

学校代码：10225



本科毕业论文

面向通信标准的大语言模型微调及知识图谱构建

罗中泽

学院：计算机与控制工程学院

专业班级：通信工程 2021-03

指导教师：王健 副教授

企业导师：张志涛

学号：2021210447

2025年06月

摘要

通信领域的标准种类繁多，传统的咨询模式周期较长且依赖专家知识经验，难以满足快速发展的技术要求。针对通信标准领域的特点，本文采用了 LoRA 低秩适应方法对大语言模型进行微调，提升其对专业语言理解和针对性内容生成的语义分析能力；提出了经过精心设计的本体结构，借助大语言模型辅助进行实体及关系高效识别与抽取，构建具有较高精确度的通信标准领域限定域知识图谱；设计了检索增强生成 RAG 智能咨询问答系统，实现微调模型与知识图谱的有机融合。实验结果表明，在本文构建的 6,587 条问答对的通信标准领域数据集上微调后，Qwen2.5-7B-Instruct 在测试集上表现出了出色的通信标准领域专业能力；根据包含 6 种实体属性与 10 种关系属性的本体框架，构建包含 13,906 个实体与 12,361 条关系的通信标准领域知识图谱，展现出良好的准确性；智能咨询问答系统结合网页服务和 API 接口，在知识检索准确度、问答响应速度、用户交互体验等方面均实现较好结果，具有良好的实际应用价值。

关键词 大语言模型；LoRA 微调；限定域知识图谱；检索增强生成；通信标准

Abstract

There are numerous types of standards in the field of communication. The traditional consulting model has a long cycle and relies on the knowledge and experience of experts, making it difficult to meet the rapidly developing technical requirements. In view of the characteristics in the field of communication standards, this thesis adopts the LoRA low-rank adaptation method to fine-tune the large language model to enhance its semantic analysis ability for professional language understanding and targeted content generation. A well-designed ontology structure was proposed. With the assistance of a large language model, efficient recognition and extraction of entities and relations were carried out to construct a knowledge graph of the communication standard domain with high accuracy. The RAG intelligent consultation and question-answering system for retrieval enhancement generation was designed to achieve the organic integration of the fine-tuning model and the knowledge graph. The experimental results show that after fine-tuning on the dataset of 6,587 question-answer pairs in the field of communication standards constructed in this thesis, Qwen2.5-7B-Instruct demonstrates outstanding professional capabilities in the field of communication standards on the test set. Based on the ontology framework containing 6 entity attributes and 10 relation attributes, a knowledge graph of the communication standard domain containing 13,906 entities and 12,361 relations was constructed, demonstrating good accuracy. The intelligent consultation and question-answering system, combining web services and API interfaces, has achieved good results in terms of the accuracy of knowledge retrieval, the response speed of question-answering, and the user interaction experience, and has good practical application value.

Keywords Large language models; LoRA fine-tuning; domain-limited knowledge graph; retrieval-augmented generation; communication standards

目录

摘要	I
Abstract	II
目录	III
1 绪论	1
1.1 研究背景及意义	1
1.2 国内外研究现状	1
1.3 本文结构安排	2
2 理论框架介绍	3
2.1 大语言模型微调	3
2.1.1 大语言模型简介	3
2.1.2 大语言模型的应用方法	4
2.1.3 LoRA 微调简介	5
2.1.4 通信标准对大语言模型的特殊需求	6
2.2 限定域知识图谱的构建	6
2.2.1 知识图谱简介	6
2.2.2 限定域知识图谱	6
2.2.3 限定域知识图谱的构建	7
2.3 本章小结	8
3 实验设计	9
3.1 微调实验	9
3.1.1 数据集构建	11
3.1.2 实验设置和损失曲线结果	12
3.1.3 微调模型指标评估	14
3.1.4 Qwen 合并模型与其他模型的指标评估	16
3.2 知识图谱构建实验	17
3.2.1 本体设计	17
3.2.2 头实体识别	18
3.2.3 潜在关系识别	20
3.2.4 尾实体识别	21
3.2.5 三元体构建	22
3.2.6 知识图谱可视化	23
3.2.7 知识图谱评估	25
3.3 微调模型搭载限定域知识图谱的推理实验	25

3.3.1 转换服务器端微调后的合并模型文件	25
3.3.2 量化模型	26
3.3.3 制作 Ollama 使用的模型	28
3.3.4 搭建检索增强生成 RAG 框架.....	28
3.3.5 更改本地电脑端 Neo4j 的 IP 访问权限.....	30
3.3.6 Windows 防火墙端口放行设置	31
3.3.7 服务器端 Ollama 模型远程访问本地电脑端的知识图谱进行推理	32
3.3.8 服务器端 Flask 部署外部可访问的网页服务	34
3.3.9 服务器端创建独立的 FastAPI 知识图谱问答服务	36
3.4 本章小节	42
3.5 开源内容说明	43
3.5.1 知识图谱构建与微调模型搭载限定域知识图谱推理的开源说明	43
3.5.2 实验环境压缩包开源内容说明	45
3.5.3 本文所有开源内容的补充说明	46
结论	47
参考文献	48
附录	51
附录 A	51
附录 B	52
附录 C	54
致谢	55

1 绪论

1.1 研究背景及意义

在通信领域中，许多工程任务涉及不同的细分方向，在方案制定中往往需要遵守种类繁多的通信标准。通信标准是指在通信系统中，确保网络设备正常工作并实现预期功能的技术要求，这通常是由国际标准化组织或行业组织所撰写发布的。

然而由于通信标准复杂的特性，如何查询并进行合规性检测成为需要解决的问题。用户通常希望能快速获得准确无误的通信标准咨询结果，这个过程一般需要长时间的学习记忆和经验积累，大语言模型的出现使得人们重新思考，这个过程是否可以通过大语言模型来替代和简化。

随着通信标准的改进与发展，大语言模型的能力飞速提升，在很大程度上影响了各行业，通过大语言模型的帮助，可以很容易地进行咨询，得到几乎完全正确的通识领域问题答案。但是大语言模型在特定领域的知识掌握情况较差，这是由于预训练数据中存在较少特定领域相关内容，但如今可以通过先进的微调技术来改善这个问题。

但本文的数据来源通常是非结构化的，即通信标准的源文件通常是复杂完整的独立文档。为了提供更加智能的咨询服务，知识图谱在管理和利用结构化数据方面成为了一种解决方案。通过微调后的特定领域大语言模型，能获得质量更高的领域问答结果，并搭载通信标准的限定域知识图谱，能够立竿见影地增强通信标准领域的咨询服务。

本文的研究背景涵盖大语言模型微调、限定域知识图谱的构建、微调模型搭载限定域知识图谱的推理，核心贡献在于融会贯通这些新兴的人工智能技术，并使之协调，共同帮助复杂真实场景下通信领域咨询服务的实现。

1.2 国内外研究现状

国内有学者在特定领域大语言模型的应用方向做出许多相关研究，沈晨晨^[1]等人提出 LawLLM 研究面向法律领域的大模型微调与应用；宋晓宁^[2]等人提出一种基于大语言模型的创新性食品知识图谱智能感知问答系统；张天鸿^[3]等人基于大语言模型构建灌浆工程知识服务系统。但是，相关技术在通信标准领域中的研究尚为空白，一般通信标准领域的研究与本论文技术的相关度也不高。

自 ChatGPT 发布以来，大语言模型^[4,5]的发展迅猛，涌现了多个有影响力的模型，包括 ChatGLM、LLaMa、BaiChuan 等。这些模型在大量数据上进行了预训练、微调以及人类反馈强化学习^[6,7]，因此能够进行流畅的对话交流，能够理解复杂的语境和用户意图，并在文本生成、情感分析、机器翻译等方面提供高质量的输出^[8-12]。

典型的特定领域大语言模型工作通常使用大量的特定领域问答数据来微调基座模型^[13]，从而得到掌握特定领域知识并具有特定领域咨询对话能力的大语言模型^[14]，医疗、法律、金融等面向特定领域的大语言模型也应运而生，并展现出卓越的领域对话能力，

满足了用户的多样化需求^[15-18]。在特定领域中，知识图谱的引入也大大提高了大语言模型问答的专业程度^[19,20]。Dorottya Demszky^[21]等人认为大语言模型具有推进心理学测量和实践的潜力，并研究了关于大语言模型在心理学中应用的主要问题；Juanming Shi^[22]等人提出 Legal-LM 通过知识图谱增强的大语言模型，专为中国法律领域的法律咨询而设计。然而国外对于相关技术在通信标准咨询服务中的研究也尚为空白。

1.3 本文结构安排

本论文拟探讨如何通过大语言模型微调、限定域知识图谱的构建等这些当前最热门的人工智能技术，来实现一个通信标准领域咨询服务流程。并且评估本文的模型在咨询服务中的优势。本文结构安排如下：

第一章为绪论，阐述了如何快速且准确地对工程任务进行通信标准查询和检测变得愈发重要，紧接着引出了大语言模型的出现对于解决咨询服务起到了关键作用。并且介绍了在帮助解决该问题的人工智能技术，以及相关的国内外研究现状。

第二章为理论框架介绍，主要介绍了本文所用技术的理论基础，主要包括两大部分，即大语言模型与微调技术的介绍，以及限定域知识图谱的理论介绍，讲述了本文所涉及的相关技术细节，以及这些技术的发展来源。

第三章为实验设计，主要介绍了本文所用技术的实验设计细节和结果评估，主要包括三大部分，即三个相关联的实验：微调实验、限定域知识图谱构建实验、微调模型搭载限定域知识图谱的推理实验，讲述了实验的相关设置和步骤细节。

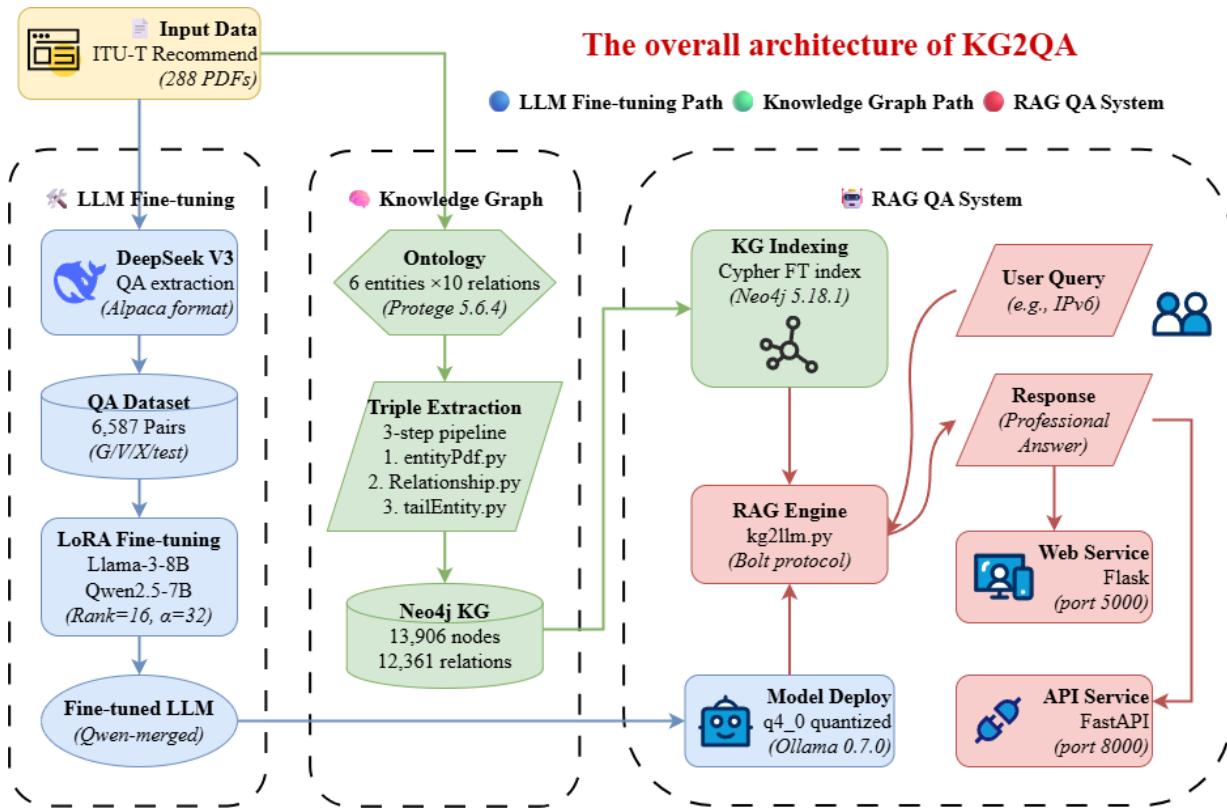


图 1.1 本文结构框架

2 理论框架介绍

2.1 大语言模型微调

2.1.1 大语言模型简介

大语言模型微调是本文研究的核心技术之一，其目的是通过针对性的训练，使预训练的大语言模型能够更好地适应通信标准领域的特定任务和需求。大语言模型是基于深度学习技术构建的人工智能模型，通常具有数十亿甚至数千亿个参数。这些模型通过在海量的文本数据上进行预训练，学习语言的语法、语义和上下文关系，从而具备强大的语言理解与生成能力。

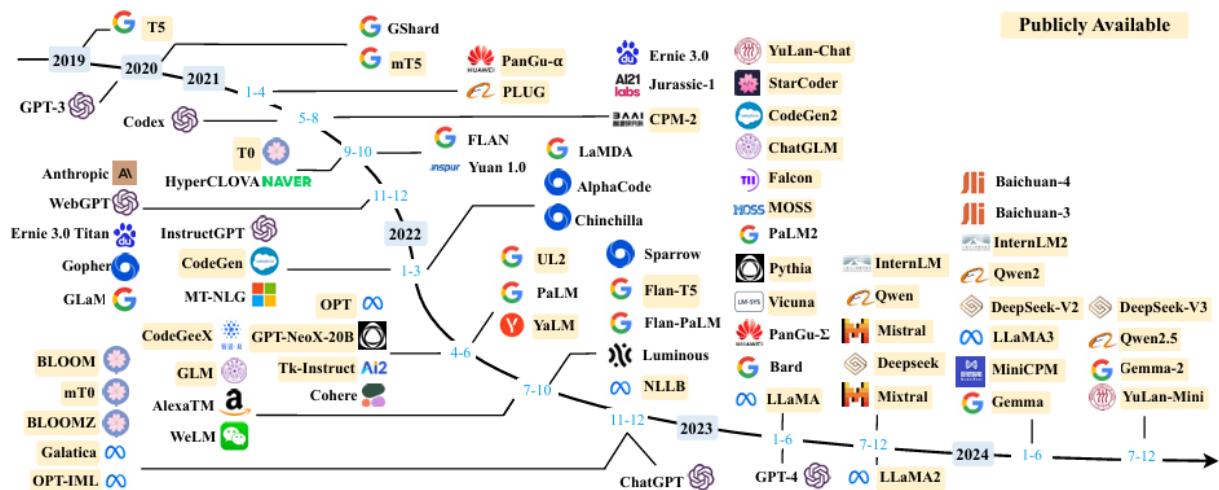


图 2.1 近年来现有大语言模型（规模大于 10B）的时间表

大语言模型的起源可以追溯到 2017 年 Vaswani 等人提出的 Transformer 架构^[4]。Transformer 提出创新性的自注意力机制，使用自注意力来权衡每个标记相对于其他标记的重要性，使得模型能够动态关注输入的相关部分，并且允许并行计算加快训练速度，并提出多头注意力等创新设计，显著提升了模型对长文本的处理能力和训练效率。此后，基于 Transformer 架构的模型不断演进，如 BERT 和 GPT 系列等，进一步推动了大语言模型的发展。

大语言模型通常基于 Transformer 架构，能够捕捉输入数据中的长距离依赖关系。模型由编码器和解码器组成，编码器将输入文本编码成潜在表示，解码器则将这些表示转换为目标文本。每个 Transformer 层包含多头自注意力机制和前馈网络，通过位置编码保留输入序列的顺序信息。此外，大语言模型通过在海量文本数据上进行预训练，学习语言的语法、语义和上下文关系，从而具备强大的语言理解与生成能力。预训练后的模型可以进一步微调，以适应特定的自然语言处理任务。

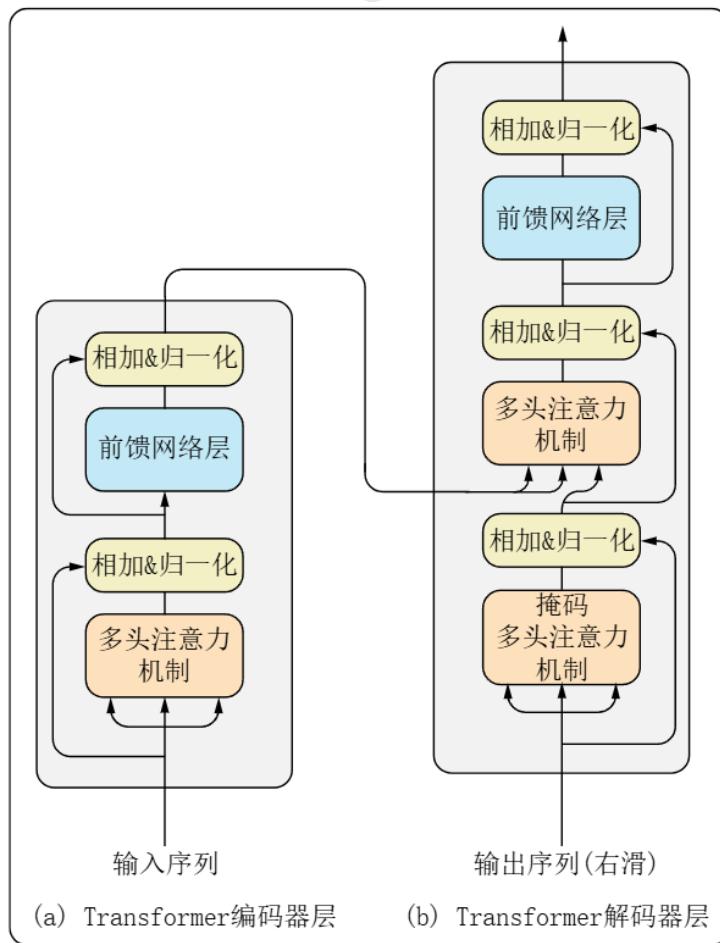


图 2.2 Transformer 架构示意图

2.1.2 大语言模型的应用方法

如今大语言模型在自然语言处理和机器学习领域取得了显著进展，其关键在于大幅提高了预训练语言模型的参数规模和训练数据量，从而展现出小型模型所不具备的涌现能力。其工作原理是基于输入提示预测下一个最可能的词，这种机制使其具备了少样本学习和指令跟随等能力，能够仅通过少量示例完成任务并准确执行用户指令。在实际应用中，大语言模型主要通过提示工程和微调两种方式应用于下游任务。

提示工程通过设计合适的提示来引导模型产生期望的输出^[23-25]，其方法包括检索增强生成 RAG、上下文学习 ICL 和推理方法等。RAG 结合外部知识库为模型提供专业知识和历史案例，使其能进行更准确的分析和推理。ICL 涵盖零样本学习和少样本学习，使模型在无需大量标注数据的情况下快速适应新任务。推理方法如链式思考 CoT 和思维树 ToT 则提升了模型解决复杂问题的能力。

微调是提升大语言模型在特定领域性能的关键技术，通过在特定领域数据上进一步训练模型来增强其针对性能力。完整的微调过程包括数据准备、预训练模型选择、模型微调和模型评估四个环节。微调方法主要分为两大类：全量微调和参数高效微调。全量微调对模型的所有参数进行调整，能够充分利用模型的全部参数以获得最佳性能，但计

