分类号：　　密级：
U D C ：　　学号：411016620319

南 昌 大 学 专 业 学 位 硕 士 研 究 生

学 位 论 文

基于知识图谱的混合式职位推荐系统的研究与设计

Design and Research of Hybrid Position Recommendation System Based on Knowledge Graph

童孝武

培养单位（院、系）：数学与计算机学院 计算中心系

指导教师姓名、职称：占传杰 教授

指导教师姓名、职称：黄水源 副教授

专业学位种类：电子信息

专业领域名称：软件工程

论文答辩日期：2023 年 5 月 25 日

答辩委员会主席：

评阅人：盲审

盲审

2023 年 5 月 25 日

摘　要

近些年来，随着信息技术的迅速发展，各行各业的企业逐步开启线上运营模式。线上服务的发展引起网络数据急剧攀升，进而引起信息过载问题。在求职领域中，求职者往往需要耗费大量的时间以及精力在富含海量岗位信息的招聘网站上筛选出符合自己要求以及自己期望的招聘岗位。另外，在职位推荐方面，新求职者存在无用户行为的问题以及职位评分信息较为稀疏的问题，这些问题会导致职位推荐的精确率和召回率等指标降低。因此，针对求职者线上推荐职位的问题，基于知识图谱的混合式职位推荐系统应运而生。该推荐系统中的混合式职位推荐算法通过求职者简历和职位的相关联系以及求职者的用户行为，计算出求职者与职位之间的关联度，进而将与求职者关联度相对较高的职位推荐给用户。该推荐系统的主要工作内容如下：

（1）本文提出了一种基于自然语言处理来识别实体的方法，使用 BERT 预训练模型和 BiLSTM-CRF 模型对求职要求信息进行知识提取。首先，使用 BERT 模型对原始求职要求信息进行编码，得到语义表示；然后，使用 BiLSTM-CRF 模型对编码后的求职要求信息进行标注，识别出能力、性格、技能等重要实体。识别出的实体可以丰富本推荐系统的职位领域知识库。本文使用的数据集来源于招聘网站公开信息，且本文将爬取的职位要求信息进行人工标注。在人工标注的数据集下，本文对多个实体识别模型进行对比。实验结果表明，该模型相对于其余模型在实体识别方面具有更高的准确性。

（2）本文提出一种混合式职位推荐方法，既考虑了求职者简历、职位以及求职者行为之间的关联信息，又考虑了时间因子对职位评分数据的影响。该推荐方法使用 TransE 获取包含用户简历与职位语义的职位相似矩阵。其次，该推荐方法使用时间衰减函数动态化降低职位评分权重，并使用隐语义模型解决职位评分矩阵数据稀疏问题，然后计算出包含用户行为与职位语义的职位相似矩阵。最后，本文融合两种语义矩阵预测求职者未评分的职位，得出各求职者的 Top-N 推荐职位。实验结果显示，本文提出的基于知识图谱的混合式职位推荐方法相较于基于内容的职位推荐方法在准确率上要高出 2%。

关键词：职位；知识图谱；语义矩阵；混合式

ABSTRACT

In recent years, with the rapid development of information technology, enterprises from all walks of life have gradually started to operate online. The development of online services has caused a sharp increase in network data, which in turn has caused the problem of information overload. In the field of job hunting, job seekers often need to spend a lot of time and energy on the recruitment websites rich in massive job information to find the job positions that meet their requirements and expectations. In addition, in terms of job recommendation, new job seekers have the problem of no user behavior and relatively sparse job rating information. These problems will lead to a decrease in the accuracy and recall of job recommendation. Therefore, for job seekers to recommend jobs online, a hybrid job recommendation system based on knowledge graphs came into being. The hybrid job recommendation algorithm in the recommendation system calculates the degree of relevance between the job seeker and the job through the relationship between the resume of the job seeker and the position and the user behavior of the job seeker, and then puts the relatively high degree of relevance to the job seeker Jobs are recommended to users. The main tasks of the recommendation system are as follows:

(1) This paper proposes a method for identifying entities based on natural language processing, using the BERT pre-training model and the BiLSTM-CRF model to extract knowledge about job requirements. First, use the BERT model to encode the original job-seeking requirement information to obtain semantic representation; then, use the BiLSTM-CRF model to mark the encoded job-seeking requirement information to identify important entities such as ability, personality, and skills. The identified entities can enrich the job domain knowledge base of this recommendation system. The data set used in this article comes from the public information of the recruitment website, and this article manually labels the crawled job requirement information. Under the manually labeled data set, this paper compares multiple entity recognition

models. Experimental results show that the model achieves higher accuracy in entity recognition than the rest.

(2) This paper proposes a hybrid job recommendation method, which not only considers the correlation information among job seekers' resumes, positions, and job seeker behaviors, but also considers the impact of time factors on job rating data. This recommendation method uses TransE to obtain a job similarity matrix that contains user resumes and job semantics. Secondly, the recommendation method uses a time decay function to dynamically reduce the weight of job ratings, and uses a latent semantic model to solve the data sparsity problem of job rating matrix, and then calculates a job similarity matrix that includes user behavior and job semantics. Finally, this paper fuses the two semantic matrices to predict the unrated positions of job seekers, and obtains the Top-N recommended positions for each job seeker. According to the experimental results, the hybrid job recommendation method based on knowledge graph proposed in this paper is 2 percentage points higher in accuracy than the content-based job recommendation method.

Key Words: job; knowledge graph; semantic matrix; hybrid

目 录

第 1 章 绪论

1.1 研究背景及意义

随着互联网技术的不断更迭以及互联网的普及，网络成为了人们获取信息、沟通的枢纽。企业为壮大公司规模纷纷采用线上招聘的方式，以更加快捷地招到合适的人才。另一方面政府同时极大鼓励和支持各大企业开展线上招聘方式，以促进就业和经济发展。因此，招聘网站将成为企业招聘人才的重要载体。

在每年的求职人群中应届毕业生占很大一部分比重，根据教育部公开数据 21 年的高校毕业生的数量已达 900 多万且高校毕业生的数量在逐年增加。教育部也曾指出研究生以及专升本的招录规模的将持续适度扩大，所以高校毕业生数量将在 22 年继续递增。对于如此庞大数量的求职者所面临的求职问题成为了求职者、就业市场以及学校热点关注问题。求职者求职与企业招聘是一个双向选择的过程，在该过程中，求职人员根据自身个人技能、项目经验、教育背景等寻找可匹配的职位，而用人单位通过岗位职责、学历要求以及岗位要求筛选出满足条件的求职者简历。在网络上每天都会有招聘岗位的发布，以至于现各大招聘网站存在着大量的招聘岗位信息。这样对求职者而言很难查找到自己相匹配的职位。当前的职位推荐大多基于点击率较高的岗位或者提供所在城市以及最近发布职位专栏进行推送。这样未考量求职者以及招聘者的双方需求就进行岗位推荐，仍然需要求职者进一步的筛选，加大求职者的寻找工作的时间。此外对用人单位来说这类职位推送的不准确性会加大用人单位与求职者的沟通成本。这类传统方式推荐出的职位列表仍然包含了与求职者不匹配的职位，影响求职用户的使用体验。为了提供更加高效、准确的求职体验给求职用户，高精准性的职位推荐成为招聘网站上不可或缺的功能。

针对上述推荐效果差的情况，本文提出的基于知识图谱的混合式职位推荐算法能够提供更高的职位推荐准确率，以此满足求职者对职位的需求。该混合式职位推荐算法通过知识图谱展示出岗位、岗位信息、求职者简历以及求职者行为之间的关联关系。然后根据职位领域的知识图谱所隐含的语义预测求职者对职位的评分，再将预测评分较高的职位推荐给求职者。使用该混合式职位推荐算法对

职位做推荐可以加快求职者的求职速率，并提高人才市场和招聘网站中人才的流动率。因此，基于知识图谱的混合式职位推荐系统对于职位推荐领域的研究具有重要贡献和实际意义。

1.2 国内外研究现状

1.2.1 知识图谱的国内外研究现状

知识图谱[1]（Knowledge Graph）的概念早在 2012 年时被互联网行业巨头谷歌公司推出，然而在最开始知识图谱用来优化谷歌的搜索引擎，进而增大信息搜索的全面性。早期的知识图谱仅局限于在知识搜索领域中使用模糊关键词来匹配图谱中的相关信息。随着时间的推移，现代研究者利用知识图谱进行知识推理，即在知识图谱中原有的显示关联信息基础上推理出实体之间的隐含信息。例如用户搜索“孟子”，搜索引擎通过历史人物领域的知识图谱响应出孟子的相关事迹、相关作品以及与儒家相似的古希腊的斯多葛学派中相关著名人物以及事迹等相关信息。知识图谱技术能够将各种信息以及信息之间的关联关系整合成为有意义的语义知识库，并使用该语义知识库获取用户需要的答案。这种方法可以帮助人们更加高效地获取并应用知识，从而促进各种智能应用的发展。除了知识图谱在知识搜索领域研究外，研究者在机器人问答、智能推荐等领域也展开了相关研究，并取得阶段性成果。

语义检索是知识图谱的一项重要应用，其后台容纳了海量半结构化和结构化的信息，便于知识检索。目前，国外相当一部分的企业开展了对百科图谱的研究，其中，开放知识库有 Wikidata[2]、YAGO[3]、OpenCyc[4]等。此外，国内的知名企业也对百科图谱展开了探索，例如，国内知名大学的实验室构建的 CN-DBpedia[5]、用于解释世界万物中词语概念的百度知心等。另一方面，知识图谱在行业中也有着重要的应用。行业领域的知识图谱依赖特定行业的信息作为数据支撑，从而被构建出来。国外有提供电影领域知识库的 IMDb，音乐领域知识库的 MusicBrainz[6]等。而国内有中医药知识库、海致星图金融知识库等。

总体来说国内研究者们对知识图谱的研究虽达到了阶段性的成果，但在知识图谱的某些行业领域中的应用仍需进一步研究，尤其在求职行业领域的知识图谱构建以及基于知识图谱的职位推荐算法研究甚少。

1.2.2 推荐方法的国内外研究现状

推荐技术早在 1995 年就被提出，起初用于电商系统中已注册用户的商品推荐。随着互联网技术的进一步发展，推荐领域的研究吸引着越来越多的研究者的探讨。然而今日的推荐技术分成基于内容、协同过滤和混合推荐技术这三大类。

基于内容的推荐算法的核心思想是通过用户之前浏览过、关注过的物品来推荐该用户可能感兴趣的类似商品。国内学者 Jiangbo Shu 等将基于内容的算法应用于多媒体学习资源系统中，然后系统会将特定学习资源定向推荐给特定学生[7]。国外研究者 Nguyen Van Dat 提出一种在高斯混合模型的基础之上的基于内容的推荐算法，且该算法在白酒数据集训练中得到较高的准确性实验结果[8]。基于内容的推荐算法的优势在于用户的物品推荐独立于其他用户之外，避免了物品的冷启动问题。但是在算法实现上不同物品的特征提取有着一定的工作量，且无法为刚注册的用户提供物品推荐，即该算法存在用户的冷启动问题。

基于协同过滤的推荐算法主要是通过当前用户的特征来寻找特征相似的用户，然后将相似用户喜欢的物品推荐给当前用户。李珊珊对协同过滤的算法中的相似度计算进行改进，然后应用于视频推荐，从而为系统提供更加精准的视频推荐功能[9]。郑策等研究人员提出针对电影数据的图结构的特点来计算影片的相似度改进基于协同过滤的推荐算法，且从实验结果上看此方法有效解决数据稀疏问题[10]。基于协同过滤的推荐算法虽相对于基于内容的推荐算法避开了获取物品的特征标签，但在用户数量大的情况下，其算法的效率较为低下。

针对以上两类算法的单一性以及存在的缺点，基于混合的推荐算法将多种推荐算法混合进行物品推荐，常见的混合算法是将基于协同过滤的推荐算法与其他算法进行组合。混合的推荐算法可以很好的解决冷启动、数据稀疏等问题。Xinyi Li 等国内研究者提出一种利用论文的特征词与浏览论文的日志关联的重新排序模型，来解决学术论文推荐系统的冷启动问题[11]。Masoumeh Riyahi 等研究者用 WordNet 词库提取小组讨论的标签[12]，再用标签语义和用户隐含分数的协同过滤，从而进行讨论组推荐。

1.2.3 基于知识图谱的推荐方法的国内外研究现状

知识图谱是一张具有丰富语义知识的网络图，将其在推荐领域应用可以更加细粒度的挖掘知识间的潜在联系，以实现精准推荐。有关知识图谱的推荐算法大

体分为基于路径、嵌入以及混合的推荐算法。目前，国内外有大量研究人员对新闻、医疗、文学、音乐、工业、农业、电商等特定领域的知识图谱进行推荐方向的研究。张丹等学者构建了新闻领域的知识图谱，并改进隐马尔科夫模型来预测用户阅读的下一篇更大概率浏览的新闻[13]。S Oramas 等采集歌曲数据以及用户对歌曲评论、浏览等信息并构建音乐领域的知识图谱，之后用知识图谱中节点间的路径进行近邻歌曲的音乐家的推荐[14]。阮小芸等对真实的简历数据构建人才领域的知识图谱，并提出用强化学习的方法在知识图谱上训练出可用于发现潜在需求和兴趣，从而进行人才工作的推荐[15]。

基于知识图谱的推荐方法包含基于路径、连接以及混合的推荐方法。基于路径的推荐通过计算实体之间的相似度来挖掘实体之间的潜在联系，然后通过实体间的相关性实现推荐。基于路径的推荐方法将知识图谱看成知识相互关联形成的异构型网络，通过基于节点之间的路径规则进行匹配计算。使用该方法进行推荐可以极大降低计算成本且推荐结果具有可解释性，但对于不同领域的知识图谱需要制定出路径规则要求。

相比基于路径的推荐，基于嵌入的推荐需要将知识图谱中的实体和关系映射成二维平面、三位平面或超平面上的低维向量，然后通过实体和关系的低维向量计算出实体间的相似关联度并实现推荐。基于嵌入的推荐算法从知识图谱的全局角度出发进行推荐，但这种推荐方式忽视间接连接的两实体之间的关联关系，且最后的推荐结果具有可解释性差的特点。

考虑到基于路径的推荐和基于嵌入的推荐方法都有各自的缺点，基于混合的推荐将两种推荐方式结合起来，可以先通过实体连接的路径规则获取用户兴趣，然后将这些兴趣作为图嵌入推荐的特征输入，再得出推荐结果。基于两种混合方式的推荐方法包括三个经典模型：RippleNet[16]、 KGCN[17]（Knowledge Graph Convolutional Networks）和 KGAT[18]（Knowledge Graph Attention Network for Recommendation）。

1.3 主要研究内容

本文主要研究内容为构建求职领域的职位知识图谱，并在此知识图谱的基础上实现职位推荐的应用。其中，在构建职位知识图谱过程中，本文针对性地对岗位信息中的职位要求的非结构化文本数据进行实体抽取，从而丰富职位知识图

谱的实体语义信息，提高后期职位推荐的推荐效果，为广大求职者提供便捷、精准推荐的求职平台。本文的具体研究内容如下：

（1）数据收集与预处理

职位数据多来源各大招聘平台或者各大公司的官网发布，考虑到各大公司的官网获取职位招聘信息的难度较大，因此本文的职位数据来源于拉勾网以及应届生求职网站的公开信息，本文采用 Scrapy[19]爬虫框架对网站公开招聘信息进行爬取。本文在爬取的过程中，遵循 Robots 协议，以适当频率发送网页请求等规范爬虫操作。这些求职网站的原始数据虽然全面，但需要进行字体反爬处理、指定职位特征数据提取、数据清洗等操作。处理后的职位相关数据放置 MySQL 进行存储以便后续知识图谱构建使用。

（2）职位知识图谱构建

构建职位领域的知识图谱首先需要分析实体与实体间的关系，从 MySQL 数据库读取职位数据并对来自多个数据源的职位数据进行知识融合，然后抽取实体与关系组成三元组集合。其中，对于职位要求的文本字段需要先进行额外的实体抽取，再组成三元组。最后将抽取的数据集合保存在 Neo4j 数据库中。

