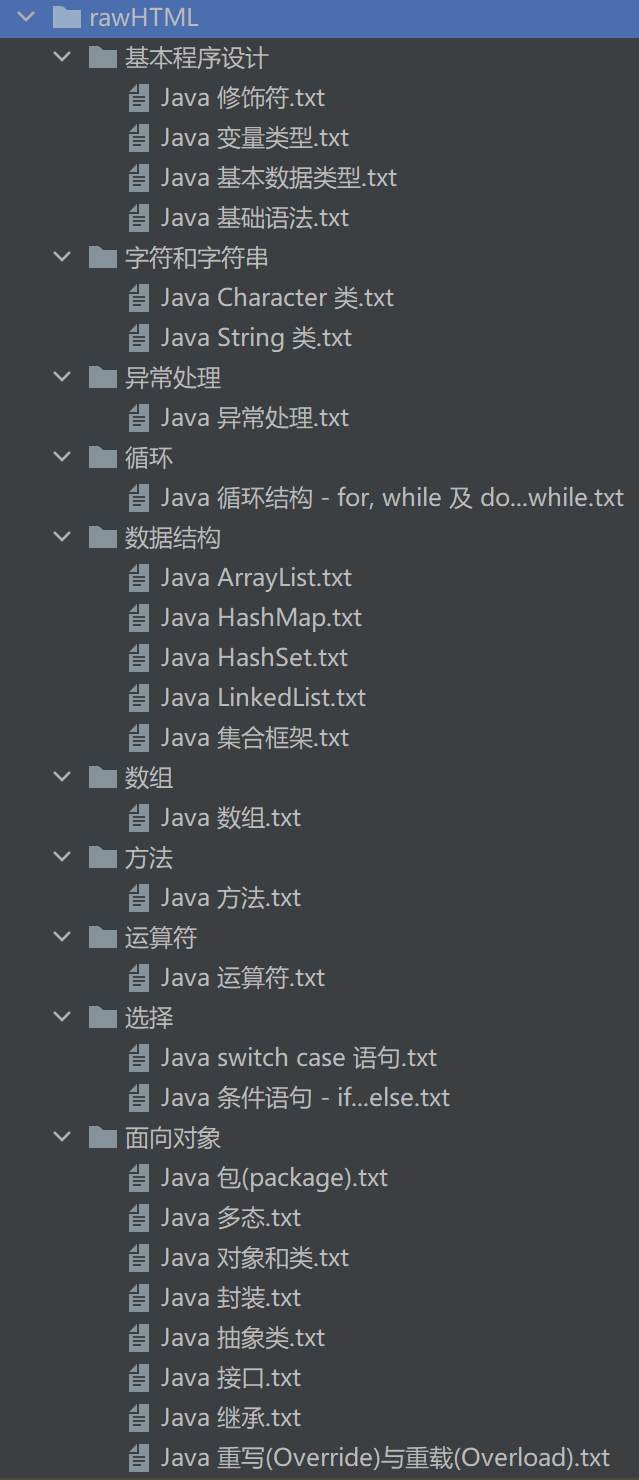
在使用Scrapy进行网站数据爬取时，首先指定目标域名（allowed\_domains）和起始爬取地址（start\_urls），然后定义parse函数指定对每个爬取到页面的处理规则。在parse函数中，每个页面都会被创建为一个item实例，包含name与content两个属性，属性的值通过XPath（XML Path Language）在网页结构中定位并调用extract()函数获取。每个item实例获取完毕后都会通过引擎交给管道，经过格式化处理后储存在本地以item.name命名的TXT格式文件中。同时，每个页面导航栏中的URL都会被制作成新的Scrapy请求并通过引擎提交给调度器再次入队，保证爬虫的不断运行，直到所需的数据全部爬取完成，调度器队列为空为止，最终获取到在线编程学习网站上Java程序设计课程概念详细信息的语料库。以菜鸟教程网站的Java教程为例，抓取完成的内容如下所示：

图4-2 爬虫抓取内容

1. **课程本体的详细设计**

如2.2节中提到的，课程本体一般使用与课程教学类似的三层结构进行设计，其中顶层为课程概念，第二层为章节概念，第三层为知识点概念，并且本文将多模态的教学资源也作为与知识点齐平的第三层概念纳入本体设计范围之内。具体的课程层次结构可以参考教材《Java语言程序设计（基础篇）》的目录结构，本文主要关注教材的前十三章，包含Java程序设计的基本编程概念与面向对象的基础知识。

本文设计的课程知识本体概念间的关系主要包括以下几种：

1. 包含关系和属于关系。包含关系存在于两个相邻层级的概念之间。章节实例属于课程实例，而知识点实例属于其对应的父章节实例。比如，知识点“Java继承”与知识点“Java多态”属于章节“面向对象”等。包含关系和属于关系是互逆的；
2. 前驱关系和后继关系。前驱关系存在于知识点之间。如前文所述，知识点之间存在学习顺序的依赖关系。例如，“if-else语句”作为程序流程的条件控制语句，需要通过判断if语句后跟随的布尔表达式的真假选择继续执行的语句，而布尔表达式则会包含“逻辑运算符”，因此“逻辑运算符”即是“if-else语句”的前驱知识点。后继关系和前驱关系是互逆的，因此相应地，“if-else语句”即是“逻辑运算符”的后继知识点；
3. 关联关系。关联关系存在于知识点与多模态教学资源之间。一个幻灯片教学资源往往包含多个主题，因此可能存在多个关联的知识点；一个视频教学资源往往具有一个比较明确的主题，但在教授过程中也可能涉及到别的知识点。比如，主题为“数组”的视频教程中涉及到数组遍历的内容，因此该视频与知识点“for循环”也相关联。关联关系是无序的双向关系。

由于包含关系与属于关系、前驱关系与后继关系之间是互相可逆的，因此本文为了让节点间的关系更为清晰，减少冗余，仅在课程知识概念图谱中存储包含关系、前驱关系与关联关系，并将关联关系根据教学资源类型细化为幻灯片关联关系与视频关联关系。

本文设计的课程知识本体概念的属性主要根据概念类型进行区分，具体如下：

1. 课程概念：包含唯一属性“名字”；
2. 章节概念：包含属性“名字”以及“优先级”；
3. 知识点概念：包含属性“名字”、“优先级”及“详细信息”。“优先级”通过知识点的前驱后续顺序计算得到，用于指定知识点在系统页面上展示的顺序；“详细信息”来自处理语料库后得到的数据，用作系统的部分知识点学习资料；
4. 代码示例概念：包含属性“详细信息”；
5. 教学资源概念：包含属性“URL”，用于指定真实教学资源在阿里云OSS平台的访问路径。

完成以上的课程知识本体详细设计后，就可以根据实际的课程知识语料库进行课程知识图谱的构建。

1. **课程概念的实例化**

本体是对实际知识图谱的一种抽象表示，因此为了获得实际课程知识点学习系统可用的底层课程知识图谱，需要以课程本体为模板，根据4.1.1节中获取的课程知识语料库完成课程概念的实例化。

本体概念实例化，或者说知识图谱的构建，一般包含实体抽取、关系抽取、知识存储等步骤。主流的知识图谱构建方式可以根据自动化程度进行分类，主要分为两种：一类知识图谱主要依靠专业人士手动构建。人工构建带来的是极高的准确度与完整性，但代价是规模偏小，因此更适合用来构建某一专业的领域类知识图谱；相应地，另一类知识图谱主要依靠机器学习、神经网络、大数据等现代人工智能技术从互联网数十亿的网页中抽取数据，进行自动化的知识图谱构建。这样的方式带来的是极大的知识与数据量，并可以不断迭代更新，但同样会导致知识驳杂，质量层次不齐，准确度也无法保证。通用类知识图谱由于其规模所限，一般只能采用这样的方式构建。因为本文构建的是课程领域类知识图谱，为了兼顾精确性与高效性，选择结合上述两种方法，采用半自动化的方式完成课程概念实体与属性的构建。

通过分析教材《Java语言程序设计（基础篇）》的目录结构，本文首先确定了各章节概念，并将Scrapy爬虫应用爬取到的网页内容分别对应到各个章节概念之下。同时，本文根据网页结构中的各级标题，初步确定每个章节下包含的部分知识点。

完成初步的章节与知识点概念确定后，本文选择使用TF-IDF算法确定每个抓取到的网页中所包含的知识点关键字。本文将所有抓取到的网页视为语料库，将每一个网页文本视为语料库中的一个文件，而将网页中潜在的知识点视为关键字。因此，对文件中每个词语计算TF-IDF后按得分逆序排列，排名越靠前的词语成为潜在知识点概念的概率也就越大。使用这种方法能够得到具有良好区分能力的关键字，并排除出现频率高但普遍性极强的词语，如“Java”。

由Scrapy应用爬取到的网页内容是包含HTML标签、文本内容、代码示例、运行结果等多种格式与信息的复合型超文本，因此为了将其转化为可用于统计计算的普通文本，本文选择使用基于Python的第三方库BeautifulSoup完成提前的数据清洗。BeautifulSoup允许程序员对HTML或XML（Extensible Markup Language）文件进行遍历，并针对指定标签作数据操作。通过分析语料库中的网页源码可以得知，该Java教程网站主要通过<h2>、<h3>标签完成页面的分段，主要的文本信息来源为<p>、<ul>下的所有<li>以及<table>标签，主要的代码示例存储于<div class=”example\_code”>中，而主要的运行结果展示在<pre>标签下，因此可以首先根据标签及class分类，将代码及运行结果与文本内容分离。代码及运行结果在后文中将被存储于与对应知识点节点关联的“代码示例”节点中。文本信息根据其所属标签不同，需要进行不同方式的处理，以下是处理<p>标签下文本内容的代码示例：

1. **def** getCorpus(html\_doc, stopwords):
2. soup = BeautifulSoup(html\_doc, 'html.parser')
3. result = []
4. **for** h2 **in** soup.find\_all('h2', class\_=False):
5. tag = h2.nextSibling
6. **while** tag **is** **not** None **and** tag.name != 'h2':
7. **if** tag.name == 'p':
8. text = tag.get\_text()
9. sentences = text.split('。')
10. **for** s **in** sentences:
11. words = list(jieba.cut(s))
12. words\_fnl = [i **for** i **in** words **if** i **not** **in** stopwords]
13. result.append(' '.join(words\_fnl))
14. tag = tag.nextSibling
15. **return** result