算成本和显存需求较高，且容易出现灾难性遗忘。参数高效微调则通过少量可训练参数进行微调，如 Prompt Tuning 和 LoRA 等方法，以较低的计算和数据成本实现显著的性能提升，且较不易出现灾难性遗忘。本文选择了高效低资源的 LoRA 微调方法，通过低秩分解的方式只调整模型中的一部分参数，既优化了模型性能，又显著降低了计算成本和显存需求^[26]。

2.1.3 LoRA 微调简介

基础模型尽管能够表现出广泛而强大的能力，然而它们庞大的参数规模使其在适应特定下游任务方面带来了重大挑战。LoRA 微调已经成为缓解这一情况的有效方法，并由于其简单性以及在多个不同模型架构和领域的泛化能力，从而得到了广泛关注^[27]。

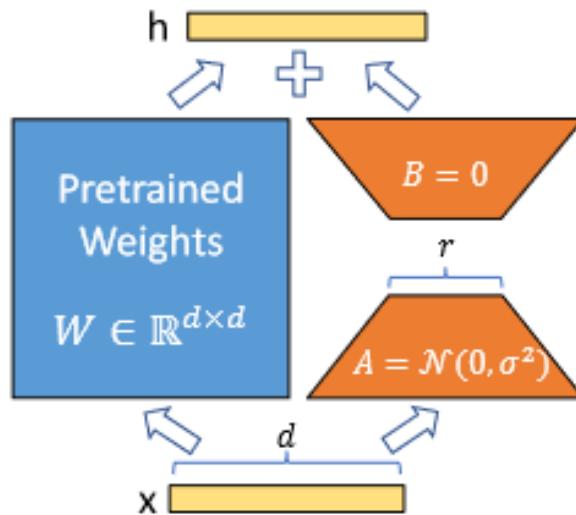


图 2.3 LoRA 算法设计示意图

在神经网络中训练一个全连接层时，权矩阵通常是满秩的。此时每个权重在模型中扮演不同的角色，并且没有冗余。然而这可能会导致过拟合，从而降低模型的泛化能力，还会导致大量的计算和存储开销。LoRA 冻结预训练模型的权重，并在每个 Transformer 块中注入秩分解矩阵，在模型的 Linear 层旁边添加分支 A 和 B。从而实现了大型模型权矩阵的隐式低秩，训练的参数更少，但是获得与全量微调基本相当的性能。

$$W_0 + \Delta W = W_0 + BA \quad (2.1)$$

$$h = W_0 x \xrightarrow{\text{LoRA}} h = W_0 x + \Delta W x = W_0 x + BA x \xrightarrow{\text{alpha}} h = W_0 x + \frac{\alpha}{r} BA x \quad (2.2)$$

$$B \in R^{d \times r}, A \in R^{r \times k} \text{ and } r \ll \min(d, k) \quad (2.3)$$

LoRA 引入了两个矩阵 A 和 B，如果参数 W 的原始矩阵的大小为 $d \times d$ ，则矩阵 A 和 B 的大小分别为 $d \times r$ 和 $r \times d$ ，其中 r 要小得多。参数 r 称为秩。如果使用秩为 $r = 16$ 的 LoRA，则这些矩阵的形状为 $16 \times d$ ，这样就大大减少了需要训练的参数数量，但是 r 较小时信息精简却不全面， r 较大时信息全面却容易含有冗余无效的知识。对于预训练权重矩阵 W_0 ，微调过程中将其更新方式限制为 $W_0 + BA$ 。在训练时，原始参数 W_0 被冻结，即虽

然 W_0 会参与前向传播和反向传播，但不会计算其对应的梯度，更不会更新其参数。这样模型不是直接更新原始的权重矩阵 W_0 ，微调过程中主要学习的是低秩矩阵B和A，并且引入了缩放超参数 α 应用于低秩自适应输出。在推理时， W_0 表示旧知识， $\frac{\alpha}{r}BA$ 表示新知识，旧知识与新知识合并，相比原始模型不存在推理延时。

2.1.4 通信标准对大语言模型的特殊需求

尽管预训练的大语言模型在通识任务上表现出色，但在通信标准领域仍面临诸多挑战。通信标准领域具有其独特的术语和规范，预训练模型的数据中可能缺乏足够的相关领域内容。例如，通信标准文档中包含大量的技术术语和复杂的规范描述，这些内容在通用文本语料库中出现的频率较低。因此，预训练模型在处理通信标准相关任务时，可能会出现理解不准确、生成内容不符合规范等问题。此外，通信标准的更新频繁，需要模型能够快速适应新的标准和规范，这些特殊需求迫切需要对大语言模型进行微调，从而显著提升其在通信标准领域的性能表现。

2.2 限定域知识图谱的构建

2.2.1 知识图谱简介

知识图谱^[28,29]作为谷歌提出的基于图结构的知识表示方法，本质是有向图结构知识库，以结构化形式描述现实世界中的实体、概念及关系，借助三元组（<实体，属性，属性值>或<实体1，关系，实体2>）作为基本存储与表示单元，通过关系和属性将不同实体连接成图状结构^[30]。其底层存储分基于表结构（用二维数据表存储数据）和基于图结构（以节点表实体、边表关系，图结构存储更直观利于查询，常见如Neo4j等图数据库）两种方式^[31]。

2.2.2 限定域知识图谱

限定域知识图谱是一种专门针对特定领域构建的知识图谱，其特点在于实体和关系类别预先定义，能够高效地组织和管理特定领域的知识。它通过有监督学习方法构建，利用预定义的结构实现精准的知识抽取和管理，广泛应用于问答系统、智能推荐和故障诊断等场景，为特定领域提供高效的知识服务^[32,33]。

知识图谱通过节点和边的形式化表达来描述现实世界中的概念、对象及其相互关系（实体和关系）。根据适用范围，知识图谱可分为开放域知识图谱和限定域知识图谱。开放域知识图谱主要涵盖常见的实体类型及其关系，然而开放域知识图谱仅包含一般性知识，缺乏特定领域的细粒度知识。相比之下，限定域知识图谱范围更窄，但提供了更深入和详细的信息，适用于不同领域的任务。例如，在医疗诊断中，知识图谱可辅助疾病、原因、药物和治疗计划之间的逻辑推理^[32]；在金融科技领域^[34]，构建了以股票为中心的知识图谱^[35]；在高校教育领域，构建了知识图谱以辅助科研管理和课程教学^[36,37]。

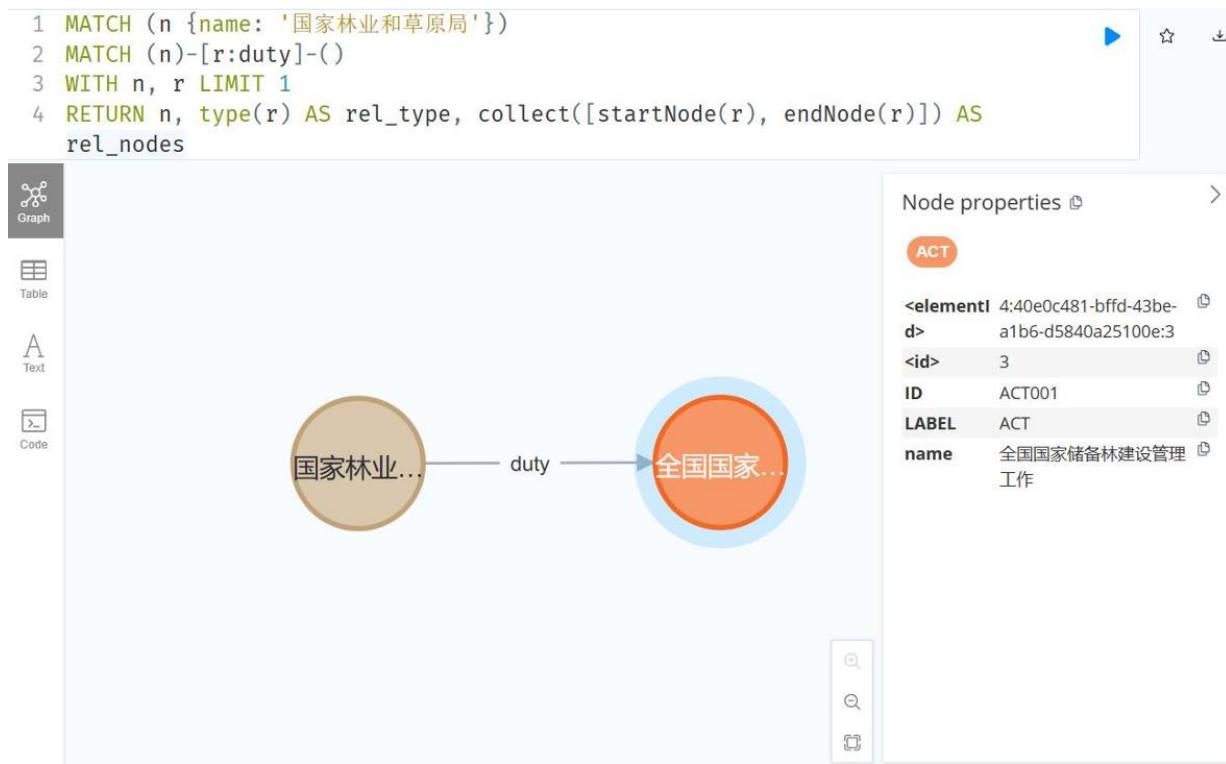


图 2.4 知识图谱查询示意图

构建的知识图谱能够有效覆盖和响应用户的特定领域查询需求。图 2.4 中显示了在 Neo4j 数据库上的一个林业政策知识图谱具体查询案例，可以看到，在查询“国家林业和草原局”实体的职责时，显示其负责“国家保护区森林建设工作”。

2.2.3 限定域知识图谱的构建

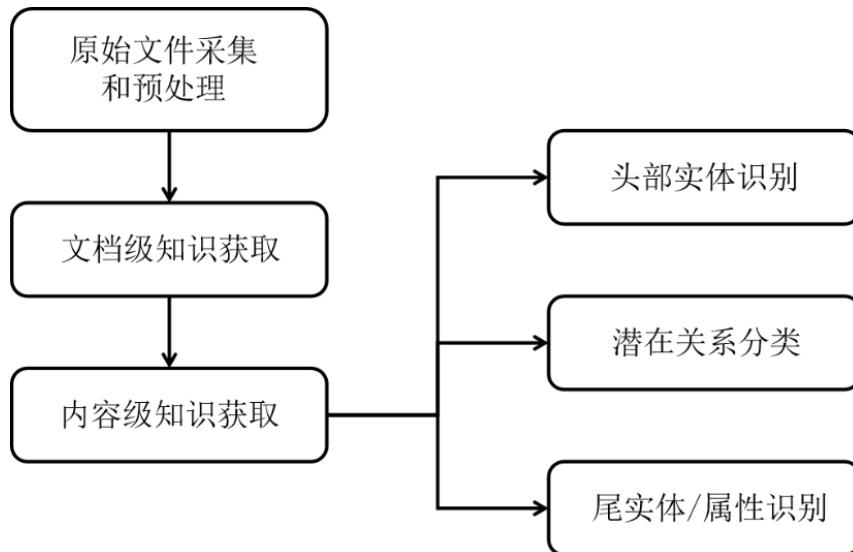


图 2.5 基于大语言模型的知识图谱构建流程

知识图谱的构建依赖于信息抽取技术，即将自由文本中的实体、关系和属性转化为结构化形式^[38,39]。经典的信息抽取方法通常将实体抽取看作是一个序列标注的机器学习问题，利用神经网络模型预测文本中实体的起始和结束位置，然后根据该实体来识别关

系类别，还有方法联合实体和关系进行同时抽取，这样可以减少实体到关系识别过程中的错误传播。尽管这些传统方法技术成熟，但它们往往需要人工进行大量数据标注，大语言模型的出现改变了这一局面，使得许多自然语言处理任务，包括信息抽取，能够在小样本甚至零样本的情况下完成^[40]。一些学者提出利用大语言模型进行实体抽取和关系识别的方法，借助许多大语言模型提供的开源 API，用户仅仅需要输入提示并上传文本便可得到抽取结果^[41]。

本体设计^[42]是构建知识图谱的关键步骤，尤其在限定域知识图谱中，其设计质量直接决定了知识图谱的应用范围。本体提供了一种形式化描述特定领域知识的机制，由一组表示概念、概念之间关系及规则的集合组成。在知识图谱中，本体实现了知识领域中数据的分类和定义，为数据之间的语义关系提供共享框架，这种结构化和语义化的表示方式不仅增加了数据的可理解性和可用性，并且对实现跨知识领域间的知识共享和集成具有深远的影响。

本体包括类 Classes、关系 Relationships 和实例 Instances。类是一组拥有相同属性与共同行为的实体集合，通过类对实体进行分类；关系是对类或实例之间的一种语义连接，为关联性的描述；实例则是知识图谱中的具体数据对象，是类的具体特例，是本体最基本的数据点，它们共同构成了知识图谱的骨架，为智能咨询服务提供支撑。

限定域知识图谱的构建首先要确定领域范围，明确知识图谱所涵盖的主题和范畴；其次设计概念层级结构，构建概念间的层次关系，保障概念的准确性和层级性；接着需要定义属性关系，属性关系包括二元关系和多元关系。

最后应用形式化的语言（如 OWL）进行本体的格式化存储与表达。本体设计的原则包括表达的明确性与完整性等原则，有助于本体的精确定义，有效反映本体中的完整信息，为知识图谱构建及应用提供支撑。

2.3 本章小结

本章详细阐述了大语言模型的技术发展与背景、LoRA 微调与知识图谱的技术简介、限定域知识图谱的构建方法等。说明了本文使用大语言模型微调技术与限定域知识图谱技术的原因，以及它们对于通信标准领域咨询服务的贡献。

3 实验设计

3.1 微调实验

本文微调实验的所有步骤都已开源至 ModelScope 平台，网址链接可见于附录 A。

The screenshot shows the ModelScope platform interface. The top navigation bar includes links for Home, Model Library, Data Sets, Create Space, AIGC专区, Documentation, Community, MCP Marketplace, GitHub, and Search. The main content area displays the 'Communication_standards_dataset' page, which includes tabs for Dataset Introduction, Preview, Dataset Files (selected), Quick Use, Feedback, and Settings. Below this, a file list shows various files like .gitattributes, dataset_infos.json, G.json, README.md, test.json, V.json, and X.json, each with download and delete options. At the bottom right of the list is a 'Download Dataset' button.

图 3.1 微调数据集的开源示意图

The screenshot shows the ModelScope platform interface. The top navigation bar includes links for Home, Model Library, Data Sets, Create Space, AIGC专区, Documentation, Community, MCP Marketplace, GitHub, and Search. The main content area displays the 'Qwen_communication_standards' model page, which includes tabs for Model Introduction, Model Files (selected), Feedback, and Settings. Below this, a file list shows files like .gitattributes, added_tokens.json, config.json, and configuration.json, each with download and delete options. At the bottom right of the list is a 'Download Model' button.

图 3.2 微调模型的开源示意图

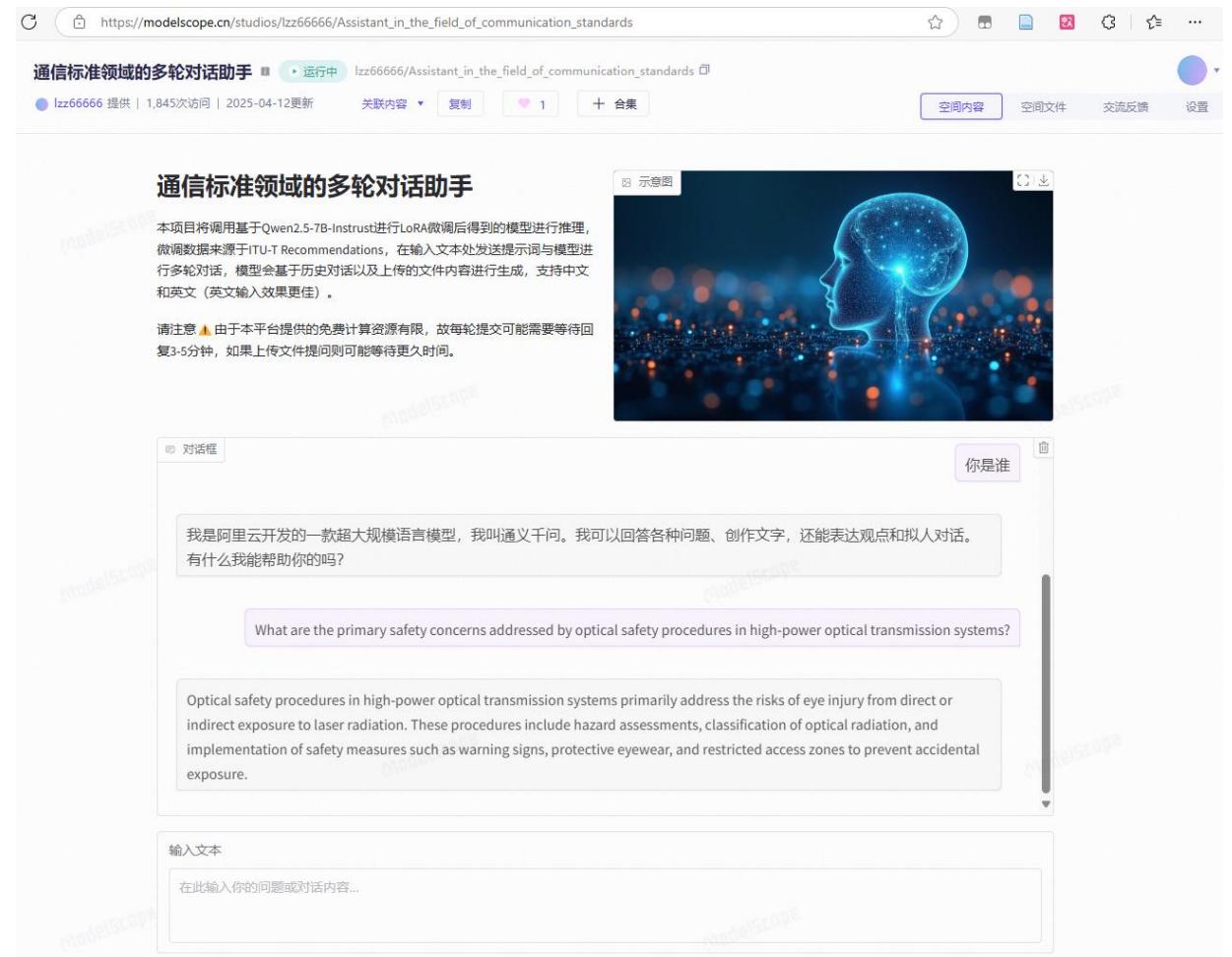


图 3.3 前端程序的开源示意图

从图 3.3 中可以清楚的看到，本项目的前端体验程序调用了基于 Qwen2.5-7B-Instruct 进行 LoRA 微调后得到的模型进行推理，可以看到对话框中先对模型进行了提问“你是谁”，模型回答“我是阿里云开发的一款大規模语言模型，我叫通义千问”，可以看到微调后模型仍然正确保留了基座模型的已有知识，并且正确回答了基座模型来源。

之后提问了一个通信标准领域问题“What are the primary safety concerns addressed by optical safety procedures in high-power optical transmission systems?”（在高功率光传输系统中，光安全程序主要关注的安全问题是什么？），模型回答“Optical safety procedures in high power optical transmission systems primarily address the risks of eye injury from direct or indirect exposure to laser radiation, These procedures include hazard assessments, classification of optical radiation, and implementation of safety measures such as warming signs, protective eyewear, and restricted access zones to prevent accidental exposure.”（高功率光传输系统中的光学安全程序主要解决直接或间接暴露在激光辐射下眼睛受伤的风险，这些程序包括危害评估、光辐射分类和安全措施的实施，如暖化标志、防护眼镜和限制进入区，以防止意外暴露。）