（3）基于知识图谱的混合式职位推荐算法研究

传统的基于物品的协同过滤算法应用在职位推荐领域会因为评分数据稀疏，从而导致推荐准确度较低。而本文提出基于知识图谱的混合式职位推荐算法，综合了求职者简历与职位静态信息以及用户对职位评分的动态信息，有效解决了职位评分数据稀疏和求职用户冷启动的问题。本文通过职位知识图谱中职位与用户之间的语义信息计算出静态职位语义矩阵，还通过求职者对职位的评分计算出动态职位语义矩阵，然后融合两种语义矩阵再预测出求职者对未有用户行为的职位进行打分，从而对各求职者进行 Top-N 职位推荐。其中求职者对职位的评分融合和时间因素以及利用隐语义模型以应对数据稀疏状况。

1.4 论文组织结构

全文共有六个章节，各个章节的简介如下：

第一章 绪论。本章介绍了本论文课题的研究背景以及分析了论文课题的研究意义，详细阐述了知识图谱、推荐方法以及基于知识图谱推荐方法的国内外研究状况。最后本文给出主要研究内容并规划出论文的结构框架。

第二章 相关技术理论。本章对职位推荐系统构建所需的相关技术理论从前往后逐一介绍。

第三章 职位知识图谱的构建。本章详细介绍如何构建职位知识图谱。首先爬取招聘网站公开的招聘数据并对数据进行处理，对职位要求的文本数据进行知识抽取并对实体抽取模型进行实验评估。然后把知识融合后的实体和关系读入到图数据库中进行可视化展示。最终本文完整地将职位知识图谱成功构建。

第四章 基于知识图谱的职位推荐算法。本章在上一章构建的职位知识图谱的基础上提出了一种结合基于隐语义和基于内容两种推荐算法的混合式职位推荐算法并将该混合式职位推荐算法与其他推荐算法在本文的实验数据下进行了相关对比实验。

第五章 系统分析与设计。本章对职位推荐系统做出需求分析、概要设计以及详细设计。需求分析给出职位推荐系统所需实现的功能和工作。概要设计给出数据流的输入输出、推荐的工作流程以及模块的划分。详细设计给出各个模块的详细设计以及数据库的详细设计。

第六章 结论与展望。本章对论文所做的相关工作做出全局性总结，给出了论文之后需进一步完善之处。

第 2 章 相关技术理论

2.1 知识图谱的构建技术

知识图谱本质上是将真实世界中的事物抽象出实体，并用关系将实体进行关联，最后以图形化的方式展现出一张巨大的语义库。知识图谱的表现形式为三元组，而三元组可由<实体 A，关系，实体 B>以及<实体，属性，属性值>构成。例如三元组<人，种类，动物>以及<张三，出生地，中国>。三元组的公式化表示为 $Graph = (E, R, T)$，其中 E 表示为知识库中的所有实体组成的集合（$E = \{E_1, E_2, E_3, \dots, E_n\}$），R 表示知识库中的所有实体之间的关系组成的集合（$R = \{R_1, R_2, R_3, \dots, R_n\}$），T 表示知识库中实体以及实体之间的关系组成的三元组的集合（$T = \{T_1, T_2, T_3, \dots, T_n\} \wedge T \subseteq E \times R \times E$）。

构建知识图谱的体系架构如下图 2.1 所示。构建知识图谱的数据源有结构化数据、非结构化数据以及半结构化数据。其中，非结构化数据以及半结构化数据通过知识抽取步骤抽取出实体以及实体之间的关系。然后将实体以及实体关系依次进行知识融合、质量评估步骤。知识融合是为了判断不同来源的实体是否指向的是相同的实体以及解决一个实体包含多种含义的问题。质量评估包含知识推理、错误检测等步骤，为了提高抽取知识的质量。最后构建出的知识图谱将应用于知识搜索、智能推荐以及智能问答等领域中。下面将对知识抽取、知识表示以及知识存储三个重要环节进行详细介绍。

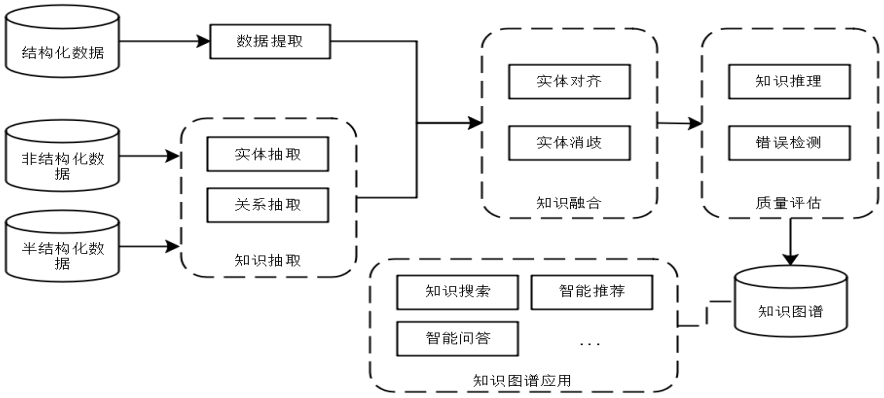

图 2.1 构建知识图谱的体系架构图

2.1.1 知识抽取

知识抽取是将非结构化数据和半结构化数据转向知识融合流程的关键环节。收集的数据往往来源于网页公开的数据或企业内部系统的数据。对于企业内部系统的数据可以直接获取并整合，而对于网页公开的数据可以采用网络爬虫进行抓取并保存。获取数据若是文本、声音等非结构化数据或 XML、HTML 等半结构化数据，则需要依赖自然语言处理技术来进行实体抽取以及关系抽取。关于实体抽取与关系抽取如下介绍。

（1）实体抽取

在知识抽取的过程中，实体抽取是最基础、最重要的模块之一，因为实体抽取的质量很大程度上会影响后续知识的获取速度和质量。实体抽取主要是将规范好的命名实体从非结构化数据或半结构化数据中区分出来，例如从一段文本中标注出对应的命名实体。抽取的实体将成为知识图谱中的节点。目前，实体抽取[20]可以根据预定义的规则来抽取实体。此外，实体抽取还可以通过机器学习算法训练标注好的样本数据从而训练出抽取实体的模型，再用训练好的模型对数据进行处理获取实体。

（2）关系抽取

这里将作为实体补充的属性也归为关系抽取的范畴。关系抽取是从文本中识别和提取出实体之间的关系，然后将实体之间进行语义关联。即关系抽取的目标是识别出文本中实体之间可能存在的语义关系。与实体抽取的技术相似，关系抽取也有基于规则的方法以及基于机器学习的方法，可用于提取实体间的语义关系。

2.1.2 知识表示

知识表示本质上是将实体以及关系用低维向量进行表示，以便构建知识图谱以及知识图谱的后续应用。例如，在构建知识图谱中的实体对齐需要先将实体进行向量化表示再计算实体之间的向量距离，从而利用向量距离进行实体对齐；在智能知识问答系统中通过将知识向量化表示然后计算实体、关系与问题的语义相似度，从而给出问题语义相关的答案。而在知识图谱的推荐系统中利用实体表示的低维向量计算出知识图谱中的实体之间的语义相似度，从而推荐相似度较高的实体。知识表示的算法有很多种，应用较为广泛的有 Trans 系列的翻译模型，

例如 TransE[21]、TransH[22]、TransR[23]等。下面将介绍几种知识向量化表示的训练模型。

（1）TransE

该模型将每个实体和关系都表示为向量。通过最小化已知三元组（头实体，关系，尾实体）与负样本三元组的距离来训练模型，其中负样本是通过替换头实体或尾实体来生成的。距离度量使用$L1$范数或$L2$范数，最小化距离可以使已知的三元组在向量空间中更接近，从而使模型可以推理出未知的实体和关系。关系作为头实体到尾实体的翻译，训练过程中需要对该三元组不停调整，使得头实体通过关系接近于尾实体。其损失函数如下：

$$Loss(head, relation, tail) = max(0, d_{pos} - d_{neg} + margin) \tag{2.1}$$

其中$d = \|head + relation - tail\|$为$L1$或$L2$范数，$d_{pos}$代表正向样本三元组，$d_{neg}$代表负向样本三元组。$margin$为正负样本之间的距离常数。

（2）TransH

TransH 与 TransE 的区别在于 TransH 可以处理一对多或者多对一三元组的关系，本质上将头实体和尾实体向量投影到规定的超面上，然后利用投影在超平面上的向量进行向量训练。例如在两个三元组分别是（赣南脐橙，特产，江西）以及（南丰蜜桔，特产，江西），经过 TransE 模型训练后会认为“赣南脐橙”约等于“南丰蜜桔”，但实际上这两个物品是江西的两大不同的特产。TransH 模型尽可能将法向量为w_r的超平面上的投影向量$head_\perp$存在关系$relation$使得$head_\perp + relation \approx tail_\perp$。其中：

$$head_\perp = head_\perp - w_r^T \cdot head \cdot w_r \tag{2.2}$$

$$tail_\perp = tail_\perp - w_r^T \cdot tail \cdot w_r \tag{2.3}$$

（3）TransR

TransR 可以处理不同类型的关系，且每种关系有自己的向量空间。因此，两实体即使在同一平面下的向量表示趋近，但在不同关系空间中表示着不同的实体。TransR 模型通过关系矩阵M_r将同一实体表示的向量映射到不同的关系空间，使得$head_r + relation \approx tail_r$，实体向量映射公式如下：

$$head_r = head \cdot M_r \, , tail_r = tail \cdot M_r \tag{2.4}$$

2.1.3 知识存储

知识存储为了将实体以及实体之间的关系存储起来，以便于知识图谱后面的知识查询、推理和分析。目前，知识图谱的存储主要是通过图数据库来存储实体以及实体之间的关系。本文主要是用 Neo4j 图数据库。

Neo4j 是一个开源的、高性能的原生图数据库[24]，它使用图形概念存储数据，图形是由节点和边组成的数据结构。Neo4j 使用 Cypher 查询语言，允许开发人员使用类似 SQL 的方式进行查询和操作图形数据。相比关系型数据库，Neo4j 能够更加灵活、高效地存储和查询复杂的关系数据。它支持在图形中存储大量的节点和边，节点和边可以拥有多种属性，这些属性可以帮助我们更好地理解和分析数据。

2.2 基于 BERT-BiLSTM-CRF 模型的知识识别

知识识别是行业领域知识图谱构建的基础。知识识别主要从评论、介绍、正文等文本数据精准地识别出“有价值的实体信息”。常见的知识抽取方法包括深度学习的方法、基于统计学习的方法以及字典的方法。基于统计学习的方法需要对行业领域的知识语料库进行统计数据分析，主要通过词频和正负情感词来训练特征识别模型，从而提取实体。然而，由于行业领域的数据大部分以非结构化和不规则性的文本为主，知识识别精度不高。对于字典方法，通过构建结构化字典来进行知识的模板匹配。虽然字典中知识的识别率很高，但是字典的识别精度并不理想。

针对上述识别精度不高的问题，本文使用了基于 BERT-BiLSTM-CRF 的深度学习模型来进行知识识别。首先，第一层 BERT 模型[25]通过多层神经网络对输入文本进行预训练并将文本转换成词向量。BERT 模型训练的词向量与传统的训练模型相比，所获得的词向量在不同的上下文中表达不同的语义，可以捕捉句子层面的隐含特征。其次，第二层 BiLSTM 模型[26]可以根据输入的词向量序列自动地提取句子的局部上下文特征。正向 LSTM 模型输出隐藏状态序列，反向 LSTM 模型将所有隐藏状态序列拼接起来，根据句子序列获得完整的隐藏状态序列。最后，第三层 CRF 模型[27]通过计算整个序列的概率分布，将局部特征归一化为全局特征，以解决部分标签偏差的问题。同时，CRF 模型可以在训练数据时获得标签的隐含约束规则。

2.3 推荐算法

2.3.1 基于协同过滤的推荐算法

协同过滤算法[28]（Collaborative Filtering，CF）的基本思想是通过用户对物品的历史用户行为，例如浏览时长、浏览次数、打分情况等，来预测其中某个用户可能感兴趣的物品。在推荐侧重点方面分为基于用户的协同过滤和基于物品的协同过滤。协同过滤算法可以挖掘出用户的潜在的兴趣与特征，同时针对用户历史行为做出推荐具有解释性强的特点。但是其也存在一些问题，比如用户数据涉及隐私问题、数据稀疏导致计算复杂的问题、新用户特征不足引起的冷启动问题等。因此，在使用协同过滤算法时通常和其他互补算法混合来进行推荐，比如深度学习、隐语义模型等。

基于用户的协同过滤算法的核心思想是通过已经给出物品评分的其他用户来对当前用户进行物品推荐[29]。如果某个用户对一个商品给予很高的评分反馈，该物品很有可能被推荐到有着类似浏览相关特征物品的用户。该算法需要建立一个用户-物品评分矩阵来存储用户之间的相似度评分。如果用户未与某物品有过交集，那么该潜在评分将通过相似的用户计算得出。如下图所示，求职者 A 关注的岗位有开发工程师岗位 1、开发工程师岗位 2 以及开发工程师岗位 4。求职者 B 关注的岗位有开发工程师岗位 1、开发工程师岗位 2 以及开发工程师岗位 3。而基于用户行为或静态属性计算出两用户相似较高，因此该算法将开发工程师岗位 3 推荐给求职者 A 同时将开发工程师岗位 4 推荐给求职者 B。图 2.2 所示基于用户的协同过滤算法推荐过程。

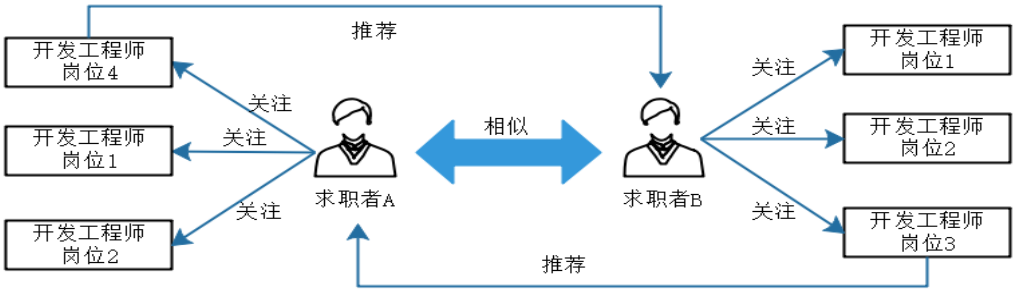

图 2.2 基于用户的协同过滤算法推荐过程

基于物品的协同过滤算法使用一个物品矩阵来存储物品之间的相似度分数[30]。该算法会推荐一组大众评分较高的相似物品。本质上，预测的物品评分取决于相邻物品之间的相似度，相似度越高，预测的评分越接近与相邻物品的评分。如下图所示，求职者 A 关注开发工程师岗位 1。求职者 D 关注开发工程师岗位 2，且求职者 B、求职者 C 和求职者 E 都在关注开发工程师岗位 1 的同时关注开发工程师岗位 3。那么通过求职者们的关注行为该算法计算出开发工程师岗位 1 和开发工程师岗位 3 相似度较大，则开发工程师岗位 3 将会推荐给求职者 A。图 2.3 所示基于物品的协同过滤算法推荐过程。

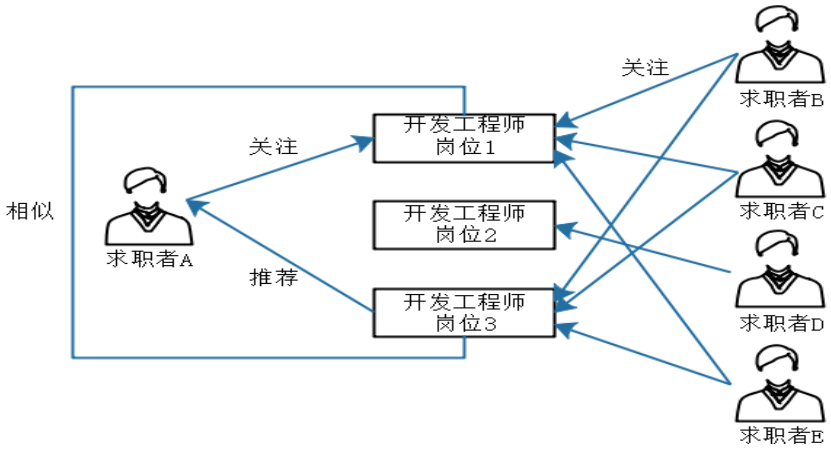

图 2.3 基于物品的协同过滤算法推荐过程

无论是基于物品的协同过滤算法还是基于用户的协同过滤算法都未考量时间因素。如下图所示，后端求职者 2 在之前一段时间关注的是后端开发岗位 1，之后这位后端求职者 2 转成全栈开发且关注了全栈开发工程师岗位 1。但基于协同过滤算法依旧将之前的关注行为程度纳入推荐的训练集中，最终推荐列表中包含后端开发工程师岗位 2 以及全栈开发工程师岗位 2，造成推荐列表的冗余。图 2.4 所示协同过滤算法推荐过程的存在问题。