首先根据传入的HTML源文件创建bs4.BeautifulSoup对象，并指定解析器为“html.parser”。接着，使用BeautifulSoup.find\_all()方法遍历所有不具有class属性的<h2>标签，得到的h2为一个Tag对象。通过Tag.nextSibling属性可以不断迭代遍历<h2>标签后的兄弟节点，并通过Tag.name属性判断其标签为<p>。通过Tag.get\_text()方法得到标签中的所有文本，然后使用基于Python的第三方库jieba[[17]](#十七)完成中文分词。在使用jieba进行分词之前，可以提前通过调用jieba.load\_userdict()，将前文初步确定的章节与知识点概念传入，作为用户自定义的分词词表使用，提高分词的准确性。完成中文分词后，再根据停止词表去除对表意没有帮助的结果。停止词表主要用于去除以下三种词语：

1. 无实意的语气助词、连词等，如“吗”、“啊”、“而已”、“也”等；
2. 过于普遍导致没有统计、学习意义的词汇，如“我”、“的”等；
3. 标点符号，如“，”、“、”等。

由于TF-IDF算法的设计本就足以应对高频出现的无意义词语及标点符号，因此停止词表对最终结果的表现提高有限。但是考虑到后续需要使用相同语料库及类似的针对网页源码的处理方式获取文本信息以对fastText模型进行训练，因而此处仍将停止词去除纳入数据清洗的步骤中。针对<ul>下的<li>标签，本文将每个<li>标签等价为<p>标签进行处理；针对<table>标签，本文将表格的每行按[表头1 单元格1 表头2 单元格2 … 表头n 单元格n]的形式组织成句子，其余处理与<p>标签类似。

完成数据清洗的操作后，就可以使用TF-IDF算法计算每个文本文档的关键字，具体核心代码如下所示：

1. **def** tf\_idf(word\_list, file\_words\_map):
2. result\_map = {}
3. **for** word **in** word\_list:
4. **if** word **in** result\_map:
5. **continue**
6. tf = word\_list.count(word) / len(word\_list)
7. num = 1
8. **for** file\_words **in** file\_words\_map.values():
9. **if** word **in** file\_words:
10. num += 1
11. idf = math.log10(len(file\_words\_map) / num)
12. result\_map[word] = tf \* idf
13. **return** sorted(result\_map.items(), key=**lambda** x: x[1], reverse=True)

以上代码遍历每个文件中每个独特的词，然后按照公式计算该词对于该文件的TF-IDF指数，并按降序排列后返回。部分文本文件的前三名关键词统计结果如下表所示：

表4-1 部分TF-IDF统计结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 文件名 | 第一关键词 | TF-IDF指数 | 第二关键词 | TF-IDF指数 | 第三关键词 | TF-IDF指数 |
| 方法 | 参数 | 0.01876 | 构造方法 | 0.01289 | 返回值 | 0.01076 |
| 继承 | 父类 | 0.03261 | 继承 | 0.02461 | 构造器 | 0.02329 |
| 抽象类 | 抽象类 | 0.05348 | 抽象方法 | 0.04722 | 子类 | 0.02365 |
| switch | switch | 0.07546 | case | 0.05606 | break | 0. 05367 |

将以上得到的关键词经过一定的人工筛选后与初步确定的知识点合并，就得到了完整的知识点概念集合。接下来需要为每个知识点填充对应的详细信息与代码实例。本文仍选择使用BeautifulSoup定位知识点的相关内容。一方面，将知识点的上下文连带HTML标签一起保存在corpus.txt文件中，作为后续系统展示的知识点详细学习信息内容；另一方面，对每段代码实例，先判断其后文是否有相邻的兄弟<pre>标签作为其运行结果的展示，然后找到其上文中最靠近的知识点，将二者进行关联，存储于code.txt文件中。最终部分得到的结果如下图所示：

(a) 代码相关内容 (b) 详细学习信息内容

图4-3 为知识点填充的详细信息与代码实例展示

1. **课程知识点间关系的实例化**

完成课程概念的实例化后，需要确定知识点之间是否存在关系，以完成课程知识点间关系的实例化。本文定义A知识点作为B知识点的前驱的必要条件是A与B间存在关联。因此，首先需要找出所有相互之间存在关联的知识点对。

本文对知识点间存在关联的定义为：若知识点B的描述文本中提到知识点A包含的关键字，或者反之，则认为知识点A与知识点B间存在相互关联。知识点的关键字主要包括知识点本身的名字、知识点描述文本中的小标题，以及4.1.3节中通过TF-IDF算法找到但在人工筛选环节中没有被确定为知识点的关键字。比如，知识点“类型转换”下属的关键字有：“自动类型转换”、“强制类型转换”及“隐含强制类型转换”等。

确定知识点包含的关键字后，问题被转化为长文本的多关键字匹配。本文选择使用AC自动机（Aho-Corasick automaton）算法加以解决。AC自动机是著名的多模式匹配算法，算法中使用到了字典树（Trie）以及KMP算法（The Knuth-Morris-Pratt Algorithm）中失败（Fail）指针的思想。构建AC自动机时，首先将所有关键字建立为一棵字典树。字典树具有以下特点：

1. 每个非根节点都代表一个字符；
2. 每个非根节点都可以代表一个字符串，值为从根节点到该节点的路径上所有字符按序串联；
3. 除根节点外的每个节点都可以被标记为模式串串尾，代表该节点对应的字符串是一个关键字。

字典树保证具有相同前缀的字符串出现在同一子树中，也可以完成多模式匹配，但缺点是匹配一旦失败就必须从根节点重新开始，浪费了之前已匹配到的内容。AC自动机在构建的第二步纳入了类似KMP算法中的失败指针，即优化了上述缺点。失败指针通过BFS（Breadth First Search，广度优先搜索）构建，简而言之字典树中每个节点的失败指针都指向与该节点对应的字符串具有最长相同后缀的其它模式串对应的节点，如果不存在则指向根节点。完成AC自动机的构建后，就可以进行类似字典树的多模式匹配时，若匹配成功，则沿着失败指针遍历，将所有标记为关键字串尾的节点也记录为匹配成功；若匹配失败，不用从根节点开始重新匹配，而是利用已匹配到的内容，从失败指针处继续匹配。构建字典树与失败指针的时间复杂度为，关键字匹配为，因此算法的总时间复杂度为，其中M为被匹配长文本的总长度，N为关键字个数，L为关键字的平均长度。

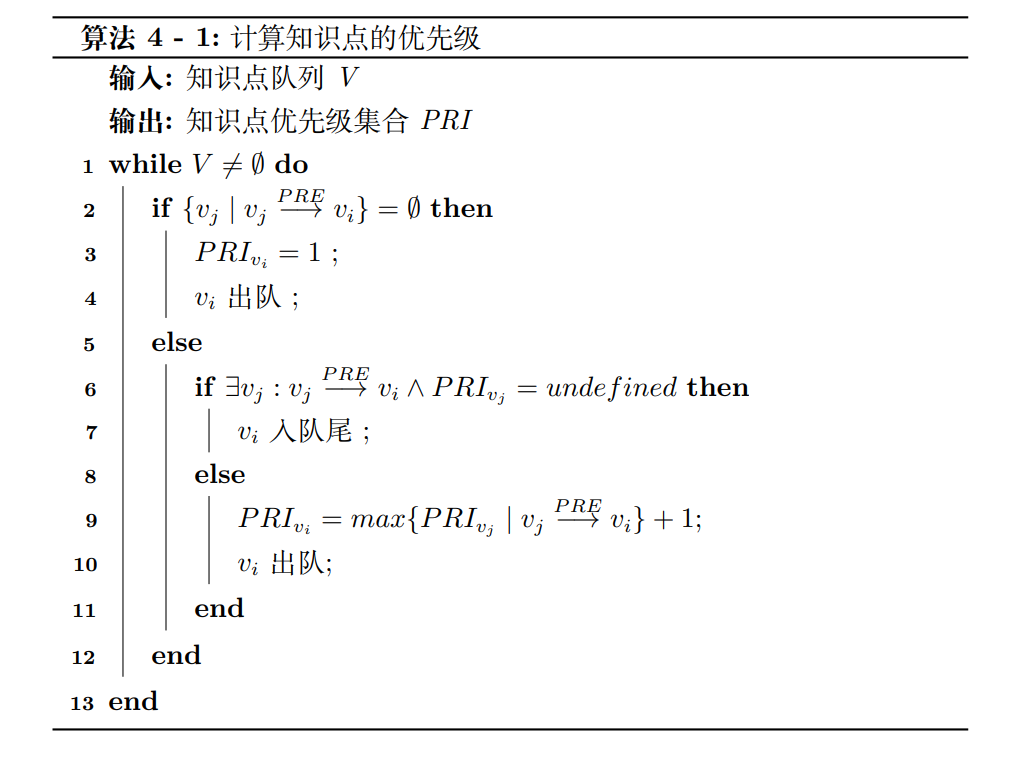
根据以上算法，本文将所有知识点包含的关键字视为模式串集，将每个知识点对应的详细文本信息视为被匹配的长文本，分别进行多模式串匹配。具体实现中，本文使用基于Python的第三方库pyahocorasick完成AC自动机的构建与匹配。通过调用ahocorasick.Automaton()创建AC自动机实例，并调用Automaton. add\_word()方法迭代传入模式串。完成AC自动机的构建后，核心的匹配代码如下所示：

1. **def** match\_keywords(ac, txt):
2. keywords = []
3. **for** i **in** ac.iter(txt):
4. keywords.append(i[1][1])
5. subwords = [subword **for** subword **in** keywords **for** word **in** keywords **if** subword **in** word **and** subword != word]
6. **return** [i **for** i **in** keywords **if** i **not** **in** subwords]

通过向Automaton.iter()方法传入长文本，能执行AC自动机的匹配算法，并得到所有出现的模式串。由于模式串间存在包含关系，比如模式串“非访问控制修饰符”包含模式串“访问控制修饰符”，因此存在“非访问控制修饰符”的文本中也必然会错误地匹配到“访问控制修饰符”，进而得到错误的关联关系，因此需要对得到的模式串进行数据清洗，将所有模式串与其子串同时被匹配到的情况中的子串去除。

完成多模式匹配后，就可以得到所有知识点间的关联对，进而通过人工筛选确定知识点间的前驱与后继关系，完成知识点间关系的实例化。最后，基于知识点间的前驱与后继关系，可以计算知识点的优先级。本文定义知识点的优先级计算公式如下：