3.1.1 数据集构建

本文数据集用于通信标准领域大语言模型微调，语言为英语。数据来源于 ITU-T Recommendations，共取自于三个系列的建议书，三个系列共选取了 288 个 PDF 源文件进行 QA 构造，具体类别来源如图 3.4。

G 系列：传输系统和介质、数字系统和网络 共34个PDF源文件

- G.600-G.699：传输介质和光学系统特性
- G.650-G.659：光纤电缆
- G.660-G.679：光学元件和子系统的特性
- G.680-G.699：光学系统的特性

V 系列：通过电话网络进行数据通信 共77个PDF源文件

- V.1-V.9：常规
- V.10-V.34：接口和语音带调制解调器
- V.35-V.39：宽带调制解调器
- V.40-V.49：错误控制
- V.50-V.59：传输质量和维护
- V.60-V.99：同时传输数据和其他信号
- V.100-V.199：与其他网络互通
- V.200-V.249：用于数据通信的接口层规范
- V.250-V.299：控制程序
- V.300-V.399：数字电路上的调制解调器

X 系列：数据网络、开放式系统通信和安全性 共177个PDF源文件

- X.1-X.199：公共数据网络
- X.1-X.19：服务和设施
- X.20-X.49：接口
- X.50-X.89：传输、信令和交换
- X.90-X.149：网络方面
- X.150-X.179：维护
- X.180-X.199：行政安排
- X.300-X.399：网络之间的互通
- X.300-X.349：常规
- X.350-X.369：卫星数据传输系统
- X.370-X.379：基于 IP 的网络
- X.1000-X.1099：信息和网络安全
- X.1000-X.1011：安全编排和服务访问
- X.1030-X.1049：网络安全
- X.1050-X.1079：安全管理
- X.1080-X.1099：远程生物识别技术
- X.1700-X.1729：量子通信
- X.1702-X.1704：量子随机数生成器
- X.1705-X.1749：量子密钥分发网络 (QKD)
- X.1750-X.1799：数据安全
- X.1750-X.1759：大数据安全
- X.1770-X.1799：数据保护

图 3.4 数据集来源说明图

数据集分为训练集和测试集，训练集由 G、V、X 三个类别的数据构成，都以 JSON 格式保存，测试集为 test.json。采用 DeepSeek V3 API 对源文件 PDF 进行 QA 抽取得到数据集，数据量如表 3.1。

表 3.1 数据量说明

数据名	QA 数量
G.json	898
V.json	1101
X.json	3779
test.json	809
All	6587

数据集格式都为标准的 Alpaca 格式，为指令监督微调数据集。Instruction 为用户给出的指令或问题，是必须提供的信息。Input 为可选的额外输入信息，用于提供给定指令的上下文或具体例子。Output 为模型根据指令和输入产生的响应，是必须提供的信息。数据集的示例如图 3.5，格式说明为图 3.6。

```

1  {
2    "instruction": "What are the chromatic dispersion coefficients at wavelengths 1264.5 nm, 1310.19 nm, and 1337.5 nm derived from linear wavelength dependence?",
3    "input": "Source: once distribution of the fibre parameter has been established, the Monte Carlo method can be used to simulate the process of building links by repeatedly sampling from measured values of the fibre parameters.", "output": "The Monte Carlo method is used to simulate the process of building links by repeatedly sampling from measured values of the fibre parameters." },
4  {
5    "instruction": "What are the Gaussian parameters for the upper and lower partial distributions of fD(x) at 1310 nm wavelength?",
6    "input": "Source: Table I.5 – Gaussian parameters for the upper and lower partial distributions of fD(x) for the reference case.", "output": "At 1310 nm wavelength, the Gaussian parameters for the lower partial distribution are μlower = -0.502 ps/(nm·km) and σlower = 0.001 ps/(nm·km). The upper partial distribution is symmetric about zero." },
7  {
8    "instruction": "What is the relationship between the number of concatenated cable pieces and the mean normalized PMD Q?",
9    "input": "Source: Table I.6 – Example dependence of mean normalized PMDQ and mean normalized DGD on the number of concatenated cable pieces.", "output": "As the number of concatenated cable pieces M increases, the mean normalized PMD Q decreases while the mean normalized DGD increases." },
10 {
11   "instruction": "What are the typical applications of optical amplifier devices and subsystems?", "input": "Source: Recommendation ITU-T G.663 (2011), Application-related aspects of optical amplifier devices and subsystems.", "output": "Optical amplifier devices and subsystems are typically used in enhancing the signal strength in optical communication systems." },
12 {
13   "instruction": "What are the key specifications for 10-Gigabit-capable passive optical networks (XG-PON) at the Physical Layer?", "input": "Source: Recommendation ITU-T G.987.2 (2023), 10-Gigabit-capable passive optical networks (XG-PON): Physical Layer Specifications.", "output": "The key specifications for 10-Gigabit-capable passive optical networks (XG-PON) at the Physical Media Dependent layer include a maximum transmission distance of 40 km, a maximum splitting ratio of 1:64, and a maximum latency of 10 ms." }

```

图 3.5 数据集示例图

```

[ {
  "instruction": "问题",
  "input": "问题来源与出处",
  "output": "模型回答"
},
...
]

```

图 3.6 数据集格式说明

3.1.2 实验设置和损失曲线结果

本文服务器 Linux 版本为 Ubuntu 22.04，使用 NVIDIA® GeForce RTX™ 4090 D 24G 采用 Llama-Factory 框架进行 LoRA 微调，对两个基座模型采取相同的参数设置进行对照实验，模型分别为 Llama-3-8b-Instruct 和 Qwen2.5-7B-Instruct。Llama-Factory 版本为 0.9.3.dev0，Python version 版本为 3.10.0，PyTorch 版本为 2.6.0+cu124(GPU)，Transformers 版本为 4.51.0，CUDA 版本为 12.8，服务器配置如图 3.7，LoRA 微调参数设置如图 3.8。

两个模型微调的过程 Loss 曲线，如图 3.9。上方两张图分别是 Llama 的训练集 Loss 曲线和验证集的 Loss 曲线，下方两张图分别是 Qwen 的训练集 Loss 曲线和验证集 Loss 曲线。可以看到，两个模型在本文的参数设置下，在训练集和验证集上都已收敛，并未存在过拟合现象，但由于本文数据集针对的是通信标准领域，由于特定领域的复杂性和所选模型的轻量特性，在验证集上收敛值略高于训练集上收敛值。

3 实验设计

```
(base) h3c@h3c-H3C-UniServer-R4900-G501:~$ nvidia-smi
Sun Apr 27 18:17:28 2025
+-----+
| NVIDIA-SMI 570.124.04      Driver Version: 570.124.04     CUDA Version: 12.8 |
+-----+
| GPU  Name      Persistence-M | Bus-Id     Disp.A  | Volatile Uncorr. ECC | | |
| Fan  Temp     Perf          Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|          |             |              |             | GPU-Util   Compute M. |
|          |             |              |             |             MIG M. |
+-----+
| 0  NVIDIA GeForce RTX 4090        Off  00000000:B1:00.0 Off |           Off
| 32%   34C   P8    15W / 450W | 15MiB / 24564MiB | 0%     Default |
|                               |                           N/A                |
+-----+
+-----+
| Processes:
| GPU  GI CI          PID  Type  Process name          GPU Memory |
| ID   ID             ID
+-----+
| 0   N/A N/A         1764   G   /usr/lib/xorg/Xorg          4MiB |
+-----+
(c) h3c@h3c-H3C-UniServer-R4900-G501:~$ llmfactory-cli env
INFO 05-23 18:38:00 [__init__.py:239] Automatically detected platform cuda.

- `llmfactory` version: 0.9.3.dev0
- Platform: Linux-5.15.0-139-generic-x86_64-with-glibc2.31
- Python version: 3.10.0
- PyTorch version: 2.6.0+cu124 (GPU)
- Transformers version: 4.51.0
- Datasets version: 3.4.1
- Accelerate version: 1.5.2
- PEFT version: 0.15.0
- TRL version: 0.9.6
- GPU type: NVIDIA GeForce RTX 4090
- GPU number: 1
- GPU memory: 23.53GB
- vLLM version: 0.8.3
```

图 3.7 服务器配置图

```
# 模型 & 方法
model_name_or_path: Qwen/Qwen2.5-7B-Instruct
stage: sft
do_train: true
finetuning_type: lora
lora_target: ["W_pack", "o_proj", "k_proj", "v_proj", "q_proj", "gate_proj", "up_proj",
"down_proj"]
lora_rank: 16
lora_alpha: 32
# 数据集
dataset: G,V,X
template: qwen
cutoff_len: 2048
max_samples: 6000
overwrite_cache: true
preprocessing_num_workers: 16
# 输出 & 日志
output_dir: saves/Qwen2.5-7B-Instruct/lora/qwen2.5-7B/1
logging_dir: saves/Qwen2.5-7B-Instruct/lora/qwen2.5-7B/1/logs
logging_steps: 10
save_steps: 500
plot_loss: true
overwrite_output_dir: true
# 训练设置
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 5.0e-5
num_train_epochs: 2
lr_scheduler_type: cosine
warmup_ratio: 0.1
bf16: true
flash_attn: auto
ddp_timeout: 180000000
# 验证
val_size: 0.1
per_device_eval_batch_size: 2
eval_strategy: steps
eval_steps: 100
load_best_model_at_end: true
metric_for_best_model: eval_loss
greater_is_better: false
```

图 3.8 LoRA 微调参数设置图

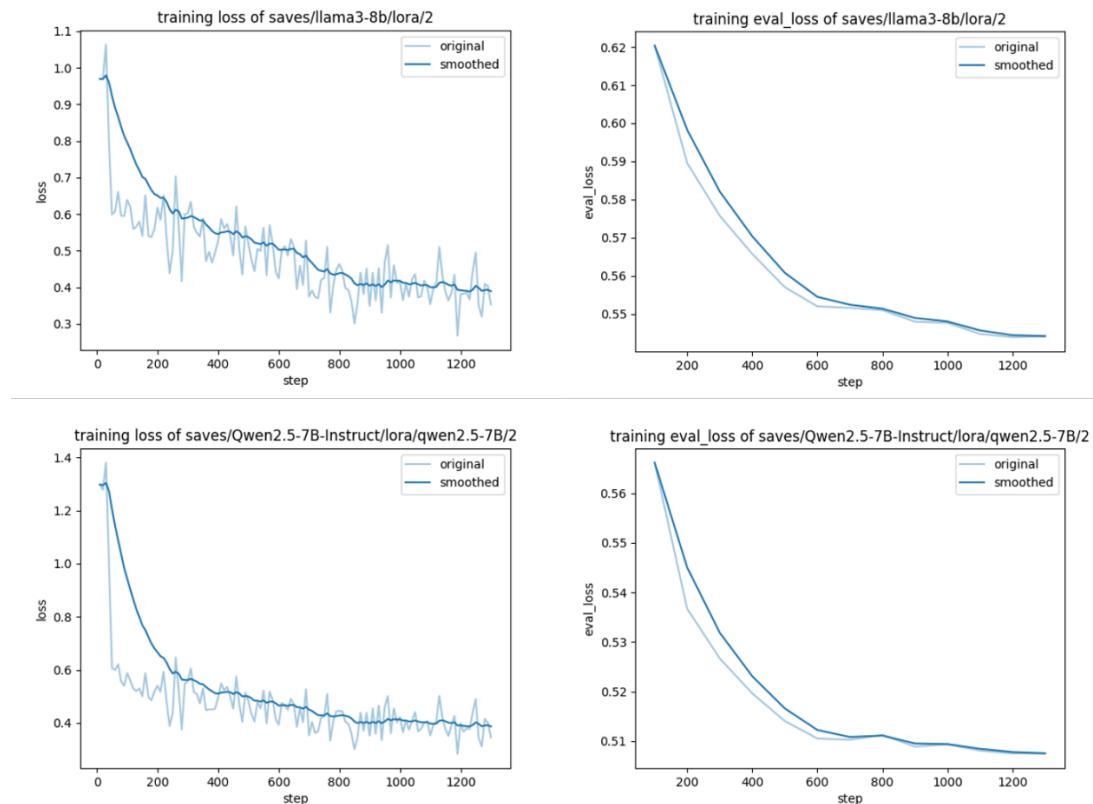


图 3.9 Llama 与 Qwen 基座模型的微调 Loss 曲线对比图

3.1.3 微调模型指标评估

将微调后的模型与原始模型合并后，得到两个合并模型。对合并模型进行评估，基座模型以及合并模型在测试集 test.json 上输出限制 2048 tokens 的评估结果如表 3.2。

表 3.2 基座模型和合并模型的评估结果对比表

指标	Qwen 基座	Qwen 合并	Llama 基座	Llama 合并
BLEU-4	18.8564	66.8993	37.3405	66.4780
predict_runtime	0.0036	0.0037	0.0037	0.0038
ROUGE-1	35.8729	70.9748	48.2548	70.2607
ROUGE-2	16.0076	52.9495	25.2638	51.7437
ROUGE-L	18.7772	61.4781	31.6350	70.7144
runtime	5301.4163	929.6552	2900.5805	1047.5824
samples_per_second	0.1530	0.8700	0.2790	0.7720
steps_per_second	0.0760	0.4360	0.1400	0.3870

BLEU-4 指标用于评估模型生成文本与参考答案在 4-gram 级别上的相似度，数值越高，代表模型生成的内容与标准答案的连续四词组匹配度越高，表明生成质量越好。predict_runtime 指标反映了推理阶段模型加载和准备的耗时，单位为秒，时间越短意味着模型上线和推理的启动速度越快。ROUGE-1 和 ROUGE-2 分别评估生成文本与参考文本在单词级（unigram）和双词组级（bigram）上的召回率，主要衡量生成内容中有多少参

考文本中的重要信息被正确覆盖，得分越高说明覆盖越充分。ROUGE-L 则考虑了生成文本与参考文本的最长公共子序列，兼顾了语义顺序的匹配性，是一种更加整体性的衡量方式。runtime 衡量整个推理过程中模型完成一次推理所需的总时间，单位为秒，数值越低越好。除此之外，samples_per_second 和 steps_per_second 进一步描述了模型推理的吞吐率，分别表示每秒完成的样本数和每秒完成的步骤数，数值越高代表推理效率越好。指标公式可见于公式 3.1-3.9，符号说明如表 3.3。

表 3.3 评估指标公式符号说明表

公式符号	含义
BP	简洁惩罚 (brevity penalty)
p_n	第 n 阶 n-gram 精确度 (precision)
n	n-gram 阶数 (对于 BLEU-4，则取 1、2、3、4)
\mathcal{G}_n	候选文本中所有 n-gram 的集合
Count _{cand} (g)	候选文本中 g 的出现次数
Count _{ref} (w)	参考文本中 g 的出现次数
Count _{clip} (g)	剪裁计数: min(Count _{cand} (g), Count _{ref} (g))
c	候选文本长度 (词数)
r	参考文本长度 (词数)，通常选最接近 c 的参考
Ref ₁	参考文本中的所有 unigram 集合
Ref ₂	参考文本中的所有 bigram 集合
w	unigram (单词)
b	bigram (连续两词组合)
L	候选文本 X 与参考文本 Y 的最长公共子序列长度 (LCS)
X, Y	候选文本序列与参考文本序列
X , Y	文本长度 (词数)
R _{LCS}	基于 LCS 的召回率
P _{LCS}	基于 LCS 的精确率
F _{LCS}	LCS 对应的 F-measure 分数 ($\beta = 1$ 时即 ROUGE-L)
β	F-measure 中的调和因子 (通常取 1，使精确率与召回率同权)

$$\text{BLEU-4} = \text{BP} \times \exp\left(\frac{1}{4} \sum_{n=1}^4 \log p_n\right) \quad (3.1)$$

$$p_n = \frac{\sum_{g \in \mathcal{G}_n} \text{Count}_{\text{clip}}(g)}{\sum_{g \in \mathcal{G}_n} \text{Count}_{\text{cand}}(g)} \quad (3.2)$$

$$\text{BP} = \min\left(1, \exp\left(1 - \frac{r}{c}\right)\right) \quad (3.3)$$

$$\text{ROUGE-1} = \frac{\sum_{w \in \text{Ref}_1} \min(\text{Count}_{\text{cand}}(w), \text{Count}_{\text{ref}}(w))}{\sum_{w \in \text{Ref}_1} \text{Count}_{\text{ref}}(w)} \quad (3.4)$$

$$\text{ROUGE-2} = \frac{\sum_{b \in \text{Ref}_2} \min(\text{Count}_{\text{cand}}(b), \text{Count}_{\text{ref}}(b))}{\sum_{b \in \text{Ref}_2} \text{Count}_{\text{ref}}(b)} \quad (3.5)$$

$$L = \text{LCS}(X, Y) \quad (3.6)$$

$$R_{\text{LCS}} = \frac{L}{|Y|} \quad (3.7)$$

$$P_{\text{LCS}} = \frac{L}{|X|} \quad (3.8)$$

$$\text{ROUGE-L} = F_{\text{LCS}} = \frac{(1 + \beta^2) R_{\text{LCS}} P_{\text{LCS}}}{R_{\text{LCS}} + \beta^2 P_{\text{LCS}}} \quad (\beta = 1) \quad (3.9)$$

从表 3.2 的数据可以看出，经过微调合并后的模型在大多数指标上均优于基座模型。以 Qwen 系列为为例，合并模型的 BLEU-4 分数从 18.8564 提升至 66.8993，ROUGE-1、ROUGE-2 和 ROUGE-L 的分数也有显著提升，分别从 35.8729 提升到 70.9748、从 16.0076 提升到 52.9495，以及从 18.7772 提升到 61.4781，表明合并模型在文本生成的准确性与覆盖度上都有明显增强。Llama 系列同样呈现出类似的提升趋势，合并模型在 BLEU-4、ROUGE-1、ROUGE-2 和 ROUGE-L 上分别达到了 66.4780、70.2607、51.7437 和 70.7144，相较基座模型均有大幅增长。

在效率指标方面，合并模型的推理时间大幅缩短。Qwen 基座模型推理一次所需时间由 5301.4163 秒降为 929.6552 秒，速度显著提升，samples per second 和 steps per second 分别提升至 0.8700 秒和 0.4360 秒。Llama 系列也表现出相似趋势，说明了合并模型在推理阶段的效率优势。故微调后的合并模型在生成质量和推理效率上均优于基座模型。

Llama 基座模型在指标上优于 Qwen 基座模型，但微调后 Qwen 合并模型整体优于 Llama 合并模型。以 BLEU-4 分数为例，Qwen 合并模型达到了 66.8993，略高于 Llama 合并模型的 66.4780；ROUGE-1、ROUGE-2 和 ROUGE-L 等指标亦是如此。这一现象表明，虽然 Llama 基座模型具有较好的原生性能，但 Qwen 模型对于本次微调任务的数据适应性更强，经过微调后能更充分地挖掘潜力，实现更高质量的文本生成，故本文上传 Modelscope 平台的模型和前端程序的模型都为合并后的 Qwen 模型。

3.1.4 Qwen 合并模型与其他模型的指标评估

由于市面上许多大语言模型仍处于闭源，或者某些开源大语言模型的规模过大，难以在本地进行部署和推理，需要很高的计算资源，故需要调用这些模型的 API 进行推理评估，本文选取市面上主流的 DeepSeek、Kimi、豆包等模型进行测试集 test.json 上的指标评估，与本文微调后的 Qwen 合并模型对比 4 个指标 BLEU-4、ROUGE-1、ROUGE-2、ROUGE-L，分析了本文微调后的模型与市面上的通识大语言模型在通信标准领域的优劣势，输出限制 2048 tokens，评估结果如表 3.4。

表 3.4 Qwen 合并模型与通识大语言模型的评估结果对比表

指标	Qwen 合并	DeepSeek	Kimi	豆包
BLEU-4	66.90	37.46	59.99	28.93
ROUGE-1	70.97	47.07	39.41	55.05
ROUGE-2	52.95	27.29	25.01	37.96
ROUGE-L	61.48	36.38	31.91	46.39