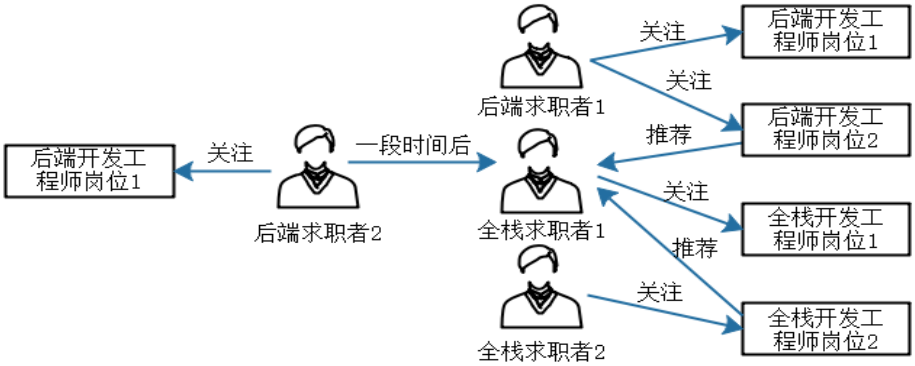

图 2.4 协同过滤算法推荐过程的存在问题

2.3.2 基于隐语义的推荐算法

隐语义分析算法[31]（Latent Semantic Analysis，简称 LSA）是一种基于矩阵分解的机器学习算法，主要应用于文本相关的任务，如信息检索、文本分类、推荐系统等，其中在推荐领域多用于预测评分。LSA 的核心思想是通过矩阵分解得到用户与物品的隐含语义信息，即本质是将稀疏的评分矩阵变成稠密矩阵。相对于 SVD 隐语义算法无需在内存中存放共现矩阵，而是用拟合的方式反复训练数据以求得近似评分矩阵。如下面公式将稀疏矩阵$G_{m\times n}$拆成一个列维度较低的矩阵$P_{m\times k}$以及一个行维度较低的矩阵$Q_{k\times n}$。

$$G_{m\times n} \approx P_{m\times k} \cdot Q_{k\times n} = \hat{G}_{m\times n} \tag{2.5}$$

$G_{m\times n}$代表求职者给职位打分的评分矩阵，$P_{m\times k}$代表求职者与隐含特征关联矩阵，$Q_{k\times n}$代表隐含特征与职位关联矩阵，$\hat{G}_{m\times n}$为预测后的职位评分矩阵。其中职位评分矩阵中的值越大则代表求职者对职位的评价越高。

隐语义算法的主要优点是可以处理高维、稀疏的文本数据[32]，并且不需要人工标注数据集。缺点是矩阵分解过程中可能出现过拟合问题，并且对于一些复杂的语义关系难以处理。

2.4 Scrapy 框架

构建职位知识图谱需要用到爬虫工具收集各大公开招聘网站的职位招聘信息。爬虫工具有很多，而本文采用 Scrapy 框架采集数据。Scrapy 框架可以异步

对网站发起请求，具有速度快的特性。此外 Scrapy 还有自动调节爬取速度的机制，以应对网站对固定 ip 访问频率限制。Scrapy 框架工作流程如下图 2.5 所示。

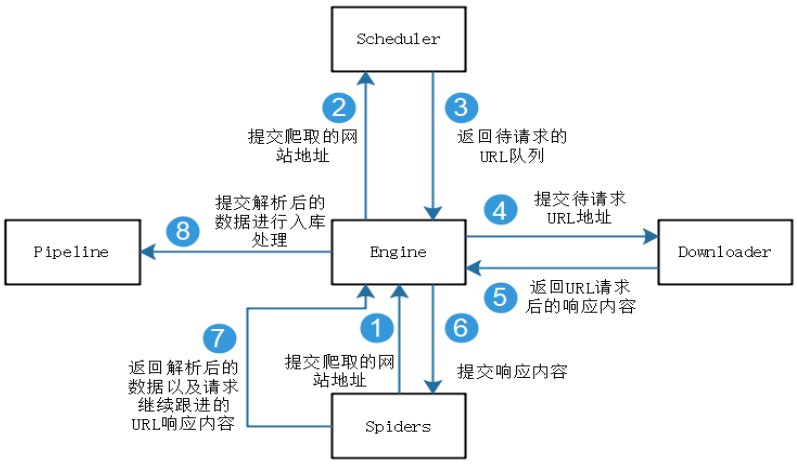

图 2.5 Scrapy 框架工作流程图

爬虫程序将待爬取的网站地址提交给引擎，引擎待爬取的网站地址所需的多个 URL 提交给调度器，调度器将链接进行排队[33]。然后引擎将排好顺序的链接封装成 requests 交于下载器，下载器对网页发起请求并将每个 requests 对应的响应下载下来。之后由爬虫程序将下载器下载的响应内容进行爬虫解析并提取数据，并将解析后的数据返回给引擎。最后管道将引擎提交的解析数据进行入库处理，其解析后的数据可以存储到 MySQL、MongoDB[34]等数据库平台中或者以 TXT、CSV 等文件格式存储。

2.5 前后端技术

本文使用前后端分离且后端功能形式以微服务的方式开发系统。前端使用 Vue[35]开发系统页面。后端使用 Spring Boot[36]搭建微服务并用 Swagger 生成微服务对外提供的接口文档，利用 OpenResty[37]的反向代理将接口文档的 URL 与系统运行时客户端发起连接进入网关的 URL 分开。系统的数据使用 MySQL[38]数据库进行数据持久化。

（1）Vue

Vue 是近年来广受欢迎的基于 JavaScript 的前端框架，其最重要的特点是数据与视图的双向绑定，从而实现数据一旦发生变化时视图会自动进行更新。Vue

的每个组件有着明确的生命周期钩子函数，包括创建、挂载、销毁等阶段[39]。钩子函数可以在指定的阶段执行相应的代码逻辑。此外，Vue 还支持前端工程师对系统页面拆分组件并对组件进行单独开发。在进行系统开发时独立的组件经过简单的入参和出参处理可快速组合到系统中，组件化开发提高了系统的代码复用性进而提高开发效率。

（2）Spring Boot

Spring Boot 是在 Spring 框架基础上研发的一套用于构建企业级应用的开源 Java 框架，它提供相较于 Spring 框架[40]更迅速、便捷的方式创建项目流程，使后端工程师专注于项目的复杂业务逻辑。Spring Boot 把注解和配置类整合在一起，减少大量的 XML 配置文件的编写所带来的的负担[41]。此外，Spring Boot 支持与 Spring 生态圈中大量组件集成，为后端工程师提供丰富的开发解决方案。

（3）OpenResty

OpenResty 是一个将 Nginx[42]和 Lua[43]结合在一起的 Web 应用服务器，还包含丰富的三方模块以及更加快速的编译器 LuaJIT。Openresty 支持 Lua 语言意味着开发人员可以进行个性脚本化定制业务逻辑，同时还提供了可视化的服务器监控和管理。OpenResty 可以为系统的服务提供反向代理功能、利用脚本化语言定制化负载均衡策略以及定制化客户端访问的安全策略。

OpenResty 具有优秀的性能和稳定性，它采用基于事件驱动以及异步 I/O 的机制，能够轻松处理 10k 以上的单机高并发的网络连接。此外，OpenResty 内置了安全模块，如上游的健康检查、IP 黑名单、支持 SSL 加密通信等[44]，具有很高的安全性。

与 Nginx 服务器相比，OpenResty 不但保留了 Nginx 的事件驱动模型还结合使用 LuaJIT 将 Lua 语言直接编译成可执行的机器代码。由于 OpenResty 支持脚本语言 Lua，开发人员可以根据需要进行定制和扩展更为复杂的功能。此外，OpenResty 社区正在不断成长，社区的热度也在不断上升。由于这些优点，OpenResty 日渐成为系统开发中广泛使用的一种 Web 服务器，特别是在高并发和业务定制化场景中表现优异。

（4）MySQL

MySQL 是一款广泛流行于中小企业的关系型数据持久化工具，它具有稳定、安全、查询速度优良、可跨平台等特性。MySQL 可以更换不同的存储引擎，以置换相应存储引擎在特定业务场景的优势。在数据库使用方面，MySQL 支持 SQL

语法，还提供触发器、事务、存储函数等机制使开发人员高效地对数据管理的操作[45]。在跨平台方面，MySQL 可以在 Windows、OpenBSD、Linux 等操作系统上安装使用。在数据库使用所需开销方面，被甲骨文公司收购后的 MySQL 依旧保持开源免费状态，且依旧维护高活跃度的社区。

2.6 本章小结

本章主要对本文研究过程中所涉及到的相关技术理论进行介绍。首先，本章介绍了构建知识图谱所需的相关技术，并对知识抽取的相关模型进行了详细介绍。其次，本章对数据采集工具所用的主流爬虫框架 Scrapy 进行简要介绍。然后，本章介绍了推荐算法的相关理论知识。最后，本章对职位推荐系统开发所需的程序实现框架以及数据库进行详细介绍。

第 3 章 职位知识图谱的构建

知识图谱是将真实世界中的事物抽象出实体并用关系将实体进行关联，富含丰富的语义知识。职位推荐算法在知识库的基础上可以很方便地抽取出职位之间的关联关系。一方面，本章将处理后的职位数据集进行实体识别实验，抽取出职位要求的技能、性格等实体；另一方面，本章利用抽取后的职位要求实体和未处理的结构化职位数据共同构建出职位知识图谱，并将构建的职位知识图谱存于图数据库中。存于图数据库的职位知识图谱为后续章节做混合式职位推荐算法提供数据支撑。

3.1 职位信息的来源和反爬处理

构建职位知识图谱首先需要收集各大招聘网站的职位招聘详细信息以及用户对职位的评分数据。在针对职位信息的真实性和准确性方面，本文的数据来源于拉勾网以及应届生求职网。本文通过 Scrapy 框架爬取职位详细信息并处理字体反爬机制。职位数据爬取工作流程如下图 3.1 所示。

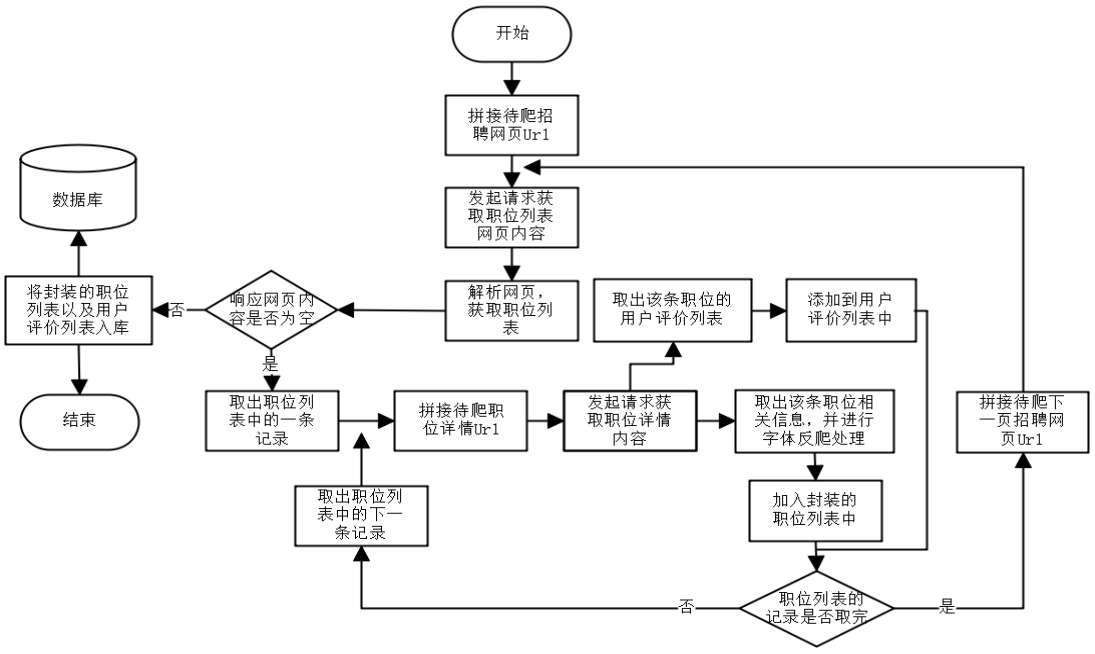

图 3.1 职位数据爬取工作流程图

本文在多个招聘网站的数据爬取工作流程和上述流程图所示步骤一致。首先，本文对招聘网站上职位检索页面的第一页开始发起请求并使用 BeautifulSoup 解析网页以获取职位列表，再依次遍历职位列表。然后利用每个职位的信息拼接出职位详情的请求链接，并发起职位详情信息的请求以获取职位详情信息。最后，对字体反爬进行处理之后再封装职位数据存入待存数据库的职位列表中。当发起职位招聘的检索的页面请求的响应内容为空时，则代表职位招聘的页数达到最大，那么爬虫程序的页面循环请求终止并将职位封装好的职位列表依次存入数据库。此外，在对获取职位详情内容的同时还需要爬取用户对职位的评价信息，以便后文利用用户对职位的反馈信息构建出用户与职位的评分矩阵并用处理后的评分矩阵推算出职位之间语义相似度。获取的各个用户对职位的评分数据将存于用户评价列表中，待页面循环请求终止后将用户评分列表存入数据库中。

针对拉勾网的网页有使用自定义字体导致爬取的数据部分中文和数字乱码且该自定义文件隔几天更新一次。因此，本文在爬取数据前，先根据 css3 的模块@font-face 指向的自定义字体文件路径下载字体文件并将 ttf 文件格式转成 xml 文件格式，然后处理 xml 文件格式的字体文件得出对应字体匹配的键值对字典，具体字体字典获取的转换处理如下图 3.2 所示。最后将爬取的数据对乱码字符进行字符替换。

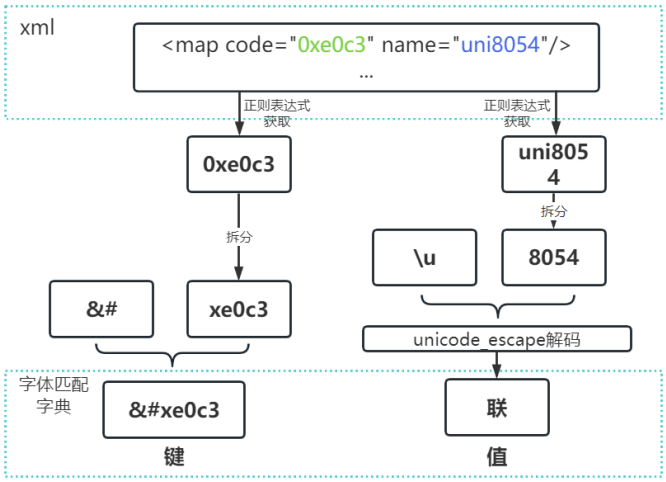

图 3.2 字体字典的转换处理图

爬取的招聘数据在存入数据库前部分字段需要进行预处理。若职位记录存在字段有异常值或大量字段缺省且无法修正，则采取丢弃该条职位记录的方式。具体字段修正规则如下表 3.1 所示。

表 3.1 字段修正规则表

字段	例子	修改后	说明
职位发布时间	2023-02-13 11:23:44	2023/02/13	规范成 yyyy/mm/dd 日期格式
薪资	8k/月	267	月工资换算成日薪，且只含纯数字
实习时长	6 个月、1 年	6、12	年换算成月，且只含纯数字
学历	缺省	不限	学历只包括不限、大专、本科、硕士
公司规模	100 以下、100-500 人、500 人以上	0、1、2	用数字代表公司规模的级别

3.2 职位信息的实体识别

针对上小节爬取地职位详细信息中职位要求为非结构化数据，本文接下来需要对该部分数据进行实体识别，提取出能力、性格、技能等实体。实体识别使用 BtBiCRF（全称为 Bert-BiLSTM-CRF）模型来进行训练。BtBiCRF 框架的结构如图 3.3 所示。第一步是利用 BERT 模型将职位要求的非结构化文本中的词转换为向量。然后，将转化的词向量输入到 BiLSTM 以捕获上下文特征。最后，根据上下文序列标记的相似性确定 CRF 层后的最终标签。