其中，代表知识点n的优先级，代表知识点m为知识点n的前驱知识点。具体实现中，本文选择使用Python的Queue（队列）构建BFS，对所有知识点进行逐层遍历计算优先级。如果当前知识点没有前驱知识点，则优先级设为1；如果当前知识点的所有前驱知识点都已有优先级，则将当前知识点的优先级设为它们的最大值加1，并将当前知识点的所有后继知识点入队；否则，则先将未计算优先级的前驱知识点入队，然后将当前知识点再次入队，重新计算。

算法伪代码如下所示：

计算完成后，知识点在系统页面进行列表显示时即根据优先级升序排列；如果两知识点优先级相等，说明彼此间没有必然的先后学习顺序要求，按知识点名称排序即可。

1. **课程知识图谱的存储**

得到了完整的课程概念、包含的属性以及课程概念间的关系后，需将其存储至Neo4j图数据库中，形成课程知识点学习系统可访问并使用的底层数据库支撑。本文选择使用基于Python的第三方库py2neo完成对Neo4j图数据库的操作。py2neo封装了大部分Cypher语句，并可以执行程序员编写的原生Cypher语句。以下初始化py2neo、插入课程概念以及添加知识点前驱关系的代码示例：

1. **class** NeoUtil:
2. **def** \_\_init\_\_(self):
3. self.graph = Graph("http://localhost:7474", auth=("user", "pwd"))
5. **def** add\_lesson(self, name):
6. lesson = Node("Lesson", name=name)
7. self.graph.create(lesson)
9. **def** make\_successor(self, predecessor, successor):
10. sql = "MERGE (m:Knowledge{{name:'{0}'}}) MERGE (n:Knowledge{{name:'{1}'}}) MERGE(m)-[r:PRE{{name:'前驱'}}]->(n)".format(predecessor, successor)
11. self.graph.run(sql)

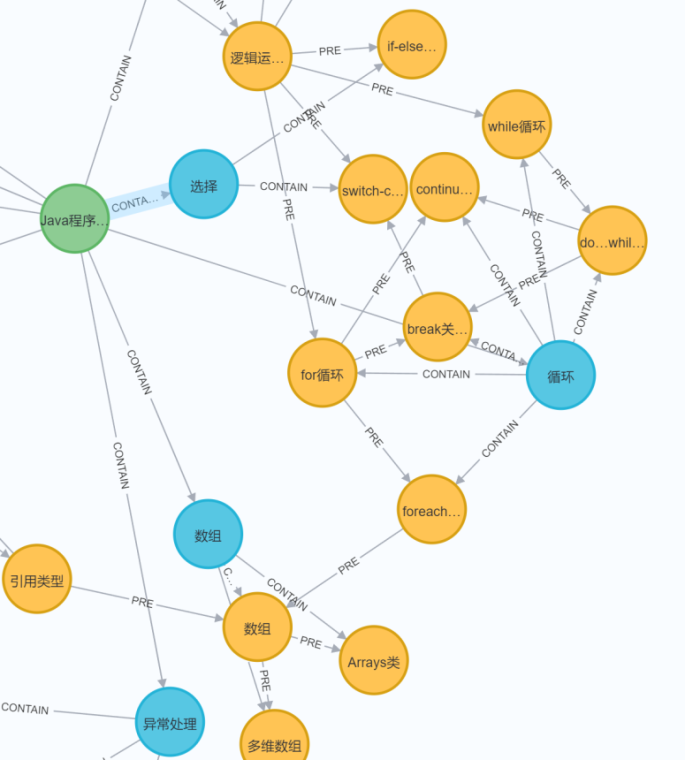
通过向py2neo.database.Graph()构造方法传入Neo4j图数据库连接地址、用户名及密码创建数据库对应的实例对象。节点的创建可以通过向Graph.create()方法传入py2neo.data.Node()对象完成，而原生的Cypher语句可以直接向Graph.run()方法传入来执行。在Cypher语句中使用到了关键字函数“MERGE”，其含义是先根据条件匹配节点/关系，如果匹配成功则直接返回，效果等同于

图4-4 存储于Neo4j中的部分课程知识图谱

“MATCH”，否则根据条件进行创建，效果等同于“CREATE”。最终将所有课程概念全部储存于Neo4j中，部分结果如图4-4所示。

图中，绿色节点代表课程概念，蓝色节点代表章节概念，黄色节点代表知识点概念。

1. **多模态教学资源链接知识点的实现**
2. **文本提取模块的实现**

多模态课程资源由于文件格式与表现形式不同，在进行统一的多分类并链接到知识点的任务之前，需先分别使用不同方式提取出其中的文本信息。本文中的课程知识点学习系统主要使用了两种不同格式的多模态教学资源：PPT资源以及视频资源。本节将介绍针对这两种资源采用的不同文本提取方式。

本文选择使用基于Python的第三方库python-pptx对PPT资源进行处理。相关示例代码如下：

1. **def** extract\_from\_ppt(path, stopwords):
2. ppt = Presentation(path)
3. result = []
4. **for** \_, slide **in** enumerate(ppt.slides):
5. **for** shape **in** slide.shapes:
6. **if** shape.has\_text\_frame:
7. **for** paragraph **in** shape.text\_frame.paragraphs:
8. words = list(jieba.cut(paragraph.text))
9. result.extend([i **for** i **in** words **if** i **not** **in** stopwords])
10. **return** result

首先通过向pptx.api.Presentation()构造方法传入要提取的PPT资源的本地路径得到PPT对应的Presentation实例对象，然后使用Presentation.slides属性遍历所有页（Slide）对象，使用Slides.shapes属性遍历每页中的所有文本框，使用SlideShapes.text\_frame属性遍历文本框中的文本片段，使用TextFrame.paragraphs遍历文本片段中的文本段落，使用\_Paragraph.text获取文本段落中的实际内容。得到文本内容后，使用类似4.1.3节中的方式使用jieba进行中文分词，去除停用词，然后返回结果。

针对视频资源的处理比较复杂，主要分为三步。首先，需要从视频资源中提取出对应的音频文件。本文选择使用基于Python的第三方库moviepy完成目标。通过向moviepy.editor.AudioFileClip()构造方法传入本地视频文件的路径得到AudioFileClip实例对象，并调用AudioFileClip.write\_audiofile()方法在本地写出对应的WAV（Waveform Audio File Format，波形声音文件）。接下来，本文选择调用百度智能云的短语音识别API完成语音文件到文本文件的转换。为了使输入的语音文件适配API，需要提前对语音文件的格式进行调整。分析可知，通过moviepy转换得到的WAV文件采样率为44.1KHz，而百度智能云的短语音识别API要求输入语音长度小于一分钟，且采样率为8KHz或16KHz，因此需要先调整音频文件的采样率，然后截取为多个长为一分钟的短语音。本文选择使用开源工具SoX（Sound eXchange），首先从官网下载SoX工具，然后在命令行使用以下命令实现采样率的转换：

1. forfiles /p .\wav /c "cmd /C for %I in (@file) do sox .\%~I -r 16000 -c 1 ..\wav2\%~I"

在Windows系统下的CMD（Command Prompt，命令提示符）中，使用命令forfiles遍历使用/p参数指定的文件夹下的所有文件，使用/c参数指定对每个文件执行的命令。针对每个文件的命令中，通过%I获取每个文件@file的文件名，并调用sox命令，通过-r参数指定采样率，-c参数指定声道数，将对应的16KHz WAV文件输出至新文件夹下。

接着，本文选择使用基于Python的第三方库pydub完成音频文件的切割。首先通过向pydub.audio\_segement.AudioSegment.from\_file()方法传入路径及音频文件类型创建AudioSegment对象，将其传入pydub.utils.make\_chunks()方法并指定切割完成后每段音频的长度（这里是59000毫秒，即59秒）完成音频的切割。接着，遍历切割完成得到的AudioSegment对象列表chunks，对其中的每一个片段调用AudioSegment.export()方法完成音频的导出。

得到格式化好的音频片段后就可以调用百度智能云的短语音识别API进行音频文件到文本文件的转换。具体实现时，借助百度提供的Python第三方库baidu-aip，通过从百度智能云中获取到的应用鉴权信息构建aip.speech.AipSpeech对象，并调用AipSpeech.asr()方法，传入二进制形式的本地音频文件片段，并指定识别模型为“普通话输入法模型”，就能得到对应的文本文件。将对应同一完整音频文件的文本内容按序拼接，就能最终完成视频资源的文本提取。

1. **fastText模型的训练**

本文选择使用fastText多分类模型完成多模态教学资源对知识点的多分类，而本文使用的程序设计课程语料库具有较强的领域性，需要提前对使用的fastText模型进行有针对性的训练，而不能直接使用官网给出的具有普适性的中文分类模型。

fastText的多分类模型需要使用有监督学习，即模型的输入训练集要求被正确地标注。根据fastText官网给出的训练集样例分析可知，fastText有监督学习的训练集的组织结构以行为单位，每行为训练集语料库中的一个句子，句子要求以被分词完成的形式呈现；每行的头部有一个或多个以“\_\_label\_\_”开头的标签标识，表示训练集中的这句话对应的正确标签。本文使用的语料库与对文本的处理方式类似4.1.3节，即选择并提取对每个知识点进行描述的详细文本。这一语料库集合完全以程序设计课程为中心，具有很强的针对性，并且每一句句子都被包含于一个确定的知识点下，不会出现语料没有任何分类的情况，非常符合本文对fastText有监督学习训练集的需要。为了实现多分类标注，对语料库的每句句

子除了标注其所属的知识点外，还需要确定其它相关的知识点。本文根据主观经验，采用人工标注的方式为每句训练集中的句子确定额外的对应知识点，最后得到了超过两千条训练集。部分fastText有监督学习语料集如下图所示：

图4-5 部分fastText训练集

得到完整的训练资料后，需要将其划分为训练集与验证集，以检验最终模型的训练情况，防止过拟合的现象发生。过拟合指模型的分类结果在训练集上能得到令人满意的表现，但在训练集之外，真实需要应用模型的数据集上，却不能很好地对数据进行拟合。过拟合现象一般源于训练参数选择的不当，导致模型将训练集中的噪声也学习为了分类特征，而使用预先划分开的验证集测试模型的预测结果能很好地觉察并预防模型的过拟合问题。本文将完整的训练资料随机划分30%作为验证集，并选择使用基于Python的第三方库fasttext完成模型的训练、测试与数据分类，具体代码示例如下所示：

1. **def** classifier\_train(path\_train, path\_test, lr, epoch, n, threshold):
2. classifier = fasttext.train\_supervised(path\_train, lr=lr, epoch=epoch, wordNgrams=n)
3. **print**(classifier.test(path\_test, k=-1, threshold=threshold))
4. **return** classifier