本文使用的 API 模型版本号为: DeepSeek-V3-0324, moonshot-v1-8k, doubao-1-5-pro-32k-250115, 使用代码 deepseek.py, kimi.py, doubao.py 进行测试, 测试后生成对应的 deepseek_v3_outputs.json, moonshot-v1-8k_outputs.json, doubao-1-5-pro-32k_outputs.json。相关代码和文件已开源至图 3.1 的 Modelscope 平台微调数据集仓库中。

由表 3.4 可以看出, 3 个通识大语言模型 DeepSeek、Kimi、豆包的 API 测试在 4 个评估指标上均不如本文微调后的 Qwen 合并模型, 可以进一步说明本文微调实验的有效性, 使得模型更好地掌握了通信标准领域知识。

3.2 知识图谱构建实验

本文在本体设计过程中充分考虑了通信标准的领域特性, 设计了简洁有效的本体, 并将本体设计利用 Protege 进行可视化操作, 有助于增强本文所构建知识图谱的合理性。本文采取基于大语言模型的知识图谱构建流程, 总共分为头实体识别、潜在关系识别、尾实体识别这三个步骤, 三个步骤调用 DeepSeek V3 API 依次进行, 最终抽取获得完整的三元体, 并将知识图谱上传至 Neo4j 平台进行可视化和评估。

3.2.1 本体设计

本体设计是构建知识图谱的重要步骤, 尤其对于限定域知识图谱来说, 本体的设计直接决定了整个知识图谱的质量和应用范围。本文首先确定通信标准领域本体所包含的实体类型, 然后确定关系类型和属性。本文针对通信标准领域的特性, 设计了 6 种实体类型, 分别为标识 (Identifier)、结构/组成 (Structure/Composition)、适用性/上下文 (Applicability/Context)、行动 (Action)、值 (Value)、功能 (Function), 如表 3.5。

表 3.5 实体类型定义表

实体类型	标识符	描述
标识	IDEN	建立实体的唯一标识, 包括其通用名称、类型和定义其来源的标准
结构/组成	STR_COM	描述实体的内部单元、组件或整体是如何构成的
适用性/上下文	APP_CON	定义应用实体的上下文、范围、约束或特定上下文
行动	ACT	执行或涉及的操作步骤或行为
值	VALUE	与实体相关联的特定数值、编码或设置
功能	FUN	执行的特定动作或功能

在定义完实体类型后，本文针对这些实体类型定义了 7 种关系类型，分别为包含（Contain）、依赖于（Be Relied on）、完成（Accomplish）、限制（Limit）、相关（Relevant）、执行（Execute）、影响（Influence）。并且定义了它们定义域与值域的实体类型，在关系类型定义表 3.6 中以实体类型的标识符表示。之后本文定义了 3 种互逆关系，分别为被包含（Be Contained）、撤销（Undo）、被影响（Be Influenced），值得注意的是，相关（Relevant）是自互逆的。故本文最终定义了 10 种潜在关系的类型，如表 3.6。

表 3.6 关系类型定义表

关系类型	标识符	定义域	值域	互逆关系
包含	contain	STR_COM/ IDEN	IDEN/ VALUE	被包含 (isContained)
依赖于	isReliedOn	FUN	VALUE	/
完成	accomplish	ACT	FUN	/
限制	limit	STR_COM/ APP_CON	FUN/ ACT	/
相关	relevant	FUN/ VALUE/ ACT/ IDEN	IDEN/ APP_CON	此关系为自互逆
执行	execute	VALUE	ACT	撤销（undo）
影响	influence	APP_CON	STR_COM	被影响 (isInfluenced)

Protege 是一款强大的本体编辑工具，用于创建和维护本体，本文利用 Protege 进行本体设计的可视化操作，Protege 版本为 5.6.4-win，本体创建结果显示如图 3.10。

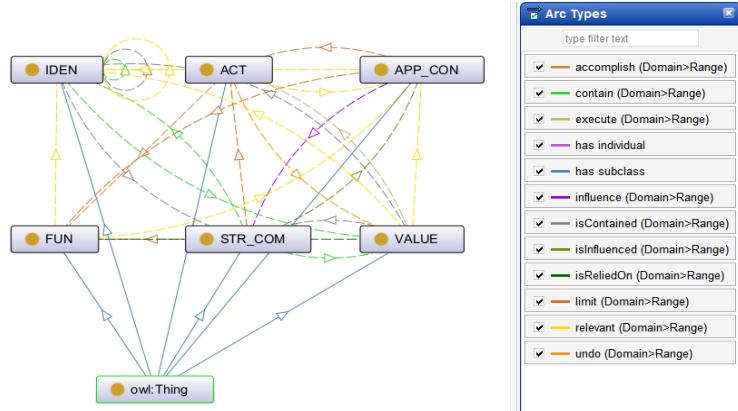


图 3.10 本体设计可视化图

3.2.2 头实体识别

头实体识别代码 entityPdf.py 从 PDF 文档中提取文本，并调用 DeepSeek V3 API 进行头实体抽取。先读取 PDF 文档文本；之后构造提示词，请求模型提取具有类型标注的头实体；每个实体包含头实体名称、头实体类型、上下文原文片段、置信度；最后输出结构化 JSON 格式的实体列表，写入输出文件夹。该模块的伪代码如图 3.11，提示词如图 3.12。

```

Input: PDF_DIR: Directory containing the PDF files.
Output: H[f]: List of head entities extracted from each PDF file f (saved as JSON).
for each file  $f \in \text{PDF\_DIR}$  do
     $text_f \leftarrow \text{extract\_text\_from\_pdf}(f)$ 
     $prompt_H \leftarrow \text{build\_prompt\_for\_head\_entities}(text_f)$ 
    for attempt  $\in [1..3]$  do
         $response_H \leftarrow \text{LLM}(prompt_H)$ 
        if  $response_H$  is valid JSON and contains entities then
             $H[f] \leftarrow \text{parse\_entities}(response_H)$ 
            break
        end
    end
     $\text{save\_to\_json}(H[f], \text{"entities\_batch\_i.json"})$ 
end

```

图 3.11 头实体识别算法伪代码

You are a professional academic information extraction expert. Please extract head entities (head entities) from the following text as comprehensively as possible. Each document should extract 30-40 entities.

```

# Extraction Requirements
1. Head entities are the starting nodes in the knowledge graph's triple (Head Entity, Relation, Tail Entity), representing the entities that initiate the relationship or act as the subject of the relationship. For example:
- In the triple "Apple Inc. - Founded - 1976", "Apple Inc." is the head entity.
- In "Beijing - Is - Capital of China", "Beijing" is the head entity.
2. Each entity must include all applicable type labels (multiple selections are possible).
3. Extract entities of the following types:
- (Identifier)
- (Structure/Composition)
- (Action)
- (Value)
- (Function)
- (Applicability/Context)

# Example (Do not include this example in actual extraction)
{
  "document": "sample.md",
  "entities": [
    {
      "index": 1,
      "entity": "ITU-T G.652 fiber",
      "type": ["Identifier", "Structure/Composition", "Applicability/Context"],
      "context": "ITU-T G.652 fiber was originally optimized for the 1310 nm wavelength region but can also be used in the 1550 nm region.",
      "confidence": 0.92,
    },
    {
      "index": 2,
      "entity": "Wavelength Division Multiplexing (WDM) technology",
      "type": ["Action", "Function"],
      "context": "Wavelength Division Multiplexing (WDM) technology can significantly increase the transmission capacity of optical fibers.",
      "confidence": 0.88,
    }
  ]
}

# Special Notes
1. Do not omit any entities that may meet the criteria in the text.
2. Label each entity with as many applicable types as possible (usually 2-3 types).
3. For long documents, ensure that the number of extractions reaches 30-40.
4. Head entities are the basic units of a knowledge graph, connecting to tail entities through relationships to form a knowledge network.
For example: (Head Entity: Apple Inc.) → (Relationship: Headquarters) → (Tail Entity: Cupertino)

# Text to be analyzed (from document: {pdf_name})
{text[:MAX_TOKENS * 4]} # Increase text truncation length

# Please return the results strictly in the following JSON format:
{
  "document": "File Name",
  "entities": [
    {
      "index": Index,
      "entity": "Entity Name",
      "type": ["Type 1", "Type 2", ...], # Must use list format
      "context": "Context of Appearance",
      "confidence": Confidence (0-1),
    }
  ],
  "extraction_stats": {
    "total_entities": Total number of entities,
    "coverage_estimate": "Document coverage estimate"
  }
}

```

图 3.12 头实体识别算法提示词

3.2.3 潜在关系识别

潜在关系识别代码 Relationship.py 用于对已抽取出的头实体，进一步分析其上下文，提取其与尾实体之间的关系。先读取头实体识别后生成的 JSON 文件；接着按上下文 index 分组处理，确保每个实体的上下文完整无更改；然后调用 API 分析上下文，抽取三元组中的关系；最后对每个实体，生成结构化关系信息，包含关系类型、对应尾实体、上下文原文片段、置信度。该模块的伪代码如图 3.13，提示词如图 3.14。

```

Input: H_JSON_DIR: Directory containing the JSON files with head entities.
θ: Confidence threshold (e.g., 0.8).
Output: R_T[f]: Extracted relations and tail candidate entities (saved as JSON).
for each file f ∈ H_JSON_DIR do
    doc ← load_json(f)
    for each h ∈ doc["entities"] do
        if h.confidence < θ then
            end
            continue
        ctx ← h.context
        prompt_R ← build_prompt_for_relations(h, ctx)
        for attempt ∈ [1..3] do
            response_R ← LLM(prompt_R)
            rels ← parse_response(response_R)
            valid_rels ← []
            for each rel ∈ rels do
                if rel.confidence ≥ θ and rel.relation ∈ REL_TYPES then
                    | Append rel to valid_rels
                end
            end
            if valid_rels ≠ ∅ then
                | Append (h, valid_rels) to R_T[f]
                break
            end
        end
    end
    save_to_json(R_T[f], "relations.batch_i.json")
end

```

图 3.13 潜在关系识别算法伪代码

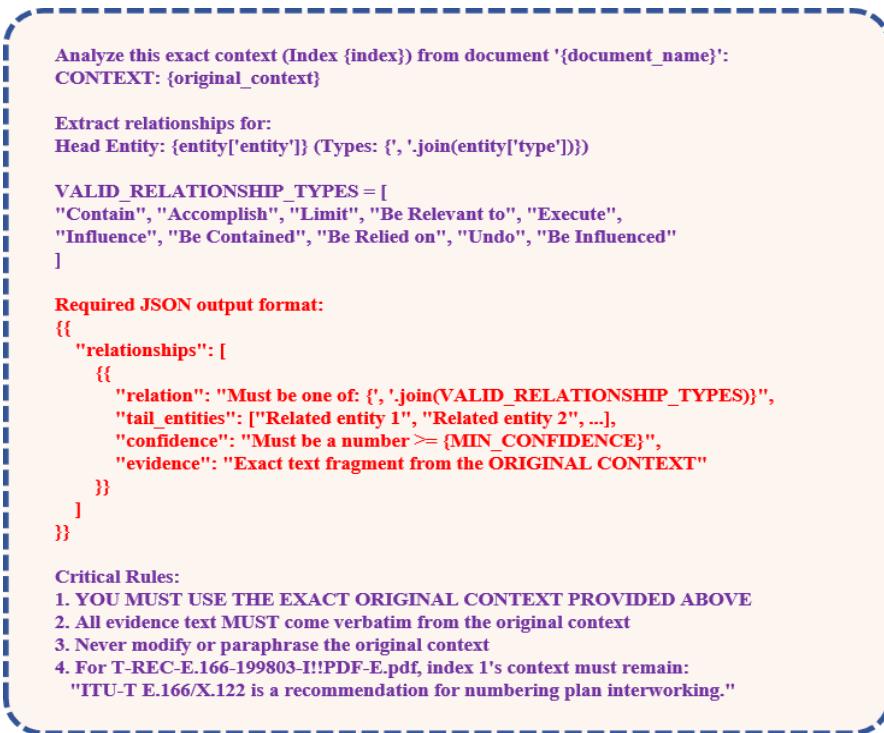


图 3.14 潜在关系识别算法提示词

3.2.4 尾实体识别

尾实体识别代码 tailEntity.py 用于进一步精炼三元组中的尾实体，确保其符合要求，并与头实体不重复。首先读取已提取关系的 JSON 文件；逐个上下文调用模型，在原始上下文中抽取尾实体，尾实体不得与头实体名称相同，仅提取类型在允许集合内的实体，并且置信度需高于阈值；返回字段包括尾实体名称、尾实体类型、上下文原文片段、置信度；最后生成新的尾实体补充结构，合并进上下文结果中。该模块的伪代码如图 3.15。

```

Input: R_T_JSON_DIR: Directory containing the candidate triples.
θ: Confidence threshold.
Output: KG[f]: Final knowledge graph triples for each file f (saved as JSON).
for each file f ∈ R_T_JSON_DIR do
    doc ← load_json(f)
    for each (h, rel_list) ∈ doc["context_relationships"] do
        ctx ← rel_list["context"]
        for each rel ∈ rel_list["relationships"] do
            if rel.confidence < θ then
                end
                continue
            t' ← rel.tail_entity
            t_type ← determine_tail_entity_type(t'.name, ctx)
            Append (h, rel.relation, t', t_type) to KG[f]
        end
    end
    save_to_json(KG[f], "formatted_f.json")
end

```

图 3.15 尾实体识别算法伪代码

本文设计了基于大语言模型从 PDF 文档中提取知识图谱三元体的完整框架。该流程分工明确，模块化实现抽取任务，确保了知识图谱的准确性与鲁棒性。该流程的伪代码如图 3.16。本文识别代码的伪代码符号说明如表 3.7。

```

Input: PDF_DIR: Directory containing the original PDF files.
θ: Confidence threshold.
Output: KG[f]: Complete knowledge graph triples, saved as JSON.
for each file f ∈ PDF_DIR do
    text_f ← extract_text_from_pdf(f)
    H[f] ← LLM(head_entity_prompt(text_f))
    for each h ∈ H[f] do
        ctx ← h.context
        Append (h, LLM(relation_prompt(h, ctx))) to R_T[f]
    end
    for each (h, rels) ∈ R_T[f] do
        for each (r, t') ∈ rels do
            if r.confidence ≥ θ then
                t_type ← determine_tail_entity_type(t'.name, ctx)
                Append (h, r, t', t_type) to KG[f]
            end
        end
    end
    save_to_json(KG[f], "final_triples_f.json")
end

```

图 3.16 基于大语言模型提取知识图谱三元体的完整框架伪代码

表 3.7 本文识别代码的伪代码符号说明

符号	解释说明
f	单个 PDF 文件名
text _f	从文件 f 中提取文本
ctx	头实体出现的上下文
h	头实体
r	潜在关系
t'	尾实体
t_type	尾实体属性类型
θ	置信阈值 (如 0.8)
H[f]	从文件 f 中提取的头实体列表
R_T[f]	文件 f 的提取关系和尾部候选项
KG[f]	文件 f 的最终知识图谱三元体

3.2.5 三元体构建

将尾实体识别后获得的 JSON 文件进行处理。分为两步骤：先进行实体处理，将补充尾实体结构后的 JSON 文件中的头实体和尾实体都按照实体类型抽取，生成单个实体类型的 CSV UTF-8 格式结果；之后进行关系处理，将关系的头实体和尾实体按照它们在对应实体类型 CSV UTF-8 文件中的 ID 进行匹配，构成三元体，将结果保存在 roles.csv 文件中。实体处理结果示例如图 3.17，关系处理结果示例如图 3.18。

A	B	C
1	ID	name
2	IDEN001	ITU-T E.166/X.122
3	IDEN002	E.164 numbering plan
4	IDEN003	X.121 numbering plan
5	IDEN004	Integrated Services Digital Networks (ISDNs)
6	IDEN005	Public Switched Telephone Networks (PSTNs)
7	IDEN006	Packet Switched Public Data Networks (PSPDNs)
8	IDEN007	World Telecommunication Standardization Conference (WTSC)
9	IDEN008	ITU-T Study Groups
10	IDEN009	Recommendation I.330
11	IDEN010	Recommendation X.121
12	IDEN011	Recommendation E.160
13	IDEN012	Recommendation E.164
14	IDEN013	Recommendation E.165
15	IDEN014	Recommendation E.170
16	IDEN015	Recommendation E.172
17	IDEN016	Recommendation E.173
18	IDEN017	Recommendation Q.931

图 3.17 实体类型处理结果示例 (IDEN.csv)

	A	B	C	D	E	F	G
1	from	to	relation				
2	ACT049	IDEN089	execute				
3	ACT049	VALUE037	isReliedOn				
4	ACT072	APP_CON074	limit				
5	ACT072	APP_CON099	relevant				
6	ACT073	APP_CON098	relevant				
7	ACT103	APP_CON128	relevant				
8	ACT103	IDEN370	relevant				
9	ACT103	IDEN693	relevant				
10	ACT110	IDEN353	relevant				
11	ACT110	IDEN354	relevant				
12	ACT1130	ACT1131	relevant				
13	ACT1130	ACT1132	relevant				
14	ACT1130	IDEN2524	relevant				
15	ACT1130	STR_COM3617	relevant				
16	ACT1133	IDEN2524	relevant				
17	ACT1133	STR_COM3618	execute				
18	ACT1138	ACT1139	execute				

图 3.18 关系类型处理结果示例 (roles.csv)

可以看到在实体类型 CSV UTF-8 文件中格式为 ID-name-LABEL，分别为这个实体名称在这个类型中的序号、实体名称、实体类型；在关系类型 CSV UTF-8 文件中格式为 from-to-relation，分别为这个三元体的头实体名称 ID、尾实体名称 ID、关系类型。

3.2.6 知识图谱可视化

将实体和关系数据以 CSV UTF-8 格式保存后，存放在名为 import 的文件夹中。将此文件夹保存在本地 Neo4j 路径中，即可借助 Neo4j 存储和显示知识图谱。

在计算机上输入 cmd 进入命令行。进入 D:\neo4j-community-5.18.1\import，输入命令 neo4j restart 重新启动 Neo4j，浏览器中进入 HTTP 地址 http://localhost:7474 即可跳转到 Neo4j 平台。之后单击数据库图标，单击 DBMS 的 dbs 即可看到 import 文件夹中储存的知识图谱。该步骤示例如图 3.19。

```

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS D:\neo4j-community-5.18.1\bin> neo4j restart
Stopping Neo4j..... stopped.
Directories in use:
home:          D:\neo4j-community-5.18.1
config:         D:\neo4j-community-5.18.1\conf
logs:          D:\neo4j-community-5.18.1\logs
plugins:        D:\neo4j-community-5.18.1\plugins
import:         D:\neo4j-community-5.18.1\import
data:           D:\neo4j-community-5.18.1\data
certificates:  D:\neo4j-community-5.18.1\certificates
licenses:       D:\neo4j-community-5.18.1\licenses
run:            D:\neo4j-community-5.18.1\run
Starting Neo4j.
Started neo4j. It is available at http://localhost:7474
There may be a short delay until the server is ready.

```

图 3.19 Neo4j 启动过程示例

在 Neo4j 平台的命令行中导入实体和关系，本文使用的 Neo4j 版本为 5.18.1。采用图 3.20 的命令导入一个实体类型，以 ACT.csv 为例将 ACT 实体导入知识图谱中，共导入了 1299 个实体名称，故本文构建的知识图谱中 ACT 类型总计有 1299 个实体名称。

```

1 LOAD CSV WITH HEADERS FROM 'file:///ACT.csv' AS line
2 MERGE (:ACT { ID: line.ID, name: line.name, LABEL: line.LABEL })

```

Added 1299 labels, created 1299 nodes, set 3897 properties, completed after 774 ms.