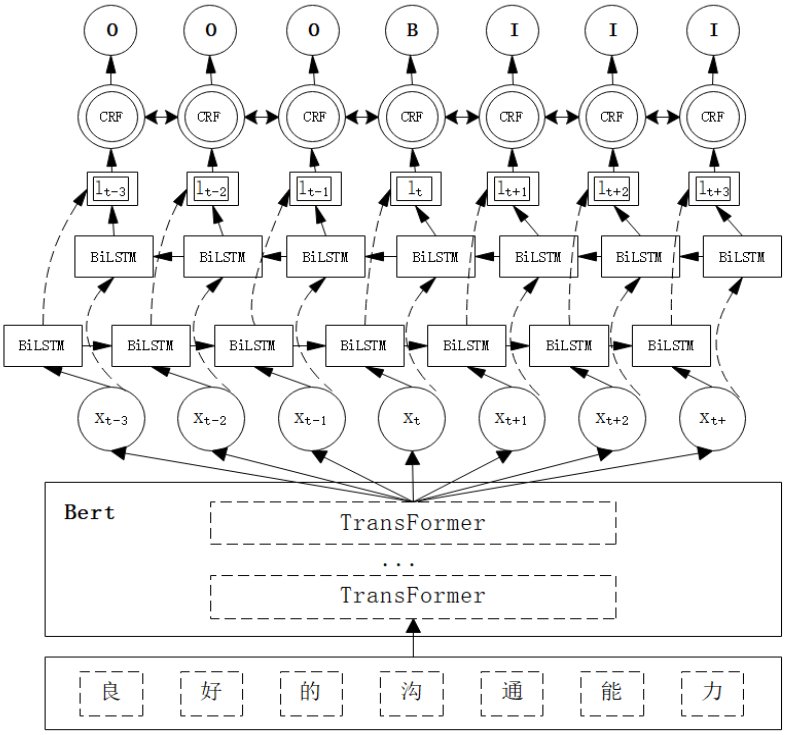

图 3.3 BtBiCRF 框架的结构图

BERT 中使用一种多层自注意机制，通过给实体增加权重来提高实体的语义信息。自注意力机制计算单词列表中每个单词与其他单词的关系，并根据这些词间关系调整每个单词与其他单词的关系权重，得出每个单词的词向量。因此，每个单词的向量不仅表示自身的语义偏向，还隐含了与其他单词之间的关系。自注意力计算公式如下所示。

$$attention(Q,K,V) = softmax(\frac{Q \cdot K^T}{\sqrt{dimension}}) \cdot V \tag{3.1}$$

其中Q表示词的自身含义向量；K表示词关键信息向量；V表示词的内容向量，$dimension$代表输入的词向量的维度，$softmax$为了进行归一化处理。

因为自注意力机制无法分辨单词在句子中的顺序，BERT 引入了位置编码和分隔编码来区分相邻的句子。在 BERT 中，输入序列中的每个单词都是通过词向量、位置向量和分隔向量相加得到的。最终的单词向量是通过深度双向模型训练生成的，然后输入到 BiLSTM 中学习上下文特征。

BiLSTM 进一步学习词在句子中的上下文特征，以获取词更具有解释性的语义信息。BiLSTM 针对信息的遗忘、更新和传递采取了记忆单元和阈值机制[46]。因此，BiLSTM 可以有效地解决神经网络训练中的梯度消失或梯度爆炸现象。LSTM 内部结构图如下图 3.4 所示。

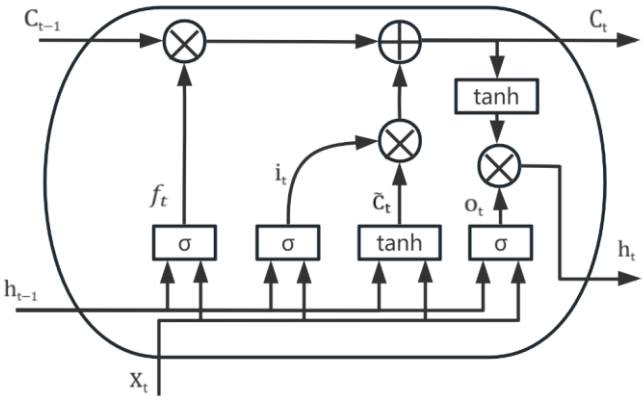

图 3.4 LSTM 内部结构图

LSTM 的记忆单元由遗忘门、输入门和输出门所控制。其中，遗忘门确定在细胞状态中哪些信息将被遗忘；输入门决定哪些新信息可以更新到细胞状态；输出门决定哪些信息部分会被输出。输入到下一处理层的记忆单元状态值的公式如下：

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{3.2}$$

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{3.3}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.5}$$

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \tag{3.6}$$

$$h_t = o_t * tanh(C_t) \tag{3.7}$$

其中σ为 sigmoid 激活函数，f、i和o分别表示遗忘门、输入门和输出门。W和b是权重矩阵和偏置项，C表示细胞状态，h是隐藏层状态。

上述 LSTM 的网络结构可以让其获取前面文本的语义信息，但却无法获取后面文本的语义信息。因此在这项工作中，获取前后文本的语义信息非常重要，因此我们使用了 BiLSTM 模型。BiLSTM 将 BERT 的输出向量作为输入，使用正向和反向的 LSTM 分别获取前面文本和后续文本中的隐藏语义信息，然后将这两部分信息拼接在一起作为最终的结果。

条件随机场（CRF）算法是该框架的最后一部分，用于捕捉上下文标签之间的依赖关系并加以约束。CRF 是一种条件概率分布模型，可以用$P(Y|X)$来表示。在这个任务中，采用了线性链条件随机场，其中X是输入变量，表示要标记的观察序列；Y是输出序列，表示一一对应于X的标签序列。它的核心公式如下：

$$p(y|x) \propto exp(\sum_{k=1}^{K} \omega_k f_k(y, x)) \tag{3.8}$$

其中，f是特征函数，ω是对应于特征函数的权重。

在训练阶段，通过使用训练集进行最大似然估计，得到条件概率模型$\hat{P} = (Y|X)$。在预测阶段，对于给定的观察序列，使用维特比算法输出具有最大条件概率$\hat{P} = (Y|X)$的标签序列Y。

3.2.1 实体识别实验数据集与环境介绍

本文使用的职位数据集来源于上小节使用爬取职位招聘网站的真实数据，且是职位要求的非结构化文本数据。原始职位要求数据集经过各领域的求职者共同进行人工标注。本文将实体划分为能力实体、性格实体、技能实体、专业实体。数据集采用 BIO 标注法，其中 B-XX 表示实体元素的开始处，I-XX 表示实体元素的中间或结尾，O 表示非实体元素。经过标注后的职位要求数据集共有 175560 个标注实体，其中能力实体有 95095 个，性格实体有 36572，技能实体有 24068 个，专业实体有 19825 个。具体标注类别如下表 3.2 所示，本文对打完标注的数

据集按照 7:2:1 的比例划分成实验训练时所需的数据、测试所需的数据以及验证所需的数据。

表 3.2 待预测的实体标注表

元素含义	开始标注	中间或结束标注	例子
能力实体	B-NL	I-NL	沟通能力、事业心、责任感等
性格实体	B-XG	I-XG	乐观、开朗等
技能实体	B-JN	I-JN	CSS、Photoshop、JAVA、Rhino 等
专业实体	B-ZY	I-ZY	软件工程、风景园林、环境设计等
非实体	O	O	标点符号或其他字词

本文实验进行的软硬件环境如下：

（1）处理器：AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

（2）内存：16G

（3）开发工具：PyCharm 2022.3.1 x64

（4）开发语言：Python3.6

（5）数据库：MySQL

（6）深度学习框架：tensorflow1.14.0、keras2.3.1

3.2.2 实体识别评价指标

本文使用召回率（Recall）、精确率（Precision）以及 F_1 值三个评价指标来对模型进行评估。召回率和精确率的取值在 0 到 1 之间。该召回率的含义是当前实体对于该模型下的识别概率，而精确率的含义是当前实体识别模型对某个实体标签正确识别的概率。召回率和精确率越接近 1 则模型实体识别性能越好，反之召回率和精确率越接近 0 则代表模型识别实体的效果差。F_1是召回率和精确率的加权平均值，综合了召回率和精确率的指标。三项指标的具体公式如下：

$$Recall = \frac{TP}{TP+FN} * 100\% \tag{3.9}$$

$$Precision = \frac{TP}{TP+FP} * 100\% \tag{3.10}$$

$$F_1 = \frac{2*Precision*Recall}{Precision+Recall} * 100\% \tag{3.11}$$

式子中TP表示模型正确识别的实体个数，FN表示模型错误地将当前标签下的实体识别成其他标签的实体或非实体的个数，FP表示模型错误地将其他标签的实体或非实体识别成当前标签下的实体个数。

3.2.3 实体识别结果与分析

为了对比不同实体识别模型的性能，本文使用了 BiLSTM 模型、CRF 模型、BiLSTM- CRF 模型以及 BtBiCRF 模型在使用本文相同的数据集来进行实验，以清晰对比实验结果。“能力”实体识别的相关实验结果如下列表 3.3 所示。

表 3.3 不同实体识别模型下“能力”实体实验结果表

模型	召回率	精确率	F_1 值
BiLSTM 模型	64.96%	71.93%	68.26%
CRF 模型	63.83%	71.82%	67.59%
BiLSTM- CRF 模型	65.21%	72.37%	68.60%
BtBiCRF 模型	66.02%	74.02%	69.79%

据上表所示，BtBiCRF 模型与 BiLSTM-CRF 模型相比在召回率、精确率以及 F1 值上都要略高，说明 BERT 模型的词向量的训练对实体识别的性能具有显著提高的效果。BiLSTM- CRF 模型同样三项评估指标比 BiLSTM 模型和 CRF 模型要高，说明 BiLSTM 模型进一步考量的上下文特征以及 CRF 模型增加训练数据的标签约束条件可以大大提高实体识别准确率。BtBiCRF 模型相比于 CRF 模型、BiLSTM 模型以及 BiLSTM- CRF 模型的实体识别在三项指标上都要高，说明 BtBiCRF 模型比其余实体识别模型要更优。

由 BtBiCRF 模型下的各实体 F_1 值的统计图 3.5 可知，能力、性格、专业实体的识别精度较高，但技能的实体识别较低。通过分析技能实体多数以英文出现，BtBiCRF 模型对英文的实体识别较差。且能力实体相比于性格和专业实体 F1 值要略高，经过分析能力实体多数的组成以“能力”结尾，因此其实体组成的特征性强。

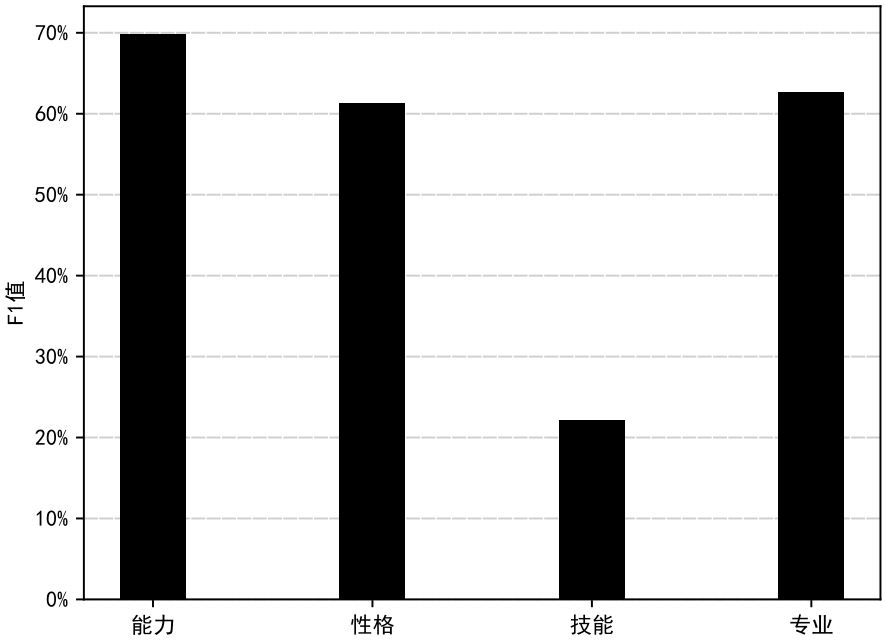

图 3.5 BtBiCRF 模型下各实体识别 F_1 值统计图

3.3 职位领域知识图谱存储

知识图谱用于记录实体以及实体之间的关系，而图数据库以图形化的方式更加直观的展现知识图谱中实体以及实体之间的关系。实体之间的关系可以是用作推荐的实体之间相似度，也可以是用于记录实体之间层次结构，例如包含关系、继承关系等。相比于关系型数据库，作为非关系型的图数据库可以在海量的知识图库中做复杂的多实体关联分析，且还能以图形结构的方式清晰观察出实体之间的间接关系。

本文利用 Neo4j 图数据库来存储前面章节处理后的结构化职位领域数据。求职领域的节点主要有职位名称、最高日薪资、最低日薪资、职位所在城市、所属公司、职位类别、职位要求等。

本文需要将 MySQL 临时表中的职位数据与职位要求进行实体识别后的文本数据按照顺序对应一并导入到 Neo4j 中。具体的过程如下：

（1）读取 MySQL 的职位临时表，逐条遍历。遍历一条记录的同时按次序从一段处理好的实体识别描述文本提取实体。

（2）将职位名称作为头节点，若数据库中不包含该头节点则创建该节点，若存在则取出该头节点。

（3）获取最高日薪资、最低日薪资、职位所在城市、所属公司、职位类别、职位要求作为尾节点。查找数据库中（2）步实体的关系是否存在对应的该尾节点，若不存在且尾结点不存在则创建该尾节点并创建头节点指向尾节点的关系，若不存在但尾结点存在则创建头节点指向尾节点的关系。

（4）重复以上（1）至（3）步，直到将 MySQL 以及已处理的实体识别文本读取完毕并组建好三元组并插入到图数据库中。

通过上述的操作，最终构造的三元组全部导入至图数据库进行可视化存储。本章通过对职位要求的实体抽取以及构造出三元组完成职位知识图谱的构建。下图 3.6 展示出构建的职位知识图谱的局部图。

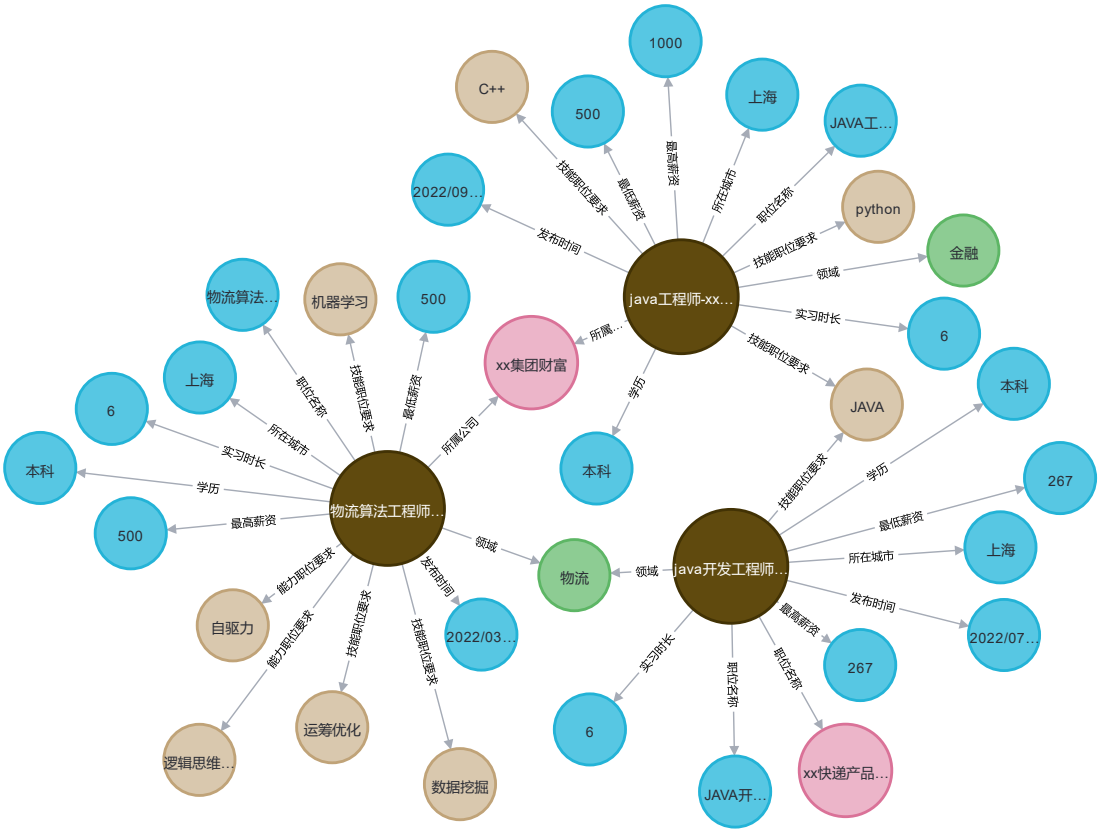

图 3.6 职位知识图谱局部图

3.4 本章小结

本章首先利用爬虫技术进行数据采集，然后将采集后的数据进行相关预处理。针对职位要求非结构化数据本章对该部分数据进行实体识别，并对该部分数据进行人工标注。本章还在该人工标注的数据前提下使用相关模型进行试验和分析试验结果。最后，本章将 MySQL 中临时表中的职位数据和已经过实体识别的职位要求数据合并存储到图数据库中，成功构建出职位知识图谱，便于后文对职位的相关推荐工作。