通过fasttext.train\_supervised()方法训练有监督学习模型，传入训练集路径及三个学习参数：lr（Learning Rate，学习率），epoch（训练轮数）以及wordNgrams（n-gram特征的n取值）；参数k表示在最后的结果中取概率最高的k个作为输出的多分类标签，取-1则代表输出所有结果，参数threshold代表仅将最终结果中概率大于threshold的项进行输出。下表中展示了部分不同学习参数取值在验证集上的预测结果：

表4-2 不同学习参数的模型效果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学习率 | 轮数 | wordNgrams | 阈值 | 精确率(%) | 召回率(%) | F1分数 |
| 0.05 | 5 | 2 | 0 | 3.45 | 100 | 0.0667 |
| 0.5 | 5 | 2 | 0.1 | 55.87 | 18.32 | 0.2759 |
| 1.0 | 5 | 2 | 0.1 | 67.03 | 44.69 | 0.5363 |
| 1.0 | 25 | 2 | 0.1 | 91.75 | 83.52 | 0.8744 |
| 1.0 | 50 | 2 | 0.1 | 92.72 | 86.26 | 0.8937 |
| 1.2 | 50 | 2 | 0.1 | 92.36 | 86.44 | 0.893 |
| 1.0 | 75 | 2 | 0.1 | 92.86 | 85.71 | 0.8914 |
| 1.0 | 50 | 3 | 0.1 | 92.69 | 86.26 | 0.8935 |
| 1.0 | 50 | 2 | 0.15 | 95.60 | 79.67 | 0.8691 |

本文采用三种不同指标来评价训练得到的模型。在一次二分类模型的分类结果中，总共有四种可能的情况，而多分类任务可以被视为多个二分类任务，每个二分类任为“属于本类”与“不属于本类”。四种情况如下所示：

1. TP（True Positive，真阳性）：正确的分类，即一个目标应该属于A类，且被模型分在A类；
2. FN（False Negative，假阴性）：遗漏的分类，即一个目标应该属于A类，但模型没有将其分在A类；
3. FP（False Positive，假阳性）：错误的分类，即一个目标不应该属于A类，但模型将其分在A类；
4. TN（True Negative，真阴性）：正确的结果，即一个目标不属于A类，且模型为将其分在A类。

使用以上四种情况在总结果数中占据的比例，可以分别计算三种模型评价指标。精确率（Precision）的计算为：

精确率代表得出的结果中正确的标签占总预测出标签的比例。越高的精确率代表模型给出的预测结果越不容易出错，而因此带来的代价可能是某些分类标签的遗漏。召回率（Recall）的计算为：

召回率代表得出的结果中正确的标签占总正确标签的比例。越高的召回率代表模型给出的预测结果越能包含每个应被分类的标签，而因此带来的代价可能是更多的错误分类出现。

分析以上可知，精确率和召回率是相互影响，相互制约的，两者有时是矛盾的。因此，F分数作为两者的综合指标被引入，其计算为：

其中，作为衡量精确率与召回率之间重要性高低的参数，当取1时，F分数则变为特例F1分数（F1 Score），作为二者的调和平均值，代表它们同等重要。因此，F1分数的具体计算为：

明确了三种评价指标的计算及含义后，就可以相应地对表4-2中不同训练参数得出的结果进行分析。第一行对应的参数是fastText模型训练的默认参数，阈值取0代表模型会将每一个目标都打上所有标签，模型因此一定能预测出所有的正确标签，召回率高达100%，代价则是极低的精确率（3.45%）。将阈值设为0.1后，随着学习率从0.05到1.0，轮数从5到50的不断增加，模型的整体性能不断提升。而将学习率从1.0提升到1.2，轮数从50提升到75，n-gram的n值从2提升到3，模型的性能几乎已经没有提升，甚至在某些指标还有略微的降低，因此确定最终的学习率为1.0，轮数为50，n-gram为bigram。而在阈值的取值方面，当阈值从0.1提升到0.15时，精确率有明显提升，但代价是召回率有更大幅度的下降，这是因为更高的阈值在筛选掉一部分错误标签的同时会漏选更多的正确标签，因此经过权衡最终将阈值定为0.1。在确定了训练参数后，就可以使用完整的训练材料训练模型并保存，为后文中的资源分类模块提供基础。

1. **资源分类与资源存储模块的实现**

完成了多模态资源文本信息的提取与fastText模型的训练后，就可以利用模型对文本信息进行多分类任务，并最终为多模态资源打上知识点的标签。使用第三方库fasttext进行分类预测的示例代码如下所示：

1. **def** classifier\_predict(classifier, content):
2. **return** classifier.predict(content, k=-1, threshold=0.1)

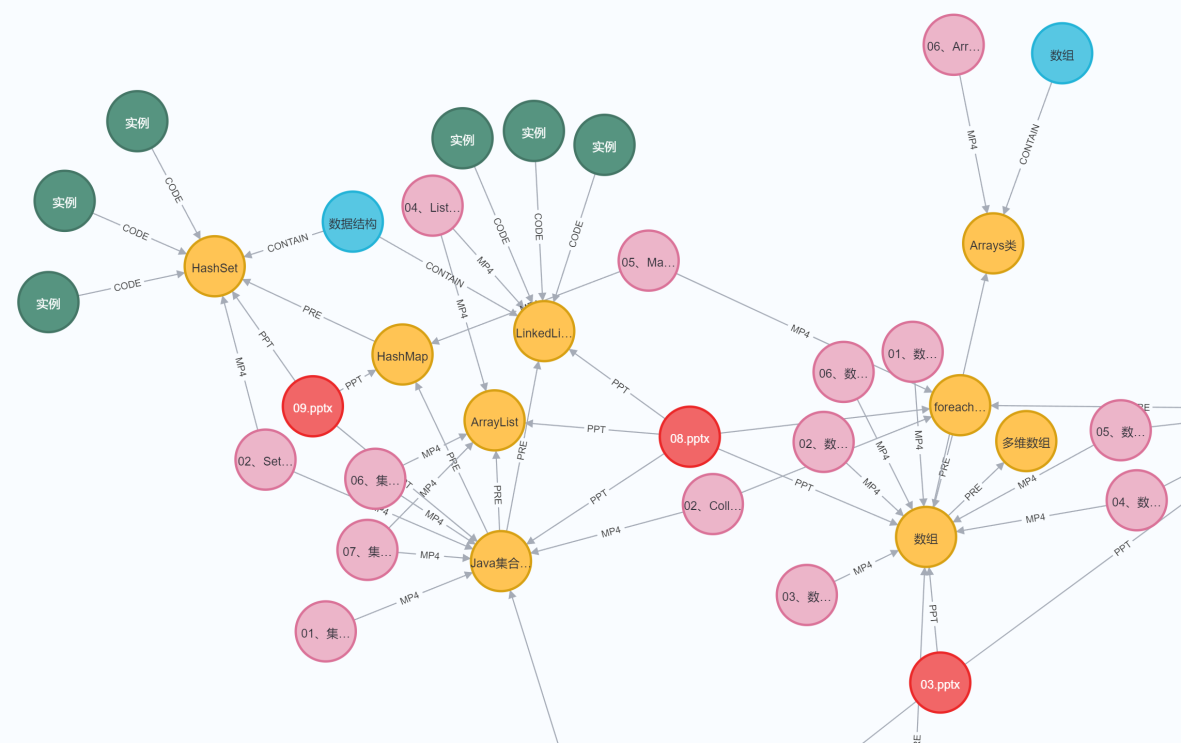
调用训练好的模型的predict()方法，对传入的content进行分类预测。参数k与threshold的取值源自上一节中的试验与分析。完成了资源对知识点的多分类后，需要将资源存储到OSS平台，便于前端系统的访问与展示。OSS提供RESTful 的接口，让用户可以随时随地通过存储空间与文件名访问任意数据。将所有教学资源都存储到OSS平台并获取到URL后，就可以类比4.1.5节，调用py2neo完成知识图谱对教学资源节点的存储与属性的设置。Neo4j中部分存储结果如图4-6所示。

图4-6 部分包含多模态教学资源的课程知识图谱

图中，蓝色节点为章节概念，黄色节点为知识点概念，粉色节点为视频教学资源，红色节点为PPT教学资源，深绿色节点为知识点相关代码示例。为了保证图的美观与清晰，没有展示所有关联的节点。