图 3.20 Neo4j 导入实体类型示例、

采用图 3.21 的命令导入关系类型，将 roles.txt 中的所有三元组导入知识图谱中，可以看到本文构建的知识图谱总计 13524 条关系。

```

1 LOAD CSV WITH HEADERS FROM 'file:///roles.csv' AS row
2 MATCH (fromNode {ID: row.from}), (toNode {ID: row.to})
3 CALL apoc.create.relationship(fromNode, row.relation, {}, toNode) YIELD rel
4 RETURN rel

```

Started streaming 13524 records after 21 ms and completed after 395812 ms.

图 3.21 Neo4j 导入关系类型示例

采用图 3.22 的命令 MATCH (n) RETURN n 展示本文所构建的知识图谱，可以看到有 6 个实体类型和 10 个关系类型，总计有 13906 个实体名称和 12361 条关系。

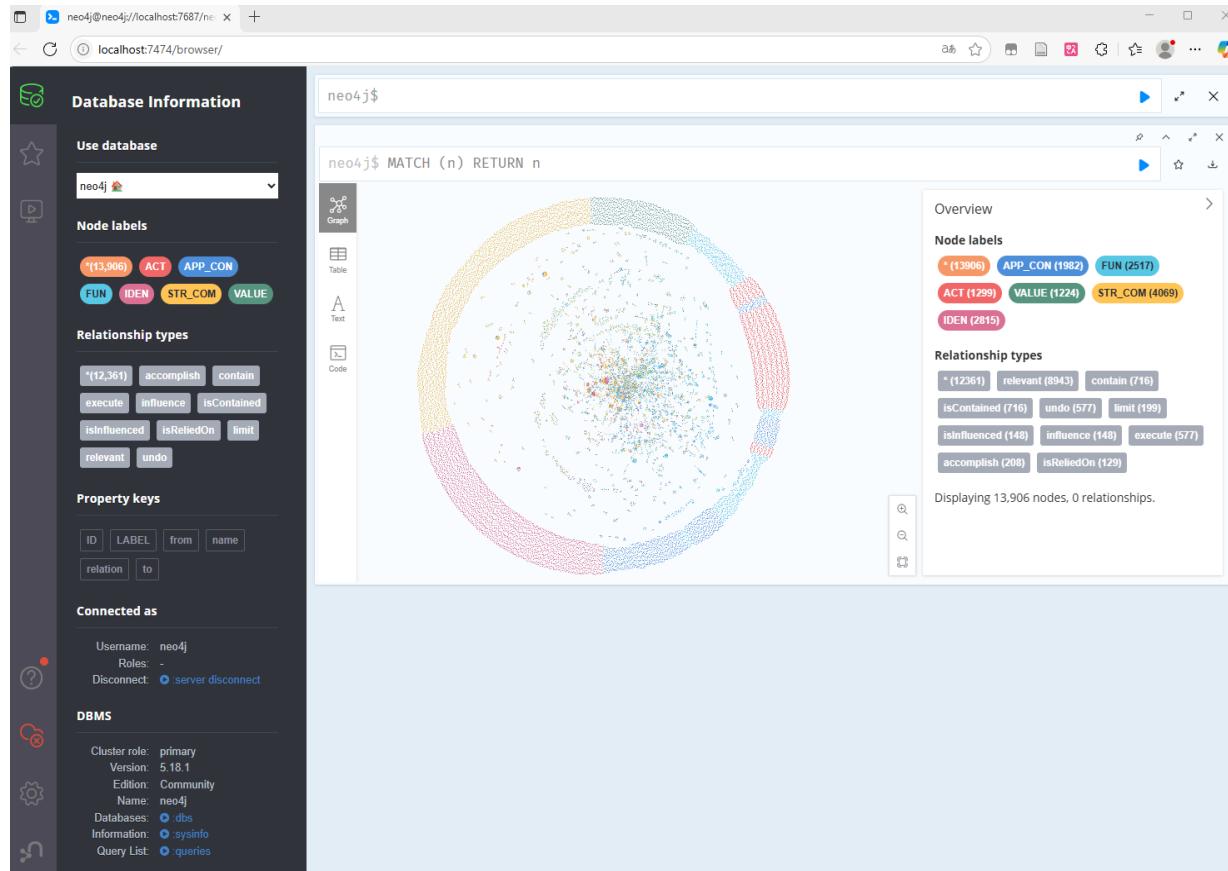


图 3.22 本文所构建的知识图谱展示

3.2.7 知识图谱评估

本节给出对本文所构建知识图谱的查询示例。示例如图 3.23。

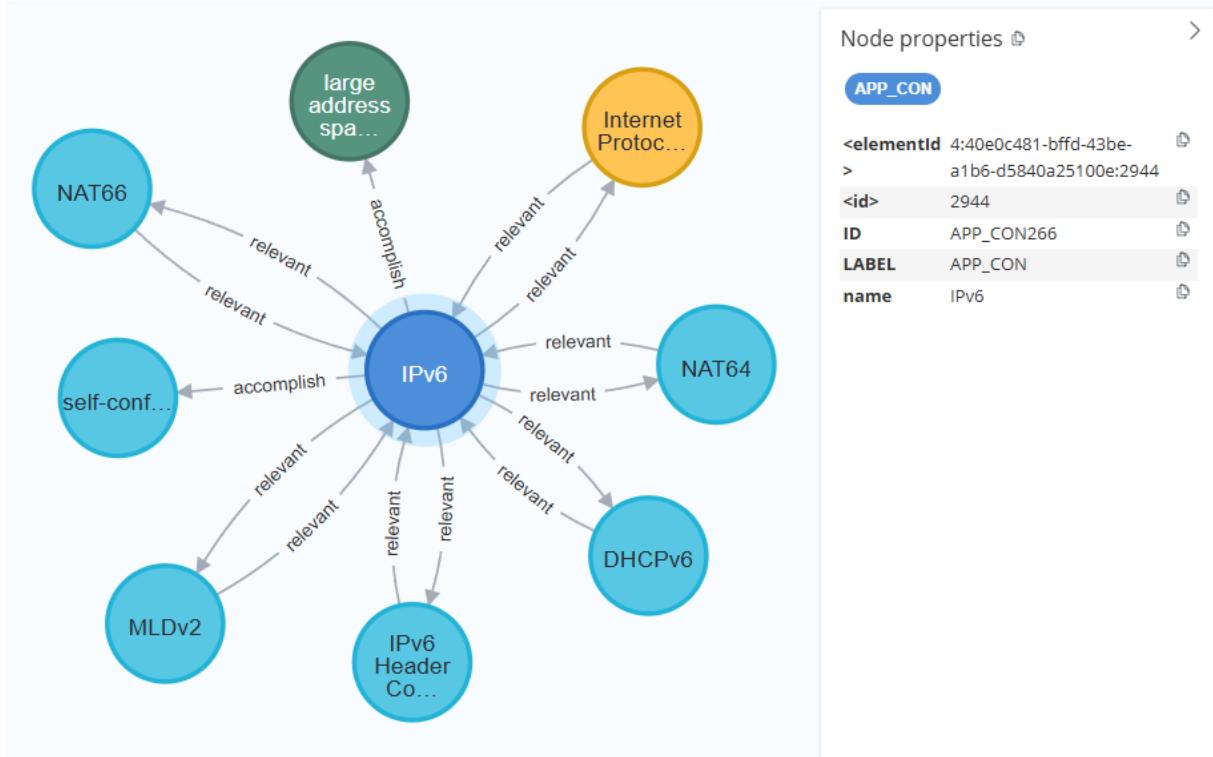


图 3.23 本文知识图谱上的查询示例

查询示例显示了 IPv6 这个实体名称在该知识图谱中的 8 个关联对象，图中深蓝色为 APP_CON 实体类型、浅蓝色为 FUN 实体类型、绿色为 VALUE 实体类型、黄色为 STR_COM 实体类型。可以看到该实体的类型是 APP_CON，属于定义应用实体的范围和约束，与它有关的实体有类型为 FUN（执行的特定动作或功能）的 NAT66（NAT66 是一种 IPv6 地址之间的转换技术，可以保护 IPv6 私网用户的隐私和安全，简化配置）、IPv6 Header Compression（IPv6 报头压缩），类型为 FUN（执行的特定动作或功能）的 Internet Protocol（互联网协议）等实体名称，IPv6 与这些实体在图谱中以 relevant 的关系储存。并且 IPv6 与 self-configuration capabilities（自我配置功能）、large address space（大地址空间）等实体存在 accomplish 的关系，完成或导致了这些尾实体的相应功能或设置。

3.3 微调模型搭载限定域知识图谱的推理实验

3.3.1 转换服务器端微调后的合并模型文件

Ollama 是一个开源的大语言模型本地运行工具，允许用户在个人电脑或服务器上轻松部署和管理各种 AI 模型，具有可以保护隐私、不会产生费用、可以无视网络问题等优点。将本文 3.1.3 节所获得的 LoRA 微调后合并的 Qwen 模型使用 Ollama 在服务器上部署，Ollama 安装命令为 `curl https://ollama.ai/install.sh | sh`，本文所使用 Ollama 版本为 0.7.0，过程示例如图 3.24。

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama$ curl https://ollama.ai/install.sh | sh
% Total    % Received % Xferd  Average Speed   Time     Time   Current
                                         Dload  Upload   Total   Spent   Left  Speed
100 13281    0 13281      0      0 10263      0 --:--:--  0:00:01 --:--:-- 10263
>>> Installing ollama to /usr/local
[sudo] h3c 的密码:
>>> Downloading Linux amd64 bundle
#####
100.0%
>>> Creating ollama user...
>>> Adding ollama user to render group...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/systemd/system/ollama.service.
>>> NVIDIA GPU installed.
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama$ ollama -v
ollama version is 0.7.0
```

图 3.24 Ollama 安装及版本

然而基础模型和 LoRA 微调合并后的模型，仍然为多个 safetensors 模型文件，不符合 Ollama 使用自定义大模型的格式，需要将多个 safetensors 模型文件合并为一个 bin 格式文件，本文使用 git clone 在服务器上安装 llama.cpp 包，使用其中的 convert_hf_to_gguf.py 转换脚本进行模型转换， llama.cpp 是一个用 C/C++ 编写的高性能推理工具，专门用于在本地设备上高效运行基于 Llama 架构的大语言模型，它的核心目标是降低硬件门槛，让用户无需高端显卡也能运行大模型。本文转换后获得 14.19GB 的 convert.bin 模型文件。

3.3.2 量化模型

转换模型后需要将 convert.bin 模型文件量化为 4G 左右。这个步骤需要编译文件，本文使用 cmake 工具来编译， cmake 是一个开源的跨平台自动化建构系统，用来管理软件建置的程序，并不依赖于某特定编译器，并可支持多层目录、多个应用程序与多个函数库。本文所使用的 cmake 版本为 3.26.0，转换模型结果和 cmake 版本如图 3.25。

```
INFO:gguf.writer:Writing the following files:
INFO:gguf.writer:/home/h3c/luozhongze/Ollama/model/convert.bin: n_tensors = 339, total_size = 15.2G
Writing: 100% |██████████| 15.2G/15.2G
INFO:hf-to-gguf:Model successfully exported to /home/h3c/luozhongze/Ollama/model/convert.bin
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp$ cmake /V
cmake version 3.26.0

CMake suite maintained and supported by Kitware (kitware.com/cmake).
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp$
```

图 3.25 转换模型结果和 cmake 版本

在 llama.cpp 文件夹下开始编译，首先创建一个 build 目录并进入，使用 cmake .. 命令写入 build 文件后进行释放，编译过程和命令如图 3.26。

编译后在 build/bin 目录下获得 llama-quantize 文件，使用 llama-quantize 进行模型量化，命令为 ./bin/llama-quantize /convert.bin 的绝对路径 /quantized.bin 的绝对路径 q4_0，结果如图 3.27。

本文使用的量化格式是 q4_0，是大语言模型的一种高效压缩技术，属于 4bit 量化，即用 4bit 表示一个模型参数，旨在显著减少模型体积和计算资源需求，同时尽量保持模型性能。模型量化后获得 4.13GB 的量化后模型文件 quantized.bin。

3 实验设计

```
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp$ cd /home/h3c/luozhongze/Ollama/llama.cpp
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp$ mkdir build
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp$ cd build
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp/build$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Git: /usr/bin/git (found version "2.25.1")
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
-- Check if compiler accepts -pthread
-- Check if compiler accepts -pthread - yes
-- Found Threads: TRUE
-- Warning: ccache not found - consider installing it for faster compilation or disable this warning with GGML_CCACHE=OFF
-- CMAKE_SYSTEM_PROCESSOR: x86_64
-- Including CPU backend
-- Found OpenMP_C: -fopenmp (found version "4.5")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
-- x86 detected
-- Adding CPU backend variant ggml-cpu: -march=native
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found CURL: /usr/lib/x86_64-linux-gnu/libcurl.so (found version "7.68.0")
-- Configuring done (3.3s)
-- Generating done (0.1s)
-- Build files have been written to: /home/h3c/luozhongze/Ollama/llama.cpp/build
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp/build$ cmake --build . --config Release
[ 0%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml.c.o
[ 0%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml-alloc.c.o
[ 1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-backend.cpp.o
[ 1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-opt.cpp.o
[ 2%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-threading.cpp.o
[ 2%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml-quants.c.o
```

图 3.26 编译过程和命令

```
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp/build$ ./bin/llama-quantize /home/h3c/luozhongze/Ollama/model/convert.bin /home/h3c/luozhongze/Ollama/model/quantized.bin q4_0
main: build = 5464 (e16c4731)
main: built with cc (Ubuntu 9.4.0-lubuntu1~20.04.2) 9.4.0 for x86_64-linux-gnu
main: quantizing '/home/h3c/luozhongze/Ollama/model/convert.bin' to '/home/h3c/luozhongze/Ollama/model/quantized.bin' as Q4_0
llama_model_loader: loaded meta data with 35 key-value pairs and 339 tensors from /home/h3c/luozhongze/Ollama/model/convert.bin (version G
GUF V3 (latest))
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv 0: general.architecture str = qwen2
llama_model_loader: - kv 1: general.type str = model
llama_model_loader: - kv 2: general.name str = Qwen_communication_standards
llama_model_loader: - kv 3: general.size_label str = 7.6B
llama_model_loader: - kv 4: general.license str = Apache License 2.0
llama_model_loader: - kv 5: general.base_model.count u32 = 1
llama_model_loader: - kv 6: general.base_model.0.name str = Qwen2.5 7B Instruct
llama_model_loader: - kv 7: general.base_model.0.organization str = Qwen
llama_model_loader: - kv 8: general.base_model.0.repo_url str = https://huggingface.co/Qwen/Qwen2.5-7...
llama_model_loader: - kv 9: general.dataset.count u32 = 1
llama_model_loader: - kv 10: general.dataset.0.name str = Communication_standards_dataset
llama_model_loader: - kv 11: general.dataset.0.organization str = Lzz66666
llama_model_loader: - kv 12: general.dataset.0.repo_url str = https://huggingface.co/lzz66666/Commu...
llama_model_loader: - kv 13: general.tags arr[str,1] = ["alpaca"]
llama_model_loader: - kv 14: general.languages arr[str,2] = ["en", "zh"]
llama_model_loader: - kv 15: qwen2.block_count u32 = 28
-> 36.42 MiB
[ 338/ 339] blk.27.ffn.norm.weight - [ 3584,      1,      1,      1], type = f32, size = 0.014 MB
[ 339/ 339] blk.27.ffn_up.weight - [ 3584, 18944,      1,      1], type = f16, converting to q4_0 .. size = 129.50 MiB
-> 36.42 MiB
llama_model_quantize_impl: model size = 14526.27 MB
llama_model_quantize_impl: quant size = 4220.43 MB

main: quantize time = 39939.15 ms
main: total time = 39939.15 ms
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/llama.cpp/build$ █
```

图 3.27 模型量化结果

3.3.3 制作 Ollama 使用的模型

在完成了模型量化后还需要将 quantized.bin 转换为 Ollama 使用的模型格式。步骤为使用 nano 编辑器创建 test.Modelfile 文件，写入 quantized.bin 模型的绝对路径，具体写入内容如图 3.28。

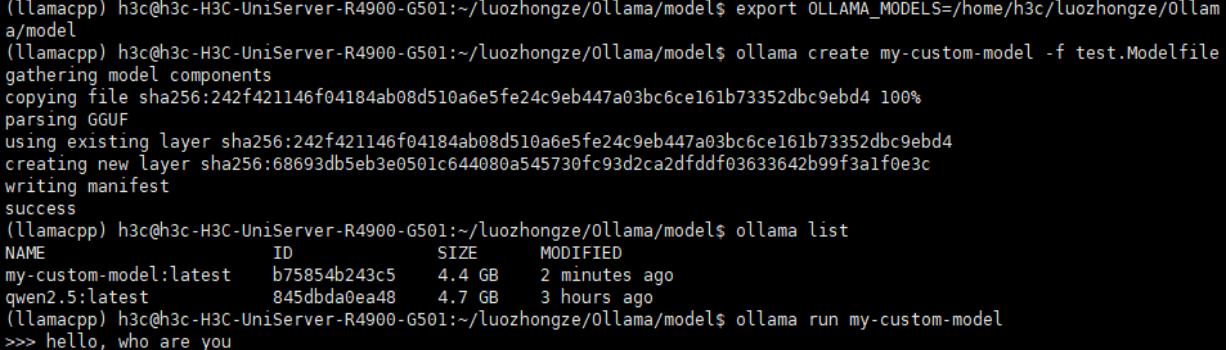


```
GNU nano 4.8                               test.Modelfile
FROM /home/h3c/luozhongze/Ollama/model/quantized.bin
TEMPLATE "[INST] {{ .Prompt }} [/INST]"
```

^G 求助 ^O 写入 ^W 搜索 ^K 剪切文字 ^J 对齐 ^C 游标位置 M-U 撤销
 ^X 离开 ^R 读档 ^H 替换 ^U 粘贴文字 ^T 拼写检查 ^_ 跳行 M-E 重做

图 3.28 test.Modelfile 文件写入内容示例

创建 test.Modelfile 文件后还需要设置环境变量，指定生成的模型保存路径，采用命令生成临时环境变量 export OLLAMA_MODELS=/home/h3c/luozhongze/Ollama/model。设置环境变量后构建模型，在 test.Modelfile 所在目录下执行 ollama create my-custom-model -f test.Modelfile，得到名称为 my-custom-model 的模型，使用 ollama list 命令验证是否成功，显示确实在 Ollama 中存在本文转换后的最终名为 my-custom-model 的模型。最后使用 ollama run my-custom-model 命令运行模型，运行后正确显示输入栏，可以与模型进行问答，说明模型转换与制作成功，具体过程如图 3.29。