第 4 章 基于知识图谱的职位推荐算法

针对基于内容的职位推荐算法存在数据稀疏而导致推荐效果差的问题，本章提出一种基于知识图谱的混合式职位推荐算法，使用时间衰减函数和隐语义模型对用户的评分数据进行调整和补充，并融合求职者简历与职位静态语义信息以及用户对职位评分的动态语义信息对用户的评分数据进行评分预测，进而提高了职位推荐的准确度。

4.1 基于知识图谱的混合式职位推荐算法

本文在上一章已成功构建了职位知识图谱，本章提出在已构建的职位知识图谱的基础上考虑时间衰减因子且使用多矩阵融合的方式给用户所需职位进行推荐。在职位推荐算法上本文解决了新建用户导致的冷启动问题以及职位多且用户评分数据稀疏问题。本文提出职位推荐算法的工作流程如下图 4.1 所示。

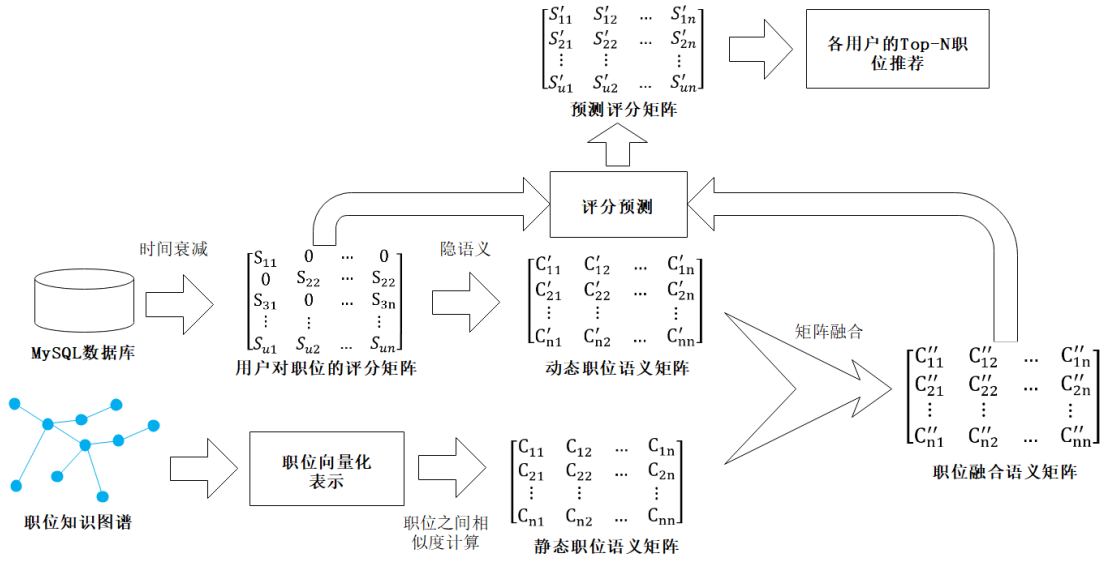

图 4.1 职位推荐算法的工作流程

本文将知识图谱中的职位实体与个人简历进行向量化表示，并计算出个人简历与职位的相似度矩阵。若用户是新创建用户且无用户行为数据，本文利用个人简历数据与职位数据训练出职位的维度向量并计算出个人简历的相似度矩阵，

通过个人简历的相似度矩阵寻找新建用户的相似用户并进行相似用户的 Top-N 职位推荐。若用户具有对职位的评分行为，本文将用户与职位的评分矩阵以及个人简历与职位的评分矩阵分别计算出不同参考状况下职位与职位的语义矩阵，再使用分配不同权重的方式将两语义矩阵进行融合，最后通过融合的语义矩阵对用户打分的职位进行评分预测并进行 Top-N 职位推荐。此外，本文考虑时间因素对职位打分的影响，规整用户与职位的评分矩阵。当时间达到给定的阈值，则用户给职位的打分退化成 0。当用户历史打分全部越过时间阈值，本文将以新建用户的情况进行 Top-N 职位推荐。

4.1.1 静态语义矩阵计算

本小节主要对知识图谱的职位实体相关数据以及个人简历数据进行训练并得出职位的向量化表征，再用职位的稠密向量构建出职位之间的相似语义矩阵。因此，本小节分成职位稠密向量计算和静态相似语义矩阵计算两步。

（1）职位稠密向量计算

TransE 不断调整知识图谱的三元组中头实体、关系以及尾实体的向量化值，即公式化表示$head + relation \approx tail$，使头实体与关系的向量和尽量逼近尾实体的向量，从而逐渐优化模型的性能以得到更加准确的实体向量。其目标函数如下：

$$S = \sum_{<h,r,t>\in G} \sum_{<h',r',t'>\in G'} [\gamma + f(h,r,t) - f(h',r',t')]_+ \tag{4.1}$$

$$G' = \{< h',r,t > | h' \in E \wedge < h',r,t > \notin G\} \cup \{< h,r,t' > | t' \in E \wedge < h,r,t' > \notin G\} \tag{4.2}$$

$$f(h,r,t) = ||h + r - t|| \tag{4.3}$$

$$[x]_+ = \begin{cases} x & ,x \geq 0 \\ 0 & ,x < 0 \end{cases} \tag{4.4}$$

其中，G表示职位知识图谱中的正三元组样本集合，G'表示知识图谱中的负三元组样本集合，此三元组在训练中修改头实体或尾实体来构建的且在原有的职位知识图谱中不存在。γ是用于控制实体和关系向量之间间隔距离大小的一个参数，需要在训练过程中反复调整。

TransE 算法如算法 1 所示经过训练和优化可以使损失函数的值最小化，以此来查看得分函数最终学习的效果。它使用连续迭代的方法升级主要的参数，直到

达到收敛抑或是已经到迭代的最大次数为止，这时知识图谱中的实体就会被映射到向量空间中的相应位置。

TransE 的具体训练步骤如下：

算法 4.1：TransE 训练算法

输入：知识图谱中的三元组集合 G={<h, r, t>}，实体集合 E，关系集合 R，向量的维度 d

输出：实体向量

1：**initialize**：r ← uniform(-6/sqrt(d), 6/sqrt(d)) for each r∈R

2：e ← uniform(-6/sqrt(d), 6/sqrt(d)) for each e∈E

// random sampling from a specified interval

3：r ← r/||r|| for each r∈R // vector normalization

4：**loop**：

5：e ← e/||e|| for each e∈E

6：G_{batch} ← sample(G，b) // take b samples from G

7：T_{batch} ← ∅

8：**for** <h，r，t>∈G_{batch} **do**

9：<h'，r，t'> ← sample($S'_{<h,r,t>}$)

10：T_{batch} ← T_{batch} ∪ {(<h，r，t>,<h'，r，t'>)}

11：**end for**

12: Update embeddings w.r.t

$$\sum_{(<h,r,t>,<h\prime,r,t\prime>)\in T_{batch}} \nabla [\gamma + d(h+r,t) - d(h'+r,t')]_+$$

13：**end loop**

14：**return** 实体向量

（2）静态语义矩阵计算

在上一步中，本文将职位知识图谱中职位实体进行向量输出。下面将根据稠密的职位向量详细计算出职位之间的静态相似语义矩阵。

在计算相似度时，主流计算方法有欧式距离计算、余弦计算、皮尔逊相关系数计算等。其中余弦相似度与其他度量方法相比，侧重考虑的是向量之间的夹角而非向量模长的大小，不会因为向量的维度较高而导致相似度计算结果不准确。

职位之间相似程度更加侧重职位向量的指向，因此计算职位之间的语义相似度采用余弦相似度计算的方式。职位之间相似度计算公式如下所示。

$$P_i = [V_{i1}, V_{i2}, \ldots, V_{id}]^T \tag{4.5}$$

$$C(P_i, P_j) = \frac{P_i \cdot P_j}{\|P_i\|^2 \times \|P_j\|^2} = \frac{\sum_{k=1}^{d} V_{ik} V_{jk}}{\sqrt{\sum_{k=1}^{d} V_{ik}{}^2} \sqrt{\sum_{k=1}^{d} V_{jk}{}^2}} \tag{4.6}$$

其中，P_i表示第i职位的d维度向量，V_{in}为第i职位的第n维的向量值。$C(P_i, P_j)$表示第i职位和第j职位的余弦相似度。

上式余弦结果值越大，则代表两职位向量之间夹角越小，那么就表明职位静态语义相似度越大。如果余弦结果值等于 0，可以认为两职位相似度很小。反之，如果余弦结果值趋于 1，则代表两职位相似度极为接近。

最后，通过两两职位向量余弦相似度计算，可以构建出职位的静态相似语义矩阵。公式如下所示：

$$C_{p\times p} = \begin{bmatrix} C(P_1, P_1) & C(P_1, P_2) & \cdots & C(P_1, P_p) \\ C(P_2, P_1) & C(P_2, P_2) & \cdots & C(P_2, P_p) \\ \vdots & \vdots & & \vdots \\ C(P_p, P_1) & C(P_p, P_2) & \cdots & C(P_p, P_p) \end{bmatrix} \tag{4.7}$$

4.1.2 动态语义矩阵计算

本小节将通过用户对职位的评分数据构建出职位之间的相似语义矩阵。本小节先对用户给职位的评分增加时间衰减因子并构建出增加时间权重的评分矩阵，然后利用隐语义模型补充用户与职位的评分矩阵。最后，本小节利用处理后的用户与职位的评分矩阵计算出职位之间的动态相似语义矩阵。此动态相似语义矩阵以及上小节计算出的静态相似语义矩阵融合之后用于后文通过评分函数做职位的 Top-N 推荐，这种利用职位之间的相似语义矩阵做 Top-N 推荐属于基于物品的协同过滤算法的范畴。

综上，本小节分成构建加权评分矩阵、隐语义模型补充评分矩阵和相似语义矩阵计算三步。

（1）构建加权评分矩阵

针对用户已对职位的评分可能受其他新职位的影响，本文对职位评分矩阵增加时间衰减的权重，即随着时间的推移降低用户已对职位所打分值。本文使用的指数级时间衰减函数，具体公式如下：

$$w(\Delta t)=\begin{cases}e^{-\lambda\Delta t}, & \Delta t<90\\ 0, & \Delta t\geq 90\end{cases} \tag{4.8}$$

其中，w表示用户给职位给出分值的权重函数，Δt表示用户给职位打出评分时间与当前时间的时间差（单位为日），λ为衰减影响因子（本文取值为 1/300）。

那么u个用户分别对p个职位的评分矩阵$R_{u\times p}$通过加权之后的评分矩阵$R'_{u\times p}$可由下公式得出。

$$R'_{u\times p}=\begin{bmatrix}w(\Delta t_{11})R_{11} & w(\Delta t_{12})R_{12} & \cdots & w(\Delta t_{1p})R_{1p}\\ w(\Delta t_{21})R_{21} & w(\Delta t_{22})R_{22} & \cdots & w(\Delta t_{2p})R_{2p}\\ \vdots & \vdots & & \vdots\\ w(\Delta t_{u1})R_{u1} & w(\Delta t_{u1})R_{u2} & \cdots & w(\Delta t_{up})R_{up}\end{bmatrix} \tag{4.9}$$

（2）隐语义模型补充评分矩阵

用户对职位的评分矩阵是一个稀疏矩阵，如果使用此稀疏矩阵进行后续的职位推荐，由于最终的推荐结果并未考虑用户与职位中间层的隐含特性，而导致推荐结果的准确率下降。因此，此步骤通过隐语义模型计算出用户与隐含特性以及隐含特性与职位的关系，然后反求预测的职位评分矩阵并对原职位评分矩阵进行补充。

隐语义模型主要将用户给职位评分的矩阵分解成用户潜在特征矩阵和职位潜在特征矩阵，且两个矩阵的特征列（或特征行）都是低维的。计算公式如下所示。

$$\hat{R}'_{u\times p}=P_{u\times k}\cdot Q_{k\times p} \tag{4.10}$$

$$L=\sum_{i=1}^{u}\sum_{j=1}^{p}(r'_{ij}-p_iq_j)^2+\lambda(\|p_i\|^2+\|q_j\|^2) \tag{4.11}$$

其中，$\hat{R}'_{u\times p}$为职位的预测评分矩阵，$P_{u\times k}$和$Q_{k\times p}$为分解的用户潜在特征矩阵和职位潜在特征矩阵，L为隐语义模型的损失函数，p_i为矩阵$P_{u\times k}$的i行向量，q_j为矩阵$Q_{k\times p}$的j列向量，λ为正则化系数[47]。正则化项$(\sum_{n=1}^{k}p_{in}+\sum_{n=1}^{p}q_{ni})$为了防止模型过度贴合训练数据而实际测试数据表现较差，以提高模型的泛化能力。

使用梯度下降法不断优化参数进行迭代，以使得损失函数尽可能的小。因此，分别对损失函数的p_i、q_j求导，得到的迭代公式如下：

$$p_i=p_i+\alpha((r'_{ij}-p_iq_j)q_j^T-\lambda p_i) \tag{4.12}$$

$$q_j=q_j+\alpha((r'_{ij}-p_iq_j)p_i^T-\lambda q_j) \tag{4.13}$$

其中，学习速率α是一个重要的超参数，其大小决定了梯度下降算法的迭代速度，即α越大，算法学习的速度就越快。为了得到一个合适的α值，通常需要进行多次试验和调整。

通过隐语义模型的不断迭代，本文最终训练出用户潜在特征矩阵和职位潜在特征矩阵相乘的预测职位评分矩阵，最后利用预测职位评分矩阵将原稀疏的职位评分矩阵中的值为 0 的项进行对应替换。记下图所示的矩阵为补充职位评分矩阵$R''_{\mathrm{u\times p}}$。

（3）相似语义矩阵计算

对于职位的评分矩阵侧重考量职位评分向量的具体数值，所以本文使用欧式距离公式来计算出职位之间的相似度。下公式中P_i表示第i职位的u维度列评分向量。$d(P_i,P_j)$表示职位向量之间的距离公式，以计算出职位之间的相似值。

$$P_i = [R''_{i1}, R''_{i2}, \ldots, R''_{ip}]^T \tag{4.14}$$

$$d(P_i,P_j) = \sqrt{\sum_{k=1}^{u}(R''_{ik} - R''_{jk})^2} \tag{4.15}$$

再将职位之间的向量值进行归一化处理，最终构建出职位之间的相似语义矩阵。具体计算公式如下：

$$sim(d(P_i,P_j)) = \frac{1}{1+d(P_i,P_j)} \tag{4.16}$$

$$C'_{p\times p} = \begin{bmatrix} sim(d(P_1,P_1)) & sim(d(P_1,P_2)) & \ldots & sim(d(P_1,P_p)) \\ sim(d(P_2,P_1)) & sim(d(P_2,P_2)) & \ldots & sim(d(P_2,P_p)) \\ \vdots & \vdots & & \vdots \\ sim(d(P_p,P_1)) & sim(d(P_p,P_2)) & \ldots & sim(d(P_p,P_p)) \end{bmatrix} \tag{4.17}$$

4.1.3 语义矩阵融合

本小节将上面两小节计算出的职位之间的静态语义矩阵和动态语义矩阵进行融合，生成的矩阵既考量用户简历中的静态特征与职位属性特征关联关系，也考虑到过往用户对职位的评分情况。因此，融合的语义矩阵给出的职位之间的相似关系更具有解释性。这里使用的是融合函数如下：

$$f(c_1,c_2) = \frac{1}{1+e^{\alpha}}c_1 + \frac{e^{\alpha}}{1+e^{\alpha}}c_2 \tag{4.18}$$