1. **系统总体功能的实现**
2. **后端功能的配置与实现**

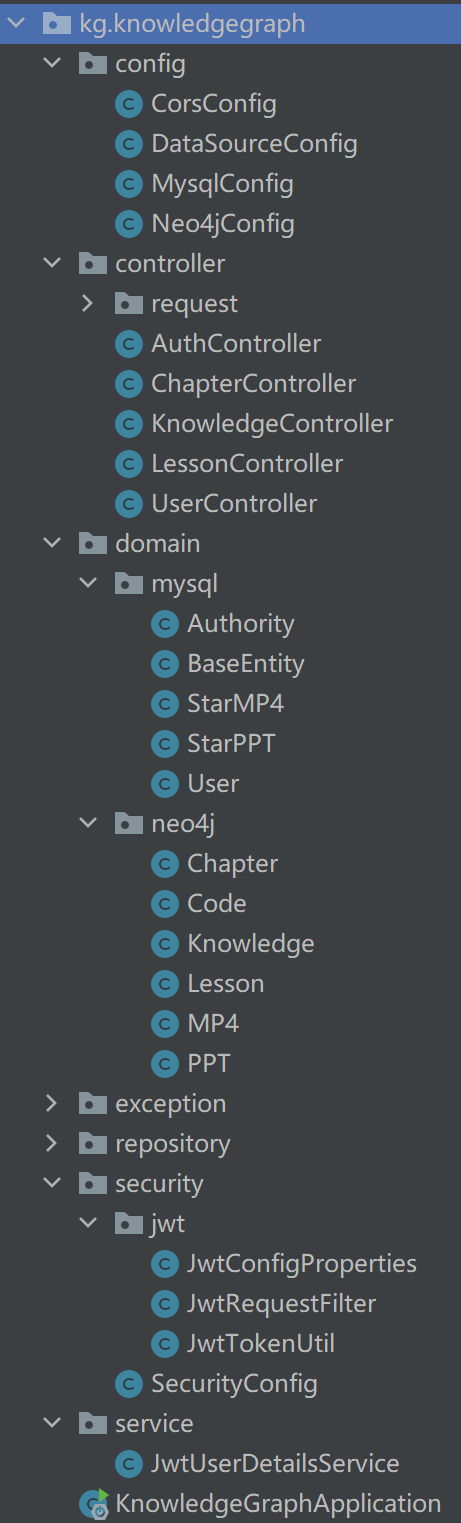
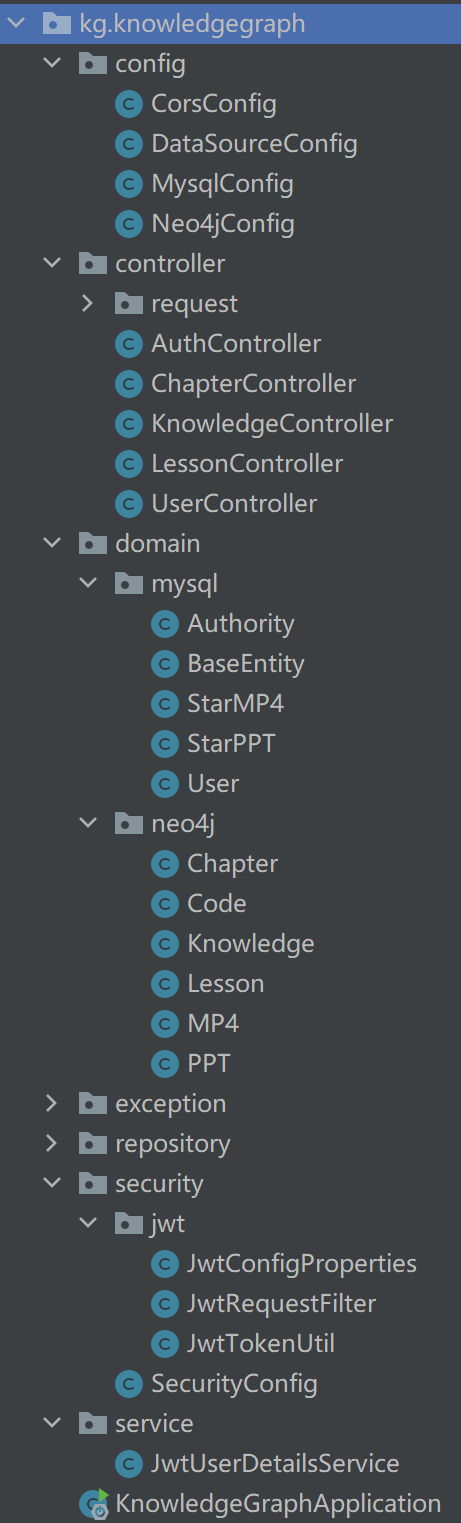
后端的整体项目结构如下图所示：

图4-7 后端项目结构

后端的功能主要分为两大模块，即用户相关功能及知识图谱相关功能。用户相关模块主要需要完成用户的鉴权，本文选择使用基于Spring Security的JWT（JSON Web Token）验证用户的登录态。具体而言，后端收到前端的登录请求并校验成功后，将用户信息与本JWT的过期时间（设置为5个小时）通过加密编码后制作成一个JWT，存储于响应头中返回给前端，前端则将这个JWT保存在浏览器中。当前端访问要求校验的页面及接口时，前端将保存的JWT放入请求头的Authorization属性中，后端通过自定义的JwtRequestFilter类对请求进行验证，如JWT格式是否正确，是否过期等。验证通过后，将用户的信息放入SecurityContextHolder类的context属性中，便于后端其它的Controller进行访问。在数据库中存储新注册用户的信息时，使用BCrypt对用户的密码进行加密，防止数据泄漏带来的隐患。

知识图谱相关模块主要完成的是对各课程概念及教学资源的查询。具体实现时，基于Neo4j为Spring Boot提供的工具类Spring Data Neo4j，首先根据图谱内节点的属性与相互关系结构构建对应的实体类。例如，知识点节点对应的实体类Knowledge，具有知识点名name、优先级priority等属性，同时因为具备前驱与后继关系，因此有以下代表节点间关系的属性。

1. @Relationship(type = "PRE", direction = Relationship.INCOMING)
2. **private** Set<Knowledge> predecessors = **new** HashSet<>();
4. @JsonIgnore
5. @Relationship(type = "PRE", direction = Relationship.OUTCOMING)
6. **private** Set<Knowledge> successors = **new** HashSet<>();

使用@Relationship注解标识关系，注解中的type标识关系名，direction标识关系的方向，且因为前驱与后继关系为多对多，关系对应的属性以集合Set类型存储。其余的关系与之类似。构建完实体类的属性，并提供相应的getter/setter后，还需要构建实体类对应的DAO类，即仓库Repository，并依照仓库接口方法的命名规则创建需要的查询方法。最后，根据逻辑需要，在Controller类中的方法中调用仓库接口，并将结果返回给前端。同时，方法还需要通过注解配置接收的参数、请求的方法与路径。后端实现的一大难点是在一个Spring Boot项目中同时连接MySQL与Neo4j两种数据库。为此，需要对两个数据库分别进行配置。以下是对Neo4j数据库进行配置的代码示例，MySQL的配置类似同理。

1. @Configuration
2. @EnableTransactionManagement
3. **public** **class** Neo4jConfig {
4. @Value("${spring.neo4j.uri}")
5. **private** String uri;
6. @Value("${spring.neo4j.authentication.username}")
7. **private** String username;
8. @Value("${spring.neo4j.authentication.password}")
9. **private** String password;
10. @Bean("configurationNeo4j")
11. **public** org.neo4j.ogm.config.Configuration configuration() {
12. **return** **new** org.neo4j.ogm.config.Configuration.Builder().
13. uri(uri).credentials(username, password).
14. withBasePackages("kg.knowledgegraph.repository.neo4j").
15. build();
16. }
17. @Bean
18. **public** SessionFactory sFactory(org.neo4j.ogm.config.Configuration c) {
19. **return** **new** SessionFactory(c, "kg.knowledgegraph.domain.neo4j");
20. }
21. @Bean("transactionManager")
22. **public** Neo4jTransactionManager neo4jTransactionM (SessionFactory s) {
23. **return** **new** Neo4jTransactionManager(s);
24. }
25. }

首先，在配置类定义时用注解@EnableTransactionManagement开启数据库的事务管理功能。在配置类中，用注解@Value获取定义于application.properties中数据库的连接地址及鉴权等信息。接着，在配置类中分别创建三个Bean对象：

1. 数据库公司提供的Configuration配置类，主要用以配置数据库的地址与鉴权信息，并通过withBasePackages()方法自定义本项目中该数据库对应的DAO层包位置；
2. SessionFactory类，主要用于自定义本项目中该数据库对应的Domain层包位置；
3. 数据库对应的TransactionManager事务管理类，用于开启数据库的事务管理功能。如果缺少这一配置项，则在通过DAO层调用预定义的CRUD接口方法时会出现错误。

本文在Domain层与DAO层将分属于两个数据库的类以不同的包进行区分，如图4-7中所示，kg.knowledgegraph.domain包含所有实体类，其中的mysql包中包含存储于MySQL关系型数据库中的实体类，如用户User，用户的收藏信息StarMP4与StarPPT等；neo4j包中包含存储于Neo4j图数据库中的实体类，如章节Chapter，知识点Knowledge等。DAO层对应的包repository也是同理。将包路径分别填入两数据库的配置类中，启动Spring Boot服务时框架就会正确地将实体类分类映射到两个数据库中，并调用底层不同的SQL/Cypher语句完成CRUD的操作，将两数据库的操作互相隔离，不会相互影响，而实现其它业务逻辑时，只需要像单数据库应用一样调用DAO层进行操作即可。此外，本文还配置了CorsConfig以允许前端请求的跨域。

1. **前端界面与交互的实现**

前端是一个基于Vue.js框架的单页应用。在用户登录并正式进入学习系统后，整个页面的页眉、页脚与页面左侧的导航栏不会因为界面的切换而刷新，只有页面中央的内容区会因为导航项的不同而不断变化。单页应用主要通过Vue Router库实现，页面中需要根据导航切换显示内容的位置由标签<router-view />代替，然后由导航栏配置具体点击后跳转的路径；另一方面，在router/index.js下配置组件与路径的关联，就可以完成组件间的跳转。同时，在router/index.js中还可以使用router.beforeEach()配置前置路由守卫，拦截未登录用户进入任何被配置为需要登录态的路由，同时拦截已登录用户进入登录/注册界面。

前端系统使用Vuex完成全局数据交互。系统将被多个视图共享的数据存储于Vuex.Store的State部分，可以被所有组件访问到。当有组件需要对数据进行修改时，组件调用对应的Action方法，Action再触发相应的Mutation，由Mutation最终修改State。Action与Mutation的区别主要在于：Action中可能存在复杂的业务逻辑，如向后端发送异步请求等，不能直接修改State；Mutation一般逻辑简单，同步执行，只专注于修改State。

前端使用Axios库完成前后端的请求交互。Axios是一个基于Promise的HTTP请求库，在设置请求方法、请求路径与参数后，可以设置两个回调函数，分别对应请求成功与失败的情况。后端传来的响应或错误信息以参数的形式传入回调函数，供前端使用。