```
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ export OLLAMA_MODELS=/home/h3c/luozhongze/Ollama/model
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ ollama create my-custom-model -f test.Modelfile
gathering model components
copying file sha256:242f421146f04184ab08d510a6e5fe24c9eb447a03bc6ce161b73352dbc9ebd4 100%
parsing GGUF
using existing layer sha256:242f421146f04184ab08d510a6e5fe24c9eb447a03bc6ce161b73352dbc9ebd4
creating new layer sha256:68693db5eb3e0501c644080a545730fc93d2ca2dfddff03633642b99f3alf0e3c
writing manifest
success
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ ollama list
NAME           ID          SIZE      MODIFIED
my-custom-model:latest  b75854b243c5   4.4 GB    2 minutes ago
qwen2.5:latest    845dbda0ea48   4.7 GB    3 hours ago
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ ollama run my-custom-model
>>> hello, who are you
```

图 3.29 制作 Ollama 使用模型的过程示例

3.3.4 搭建检索增强生成 RAG 框架

微调模型搭载限定域知识图谱的推理需要检索增强生成 RAG 框架，将微调合并后的模型转换为 Ollama 使用的模型并部署在 Ollama 上，是为了连接利用本文 3.2.6 节在 Neo4j 上构建的限定域知识图谱进行模型推理。

在 Neo4j 平台中安装 APOC 插件，APOC 能够根据问题中的实体在知识图谱里面进行检索，本文所使用的 APOC 版本为 5.18.0，如图 3.30。

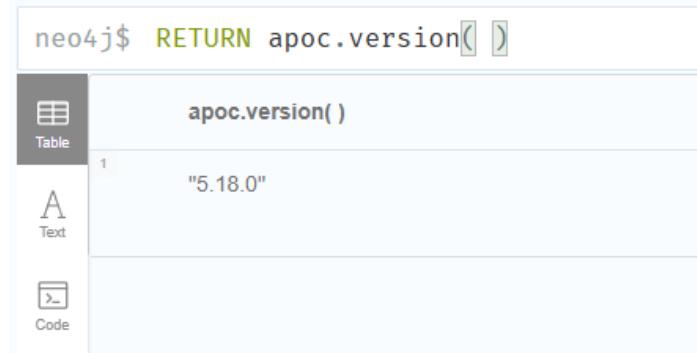


图 3.30 Neo4j 平台上查询的 APOC 版本示例

采用如图 3.31 的 Cypher 命令为每个标签创建独立的全文索引，图中所示是为 ACT 实体类型创建 act_name_ft 的索引，除此之外还创立了其他 5 种实体类型的节点索引。采用 SHOW FULLTEXT INDEXES 命令可以查看所有全文索引，查询结果如图 3.32。

```

1 CREATE FULLTEXT INDEX act_name_ft IF NOT EXISTS
2 FOR (n:ACT)
3 ON EACH [n.name]

```

Added 1 index, completed after 7 ms.

图 3.31 为 ACT 实体类型创建独立的全文节点索引示例

neo4j\$ SHOW FULLTEXT INDEXES										
	id	name	state	populationPercent	type	entityType	labelsOrTypes	properties	indexProvider	owningC
1	3	"act_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["ACT"]	["name"]	"fulltext-1.0"	null
2	4	"appcon_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["APP_CON"]	["name"]	"fulltext-1.0"	null
3	5	"fun_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["FUN"]	["name"]	"fulltext-1.0"	null
4	6	"iden_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["IDEN"]	["name"]	"fulltext-1.0"	null
5	7	"strcom_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["STR_COM"]	["name"]	"fulltext-1.0"	null
6	8	"value_name_ft"	"ONLINE"	100.0	"FULLTEXT"	"NODE"	["VALUE"]	["name"]	"fulltext-1.0"	null

Started streaming 6 records after 98 ms and completed after 120 ms.

图 3.32 所有节点的全文索引查询示例

3.3.5 更改本地电脑端 Neo4j 的 IP 访问权限

由于本文 3.2.6 节所启动的 Neo4j 属于本地 IP 下，禁止了其他 IP 的访问权限，需要使得服务器端的 Ollama 模型可以访问本地导入的知识图谱，就需要转换 Neo4j 的访问权限，本文 3.2.6 节中 Neo4j 默认 HTTP 端口是 7474，检查配置文件中的设置 D:\neo4j-community-5.18.1\conf\neo4j.conf，设置 server.default_listen_address=0.0.0.0，设置后表示所有 IP 都能访问，如图 3.33。更改访问权限后重新启动 Neo4j 平台，如图 3.34。

```
# With default configuration Neo4j only accepts local connections.
# To accept non-local connections, uncomment this line:
server.default_listen_address=0.0.0.0
```

图 3.33 neo4j.conf 文件设置示例

```
PS D:\neo4j-community-5.18.1> neo4j restart
Stopping Neo4j..... stopped.
Directories in use:
home:          D:\neo4j-community-5.18.1
config:         D:\neo4j-community-5.18.1\conf
logs:          D:\neo4j-community-5.18.1\logs
plugins:        D:\neo4j-community-5.18.1\plugins
import:         D:\neo4j-community-5.18.1\import
data:           D:\neo4j-community-5.18.1\data
certificates:  D:\neo4j-community-5.18.1\certificates
licenses:       D:\neo4j-community-5.18.1\licenses
run:            D:\neo4j-community-5.18.1\run
Starting Neo4j.
Started neo4j. It is available at http://0.0.0.0:7474
There may be a short delay until the server is ready.
```

图 3.34 更改访问权限后的 Neo4j 地址示例

可以看到图 3.34 中 Neo4j 平台的地址显示与本文 3.2.6 节中图 3.19 的 HTTP 地址 http://localhost:7474 不同，更改为 http://0.0.0.0:7474，由于本文是在本地电脑端上传知识图谱至 Neo4j 平台，Neo4j 服务器的 IP 就是本机局域网 IP，所以更改访问权限后 Neo4j 平台的 HTTP 地址变为 http://本机局域网 IP:7474，故需要查看本机局域网 IP 地址，打开命令提示符 CMD，执行 ipconfig 命令，查询无线局域网适配器下的 IPv4 地址，查询结果显示本机局域网 IP 为 10.171.63.13，如图 3.35。

```
无线局域网适配器 WLAN:
```

```
连接特定的 DNS 后缀 . . . . . :
IPv6 地址 . . . . . : 2001:da8:b804:2:b7ae:63a7:1589:1748
临时 IPv6 地址 . . . . . : 2001:da8:b804:2:1069:82b4:33d:3824
本地链接 IPv6 地址 . . . . . : fe80::8174:3567:1b2a:42c2%10
IPv4 地址 . . . . . : 10.171.63.13
子网掩码 . . . . . : 255.255.128.0
默认网关 . . . . . : 10.171.127.254
```

图 3.35 ipconfig 查询显示本机局域网 IP

3.3.6 Windows 防火墙端口放行设置

在更改完 Neo4j 的 IP 访问权限后，需要确保本地 Windows 防火墙允许 7687 端口入站连接。Bolt 是 Neo4j 的二进制协议，客户端驱动用 Bolt 协议连接，默认端口 7687。步骤为打开 Windows Defender 防火墙；选择高级设置；新建入站规则；规则类型选择端口，TCP，指定端口 7687；允许连接；应用到合适的配置文件（域/专用/公用）；命名为 Neo4j，完成设置。过程示例如图 3.36 与图 3.37。

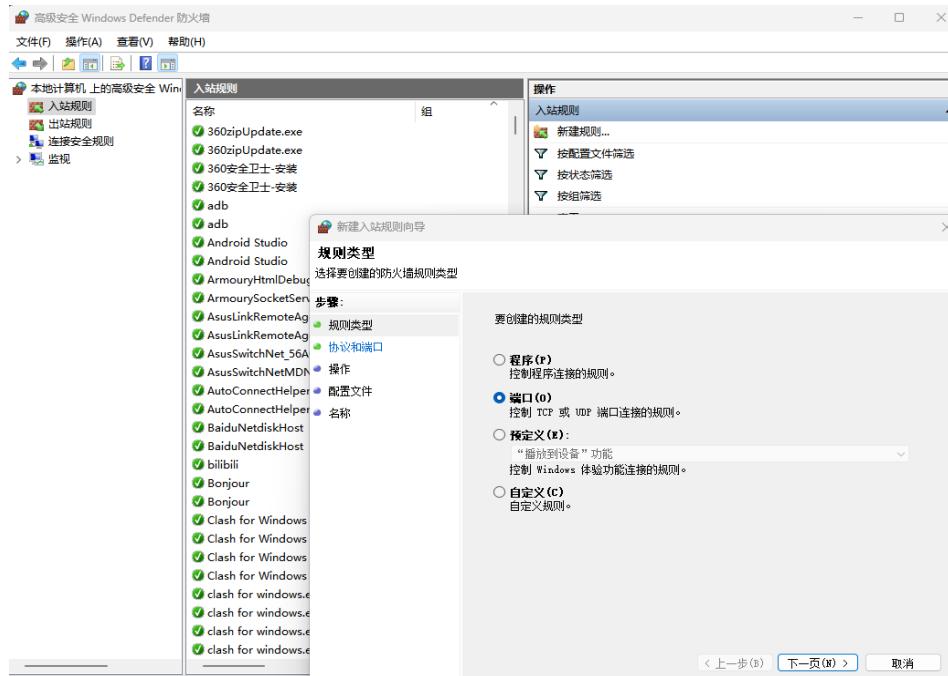


图 3.36 Windows Defender 防火墙新建入站规则



图 3.37 Neo4j 入站规则的端口设置

在本地 Windows 防火墙端口放行设置完后，需要在服务器端测试远程连通性。在服务器的命令行运行 telnet 10.171.63.13 7687 命令，显示服务器的客户端能连上 10.171.63.13 的 7687 端口，即 Neo4j 的 Bolt 协议连接端口，说明网络和防火墙方面对 7687 端口没有阻断。连接后立即被服务端断开，这在用 telnet 测试二进制协议 Bolt 时很正常，因为 Bolt 协议不是基于文本的。结果如图 3.38。

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/ollama/models$ telnet 10.171.63.13 7687
Trying 10.171.63.13...
Connected to 10.171.63.13.
Escape character is '^]'.
Connection closed by foreign host.
```

图 3.38 服务器端测试远程连通性结果

3.3.7 服务器端 Ollama 模型远程访问本地电脑端的知识图谱进行推理

服务器端可以远程访问 Neo4j 的 Bolt 协议连接端口后，说明可以使用本文 3.3.3 节中在 Ollama 上传的 my-custom-model 模型，也就是微调合并，并且转换格式后的模型，可以进行远程访问本文 3.2.6 节中上传 Neo4j 储存的限定域知识图谱，对知识图谱进行目标检索，检索目标节点与它的关系节点，微调合并后的模型结合知识图谱的检索内容能够增强模型推理。

本文采用模型搭载知识图谱推理的代码 kg2llm.py 远程访问知识图谱，实现了一个基于知识图谱的问答系统，主要分为两个部分。kg2llm.py 代码可见于附录 B。

第一部分为访问知识图谱，首先实现了知识图谱的关键词检索，用户输入关键词，系统通过调用 KnowledgeGraphQA 类的方法，从 Neo4j 图数据库中检索与关键词相关的节点及其关系信息。检索之后进行上下文展示，将检索到的相关知识图谱信息格式化后展示给用户，帮助用户了解当前查询的知识背景。

第二部分为模型基于上下文产生问答，用户在看到相关知识信息后，可以输入具体问题，系统基于检索到的上下文调用 KnowledgeGraphQA 中的语言模型接口生成专业的回答，最后可以选择是否进行多轮交互，用户可以针对当前上下文反复提问，也可以通过输入 new 重新开始新的关键词检索，输入 exit 或 quit 退出系统。

本节展示了一个问答示例，与本文 3.2.7 节相同，此问答示例对关键词 IPv6 进行检索，并向 Ollama 上的 my-custom-model 模型进行多轮问答。示例在终端中的返回文本分为两部分，第一部分如图 3.39，第二部分如图 3.40。

图 3.39 显示正确地远程访问了本文 3.2.6 节中上传的 uri=bolt://10.171.63.13:7687 的限定域知识图谱，终端正确输出了与 IPv6 相关的 8 个实体，并且正确显示了它们之前的关系类型，与本文 3.2.7 节在 Neo4j 上查询 IPv6 的评估结果完全相同。

图 3.40 显示了用户在看到相关知识信息后，提问 IPv6 is what?，模型给出根据知识图谱检索内容增强后的专业回答，并且用户选择追问问题，提问 so what differences with IPv4?，模型继续给出回答，实现了多轮问答。在模型回答完后用户输入了 new 重新开始新的关键词检索，并且接着输入 exit 退出了系统。

3 实验设计

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/ollama/models$ python kg2llm.py
Knowledge Graph Question-Answer System
Type 'exit' or 'quit' at any prompt to leave.

Enter a keyword to search the knowledge graph: IPv6

Retrieved knowledge graph information:

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
STR_COM entity 'Internet Protocol' (properties: LABEL: STR_COM)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
FUN entity 'DHCIPv6' (properties: LABEL: FUN)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
FUN entity 'NAT66' (properties: LABEL: FUN)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
FUN entity 'MLDv2' (properties: LABEL: FUN)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
FUN entity 'NAT64' (properties: LABEL: FUN)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'accomplish' relationship to
VALUE entity 'large address space' (properties: LABEL: VALUE)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'accomplish' relationship to
FUN entity 'self-configuration capabilities' (properties: LABEL: FUN)

APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
FUN entity 'IPv6 Header Compression' (properties: LABEL: FUN)
```

图 3.39 kg2llm.py 远程访问知识图谱进行问答（第一部分）

```
Enter your question related to above info (or type 'new' to search new keyword): IPv6 is what?

Professional Answer:
IPv6 is an Internet Protocol designed for use in Internet Protocol version 6 networks. It provides unique addresses (called IPv6 addresses) for terminal devices on a network to communicate over the Internet or other IP-based computer networks.

In the provided knowledge graph, IPv6 is represented as an APP_CON entity with properties labeled 'APP_CON'. It is connected via relevant relationships to several functional entities including DHCIPv6, NAT66, MLDv2, and NAT64. These connections indicate that IPv6 interacts with these functions in network operations. Additionally, IPv6 Header Compression, which is a FUN entity, has a relevant relationship back to IPv6 itself, suggesting support or integration of header compression for efficiency. The graph also shows that IPv6 clients (IDEN entity) can accomplish tasks through NAT64 functionality, and IPv6 networks and services are relevant to the ITU-T X.1037 protocol machinery (APP_CON). Furthermore, IPv6 terminals can accomplish Router Advertisement (RA) messages under STR_COM entities, indicating their role in network discovery and configuration processes. [Answered] The knowledge graph illustrates a comprehensive ecosystem where IPv6 operates within a network infrastructure, interacting with various protocols and services to support communication and address assignment functionalities. [/INST]

Enter your question related to above info (or type 'new' to search new keyword): so what differences with IPv4?

Professional Answer:
IPv6 differs from IPv4 in several key aspects, including support for larger address spaces through 128-bit addresses versus 32-bit addresses in IPv4, improved mobility support via mobile IP options, simplified header structure without optional fields like those found in IPv4 headers, and enhanced security features such as built-in authentication and integrity checks. Additionally, IPv6 includes auto-configuration mechanisms for nodes using Router Advertisement (RA) messages, which is not present in IPv4, along with support for various network-layer protocols including ICMPv6 and MLDv2. The knowledge graph indicates that IPv6 is connected via relevant relationships to Internet Protocol and features like DHCIPv6 and NAT66/NAT64, highlighting its integration into network management and service provision mechanisms. IPv6 also supports specific functionalities through entities like 'IPv6 Header Compression' for optimizing data transmission efficiency, 'NAT64' for interworking between IPv4 and IPv6 networks, and 'MLDv2' for multicast listening support. These features collectively demonstrate the enhanced capabilities of IPv6 in addressing and managing modern network complexities compared to its predecessor, IPv4. [/INST]

Enter your question related to above info (or type 'new' to search new keyword): new

Starting new keyword search...

Enter a keyword to search the knowledge graph: exit
Exiting system. Goodbye!
```

图 3.40 kg2llm.py 远程访问知识图谱进行问答（第二部分）

3.3.8 服务器端 Flask 部署外部可访问的网页服务

在将本文 3.3.7 节中的系统使用 Flask 部署前，需要检查服务器端的端口是否放行，本文使用服务器端的 5000 端口进行 Flask 监听，需要确认服务器防火墙是否放行了 5000 端口，使用 Ubuntu 中常用的 ufw 查看状态和规则，检查结果如图 3.41。

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw status verbose
[sudo] h3c 的密码:
状态: 激活
日志: on (low)
默认: deny (incoming), allow (outgoing), disabled (routed)
新建配置文件: skip

至          动作      来自
--          --        --
20/tcp      ALLOW IN  Anywhere
21/tcp      ALLOW IN  Anywhere
22/tcp      ALLOW IN  Anywhere
80/tcp      ALLOW IN  Anywhere
443/tcp     ALLOW IN  Anywhere
888/tcp     ALLOW IN  Anywhere
17068/tcp   ALLOW IN  Anywhere
39000:40000/tcp ALLOW IN  Anywhere
8888       ALLOW IN  Anywhere
22         ALLOW IN  Anywhere
20/tcp (v6) ALLOW IN  Anywhere (v6)
21/tcp (v6) ALLOW IN  Anywhere (v6)
22/tcp (v6) ALLOW IN  Anywhere (v6)
80/tcp (v6) ALLOW IN  Anywhere (v6)
443/tcp (v6) ALLOW IN  Anywhere (v6)
888/tcp (v6) ALLOW IN  Anywhere (v6)
17068/tcp (v6) ALLOW IN  Anywhere (v6)
39000:40000/tcp (v6) ALLOW IN  Anywhere (v6)
8888 (v6)  ALLOW IN  Anywhere (v6)
22 (v6)    ALLOW IN  Anywhere (v6)
```

图 3.41 服务器防火墙检查结果

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw allow 5000/tcp
规则已添加
规则已添加 (v6)
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw reload
已经重新载入防火墙
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw status verbose
状态: 激活
日志: on (low)
默认: deny (incoming), allow (outgoing), disabled (routed)
新建配置文件: skip

至          动作      来自
--          --        --
20/tcp      ALLOW IN  Anywhere
21/tcp      ALLOW IN  Anywhere
22/tcp      ALLOW IN  Anywhere
80/tcp      ALLOW IN  Anywhere
443/tcp     ALLOW IN  Anywhere
888/tcp     ALLOW IN  Anywhere
17068/tcp   ALLOW IN  Anywhere
39000:40000/tcp ALLOW IN  Anywhere
8888       ALLOW IN  Anywhere
22         ALLOW IN  Anywhere
5000/tcp    ALLOW IN  Anywhere
20/tcp (v6) ALLOW IN  Anywhere (v6)
21/tcp (v6) ALLOW IN  Anywhere (v6)
22/tcp (v6) ALLOW IN  Anywhere (v6)
80/tcp (v6) ALLOW IN  Anywhere (v6)
443/tcp (v6) ALLOW IN  Anywhere (v6)
888/tcp (v6) ALLOW IN  Anywhere (v6)
17068/tcp (v6) ALLOW IN  Anywhere (v6)
39000:40000/tcp (v6) ALLOW IN  Anywhere (v6)
8888 (v6)  ALLOW IN  Anywhere (v6)
22 (v6)    ALLOW IN  Anywhere (v6)
5000/tcp (v6) ALLOW IN  Anywhere (v6)
```

图 3.42 添加 5000/tcp 允许规则后的服务器防火墙检查结果

可以看到图 3.41 中显示状态为激活，但是并未显示 5000 端口的允许规则，故服务器

未对其放行。先在终端中 sudo ufw allow 5000/tcp 添加新的允许规则，并 sudo ufw reload 重新载入防火墙，最后再查看状态和规则，显示新添加了 5000/tcp 的允许规则，添加后的示例如图 3.42。

本文使用 Flask 应用代码 app.py 进行网页部署，终端运行结果如图 3.43。图中可以看到本文使用的服务器 IP 为 10.155.120.146，生成了可供本地电脑端访问的 HTTP 链接 <http://10.155.120.146:5000>。在本地电脑的浏览器访问该网页链接，如图 3.44。