这里α代表融合参数，它的区间在$(-\infty,+\infty)$。α趋近于正无穷则表示f的值只考量c_2的数值大小；α趋于负无穷则表示f值只考量c_1的数值大小。应用于职位之

间的相似语义矩阵融合，α值约大则职位之间的相似值偏向于动态语义的相似值，反之，α值约小则职位之间的相似值偏向于静态语义的相似值。

4.2 职位推荐列表的生成

接下来通过融合的语义矩阵，进行用户的未评分的职位进行预测打分。职位的预测打分公式如下：

$$s_{ui} = \frac{\sum_{j \in M} \hat{c}_{ij} \times R'_{ui}}{\sum_{j \in M} \hat{c}_{ij}} \tag{4.19}$$

其中，s_{ui}表示用户u对第i职位的预测打分。M表示用户已对职位打分的职位编号集合。根据上述公式计算出的职位预测评分值与用户对其他职位的打分以及自身简历特征与该职位特征有着紧密的关系，预测打分的值较高的职位会在系统的职位推荐列表中，且分值越高的职位会被优先排列在越靠前的位置。

4.3 实验与结果分析

这里帮助求职者进行偏好职位推荐的基本思想是，首先根据前面小节计算获取到的简历信息与职位之间生成的相似度矩阵和根据求职者对于职位的用户行为生成的矩阵，通过估算每个求职者给未评价过的职位内容的打分数值，以此根据求职者对于职位内容的预估分值在系统中生成自带排序的职位列表。

由于融合参数的取值范围是$(-\infty, +\infty)$，考虑两矩阵权重在(0.2, 0.8)区间内，则本文实验的融合参数取值样本为{-1.2, -1, -0.8, -0.6, -0.4, -0.2, 0 , 0.2, 0.4, 0.6, 0.8, 1, 1.2}。然后观察混合职位推荐算法与本文职位推荐算法不同融合因子下准确率指标的变化情况。若融合参数越大，则基于用户行为动态信息的语义在推荐中的权重越大；若融合参数越小，则基于用户与职位静态信息的语义在推荐中的权重越大；若融合参数为 0，则代表基于用户行为动态信息的语义以及基于用户与职位静态信息的语义权重各占一半。

如下图 4.2 所示两种职位推荐算法在不同融合度参数下准确率指标变化情况。通过下图观察，融合因子在-0.2 时两种职位推荐算法的推荐准确率均表示最佳，即代表基于用户行为动态信息的语义权重约占 0.55 时推荐效果在准确率指标下表现最佳。且本文推荐算法相较于基于内容的职位推荐算法在准确率最高峰时

提升了接近 2%。此外不难发现。随着融合因子的不断增大，其两种职位推荐算法在准确率下的间隔不断拉大，这表明时间衰减用户对职位的评分影响着用户对职位的喜爱程度。

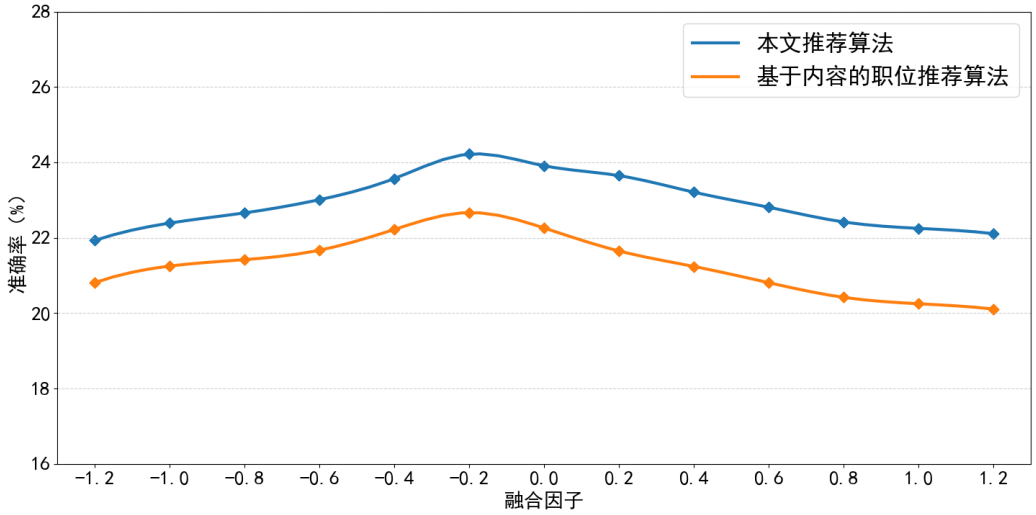

图 4.2 不同算法且各融合因子下的准确率

4.4 本章小结

本章使用上章构建的职位领域知识图谱进行职位推荐，首先利用职位知识图谱中的实体与关系进行职位实体的向量化表示，以计算出职位之间的语义关系并得出职位的静态语义矩阵。然后，本文利用用户给职位评分的数据同样计算出职位之间的语义关系并得出动态语义矩阵，但考虑到用户给职位的评分受新旧职位的影响以及职位的评分数据稀疏，本文使用时间衰减函数以及隐语义模型分别对应解决相关问题。最后，本文将职位的静态语义矩阵和动态语义矩阵使用融合函数进行矩阵融合并调整融合函数的参数来进行实验，以及分析不同参数下的融合产生的不同推荐效果的实验结果分析。

第 5 章 系统分析与设计

本章主要对职位推荐系统进行系统分析，分析不同用户的功能模块、系统的数据流向分析、数据库的设计以及系统架构设计。然后实现不同用户的功能模块，并做出推荐系统的功能测试。

5.1 需求分析

5.1.1 功能分析

本文的职位推荐系统为了给求职者提供寻找职位更加迅速的招聘平台。系统可以根据求职者的显式简历数据和隐式职位评分数据推荐求职者可能感兴趣的职位。系统的用户包括求职用户和招聘用户。招聘用户可以为企业发布招聘的职位，可以对发布的职位进行管理。而求职用户可分为首次创建的求职用户和现有的求职老用户。对于首次创建的求职用户，本文的推荐利用用户创建时的简历信息与职位信息的相关联系从而进行推荐；对于现有的求职老用户，本文同时考虑简历信息和评论职位的相关数据并推出用户查找职位的偏向并进行推荐。本文主要为求职者求职过滤匹配度较低的职位信息，减少求职用户筛选职位时间。

5.1.2 用例分析

系统的使用角色有求职用户、招聘用户以及后台管理员。以下分别对这两种用户进行用例分析：

（1）求职用户通过系统的注册页面进行账号注册，然后用户可以使用成功注册的账号在系统进行登陆。首次成功登陆后的求职用户需要对填写简历信息，然后才能进入系统。求职用户在系统中可以浏览企业发布的职位信息，并且求职用户可以根据自己的符合程度对职位发起评论以及评分。其中，求职用户在系统首页可以查看到系统为求职用户的推荐职位列表，也可以在主页按条件检索相

关职位。除此之外，求职用户可以查看以及修改个人信息。下图 5.1 为求职用户的用例图。

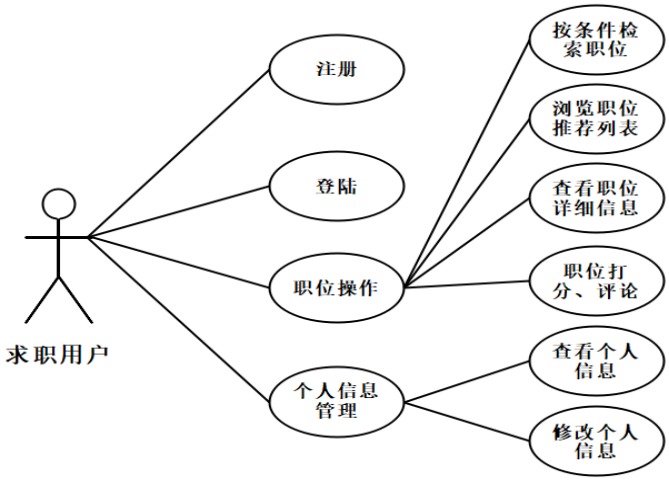

图 5.1 求职用户用例图

（2）招聘用户同样具有和求职用户一样的注册、登陆流程。但与求职用户不同的是招聘用户的账号需要后台管理员审核，只有审核通过的招聘用户账号才可以对系统进行登陆。此外，招聘用户可以代表企业发布公司招聘岗位信息，同时可以删除职位、修改职位信息以及改变职位发布状态。对于求职用户对该职位的评论以及打分情况，招聘用户也可以查看。下图 5.2 为招聘用户的用例图。

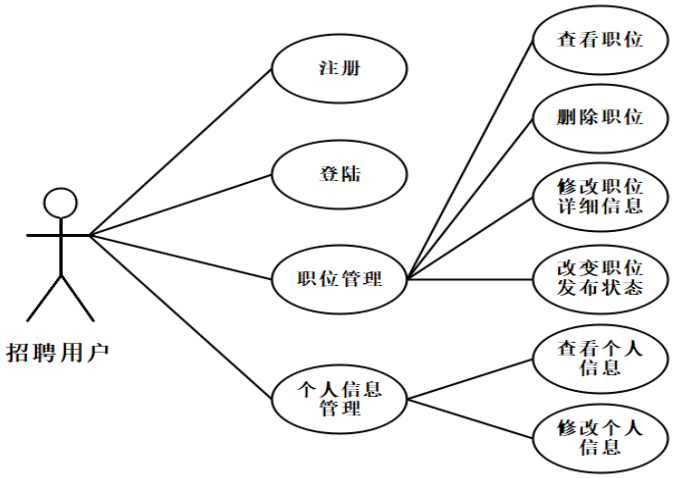

图 5.2 招聘用户用例图

（3）后台管理员有角色管理、账号管理以及职位管理的操作。对于角色管理，管理员登录后台系统后可以创建角色、删除角色以及分配角色的菜单权限，

创建的角色用于后台管理员给下级管理员分配权限角色使用的。后台管理员还可以创建、删除下级管理员账号，也可以删除用户账号以及审核招聘用户账号的申请。此外，后台管理员还可以删除不当职位以及不当言论的职位评论的功能。下图 5.3 为后台管理员的用例图。

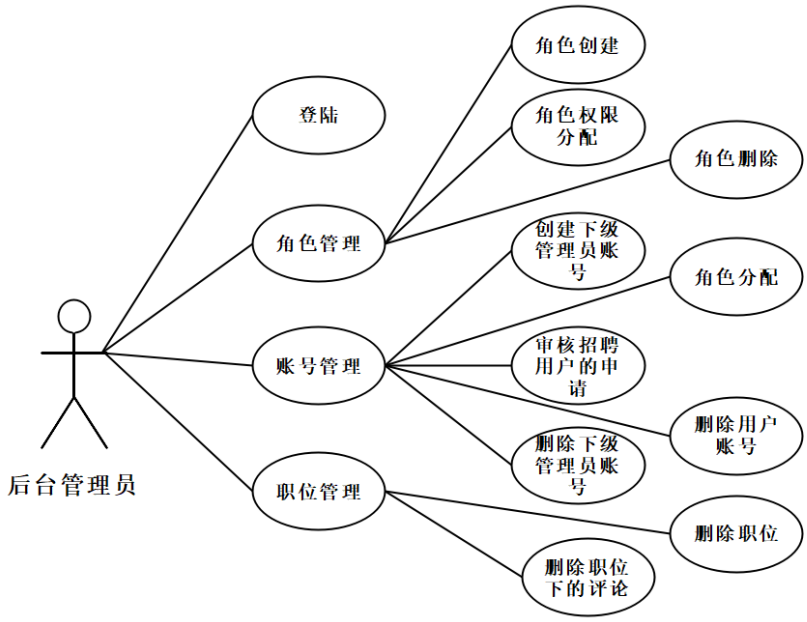

图 5.3 后台管理员用例图

5.1.3 数据流分析

职位招聘系统的数据流入主要是求职用户注册后提供简历数据和对职位反馈数据以及招聘用户提供的职位数据。这两部分数据会流入推荐引擎，然后推荐引擎利用这些数据产生用户的职位推荐列表，最后由系统将职位推荐列表数据流出到求职用户。另一部分数据流入是后台管理员。后台管理员可以创建角色、分配角色权限、下级管理员账号创建等操作，其中职位评论与招聘职位信息分别由求职用户和招聘用户流入的。职位推荐系统的数据流图如图 5.4 所示。

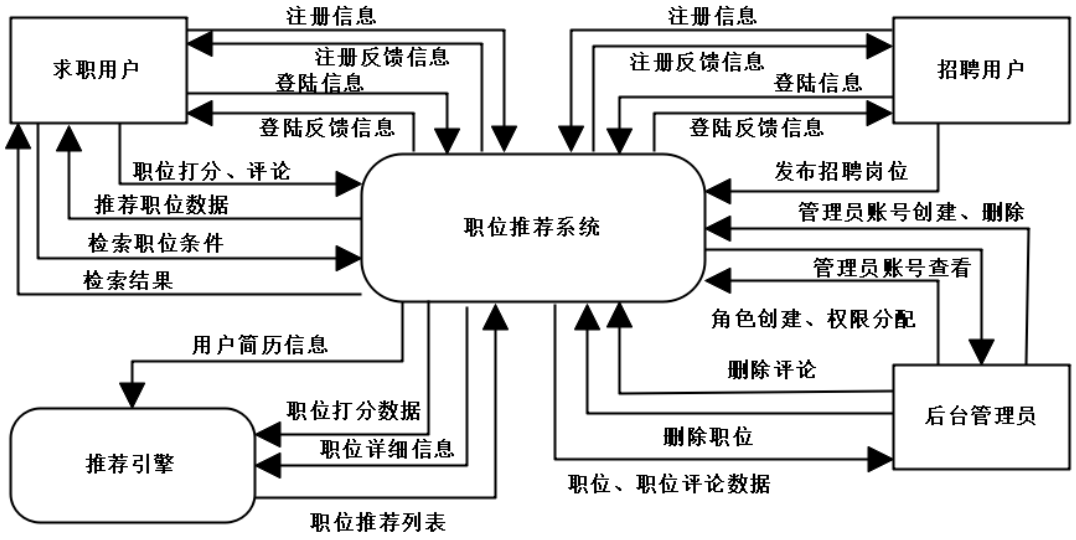

图 5.4 职位推荐系统数据流图

5.2 概要设计

5.2.1 系统架构设计

系统架构是指将系统划分成多个组件，并规定它们之间的协作方式来实现系统的功能和目标[48]。系统架构的特点在于，将软件分成多个水平层，每层都有特定的职责和功能，通过接口进行通信，可以使得各个层之间分工明确、依赖关系清晰，从而实现系统功能的高效实现。此外，分层还可以限制子系统间的依赖关系，使系统更加低耦合[49]，便于维护和开发。整个系统架构更为简单明了，为开发和维护工作提供了便利。本文的职位推荐系统的架构分为三个层次：表现层、业务层和持久层。表现层负责把后台渲染后的数据展现给用户以及管理员，而用户或管理员通过发起 HTTP 请求来获取后台的渲染数据并与后台进行数据交互。职位推荐系统的表现层采用 Vue.js 来显示页面数据。业务层处于职位推荐系统的重要位置，代表着职位推荐系统处理数据的逻辑。业务层根据不同的客户端请求从而执行不同的业务逻辑并响应结果给客户端[50]。职位推荐系统的业务层采用 Spring Boot、MyBatis-Plus[51]技术来实现表现层与持久层的中间业务处理层。持久层主要将用户和管理员产生的数据进行落盘。职位推荐系统的职位推荐列