本文使用D3.js搭配Vue.js完成课程知识图谱的可视化，主要依靠D3.js提供的力导向图进行实现。图谱的部分初始化JavaScript代码如下所示：

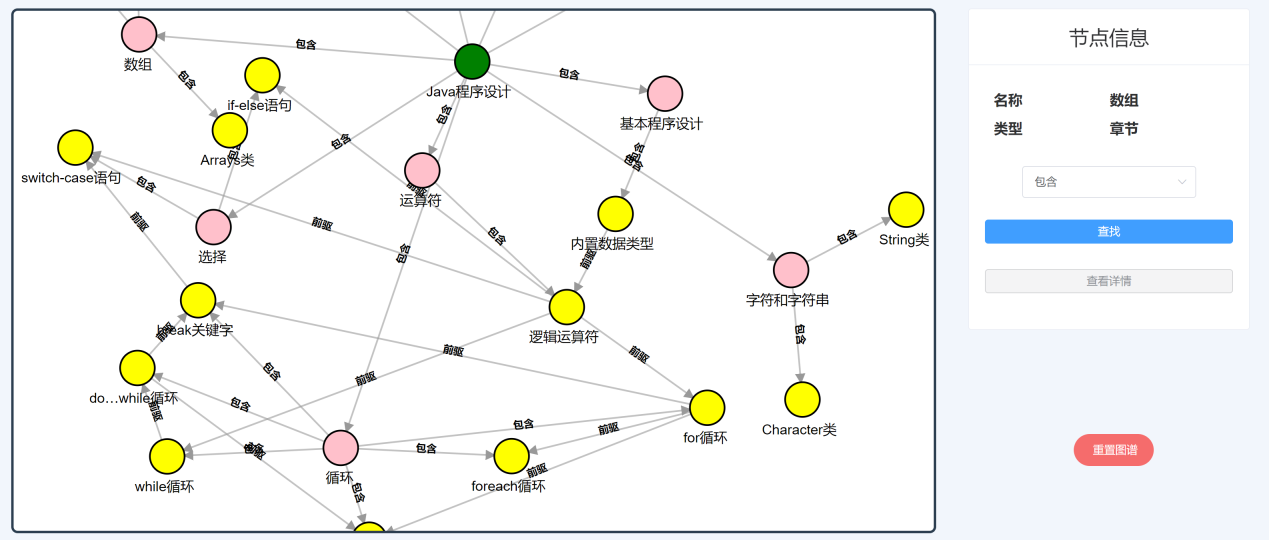
1. **function** initGraph () {
2. **const** links = data.links.map(d => Object.create(d))
3. **const** nodes = data.nodes.map(d => Object.create(d))
4. **this**.simulation = d3.forceSimulation(nodes)
5. .force('link', d3.forceLink(links).id(d => d.id).distance(50))
6. .force('collide', d3.forceCollide().radius(50))
7. .force('charge', d3.forceManyBody().strength(-50))
8. .force('center', d3.forceCenter(width / 2, height / 2))
9. **const** svg = d3.select('svg').attr('viewBox', [0, 0, width, height])
10. **const** g = svg.append('g')
11. svg.call(d3.zoom().on('zoom', **function** () {
12. g.attr('transform', d3.event.transform)
13. }))
14. }

links与nodes是需要绘制的知识图谱的底层数据，nodes是节点信息，而links是节点间的关系信息，这些信息在进入图谱展示界面时由系统使用Axios向后端请求得来。simulation是D3.js提供的力导向图模拟，提供了一些节点与边的初始化配置定义。forceSimulation初始化时传入了全部的节点信息；forceLink传入了全部的边信息，并指定作为边端点的节点的标识符（本文中是id）及边的初始长度；forceCollide指定了节点的碰撞体积，本文设定与节点的半径一致，防止节点的重叠；forceManyBody指定节点间的相互作用力，负值代表节点间自然存在斥力，斥力利于整体图像结构的平均展示，使图谱对用户更清晰地呈现；forceCenter则指定了力图的重心位置。svg与g标签由D3.js进行动态添加，作为知识图谱展示的底层画布。最后，通过设置g标签的transform属性，使其能够响应用户的缩放指令。在画布上展示的节点、边、文字、箭头等都通过append()方法添加到g标签上，并通过attr()方法设置相应的属性，如节点的半径、颜色、拖拽与点击的响应事件等。当用户查询与节点存在关系的其他节点时，系统先向后端发送请求，收到数据后将新的节点与边填入增加到nodes与lists属性中，而这两个属性已被设置为Vue.js框架的监听属性，Vue.js检测到监听属性的变化后就会调用设定好的handler函数进行处理，对新增部分重新进行渲染与绘制。

由于节点间存在斥力，且用户可能拖动节点，因此整个画布需要根据每一帧每个节点新的位置重新渲染。监听每帧变化的示例代码如下所示：

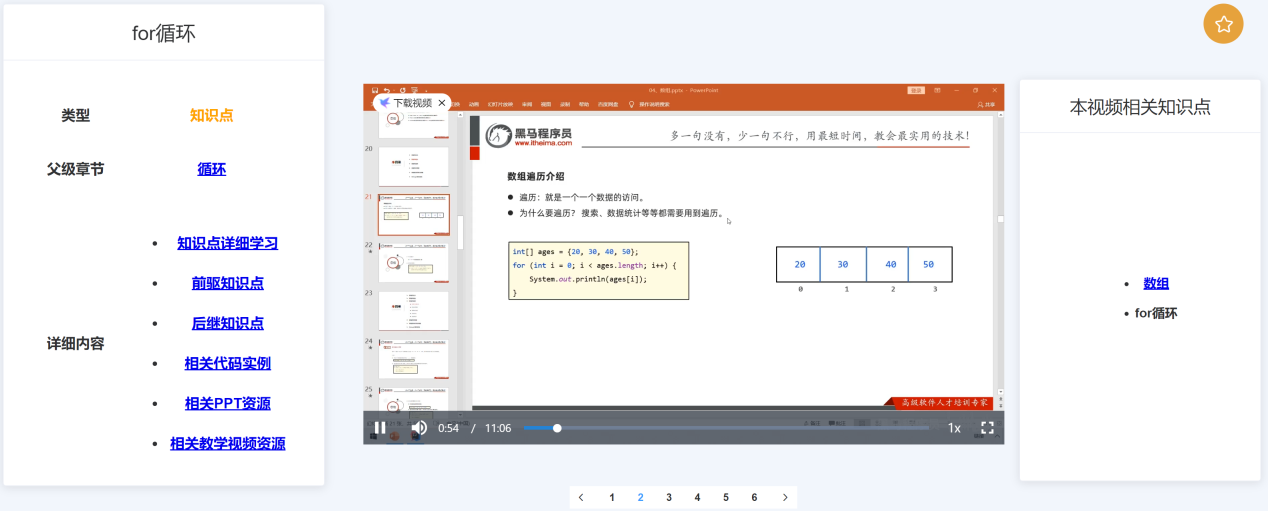
1. **function** onTick () {
2. **this**.simulation.on('tick', () => {
3. **this**.nodeLinks.attr('d', **function** (d) {
4. **if** (d.source.x < d.target.x) **return** 'M ' + d.source.x + ' ' + d.source.y + ' L ' + d.target.x + ' ' + d.target.y
5. **else** **return** 'M ' + d.target.x + ' ' + d.target.y + ' L ' + d.source.x + ' ' + d.source.y
6. })
7. .attr('marker-end', **function** (d) {
8. **if** (d.source.x < d.target.x) **return** 'url(#positiveMarker)'
9. **else** **return** **null**
10. })
11. .attr('marker-start', **function** (d) {
12. **if** (d.source.x < d.target.x) **return** **null**
13. **else** **return** 'url(#negativeMarker)'
14. })
15. **this**.nodeList.attr('cx', d => d.x).attr('cy', d => d.y)
16. **this**.nodeNameTexts.attr('x', d => d.x).attr('y', d => d.y)
17. })
18. }

其中，节点列表nodeList与节点名字nodeNameTexts都只需要直接根据节点位置修改相应画布上的图像的位置参数即可；边列表nodeLinks的修改比较复杂，因为边上同样需要展示名字，而边的其中一个端点在绕另一个端点旋转一周的过程内，其中的半圈时间该边的名字会被倒置，影响展示效果与用户体验。为此，本文通过比较两端点x坐标的大小判断二者的位置信息，然后在本来文字倒置的半圈中将边的首尾关系互换，即将原来的改为，以解决上述问题。但这样会引入一个新的问题：由于设计中边上会显示箭头以标识关系的方向，首尾关系的互换会导致箭头方向与预期相反。因此，本文将箭头的设计修改为在边的两端都预先绘制，然后根据当前边的方向选择显示正向箭头（positiveMarker）还是反向箭头（negativeMarker），最终解决在移动过程中各部分的方向问题。系统呈现的课程知识图谱可视化如图4-8所示。

图4-8 前端课程知识图谱可视化

除此之外，前端还需要完成多模态教学资源的展示。对于PPT教学资源，本文使用Microsoft为网页应用提供的在线预览PPT的功能组件，使用<iframe>标签，在其中的src属性填入"http://view.officeapps.live.com/op/view.aspx?src="，并附上与需要浏览的PPT对应的公网可访问的URL（来自阿里云OSS平台），即可实现PPT的在线预览，并同时附带资源的下载等功能。

对于视频教学资源，本文选择使用Vue.js框架的扩展库VideoPlayer完成实现。VideoPlayer库提供新的标签<video-player>，具有options属性，可以在其中配置访问视频的URL、视频可选播放速度、视频展示比例、是否自动播放与循环播放等属性。视频资源的在线预览界面如图4-9所示。

图4-9 前端视频教学资源预览

1. **系统的部署**

本文以数据库-后端-前端的顺序部署整个课程知识点学习系统，总体而言是沿系统架构自底向上，确保高层应用在部署时已有可用的底层基础。

1. **数据库的部署**

本文分别采用Docker部署两种数据库。部署MySQL数据库的命令如下所示：

1. docker pull mysql
2. docker run -p 3306:3306 --name m-grad -e MYSQL\_ROOT\_PASSWORD=java -d mysql

首先从仓库拉取mysql对应镜像，不指定版本则默认为latest版本。由于MySQL数据库不需要进行额外配置，也不需要提前插入数据，因此可以直接运行容器。命令参数中，-p指定容器内到服务器主机的端口映射， --name与-e分别指定容器名和MySQL root的密码，-d令容器在后台以守护进程形式保持运行，最后的mysql则指定容器创建时基于的镜像。Spring Boot框架在初次运行时，会自动为属于MySQL的实体类创建对于数据表，并按其中属性创建对应的表字段，因此只需要将数据库相关信息填入Spring Boot框架的配置信息即完成MySQL的数据库配置。

部署Neo4j图数据库时，除了需要在服务器上运行数据库，还需要将在本地创建好的课程知识图谱信息迁移到服务器。具体的命令如下所示：

1. neo4j-admin dump --database "java" --to "D:\java.db"
2. scp D:\java.db root@139.224.193.157:/home/neo4j/import
3. docker pull neo4j
4. docker run -d --name neo4j-grad \
5. -p 7474:7474 -p 7687:7687 \
6. -v /home/neo4j/data:/data \
7. -v /home/neo4j/logs:/logs \
8. -v /home/neo4j/conf:/var/lib/neo4j/conf
9. -v /home/neo4j/import:/var/lib/neo4j/import \
10. --env NEO4J\_AUTH=neo4j/java \
11. neo4j
12. docker exec -it neo4j-grad /bin/bash
13. bin/neo4j-admin load --from=./import/java.db --database=graph.db --force