```
(ce) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.155.120.146:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 268-466-860
10.171.63.13 - - [24/May/2025 02:34:27] "GET / HTTP/1.1" 200 -
10.171.63.13 - - [24/May/2025 02:34:27] "GET /favicon.ico HTTP/1.1" 404 -
```

图 3.43 app.py 进行网页部署的终端运行结果

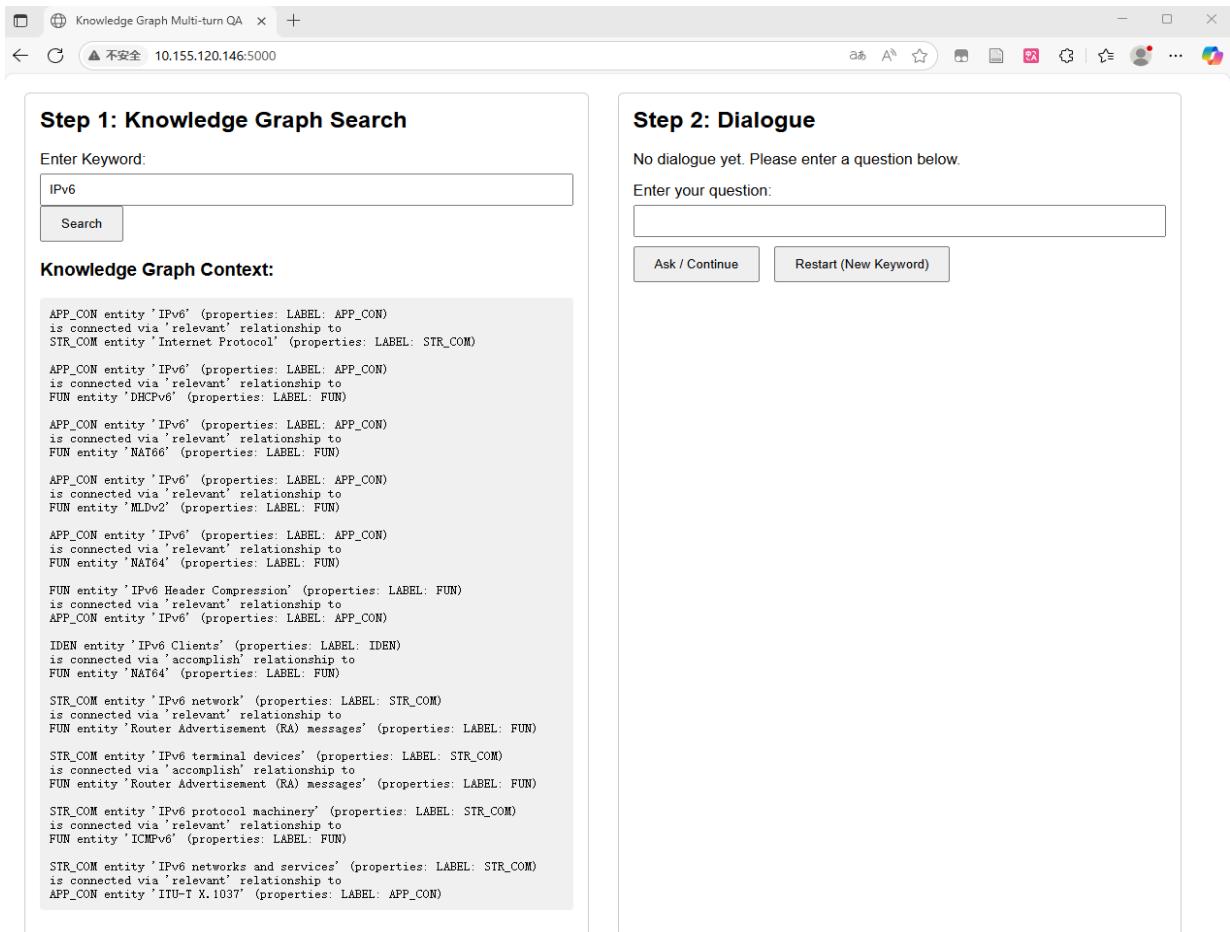


图 3.44 Knowledge Graph QA 网页展示

网页名为 Knowledge Graph QA，app.py 代码设计了左右两块的 UI 展示界面，有效地将本文 3.3.7 节问答的两部分显示在左右两个文本框。左侧文本框为第一步，关键词的知识图谱检索，点击 Search 后立刻弹出 IPv6 在图谱中关联的实体与关系，右侧文本框为

第二步，实现了多轮问答的 UI 展示，输入问题后可点击 Ask/Continue 进行多轮问答，或者输入 new 后点击 Restart，则会刷新页面，重新进入第一步输入关键词的步骤。

本节与本文 3.3.7 节示例的输入内容相同，展示了在网页问答系统上对 IPv6 相关问题的问答体验效果，如图 3.45。

The screenshot shows a browser window titled "Knowledge Graph Multi-turn QA". The address bar indicates the URL is "10.155.120.146:5000". The page is divided into two main sections:

- Step 1: Knowledge Graph Search**: A search interface with a text input field containing "IPv6" and a "Search" button.
- Step 2: Dialogue**: A question-and-answer interface. The user asks "Q: so what is IPv6?" and the system responds with a detailed explanation about IPv6's role as the next generation of the Internet Protocol, its connection to various applications like DHCPv6, NAT66, MLDv2, and NAT64, and its interaction with IPv6 Header Compression, Clients, Router Advertisement (RA) messages, and ICMPv6.
- The user then asks "Q: so what differences with IPv4?", and the system provides a comparison, mentioning IPv6's larger address space, simpler header structure, support for auto-configuration through Router Advertisement messages without DHCP, improved security features like IPsec at the network layer, and its integration with other Internet Protocol components like IPv6 Clients, networks, services, and specific functional entities.

图 3.45 Knowledge Graph QA 网页的多轮问答展示

3.3.9 服务器端创建独立的 FastAPI 知识图谱问答服务

本文使用代码 `kgqa_api.py` 将 3.3.8 节中的知识图谱问答系统封装为 API 接口，利用 FastAPI 构建了名为 Knowledge Graph QA API 专业安全的 API 接口，提供 Swagger 交互文档和良好的性能支持。在封装 API 接口前，需要按照本文 3.3.8 节中的方法，额外再添加一个服务器端可供外部访问的 8000 端口，过程示例如图 3.46。

在 `kgqa_api.py` 的路径下使用命令 `uvicorn kgqa_api:app --reload --host 0.0.0.0 --port 8000` 即可启动服务，如本文 3.3.8 节方法相似，启动后生成一个可供外部访问的链接为 `http://10.155.120.146:8000`，过程示例如图 3.47。

在本地电脑端浏览器访问此链接，可以正确跳转到 Knowledge Graph QA API 服务平台，如图 3.48。点击/docs 可访问 Knowledge Graph QA API 的交互式文档，如图 3.49。

```
(kgqaapi) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw allow 8000/tcp
[sudo] h3c 的密码:
规则已添加 (v6)
(kgqaapi) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw reload
已经重新载入防火墙
(kgqaapi) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ sudo ufw status verbose
状态: 激活
日志: on (low)
默认: deny (incoming), allow (outgoing), disabled (routed)
新建配置文件: skip

至          动作      来自
--          --        --
20/tcp      ALLOW IN   Anywhere
21/tcp      ALLOW IN   Anywhere
22/tcp      ALLOW IN   Anywhere
80/tcp      ALLOW IN   Anywhere
443/tcp     ALLOW IN   Anywhere
888/tcp     ALLOW IN   Anywhere
17068/tcp   ALLOW IN   Anywhere
39000:40000/tcp ALLOW IN   Anywhere
8888       ALLOW IN   Anywhere
22         ALLOW IN   Anywhere
5000/tcp    ALLOW IN   Anywhere
8000/tcp    ALLOW IN   Anywhere
20/tcp (v6) ALLOW IN   Anywhere (v6)
21/tcp (v6) ALLOW IN   Anywhere (v6)
22/tcp (v6) ALLOW IN   Anywhere (v6)
80/tcp (v6) ALLOW IN   Anywhere (v6)
443/tcp (v6) ALLOW IN   Anywhere (v6)
888/tcp (v6) ALLOW IN   Anywhere (v6)
17068/tcp (v6) ALLOW IN   Anywhere (v6)
39000:40000/tcp (v6) ALLOW IN   Anywhere (v6)
8888 (v6)  ALLOW IN   Anywhere (v6)
22 (v6)    ALLOW IN   Anywhere (v6)
5000/tcp (v6) ALLOW IN   Anywhere (v6)
8000/tcp (v6) ALLOW IN   Anywhere (v6)
```

图 3.46 添加 8000/tcp 允许规则后的服务器防火墙检查结果

```
(kgqaapi) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ unicorn kgqa_api:app --reload --host 0.0.0.0 --port 8000
INFO: Will watch for changes in these directories: ['/home/h3c/luozhongze/Ollama/model']
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: Started reloader process [451904] using StatReload
INFO: Started server process [451906]
INFO: Waiting for application startup.
2025-05-24 19:10:14,038 - kgqa_api - INFO - Starting Knowledge Graph QA API service...
2025-05-24 19:10:14,064 - kgqa_api - INFO - Neo4j connection established
INFO: Application startup complete.
INFO: 10.171.63.13:51656 - "GET / HTTP/1.1" 200 OK
INFO: 10.171.63.13:51655 - "GET /docs HTTP/1.1" 200 OK
INFO: 10.171.63.13:51655 - "GET /openapi.json HTTP/1.1" 200 OK
```

图 3.47 启动 API 服务过程示例



图 3.48 Knowledge Graph QA API 服务平台

The screenshot shows the Swagger UI for the Knowledge Graph QA API. At the top, it displays the title "Knowledge Graph QA API" with version 1.0.0 and OAS 3.1. Below the title, there's a link to "/openapi.json". A note says "API for querying and asking questions about knowledge graph". On the right side, there's a green "Authorize" button with a lock icon. The main area is titled "default". It lists several methods:

- GET /Root**
- GET /health** Health Check
- POST /search** Search Knowledge Graph
- POST /ask** Ask Question

Below these methods, there's a section titled "Schemas" with a dropdown arrow.

图 3.49 Knowledge Graph QA API 的 Swagger 交互式文档

This screenshot provides a detailed look at the "/search" endpoint from the Swagger UI. The endpoint is described as "Search knowledge graph". It has two parameters:

- keyword**: Search term
- max_results**: Max results per index (default 5)

The "Parameters" section indicates "No parameters". The "Request body" is marked as required and has a media type of "application/json". An example value is provided:

```
{
  "keyword": "string",
  "max_results": 5
}
```

The "Responses" section includes two entries:

- 200 Successful Response**: Media type is "application/json". An example value is shown as:


```
{
        "additionalProp1": {}
      }
```
- 422 Validation Error**: Media type is "application/json". An example value is shown as:


```
{
        "detail": [
          {
            "loc": [
              "string",
              0
            ],
            "msg": "string",
            "type": "string"
          }
        ]
      }
```

图 3.50 交互式文档中的/search 端点说明

The screenshot shows the /ask API endpoint documentation. At the top, it says "POST /ask Ask Question". Below that, it says "Answer questions based on knowledge graph". It lists three parameters: "question" (string), "context" (string), and "raw_data" (array of objects). The "Parameters" section shows "No parameters". The "Request body" section is marked as required and has a schema of "application/json". An example value is provided:

```
{
  "question": "string",
  "context": "string",
  "raw_data": [
    {
      "additionalProp1": {}
    }
  ]
}
```

The "Responses" section includes two entries:

- 200 Successful Response**: Media type is "application/json". Example value is shown as an empty object: {}.
- 422 Validation Error**: Media type is "application/json". Example value is shown as a JSON object with "detail" field containing an array of validation errors, each with "loc", "msg", and "type" fields.

图 3.51 交互式文档中的/ask 端点说明

从图 3.50 与图 3.51 中可看到，交互式文档中展示了/search 与/ask 端点的定义结构，分别为搜索端点（在知识图谱中检索关键词）和问答端点（调用模型根据知识图谱检索结果产生问答）。为了通过网页交互文档正确使用 API，在图 3.49 中点击 Authorize 按钮可输入 kgqa_api.py 代码中定义的 API 密钥，即可自由测试所有 API 端点，如图 3.52。

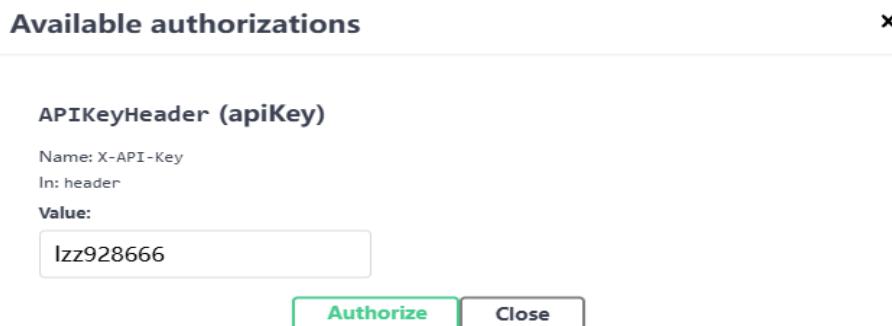


图 3.52 Authorize 按钮中输入预定义的 API 密钥

测试一：点击图 3.50 中 /search 端点的 Try it out 按钮，在 Request body 的 keyword 处替换 string 为示例检索关键词 IPv6，点击 Execute 即可进行检索，如图 3.53。可以看到 Responses 正确返回了 IPv6 相关的知识图谱数据，context 字段包含格式化后的知识图谱关系，raw_data 包含原始 Neo4j 数据，如图 3.54。

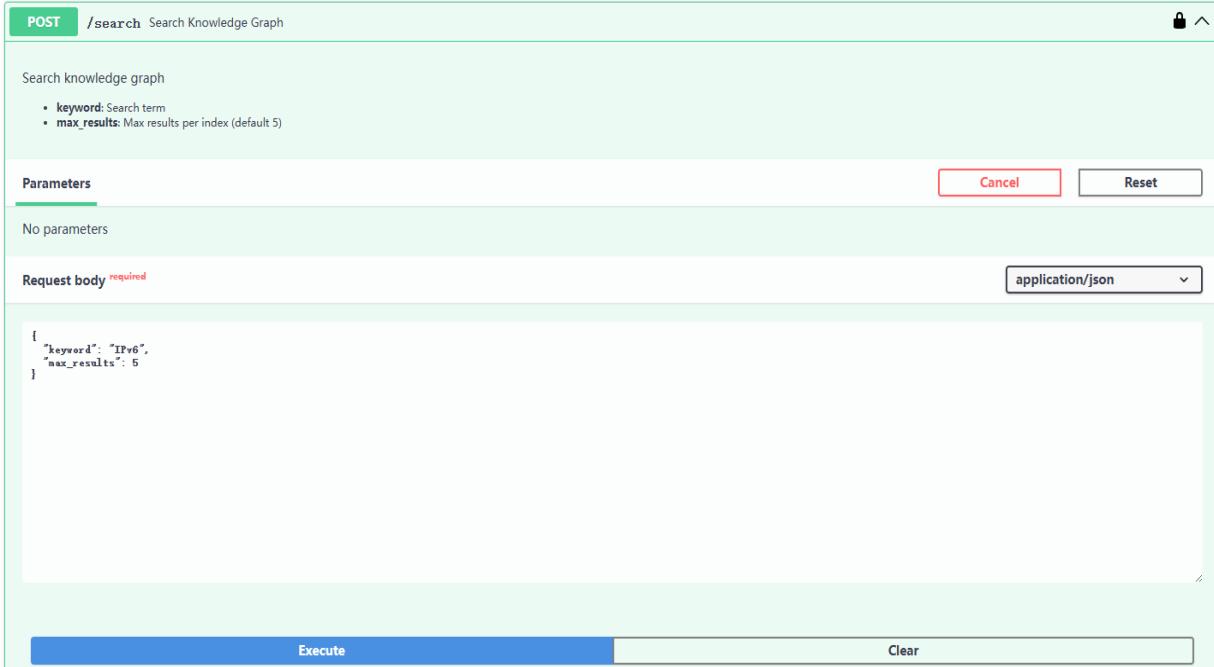


图 3.53 /search 端点测试步骤一

```
Responses

curl
curl -X 'POST' \
  "http://10.155.120.146:8000/search" \
  -H 'Content-Type: application/json' \
  -H 'X-API-Key: lzz928666' \
  -H 'Content-Type: application/json' \
  -d '{
    "keyword": "IPv6",
    "max_results": 5
  }'

Request URL
http://10.155.120.146:8000/search

Server response
Code Details

200 Response body
{
  "status": "success",
  "keyword": "IPv6",
  "results_count": 11,
  "context": {
    "APP_COM": {
      "entity": "IPv6",
      "properties": {
        "LABEL": "APP_COM"
      }
    }
  }
}

Response headers
content-length: 4537
content-type: application/json
date: Sat, 24 May 2025 11:38:59 GMT
server: unicorn

Download
```

图 3.54 /search 端点测试步骤一

测试二：点击图 3.51 中/ask 端点的 Try it out 按钮，在 Request body 的 question 处替换 string 为示例问题 so what is IPv6?，并且复制测试一中/search 返回的 context 粘贴在 question 后的 context，点击 Execute 即可进行问答，如图 3.55。可以看到 Responses 正确返回了包含调用模型生成的详细回答的 answer 字段，以及显示实际使用上下文片段的 context_used 字段，如图 3.56。

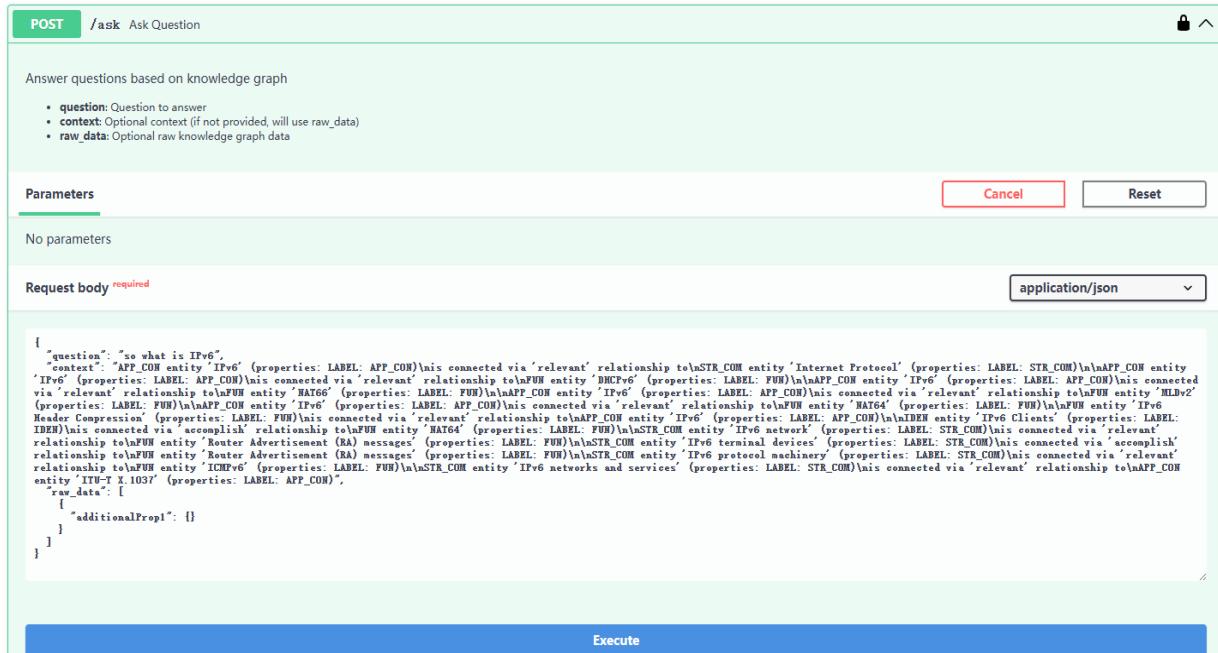


图 3.55 /ask 端点测试步骤一

This screenshot shows the detailed response body for a successful 200 status code. It includes the raw curl command used to make the request, the request URL, and the server response. The response body itself is a large JSON object containing the question, answer (a detailed explanation of IPv6), context_used (the original context from the search), and processing_time. The response headers are also listed at the bottom.

图 3.56 /ask 端点测试步骤二

本文也给出了测试 API 的 python 代码运行示例，使用代码 testapi.py。健康检查测试用于验证 API 服务是否正常运行，检查 Neo4j 和 Ollama 的连接状态，可以看到 API 服务正常通过，并且正确对关键词 IPv6 进行知识图谱搜索与问答功能测试，如图 3.57。

```
(kgqaapi) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze/Ollama/model$ python testapi.py
=====
开始知识图谱问答API测试
=====

==== 测试健康检查 ====
服务状态:
- 整体状态: Service healthy
- Neo4j连接: 正常
- Ollama可用性: 正常
- 时间戳: 2025-05-24T20:00:23.073310

==== 测试知识图谱搜索 (关键词: IPv6) ====
找到 11 条结果
上下文预览:
APP_CON entity 'IPv6' (properties: LABEL: APP_CON)
is connected via 'relevant' relationship to
STR_COM entity 'Internet Protocol' (properties: LABEL: STR_COM)

APP_CON entity 'IPv6' (properties: LABEL...