表数据等热点数据将会存放于 Redis 中，而职位信息、个人信息等查询量较低的数据会存放于 MySQL 中。具体的职位推荐系统架构情况如下图 5.5 所示。

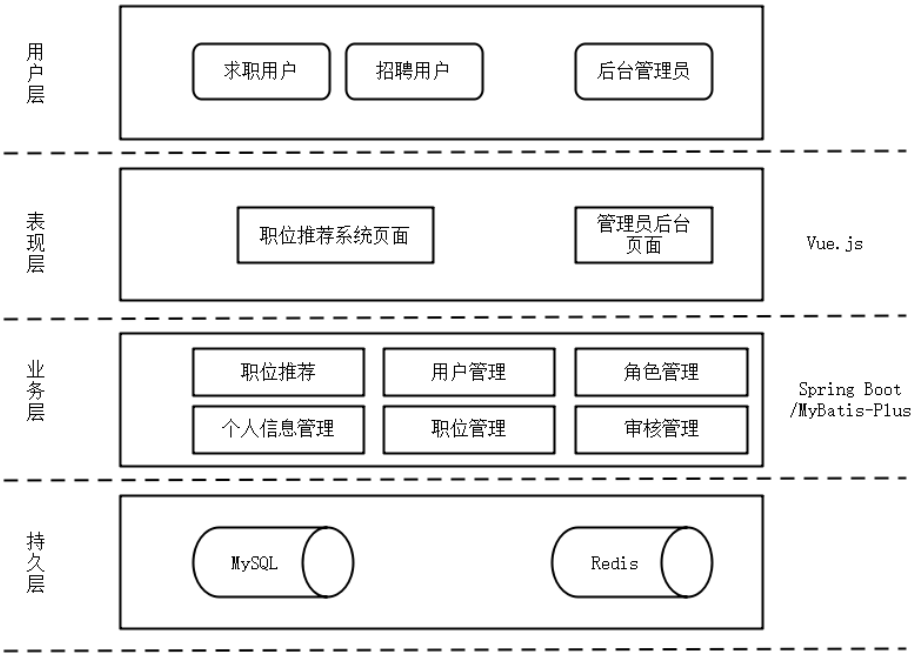

图 5.5 职位推荐系统架构图

5.2.2 模块结构设计

根据职位推荐系统的功能以及用例的分析，本文把职位推荐系统划分成五大重要功能模块，模块分别是求职用户模块、招聘用户模块、推荐模块、职位模块以及后台管理员模块。具体各模块的功能如下图 5.6 所示。

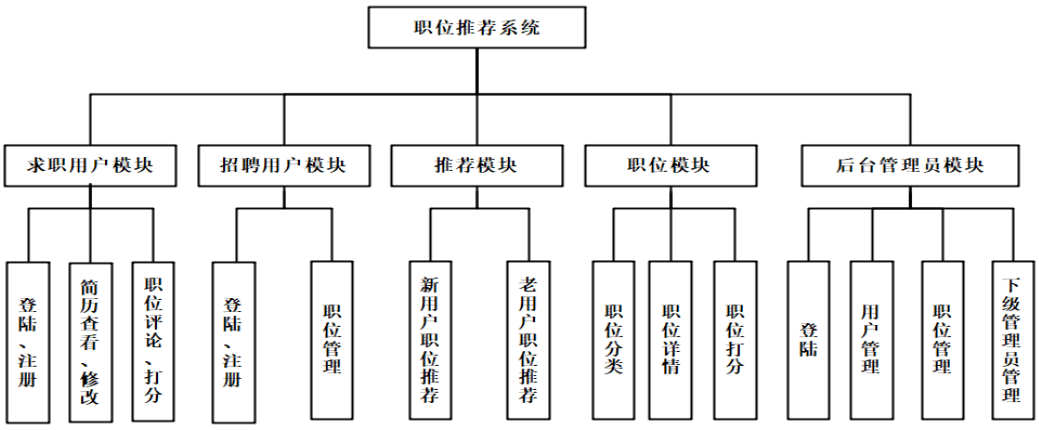

图 5.6 职位推荐系统模块功能图

（1）求职用户有登陆、注册以及个人简历查看与修改等功能。求职用户的主页可以用来浏览职位相关信息，点击其中的职位可以查看职位的详细信息以及对该职位的评论和打分。

（2）招聘用户模块包含招聘用户登陆、系统账号申请注册以及职位管理功能。对于招聘用户的账号需要提交账号注册的申请。其中职位管理主要是发布职位信息、改变职位发布状态以及职位的删除和修改。

（3）推荐模块对于无打分且无评论记录的新用户主要推荐用户的简历信息与职位信息更为相似的职位。而对于已具用户行为的老用户本推荐模块先利用时间衰减将用户评分进行规整并推出动态职位语义矩阵，然后融合用户静态职位语义矩阵和动态职位语义矩阵并预测其余未打分的职位评分，最后再利用预测的职位评分进行职位推荐。

（4）职位模块主要有展示各大招聘的职位信息、职位的打分情况等功能，用户还可以根据职位的所在城市、领域、职位名称关键字等来搜索职位。

（5）后台管理员主要是用户管理、职位管理以及下级管理员管理。用户管理是对招聘用户注册申请审核、用户账号的查看、删除等功能；职位管理是职位删除、职位评论删除等功能；下级管理员管理是下级管理员账号删除、角色分配等功能。

5.2.3 数据库 E-R 图

根据推荐系统的功能以及用例的分析，本小节画出 E-R 图来展示系统中各实体之间的关系，如下图 5.7 所示。

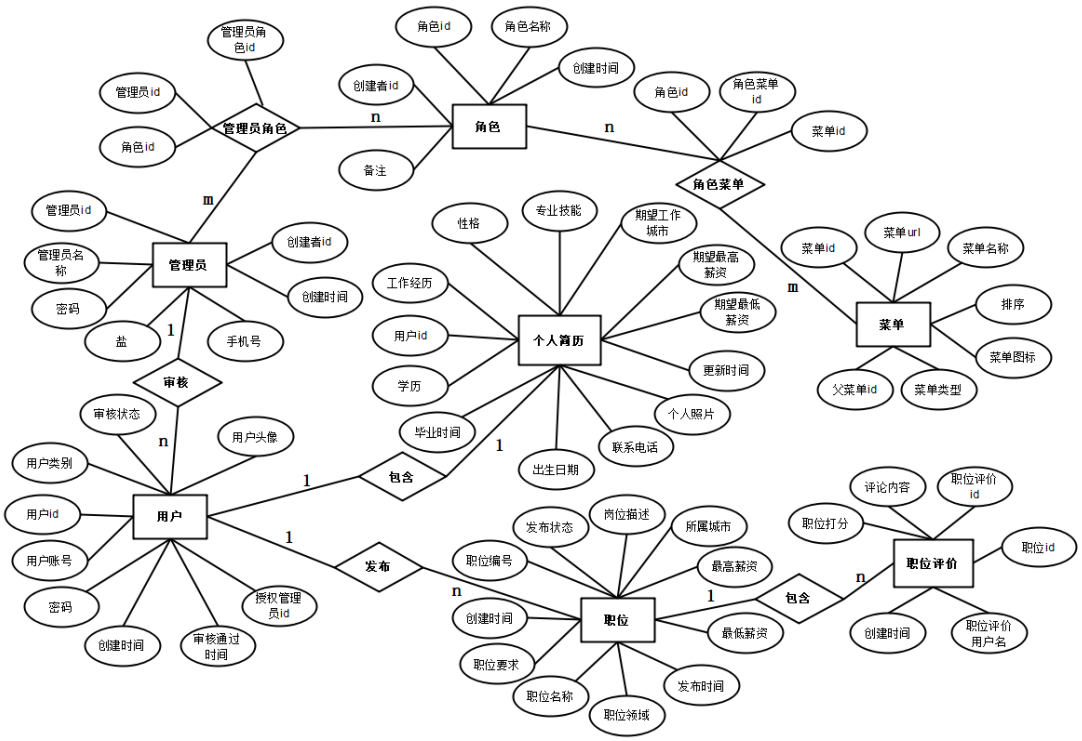

图 5.7 职位推荐系统的 E-R 图

5.2.4 求职数据库表设计

在进行数据库表设计前，本文需要规范数据库中命名规则。数据库中命名规范使命名风格一致有利于不同开发人员共同协作开发，以及后期维护见字知其意。本职位推荐系统的数据库命名规范如下：

（1）表名命名规则

① 统一采用 jw_为表名前缀，如用户表 jw_user；

② 全小写英文单词组成，多个单词之间使用“_”分隔；

③ 表名长度控制在 20 个字母以内。

（2）字段命名规则

① 全小写英文单词组成，多个单词之间使用“_”分隔；

② 单词过长取单词缩写；

③ 字段名长度控制在 20 个字母以内。

根据上小节的实体关系图，现给出职位推荐系统的数据库的表结构设计。用户表用于存放求职用户以及招聘用户的账户信息。具体字段信息如下表 5.2 所示：

表 5.2 求职系统用户表

字段名	类型	是否可空	约束	备注
user_id	int	N	P	用户编号
admin_id	int	N	F	授权管理员编号
user_name	varchar(20)	N		用户名称
user_ account	varchar(20)	N		用户账号
user_pwd	varchar(64)	N		用户密码
create_time	datetime	N		创建时间
pass_time	datetime	N		审核通过时间
user_avatar_src	varchar(255)	N		用户头像地址
audit_status	tinyint	N		审核状态，1 通过 0 未通过
user_type	tinyint	N		用户类别，1 招聘用户 0 求职用户

个人简历表用于存放求职用户的个人简历信息。具体字段信息如下表 5.3 所示：

表 5.3 职位推荐系统个人简历表

字段名	类型	是否可空	约束	备注
resume_id	int	N	P	简历编号
user_id	int	N	F	用户编号
phone	varchar(20)	N		联系电话
highest_degree	varchar(10)	N		最高学历
birthday	date	N		出生日期
resume_picture_src	varchar(255)	N		简历照片地址
update_time	datetime	N		更新时间
min_expec_salary	int	N		期望最低薪资

（续）表 5.3 职位推荐系统个人简历表

字段名	类型	是否可空	约束	备注
max_expec_salary	int	N		期望最高薪资
work_city	varchar(20)	N		期望工作城市
skills	varchar(255)	N		专业技能
personality	varchar(255)	N		性格
business_experience	varchar(255)	N		工作经历
education_experience	varchar(255)	N		教育经历

职位表用于存放招聘用户发布的职位信息。具体字段信息如下表 5.4 所示：

表 5.4 职位推荐系统职位信息表

字段名	类型	是否可空	约束	备注
position_id	int	N	P	职位编号
user_id	int	N	F	用户编号
create_time	datetime	N		创建时间
position_requirements	varchar(255)	N		职位要求
position_name	varchar(64)	N		职位名称
resume_picture_src	varchar(255)	N		简历照片地址
position _area	varchar(20)	N		职位领域
publish_time	datetime	N		发布时间
max_salary	int	N		最高薪资
min_salary	int	N		最低薪资
city	varchar(20)	N		所在城市
position_description	varchar(255)	N		岗位描述
publish_status	tinyint	N		发布状态，0 未发布 1 已发布

职位评价表用于存放对应职位的求职用户的评价信息。具体字段信息如下表 5.5 所示：

表 5.5 职位推荐系统职位评价表

字段名	类型	是否可空	约束	备注
position_eval_id	int	N	P	职位评价编号
position_id	int	N	F	职位编号
create_time	datetime	N		创建时间
evaluator_name	varchar(20)	N		评价用户名
position_eval_value	varchar(64)	N		职位评价分值
position_eval_content	varchar(255)	N		职位评价内容

管理员表用于存放后台管理员的相关信息。具体字段信息如下表 5.6 所示：

表 5.6 职位推荐系统管理员信息表

字段名	类型	是否可空	约束	备注
admin_id	int	N	P	管理员编号
admin_name	int	N		管理员名称
create_time	datetime	N		创建时间
admin_pwd	varchar(64)	N		管理员密码
salt	varchar(64)	N		盐，用于密码加密
phone	varchar(20)	N		手机号
creator_id	int	Y		创建者编号

管理员角色表位于角色和管理员的中间表。具体字段信息如下表 5.7 所示：

表 5.7 职位推荐系统角色管理员表

字段名	类型	是否可空	约束	备注
admin_role_id	int	N	P	管理员角色编号
admin_id	int	N	F	管理员编号
role_id	int	N	F	角色编号

角色表用于存放管理员创建的角色的信息。具体字段信息如下表 5.8 所示：

表 5.8 职位推荐系统角色表

字段名	类型	是否可空	约束	备注
role_id	int	N	P	角色编号
creator_id	int	N		创建者编号
create_time	datetime	N		创建时间
role_name	varchar(20)	N		角色名称
remark	varchar(255)	N		备注

菜单表用于存放职位推荐系统的功能菜单。具体字段信息如下表 5.9 所示：

表 5.9 职位推荐系统菜单表

字段名	类型	是否可空	约束	备注
menu_id	int	N	P	菜单编号
parent_menu_id	int	N		父菜单编号
menu_type	tinyint	N		菜单类型，0 目录 1 菜单 2 按钮
menu_icon_src	varchar(255)	N		菜单图标地址
sort	tinyint	N		排序
menu_name	varchar(64)	N		菜单名称
menu_url	varchar(64)	Y		菜单链接地址

角色菜单表位于角色和菜单的中间表。具体字段信息如下表 5.10 所示：

表 5.10 职位推荐系统角色菜单表

字段名	类型	是否可空	约束	备注
role_menu_id	int	N	P	角色菜单编号
role_id	int	N	F	角色编号
menu_id	int	N	F	菜单编号

5.3 详细设计

本职位推荐系统主要实现了求职用户和招聘的登录以及注册、求职用户职位推荐、系统检索职位首页、求职用户个人简历设置、招聘用户职位管理等功能。

新招聘用户和新求职用户在进入推荐系统前需要注册系统的账号。注册页面如下图 5.8 所示，用户填写用户密码、用户昵称、手机号、手机的验证码以及勾选注册的用户类型。其中，手机号即为用户的登陆账号；同一个手机号在每种类型的用户只能注册一个账户；用户类型包含求职用户以及招聘用户两种类型。注册完成后用户需要返回跳转到登陆页面进行账号登陆。用户登陆页面如下图 5.9 所示，用户登陆需要填写手机号以及账号对应的密码，待后台验证正确后系统将自动跳转到职位系统的首页。

图 5.8 求职和招聘用户的注册页面

图 5.9 求职和招聘用户的登陆页面

求职用户成功登陆后即可进入职位推荐系统的的首页。职位推荐系统的首页如下图 5.10 所示，导航栏上有首页、职位推荐以及我的简历三项菜单。求职用户在首页可以通过职位检索框模糊通过职位名关键字模糊搜索相关职位，还可以通过搜索框下方的职位划分职位类别寻找相关职位。首页最下方显示的是点击量较多的热门职位，求职用户可以点击职位框查看职位详细信息。

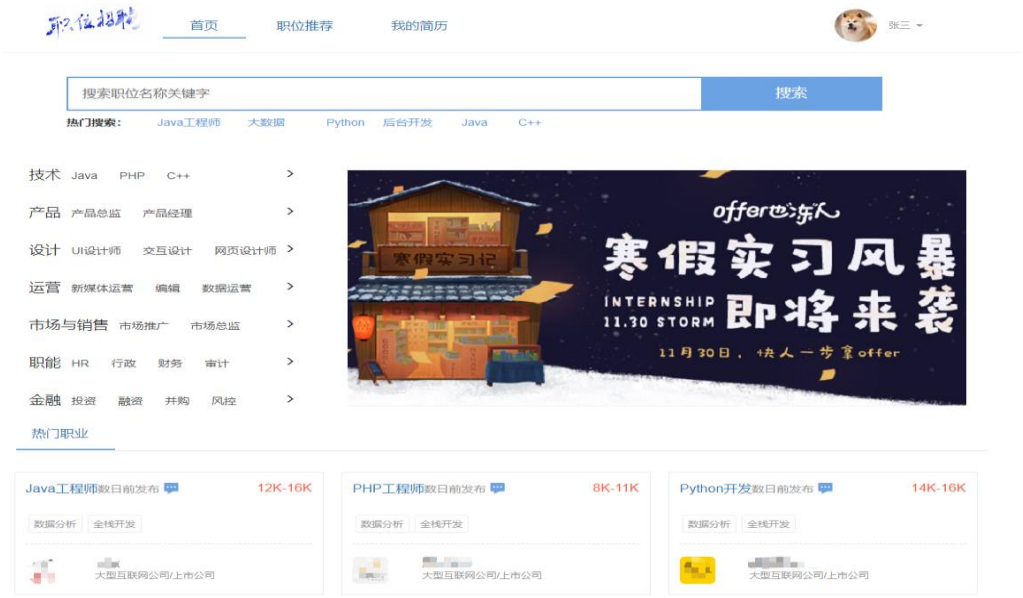

图 5.10 职位推荐的系统首页

求职用户可以点击职位推荐系统上方导航栏的“职位推荐”菜单，查看职位推荐系统根据本文推荐算法列出的 Top-N 职位推荐列表，且职位列表中越靠前的职位其推荐强度越高。职位推荐页面如下图 5.11 所示。

图 5.11 职位推荐系统的职位推荐页面

求职用户可以点击职位推荐系统上方导航栏的“我的简历”菜单，进入个人简历的设置。求职用户可以修改用户昵称、真实姓名、出生年月、性别、毕业年份等个人信息，但不能修改手机号码。其中最高学历、专业名称、所会技能、自我评价四个字段会用于求职用户的职位推荐算法中，以增强用户简历与职位的语义信息。求职用户修改好后点击保存按钮即可完成对个人简历的修改。求职用户个人简历设置页面如下图 5.12 所示。