首先，在本地主机执行前两行命令：通过neo4j-admin dump命令将一个图数据库中所有信息（节点、关系及属性）导出，并使用scp（Secure Copy）命令在主机与服务器间完成数据库导出文件的传递。接着，在服务器端类似地拉取neo4j对应镜像，然后运行容器。运行命令中有一些与运行MySQL容器时不同的参数：有两个-p是因为Neo4j数据库一般向外暴露两个不同端口，以提供两种不同的连接方式；其余的-v则是在运行主机与容器的目录间建立联系，分别将容器内的数据、日志、配置与导入目录挂载在了主机对应目录下；最后--env则是鉴权信息。容器运行完成后，使用docker exec –it 容器名 /bin/bash命令开启一个运行时容器内的命令行终端，为了执行最后一行命令，即通过neo4j-admin load将在本地导出的数据库信息载入进在服务器运行的Neo4j中。

接着，还需要对服务器Neo4j数据库的配置文件neo4j.conf进行修改，具体如下所示：

1. dbms.connectors.default\_listen\_address=0.0.0.0
2. dbms.connector.bolt.listen\_address=0.0.0.0:7687
3. dbms.connector.http.listen\_address=0.0.0.0:7474
4. dbms.active\_database=graph.db

前三行命令将连接器的默认监听IP地址改为0.0.0.0，即允许任意IP地址连接数据库；最后一行声明默认打开的数据库，这与之前导入数据的数据库是相同的。完成配置文件的修改后，就可以在Spring Boot框架填入对应的数据库信息，完成Neo4j数据库的配置与连接。

1. **后端应用的部署**

后端整体上是一个使用Maven进行包管理的Java Spring Boot应用，因此本文在打包后端系统时使用maven package命令将所有项目中的类及相关依赖打包成一个JAR（Java Archive，Java归档）包，同样使用scp命令复制到服务器端。接着，编写针对Spring Boot应用部署的Dockerfile文件，示例如下：

1. FROM java:8
2. ADD KnowledgeGraph-0.0.1.jar /kg.jar
3. EXPOSE 8080
4. ENTRYPOINT["java","-Djava.security.egd=file:/dev/./urandom","-jar","/kg.jar"]

其中，FROM指定了容器依赖的镜像，ADD将主机中的内容拷贝至容器内，EXPOSE指定容器暴露的接口，而ENTRYPOINT指定了运行容器时执行的命令。参数中，-Djava.security.egd=file:/dev/./urandom指定Tomcat启动时生成随机数的算法。默认的随机数生成算法可能因应用多次启动，熵池被用空而导致阻塞。使用参数指定后能避免这种阻塞的产生，以保证后端应用正常启动。最后，基于Dockerfile创建镜像，将后端应用运行在基于新镜像的容器内。

1. **前端应用的部署**

前端整体上是一个使用NPM（Node Package Manager）进行包管理，基于Vue.js的应用，以Node.js作为底层运行环境。部署时，首先使用命令node build/build.js，调用webpack打包工具完成前端整体应用的打包，在dist目录下生成静态的HTML、CSS及JS文件，并包含前端应用到的所有静态资源。然后将dist目录上传到服务器。

在服务器端，本文选择基于Nginx镜像构建前端应用容器。为了使用Nginx完成前端到后端请求的转发，首先需要编写配置文件default.conf，示例如下所示：

1. server {
2. listen 80;
3. server\_name vue-grad;
4. location /api/ {
5. rewrite /api/(.\*) /$1 **break**;
6. proxy\_pass http://139.224.193.157:8080;
7. }
8. }

配置文件中指定了端口号、服务器名，以及转发请求的规则。前端发送给后端的请求都以/api/作为起始路径，以与其它请求进行区分，而后端接收请求时的路径是/api/后面的部分，因此在转发请求时需要将路径中起始的/api去除，并用proxy\_pass指定后端服务器的IP地址与端口号，完成请求的转发。接着，为前端容器编写如下的Dockerfile文件：

1. FROM nginx
2. COPY dist/ /usr/share/nginx/html/
3. COPY default.conf /etc/nginx/conf.d/default.conf

同样基于Dockerfile创建镜像并运行容器，完成前端应用及整个系统的部署。

1. 课程学习系统的使用测试

系统设计与功能的合理性与系统对程序设计课程学习效果提升的有效程度都需要通过分析用户实际使用后给出的反馈与意见来确定。因此，本文根据第三章与第四章中的整体流程与实现细节，对基于Java语言的程序设计课程完成了多模态知识图谱的构建与课程学习系统的实现，为用户提供在线的程序设计课程学习，并设计了调查问卷作为用户反馈的收集渠道。

1. **问卷设计**

本文的用户使用体验调查问卷设计如下：

您对Java语言的掌握程度？（1-完全不会、2-正在学习基础语法、3-正在学习高级应用、4-熟练掌握）

请您对系统知识点学习模块进行评价（1-5）

请您对系统知识图谱可视化模块进行评价（1-5）

请您对系统的可用性（用户友好程度）进行评价（1-5）

如果问题1)选择了1/2，您认为本系统的多模态式学习方法是否有助于您掌握程序设计课程？（有帮助，无帮助，不确定）

如果问题1)选择了3/4，您认为对比您当时学习的程序设计，本系统的多模态式学习方法是否对课程知识点的掌握更有效果？（效果更好，效果更差，差不多）

您对本系统还有什么意见或建议？

1. **结果分析**

系统测试与问卷发放主要针对多所不同大学与不同专业的大学生群体，最终收回42份问卷结果，具体结果如下所示：

表5-1 问卷统计结果

|  |  |  |
| --- | --- | --- |
| 题目 | 选项 | 人数 |
| 您对Java语言的掌握程度？ | 完全不会 | 10 |
| 正在学习基础语法 | 5 |
| 正在学习高级应用 | 25 |
| 熟练掌握 | 2 |

续表5-1

|  |  |  |
| --- | --- | --- |
| 题目 | 选项 | 人数 |
| 请您对系统知识点学习模块进行评价 | 1（很差） | 0 |
| 2（较差） | 2 |
| 3（一般） | 17 |
| 4（较好） | 18 |
| 5（很好） | 5 |
| 请您对系统知识图谱可视化模块进行评价 | 1（很差） | 0 |
| 2（较差） | 2 |
| 3（一般） | 12 |
| 4（较好） | 22 |
| 5（很好） | 6 |
| 请您对系统的可用性（用户友好程度）进行评价 | 1（很差） | 0 |
| 2（较差） | 2 |
| 3（一般） | 22 |
| 4（较好） | 15 |
| 5（很好） | 3 |
| 如果问题1)选择了1/2，您认为本系统的多模态式学习方法是否有助于您掌握程序设计课程？ | 有帮助 | 8 |
| 无帮助 | 1 |
| 不确定 | 6 |
| 如果问题1)选择了3/4，您认为对比您当时学习的程序设计，本系统的多模态式学习方法是否对课程知识点的掌握更有效果？ | 效果更好 | 20 |
| 效果更差 | 0 |
| 差不多 | 7 |
| 总计 | | 42 |

调查问卷首先将调研群体用户通过问题1)分为两大类：第一类是完全不会或正在学习Java基础语法的用户（36%/15人），这类用户是系统主要针对的用户群体；第二类是正在学习Java高级应用或已熟练掌握Java的用户（64%/27人），这类用户虽然并不是本次系统测试针对的用户群体，但是他们拥有充足的对程序设计课程学习的经验，可以通过对比自身学习与系统的多模态式学习，对系统提出有效反馈。

第一类用户中，认为多模态式学习方法对程序设计课程的掌握有帮助的用户占比达53%（8人），说明系统能为目标用户提供预期的服务与效果；选择不确定是否有帮助的有40%（6人），这可能源于对系统体验的时间较短，功能体验不够充分，也可能源于系统对用户的引导还不够充分，尤其对于完全不熟悉Java语言的用户而言，可以作为系统后续的优化点之一。第二类用户中，认为多模态式学习方法对比自己当时的程序设计学习，对知识点的掌握更有效果的达74%（20人），没有人认为效果更差，说明基于多模态知识图谱的课程学习系统对比现有的传统教育方式，对知识点整体层面上的学习与理解确实更有效果。

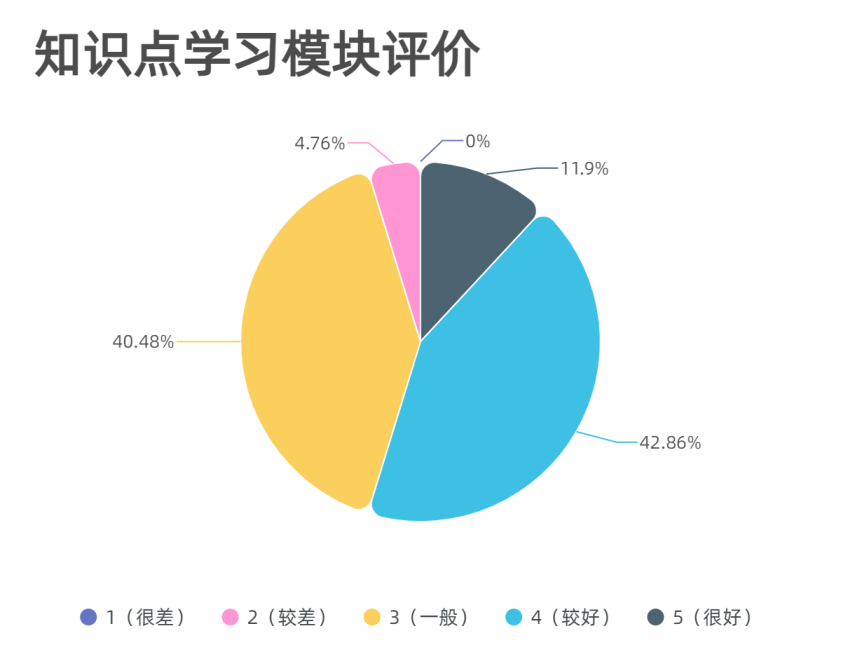
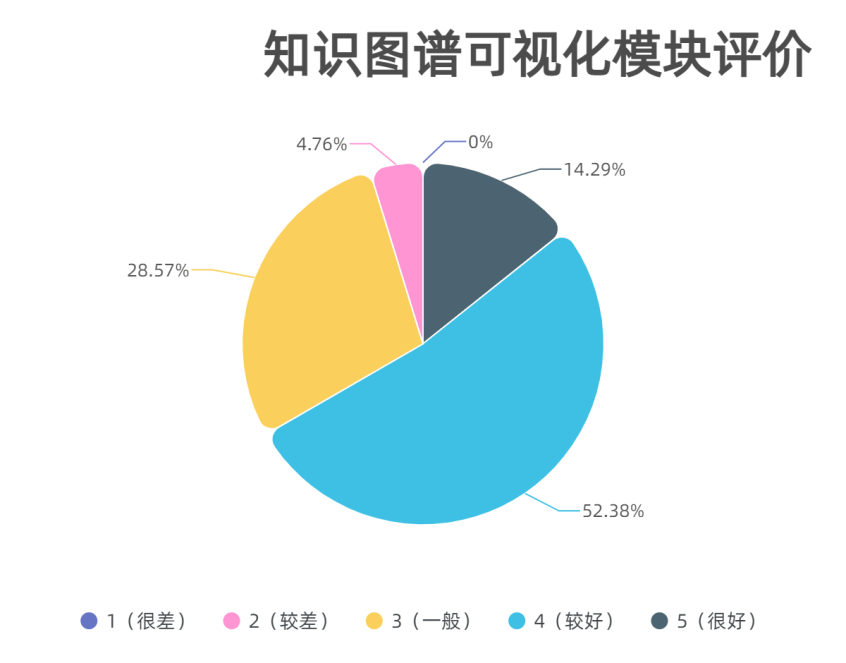
问卷的剩下四个问题对所有用户进行统一调研。对于系统的核心功能之一，知识点学习模块，有40%（17人）认为一般，43%（18人）认为较好，一般及以上占比达到95%（40人）；对于另一个系统核心功能，知识点图谱可视化模块，有29%（12人）认为一般，52%（22人）认为较好，一般及以上占比达到95%（40人）；对于系统的总体可用性，同样有95%（40人）打分为一般及以上。对用户统一调研系统核心功能的数据饼状图如下图所示：