图 5.12 求职用户个人简历设置页面

招聘用户登陆成功后即可进入招聘管理后台，招聘用户可以查看公司信息以及职位管理。招聘用户点击左侧导航栏的“公司信息”菜单，可查看招聘用户所在的公司相关信息。招聘用户点击左侧导航栏的“职位管理”菜单，则进入职位管理的页面。招聘用户可以修改职位的发布状态、职位名称、招聘人数、学历要求等字段。其次，招聘用户还可以删除已发布的职位和发布新的职位。招聘用户职位管理的职位列表页面和职位发布如下图 5.13 和图 5.14 所示。

图 5.13 职位管理的职位列表页面

图 5.14 职位管理的职位发布页面

5.4 系统测试

软件测试是软件开发生命周期中重要的一环，可以帮助开发团队在软件发布之前发现和修复问题，提高软件质量和可靠性。软件测试的主要目的是确保软件系统或应用程序能够满足预期的需求、设计和功能，并且在预期的环境中稳定、可靠地运行。本文对基于知识图谱的混合式职位推荐系统进行了功能测试。

职位推荐系统的功能测试是为了保障用户端以及管理端能够正常使用各项功能，本推荐系统功能测试分为求职用户功能测试、招聘用户功能测试以及后台管理员功能测试。其中求职用户功能测试结果如下表 5.11 所示。

表 5.11 求职用户功能测试结果

测试模块	测试功能	输入条件	预期输出	测试结果
账号注册登陆	账号注册	合法注册信息	账号注册成功	测试通过
		非法注册信息	账号注册失败	测试通过
	账号登陆	正确登陆信息	登陆成功	测试通过
		错误登陆信息	登陆失败	测试通过
个人简历浏览修改	简历浏览	无	正确显示个人简历信息	测试通过
	简历修改	合法简历信息	修改成功	测试通过

（续）表 5.11 求职用户功能测试结果

测试模块	测试功能	输入条件	预期输出	测试结果
个人简历浏览修改	简历修改	非法简历信息	修改失败	测试通过
职位推荐	职位推荐	无	显示推荐职位列表	测试通过
	热门职位浏览	无	显示职位列表	测试通过
	职位搜索	职位关键字	显示对应的职位列表	测试通过

招聘用户功能测试结果如下表 5.12 所示。

表 5.12 招聘用户功能测试结果

测试模块	测试功能	输入条件	预期输出	测试结果
账号注册登陆	账号注册	合法注册信息	注册成功，待管理员审核	测试通过
		非法注册信息	注册失败	测试通过
	账号登陆	正确登陆信息	登陆成功或管理员审核中	测试通过
		错误登陆信息	登陆失败	测试通过
职位管理	职位发布	合法职位信息	发布成功	测试通过
		非法职位信息	发布失败	测试通过
	职位状态修改	合法简历信息	修改成功	测试通过
		非法简历信息	修改失败	测试通过

后台管理员功能测试结果如下表 5.13 所示。

表 5.13 后台管理员功能测试结果

测试模块	测试功能	输入条件	预期输出	测试结果
登陆	账号登陆	正确登陆信息	登陆成功	测试通过
		错误登陆信息	登陆失败	测试通过
用户管理	招聘用户审核	通过	审核通过	测试通过
		不通过以及原因	审核不通过	测试通过

（续）表 5.13 后台管理员功能测试结果

测试模块	测试功能	输入条件	预期输出	测试结果
用户管理	用户账号浏览	无	显示用户账号列表	测试通过
职位管理	职位浏览	无	显示职位列表	测试通过
	职位评论浏览	无	显示职位评论列表	测试通过
	职位删除	无	删除成功	测试通过
	评论删除	无	删除成功	测试通过
角色管理	查询角色	无	显示角色列表	测试通过
	添加角色	合法角色名称	添加成功	测试通过
		非法角色名称	添加失败	测试通过
	删除角色	无	删除成功	测试通过
	赋予权限	对应菜单权限	赋权成功	测试通过

5.5 本章小结

本章对基于知识图谱的混合式职位推荐系统进行了详细设计。主要对推荐系统的需求分析、概要设计以及详细设计。其中，需求分析包含对推荐系统的功能展开分析、推荐系统各个用例所包含的各项操作分析以及整个系统中的数据流向展开分析。而概要设计包含了推荐系统的系统架构、模块架构以及系统所用的相关表设计。最后，本章对职位推荐系统中的各个页面进行详细设计。

第 6 章 结论与展望

6.1 结论

本文提出了基于知识图谱的混合式职位推荐算法，现有的大部分职位推荐算法对用户行为信息单一偏向，这样做出的推荐不仅存在用户对职位评分的数据稀疏问题还未考量提取用户简历与职位的相关性。因此，为了更加有效的挖掘出求职者与职位的相关联系，以提高职位推荐的精确度，本文的所完成的具体工作有以下几点：

（1）收集公开招聘数据。本文利用 Scrapy 爬虫框架合理爬取拉勾网以及应届生求职网站的公开招聘的职位信息以及求职者对职位的评分信息。本文对爬取数据的字体进行了字体反爬处理。然后提取出职位招聘的详细信息并进行数据清洗、数据规范等操作。处理后的职位相关数据存储于 MySQL 数据库以便后续知识图谱构建使用。

（2）职位知识图谱构建。本文从存放在 MySQL 数据库中读取职位数据构建职位领域的知识图谱，对于职位数据中职位要求的文本字段需要先进行额外的实体抽取操作，再组成三元组。本文对职位要求的文本字段进行实体抽取展开实验研究，分析了不同模型在本文人工标注的数据集前提下的实体抽取效果。最后将抽取的职位要求实体数据集合保存在 Neo4j 数据库中，构建出职位领域的知识图谱。

（3）基于知识图谱的混合式职位推荐算法。本文考虑求职者简历与职位静态信息以及用户对职位评分的动态信息，综合两者来对求职者进行职位推荐。通过职位知识图谱中职位与用户之间的语义信息计算出静态职位语义矩阵，还通过求职者对职位的评分计算出动态职位语义矩阵。最后融合两种语义矩阵再预测出求职者对未有用户行为的职位进行打分，从而对各求职者进行 Top-N 职位推荐。其中求职者对职位的评分融合和时间因素以及利用隐语义模型应对数据稀疏状况。通过本文提出的推荐算法相较于其他推荐算法具有更好的推荐效果。

（4）混合式职位推荐系统设计与实现。本文以基于知识图谱的混合式职位推荐算法为核心，分析了职位推荐系统的需求，设计了职位推荐系统的整体架构、各个功能模块以及数据库中所含的表。最后实现了职位推荐系统的相关页面。

6.2 未来展望

本文的相关研究存在一些不足的地方，因此本文仍然需要更深一步的改进，这里简要讨论下未来工作可从以下几个方面进行：

（1）职位推荐算法的速度方面优化。目前的职位推荐需要从图数据库以及 MySQL 数据库中读出大量数据，其次职位推荐的步骤较多。未来工作可考虑简化职位推荐的步骤或提升数据库读取职位数据的速度，从而加快职位推荐的计算速度。

（2）职位推荐算法的推荐效果方面优化。本文的职位系统在收集用户行为数据只考量求职者对职位评分的信息，未来工作可考虑融合求职者对职位的浏览次数、浏览时间以及收藏等用户行为信息到职位推荐算法中，从而提升职位推荐算法的准确率。

（3）职位推荐系统的优化。本文设计的基于知识图谱的混合式职位推荐系统在功能上存在优化空间，比如增加求职者与招聘者聊天室功能、求职者之间在同一职位下的相互评论功能等。

参考文献

[1] Le T, Tran H, Le B. Knowledge graph embedding with the special orthogonal group in quaternion space for link prediction[J]. Knowledge-Based Systems, 2023, 266: 110400.

[2] Vrandečić D, Krötzsch M. Wikidata: a free collaborative knowledgebase[J]. Communications of the ACM, 2014, 57(10): 78-85.

[3] Suchanek F M, Kasneci G, Weikum G. Yago: A large ontology from wikipedia and wordnet[J]. Journal of Web Semantics, 2008, 6(3): 203-217.

[4] Lenat D B. CYC: A large-scale investment in knowledge infrastructure[J]. Communications of the ACM, 1995, 38(11): 33-38.

[5] Xu B, Xu Y, Liang J, et al. CN-DBpedia: A never-ending Chinese knowledge extraction system[C]//Advances in Artificial Intelligence: From Theory to Practice: 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017, Proceedings, Part II. Cham: Springer International Publishing, 2017: 428-438.

[6] Irwin K. Musipedia: The open music encyclopedia[J]. Reference Reviews, 2008, 22(4): 45-46.

[7] Shu J, Shen X, Liu H, et al. A content-based recommendation algorithm for learning resources[J]. Multimedia Systems, 2018, 24(2): 163-173.

[8] Van Dat N, Van Toan P, Thanh T M. Solving distribution problems in content-based recommendation system with gaussian mixture model[J]. Applied Intelligence, 2022, 52(2): 1602-1614.

[9] 李姗姗. 基于协同过滤的视频推荐系统设计[D]. 南京邮电大学, 2017.

[10] 郑策, 尤佳莉. 电影推荐系统中基于图的协同过滤算法[J]. 计算机与现代化, 2019 (11): 38-43.

[11] Li X, Chen Y, Pettit B, et al. Personalised reranking of paper recommendations using paper content and user behavior[J]. ACM Transactions on Information Systems (TOIS), 2019, 37(3): 1-23.

[12] Riyahi M, Sohrabi M K. Providing effective recommendations in discussion groups using a new hybrid recommender system based on implicit ratings and semantic similarity[J]. Electronic Commerce Research and Applications, 2020, 40: 100938.

[13] 张丹, 周从华. 基于改进的隐马尔可夫模型的新闻推荐算法[J]. 计算机与数字工程, 2020, 48(10): 2332-2337.

[14] Oramas S, Ostuni V C, Noia T D, et al. Sound and music recommendation with knowledge graphs[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2016, 8(2): 1-21.

[15] 阮小芸, 廖健斌, 李祥, 等. 基于人才知识图谱推理的强化学习可解释推荐研究[J]. 数据分析与知识发现, 2021, 5(6): 36-50.

[16] Wang H, Zhang F, Wang J, et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems[C]//Proceedings of the 27th ACM international conference on information and knowledge management. 2018: 417-426.

[17] Kojima R, Ishida S, Ohta M, et al. kGCN: a graph-based deep learning framework for chemical structures[J]. Journal of Cheminformatics, 2020, 12: 1-10.

[18] Wang X, He X, Cao Y, et al. Kgat: Knowledge graph attention network for recommendation[C]//Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 950-958.

[19] Gao X, Yuan F, Fan J. How to circumvent common anti-crawler mechanism of target websites via Scrapy[C]//6th International Workshop on Advanced Algorithms and Control Engineering (IWAACE 2022). SPIE, 2022, 12350: 418-423.

[20] Wang J, Lu W. Two are better than one: Joint entity and relation extraction with table-sequence encoders[J]. arXiv preprint arXiv:2010.03851, 2020.

[21] Bordes A, Usunier N, Garcia-Durán A, et al. Translating embeddings for modeling multi-relational data[C]//Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2. 2013: 2787-2795.

[22] Wang Z, Zhang J, Feng J, et al. Knowledge graph embedding by translating on hyperplanes[C]//Proceedings of the AAAI conference on artificial intelligence. 2014, 28(1).

[23] Lin Y, Liu Z, Sun M, et al. Learning entity and relation embeddings for knowledge graph completion[C]//Proceedings of the AAAI conference on artificial intelligence. 2015, 29(1).

[24] 王卓岚, 张雨琦, 陈鸣宇, 等. 基于 Neo4j 图数据库的电影知识图谱构建与电影推荐研究[J]. 现代电影技术, 2022(03): 29-36.

[25] Masala M, Ruseti S, Dascalu M. Robert–a romanian bert model[C]//Proceedings of the 28th International Conference on Computational Linguistics. 2020: 6626-6637.

[26] Hameed Z, Garcia-Zapirain B. Sentiment classification using a single-layered BiLSTM model[J]. Ieee Access, 2020, 8: 73992-74001.

[27] Tang H, Ji D, Zhou Q. End-to-end masked graph-based CRF for joint slot filling and intent detection[J]. Neurocomputing, 2020, 413: 348-359.

[28] Chen C, Zhang M, Zhang Y, et al. Efficient heterogeneous collaborative filtering without negative sampling for recommendation[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(01): 19-26.

[29] 史达, 于淼川, 李梦琪. 基于用户隐式数据的个性化酒店推荐算法[J]. 山东大学学报(理学版), 2021, 56(7): 1-10.

[30] 石弘利, 王从瑜, 谢维奇. 基于协同过滤算法的教育平台课程推荐的研究[J]. 电脑知识与技术, 2021,17(34): 19-22.

[31] 陈钢, 常笑, 胡枫. 基于隐语义模型的学生选课推荐算法[J]. 计算技术与自动化, 2021, 40(03): 88-93.

[32] Yang T, Kumoi G, Yamashita H, et al. Transfer learning based on probabilistic latent semantic analysis for analyzing purchase behavior considering customers' membership stages[J]. Journal of Japan Industrial Management Association, 2022, 73(2E): 160-175.
[33] 林观德. 基于 Scrapy 爬取招聘信息的研究[J]. 电脑知识与技术, 2020, 16(35): 54-55.
[34] Győrödi C A, Dumşe-Burescu D V, Zmaranda D R, et al. A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management[J]. Big Data and Cognitive Computing, 2022, 6(2): 49.
[35] Bielak K, Borek B, Plechawska-Wójcik M. Web application performance analysis using Angular, React and Vue. js frameworks[J]. Journal of Computer Sciences Institute, 2022, 23: 77-83.
[36] Hagos T. Creating a Spring Boot Project[M]//Beginning Kotlin: Build Applications with Better Code, Productivity, and Performance. Berkeley, CA: Apress, 2022: 213-224.
[37] Canakci B. Supporting Distributed Systems of Distributed Systems[D]. Cornell University, 2022.
[38] Zhang Z, Yuan M, Qian H. Research on mysql database recovery and forensics based on binlog[C]//Proceedings of the 11th International Conference on Computer Engineering and Networks. Springer Singapore, 2022: 741-750.
[39] Steyer R. Behind the Scenes: How and Why Does Vue. js Work?[M]//Building web applications with Vue. js: MVVM patterns for conventional and single-page websites. Wiesbaden: Springer Fachmedien Wiesbaden, 2022: 25-41.
[40] Karthik R, Sridhar T S, Sriram R. Digital Food ordering system based on Spring Framework[J]. International Journal of Recent Technology and Engineering, 2020, 8(6): 4795-4798.
[41] Menezes G, Cafeo B, Hora A. How are framework code samples maintained and used by developers? The case of Android and Spring Boot[J]. Journal of Systems and Software, 2022, 185: 111146.
[42] Wang J, Kai Z. Performance Analysis and Optimization of Nginx-based Web Server[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1955(1): 012033.
[43] Gualandi Hugo Musso,Ierusalimschy Roberto. A surprisingly simple Lua compiler—Extended version[J]. Journal of Computer Languages,2022,72.
[44] 张会奇. WAF 系统在 OpenResty 上的构建[J]. 福建电脑, 2020, 36(12): 142-144.
[45] 黄冰. 基于 Docker 的 MySQL 数据库性能分析[J]. 无线互联科技, 2021, 18(06): 69-70.
[46] 何晓霞, 古兰拜尔·吐尔洪, 买日旦·吾守尔, 等. 融合 BERT 词嵌入和 BiLSTM 的微博谣言持续检测模型[J]. 东北师大学报(自然科学版), 2023, 55(01): 65-71.
[47] 孔欢, 黄树成. 基于隐语义模型推荐算法的优化[J]. 计算机与数字工程, 2022, 50(10): 2197-2201.
[48] 易佳丽. 面向企业信息化系统集成的中台架构研究[J]. 中国新通信, 2022, 24(24): 84-86.
[49] 成冰洁, 林秀燕. 基于三层架构的考核管理系统设计[J]. 电脑知识与技术, 2019, 15(35): 50-51.

[50] 高志平. 基于 SpringBoot 框架与 ITIL 方法的运维管理系统的设计与实现[D]. 华东师范大学, 2021.

[51] Li Y Z, Gao S, Pan J, et al. Research and application of template engine for web back-end based on MyBatis-Plus[J]. Procedia Computer Science, 2020, 166: 206-212.