图5-1 系统核心功能调研数据饼状图

通过对以上数据及对最后用户提出的意见与建议分析可知，系统整体可用性与核心功能模块基本达到预期标准，可以给用户提供较高质量的体验。值得注意的是有两个用户对系统给出“较差”的评价，这主要源于他们使用手机访问了系统，没有收获很好的用户使用体验。此外，其它的意见和建议还提到希望能增加包含的知识点，包含的教育资源及更多的功能等，因此移动端的适配、服务器性能的提升与更多功能的开发也是系统未来可以优化的方向。总而言之，目标群体的反馈与对比反馈都有力地证明了系统的有效性与可用性。

1. 总结与展望
2. **总结**

本文首先针对基于Java语言的程序设计课程设计了课程知识本体，并将其实例化后存储于Neo4j图数据库中，并采用基于机器学习的多分类模型完成多模态课程教学资源与知识点概念间的链接，将多模态教学资源也存储于课程知识图谱中，将知识图谱扩展为多模态知识图谱。在多模态知识图谱的基础上，构建了课程知识点学习系统来为Java程序设计的初学者提供服务。该系统基于前后端分离的模式，前端使用Vue.js框架，搭配D3.js框架实现课程图谱的可视化；后端基于Spring Boot框架，完成用户信息管理与图谱信息的查询。

本文实现了程序设计课程知识图谱的构建，首先参考《Java语言程序设计（基础篇）》的课程体系结构设计课程本体的层次结构，然后通过爬虫抽取在线编程学习网站的信息，使用统计学习的方式确定课程相关概念，并基于多模匹配确定课程知识点间的关系，完成知识图谱的实例化，并将知识图谱存储于图数据库中。

本文实现了课程教学资源分类模块，首先根据教学资源类型采用对应的不同方式抽取其中的文本信息，然后利用已有的课程知识点详细信息作为语料库对fastText模型进行有监督训练，并通过试验调优模型参数获得较为高性能的针对课程的文本多分类模型。运用模型完成资源文本的多分类，即完成多模态教学资源到知识点概念的多对多链接与多模态知识图谱的实例化，为后续系统的功能提供底层支持。

本文设计并实现了多模态课程知识点学习系统，为课程的初学者或自学者提供服务。学生能准确地获得知识点的相关多模态学习信息，并减少用于重复搜索资料而浪费的时间，以提升学生在自学过程中的积极性；同时，通过系统中课程知识图谱可视化的功能，学生能明晰课程自学的路线，明确章节与知识点间的包含关系及知识点间的前后学习顺序，避免知识点的遗漏或跳跃学习，有助于构建完整的课程知识体系，也有利于多个知识点间的融合运用，并进行查漏补缺。

综上所述，本文介绍了多模态课程知识点学习系统的设计理念与整体思路，以程序设计课程为例给出了学习系统具体实现的技术要点、关键模块的功能实现及整体构建流程，并针对该系统进行了使用测试与问卷调研，结果令人比较满意，说明系统能为学生提供更好的学习环境，有较好的使用体验与效果，具备实际应用意义。

1. **展望**

本文虽然设计并实现了多模态课程知识点学习系统，但仍有一些不完善的地方可以改进，各个模块的功能也可以继续进行扩展。

从设计角度而言，首先，本文在构建课程知识图谱时使用了自动化与人工结合的方式，且仅针对单一课程。未来如果要将范围扩展到更多的课程领域，这一构建方式的效率会有所不足，可能需要更多地依赖人工智能全自动地完成课程概念与关系的提取；其次，本文仅将有限的课程教学资源纳入了系统范围，还有更多形式的多模态教学资源，如图片、练习题等，可以通过采用不同的知识链接方式与算法置入多模态知识图谱，提供更全面的知识点的学习角度。

从模块实现的角度，首先，本文采用的模型标注训练集仍十分有限，因此可能会对模型的分类效果产生影响。未来如果能获取更多更高质量的标注语料库，资源分类模块的表现可能有所提升；其次，本文中系统的功能还可以进一步扩展，比如根据学生学习情况实现因人而异的学习路径推荐，将更多计算机领域学科的课程知识图谱纳入系统后实现不同课程间知识点的联系，或者实现系统多模态知识图谱的自动更新等。这些功能都能为学生提供更好的学习条件。此外，还可以考虑将移动端适配、服务器性能、负载均衡与带宽提升纳入考虑范围之内。

总而言之，未来仍可以在系统的基础上进行进一步的研究与改进，并实现更多更全面的功能。

# 参考文献

1. Berners-Lee, Tim, et al. The Semantic Web. Scientific American, vol. 284, no. 5, pp. 34–43. 2001.
2. M. Bucos, B. Dragulescu and M. Veltan. Designing a Semantic Web Ontology for E-learning in Higher Education. 2010 9th International Symposium on Electronics and Telecommunications, pp. 415-418. 2010.
3. Yang, Z., Wang, Y., Gan, J., et al. Design and Research of Intelligent Question-Answering (Q&A) System Based on High School Course Knowledge Graph. Mobile Netw Appl 26, 1884–1890. 2021.
4. Penghe Chen, Yu Lu, Vincent W. Zheng, et al. An Automatic Knowledge Graph Construction System for K-12 Education. Proceedings of the Fifth Annual ACM Conference on Learning at Scale (L@S '18). Association for Computing Machinery, New York, NY, USA, Article 40, 1–4. 2018.
5. P. Chen, Y. Lu, V. W. Zheng, et al. KnowEdu: A System to Construct Knowledge Graph for Education. IEEE Access, vol. 6, pp. 31553-31563. 2018.
6. Chao Huang, Quanlong Li, Yuanlong Chen, et al. An Effective Method for Constructing Knowledge Graph of Online Course. 2021 4th International Conference on Big Data and Education (ICBDE 2021). Association for Computing Machinery, New York, NY, USA, 12–18. 2021.
7. Thushari Atapattu, Katrina Falkner, and Nickolas Falkner. A Comprehensive Text Analysis of Lecture Slides to Generate Concept Maps. Computers & Education, Volume 115, Pages 96-113. 2017.
8. Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification[J]. arXiv preprint arXiv:1607.01759. 2016.
9. 李直旭,何芙珍,刘安. 多模态教学知识图谱的构建与应用[J]. 福建电脑,35(8):5-8. 2019.
10. Zhu X, Li Z, Wang X, et al. Multi-Modal Knowledge Graph Construction and Application: A Survey[J]. arXiv preprint arXiv:2202.05786. 2022.
11. 王岩,欧歌,向健,等. Java课程体系本体系统构建探索[J]. 计算机教育, 2016(6): 108-111, 115.
12. 梁勇（Liang, Y. D.）著, 戴开宇译. Java语言程序设计（基础篇）（原书第10版）[M]. 北京：机械工业出版社, 2015.6.
13. Java教程[EB/OL]: https://www.runoob.com/java/java-tutorial.html. 2013-2022.
14. Java教程视频[EB/OL]: https://www.bilibili.com/video/BV18J411W7cE. 2019-2022.
15. Scrapy框架[EB/OL]: https://scrapy.org/. 2008-2022.
16. Google TF-IDF[EB/OL]: https://www.searchenginejournal.com/google-tf-idf/304 361/. 2019.
17. Junyi Sun. 2013. jieba. https://github.com/fxsjy/jieba/. (2022).

# 致 谢

这篇论文的完成首先要感谢我的指导教师戴开宇老师。起初我的研究经验较少，不知道如何选定题目，也不知道如何着手开始工作，是戴老师通过微信、线上会议等多种方式结合与我讨论，确定了感兴趣的方向与工作的目标，并提供了历年指导的论文题目以供我参考；当我在学期初对毕业设计态度有所懈怠时，是戴老师及时地点醒我，并令我意识到我应当为自己负责，开始以端正的态度对待这份毕业论文。在具体的工作开展环节，戴老师也多次为我指明继续前进的方向，让我找到明确的目标。我非常庆幸戴老师成为了我大一上的第一门课的任课老师，也衷心地感谢老师对我这段时间毕业论文工作的指导与帮助。

其次，我要感谢复旦大学与软件学院，为我提供了四年宝贵的学习经验与许多非常优秀的同学、老师。这段大学的经历对我至关重要，不仅令我明确了这篇论文的主题，我自己所学习与热爱的专业，更可能影响我这一生之后几十年的选择与道路。我会永远将复旦大学视作我的母校。

再次，我想感谢这段困难的时间以来一直默默付出与奉献着的医务工作人员与志愿者们。在我完成这篇毕业论文的时间里，恰逢上海疫情的一次反复，虽然时局艰难，但在这困难的大环境下却有着无数“大白”们奋战在第一线，不计时间地完成核酸检测，提供物资等等。他们是病毒与人民之间的第一道防线，也是我能安心完成这篇论文的一大理由。

我还想感谢我的母亲，外公与外婆。他们是我最亲近的家人，从小到大一直无条件地支持着我。在因疫情封控的时间里，他们也在尽一切努力为我创造不受影响的环境。是他们对我的照顾让我能够顺利地完成学习与研究。

最后，我由衷地感谢各位评审老师与专家为答辩与审阅这篇论文所付出的时间与劳动。