使用搜索结果进行问答测试:

==== 测试问答 (问题: so what is IPv6?) ====
问答结果:
问题: so what is IPv6?
答案:
IPv6 stands for Internet Protocol version 6. It is the latest major version of the Internet Protocol (IP) that replaced IPv4 to support more devices on networks by providing addresses with a much larger address space, effectively solving the issue of IPv4 address depletion. The knowledge graph connects 'IPv6' as an application concept ('APP_CON') entity relevant to various functional entities such as DHCPv6 for dynamic host configuration, NAT66 and NAT64 for network address translation, MLDv2 for multicast listener discovery, ICMPv6 for internet control messages, and Router Advertisement (RA) messages for IPv6 networking. It also links 'IPv6' with specific IDEN entities like IPv6 Clients, indicating its utility in client management within networks. The graph further illustrates the relationship between IPv6 and other protocols or functionalities through relevant relationships, such as those involving NAT64 which supports IPv4/IPv6 translation for clients, making it a critical component in modern network infrastructure. [$/INST]
```

图 3.57 测试 API 的 testapi.py 代码运行示例

将知识图谱问答系统用 FastAPI 独立封装为 API 服务，相较于本文 3.3.8 节使用 Flask 部署可外部访问的网页服务，具有多方面的优势。FastAPI 支持异步编程，能够更高效地处理并发请求，对于知识图谱查询和模型推理这种 I/O 密集型任务，响应速度更快；并且 FastAPI 通过类型注解自动生成清晰的 Swagger 文档，极大地方便了接口的使用和维护，Flask 虽然也可以支持 API 开发，但需要额外配置文档和异步支持，维护成本较高；FastAPI 设计上更强调将服务以 API 形式模块化，适合构建灵活的后端系统，Flask 部署的网页服务更侧重于提供用户界面，适合小型项目或快速开发可交互的页面，但在高并发和分布式扩展方面相对局限。不过 Flask 部署网页服务的优势在于其生态成熟，开发者对模板渲染和传统 Web 开发支持更完善，适合需要丰富交互界面的应用场景。FastAPI 则更适合 API 后端，尤其是将知识图谱问答能力作为服务被其他应用调用时，FastAPI 提供的灵活且标准化的 RESTful 接口体验更好。

3.4 本章小节

本章主要分为三部分，分别为微调实验、知识图谱构建实验、微调模型搭载限定域知识图谱的推理实验。

微调实验部分详细介绍了本文微调实验的实验细节，包括所选基础模型、所选微调方法，以及本文微调使用的数据集构建方法。最后展示了两组基础模型微调的实验结果，并对结果进行了对比评估，结果显示在 Qwen2.5-7B-Instruct 基础模型上微调后的效果更好，并使用它微调后的模型搭建了一个前段体验程序，将所有模型文件和数据集进行了开源平台上传，展示了前段体验的一个问答示例。

知识图谱构建实验部分详细介绍了本文知识图谱的构建流程和方法，并且说明了利用大语言模型如何进行头实体识别、潜在关系识别、尾实体识别，如何构建三元体。并且对本文所构建的知识图谱进行了构建结果的分析，包括如何将三元体导入 Neo4j 平台进行知识图谱的储存和展示，最后展示了一个有关 IPv6 实体的查询示例。

微调模型搭载限定域知识图谱的推理实验部分详细介绍了本文如何在服务器上使用 Ollama 部署微调后的模型，并且远程访问本地电脑上传 Neo4j 的限定域知识图谱，微调合并后的模型通过对知识图谱的检索，能够增强模型推理的效果，本文在终端上再次展示了有关 IPv6 的多轮问答示例。并且本文利用 Flask 部署网页服务，创建了一个名为 Knowledge Graph QA 的问答系统网页，有效解决了多轮问答的 UI 展示问题，利用 FastAPI 封装 API，创建了一个名为 Knowledge Graph QA API 的问答服务 API。同样的，本文依旧使用 IPv6 作为网页端和 API 测试的多轮问答示例。

3.5 开源内容说明

3.5.1 知识图谱构建与微调模型搭载限定域知识图谱推理的开源说明

本文已将所有内容全开源。除本文 3.1 节展示的微调实验开源外，本章知识图谱构建实验的所有过程文件也已开源至 ModelScope 平台。包括构建的 Protege 本体文件 on.rdf；三元体识别代码 entityPdf.py、Relationship.py 与 tailEntity.py，和运行这些代码产生的所有 JSON 文件；三元体构建中的实体类型抽取代码和关系抽取代码 relationshipText.py；三元体 CSV 文件，包括 6 个实体类型的 CSV 文件与关系文件 roles.csv。

可以看到本仓库包含 4 个子文件夹，分别为 code 存放代码，import 存放 CSV 文件，json 存放过程 JSON 文件，txt 存放过程 TXT 文件。如图 3.58。

具体的代码和三元体 CSV 文件内容展开如图 3.59，网址链接可见于附录 A。

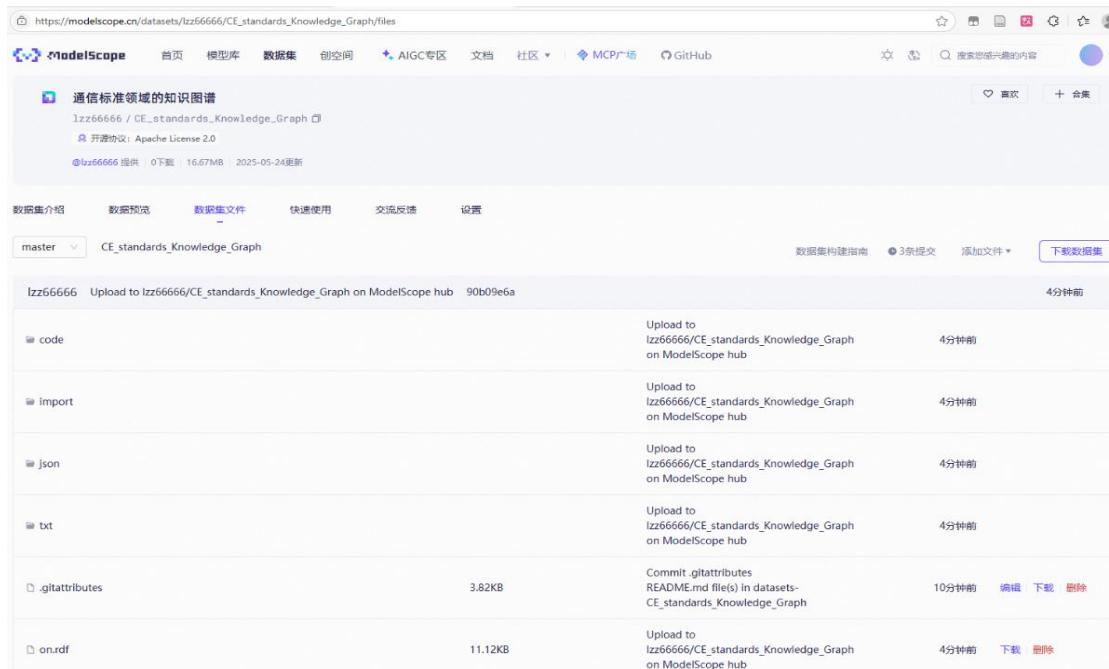


图 3.58 通信标准领域知识图谱的开源示意图

通信标准领域的知识图谱 lzz66666 / CE_standards_Knowledge_Graph					
数据集介绍	数据预览	数据集文件	快速使用	交流反馈	设置
code			Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	
ACTest.py	3.53KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
APP_CONtest.py	3.66KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
entityPdt.py	6.52KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
FineTwo.py	2.40KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
FUNtest.py	3.55KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
IDENTest.py	3.59KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
RelationFinal.py	9.76KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
relationshipText.py	4.73KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
STR_COMtest.py	3.65KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
tailEntity.py	8.03KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
VALUEtest.py	3.52KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	12分钟前	编辑 下载 删除
import			Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	
ACT.csv	46.13KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
APP_CON.csv	92.34KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
FUN.csv	95.70KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
IDEN.csv	96.06KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
roles.csv	394.89KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
STR_COM.csv	181.20KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除
VALUE.csv	44.05KB		Upload to lzz66666/CE_standards_Knowledge_Graph on ModelScope hub	14分钟前	编辑 下载 删除

图 3.59 通信标准领域知识图谱的代码和三元体 CSV 文件内容展开示意图

本章微调模型搭载限定域知识图谱推理实验的所有过程文件也已开源至 Modelscope 平台。包括所使用的 llama.cpp 包；产生的 test.Modelfile 文件；过程模型文件 convert.bin 与 quantized.bin；以及所使用的代码文件 kg2llm.py, app.py, kgqa_api.py, testapi.py。如图 3.60，网址链接可见于附录 A。

文件名	大小	操作
__pycache__	intree	Upload to lzz66666/Ollma_model on ModelScope hub
llama.cpp	intree	Upload to lzz66666/Ollma_model on ModelScope hub
.gitattributes	2.22KB	Upload to lzz66666/Ollma_model on ModelScope hub
app.py	7.68KB	Upload to lzz66666/Ollma_model on ModelScope hub
configuration.json	48.00B	System init configuration.json
convert.bin	15.24GB	Upload to lzz66666/Ollma_model on ModelScope hub
kg2llm.py	5.73KB	Upload to lzz66666/Ollma_model on ModelScope hub
kgqa_api.py	11.63KB	kgqa_api
quantized.bin	4.43GB	Upload to lzz66666/Ollma_model on ModelScope hub
README.md	1.40KB	System update meta information
test.Modelfile	93.00B	Upload to lzz66666/Ollma_model on ModelScope hub
testapi.py	6.08KB	testapi

图 3.60 微调模型搭载限定域知识图谱推理实验的开源示意图

3.5.2 实验环境压缩包开源内容说明

本文在服务器采用了 3 个环境，分别为 ce 环境，llamacpp 环境，kgqaapi 环境。

llamacpp 用于本文 3.3.2 节量化模型与 3.3.3 节制作 Ollama 使用模型的内容，kgqaapi 环境用于本文 3.3.9 节创建独立的 API 接口，其余章节使用的是 ce 环境。kgqaapi 环境由于并不复杂故未上传，只需新建 python=3.10 新环境，并使用命令 pip install fastapi uvicorn python-dotenv neo4j ollama requests，即可完成配置。

本文将 ce 环境与 llamacpp 环境在服务器/home/h3c/anaconda3/envs 路径中的文件夹打包为压缩包 ce.zip 与 llamacpp.zip，并开源至 Modelscope 平台。如图 3.61，链接可见于附录 A。服务器终端的环境展示，ce 环境可见于附录 B，llamacpp 环境可见于附录 C。

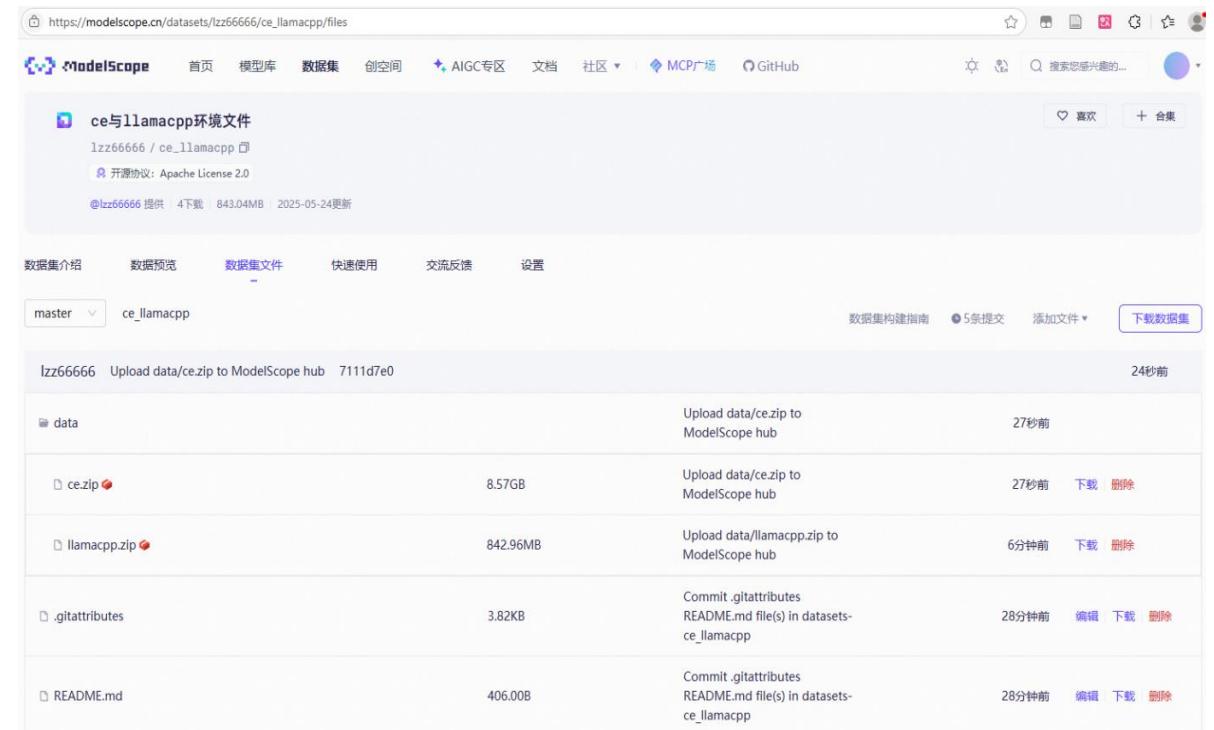


图 3.61 实验环境压缩包的开源示意图

3.5.3 本文所有开源内容的补充说明

本文所有开源内容都归类在 Modelscope 平台的 CEstandards-LLM 合集，如图 3.62，网址链接可见于附录 A。



图 3.62 本文所有开源内容 CEstandards-LLM 合集的示意图

结论

本文根据通信标准领域知识复杂度高、咨询服务效率低的问题，提出了基于大语言模型微调与限定域知识图谱构建的通信标准领域智能咨询服务系统。

LoRA 微调降低了模型微调所需求的计算量，并且明显改善了模型对于通信标准领域知识的理解与生成能力，测试结果显示微调后的 Qwen 合并模型在通信标准领域的问答质量与推理速度均有明显提升。

构建了通信标准领域的限定域知识图谱，采用合理的本体设计和高效的实体识别及关系抽取方法，基于大语言模型抽取并构建了高质量三元组数据，并构建了结构清晰的通信标准领域知识图谱，便于知识的管理与检索。

设计并实现了基于检索增强生成 RAG 的智能问答系统，将微调后的合并模型与构建的限定域知识图谱集成，构建交互的网页服务和 API 接口。结果表明，系统实现了知识检索和多轮问答的准确高效，大幅提升了用户在通信标准领域的咨询体验。

本文提出的通信标准智能咨询系统具有较好的应用推广价值，提升了通信标准领域咨询的处理效率，减轻通信工程师在日常工作中的信息搜索负担，并且具备一定的通用性，可将该方法推广应用至其他专业的标准化咨询服务。

本文也存在一些不足。当前构建的知识图谱主要为静态知识，实时性的知识更新并未设置，未来知识图谱动态化更新服务需要继续完善；如何在微调过程中降低其标注数据的依赖程度，实现在有限的标注数据下模型仍然能具有较好的鲁棒性，是值得未来改进工作中研究的问题；此外还需要持续提高系统在更加复杂的语境下进行知识理解与推理的能力，使得交互效果和应用场景效果更加完善。

参考文献

- [1] 沈晨晨, 岳圣斌, 刘书隽, 等. 面向法律领域的大模型微调与应用[J]. 大数据, 2024, 10(05): 12-27.
- [2] 宋晓宁, 蒋启一, 张文杰, 等. 基于食品知识图谱的智能感知问答[J]. 中国食品学报, 2024, 24(12): 1-12.
- [3] 张天鸿, 王晓玲, 余红玲, 等. 基于大语言模型的灌浆工程知识服务系统[J]. 水利学报, 2025, 56(01): 130-142.
- [4] 李戈, 彭鑫, 王千祥, 等. 大模型: 基于自然交互的人机协同软件开发与演化工具带来的挑战[J]. 软件学报, 2023, 34(10): 4601-4606.
- [5] 王文奇, 郭梦帆, 杨杜祥, 等. 大语言模型发展与应用综述[J]. 中原工学院学报, 2025, 36(02): 1-8.
- [6] 吴春志, 赵玉龙, 刘鑫, 等. 大语言模型微调方法研究综述[J]. 中文信息学报, 2025, 39(02): 1-26.
- [7] 杨毛加, 柔特, 才智杰, 等. 基于参数高效微调的藏文大模型研究[J]. 中文信息学报, 2024, 38(12): 106-115.
- [8] 蔡子杰, 方荟, 刘建华, 等. 基于大型语言模型指令微调的心理健康领域联合信息抽取[J]. 中文信息学报, 2024, 38(08): 112-127.
- [9] 李诗晨, 王中卿, 周国栋. 大语言模型驱动的跨领域属性级情感分析[J]. 软件学报, 2025, 36(02): 644-659.
- [10] 庞杰, 闫晓东, 赵小兵. Ko-LLaMA: 基于 LLaMA 的朝鲜语大语言模型[J]. 外语学刊, 2025(01): 1-8.
- [11] 赵京胜, 宋梦雪, 高祥, 等. 自然语言处理中的文本表示研究[J]. 软件学报, 2022, 33(01): 102-128.
- [12] 朱君辉, 王梦焰, 杨尔弘, 等. 大模型生成回答与人类回答文本的语言特征比较研究[J]. 中文信息学报, 2024, 38(04): 17-27.
- [13] Lin X, Wang W, Li Y, et al. Data-efficient Fine-tuning for LLM-based Recommendation[C]. Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2024: 365-374.
- [14] Wu X-K, Chen M, Li W, et al. LLM Fine-Tuning: Concepts, Opportunities, and Challenges[J]. Big Data and Cognitive Computing, 2025, 9(4): 87.
- [15] 胡忠义, 税典程, 吴江. 基于大模型微调的生成式文献层次分类标引[J]. 情报学报, 2025, 44(04): 425-437.
- [16] 罗鹤, 张廷, 孙媛, 等. NCIFD: 面向大模型的民族文化微调数据集[J]. 中文信息学报, 2025, 39(02): 41-51.

- [17] 魏炜, 张坤, 徐哲淇. 从通用到垂直: 大模型赋能管理学研究的新路径[J]. 管理学报, 2025, 22(01): 1-11.
- [18] 叶淋潮, 邵会会, 谢振平. 基于精调LLaMA模型的中西医概念关系对比分析方法[J]. 中文信息学报, 2025, 39(02): 162-170.
- [19] Chen X, Jia S, Xiang Y. A review: Knowledge reasoning over knowledge graph[J]. Expert systems with applications, 2020, 141: 112948.
- [20] Fensel D, Şimşek U, Angele K, et al. Introduction: what is a knowledge graph?[J]. Knowledge graphs: Methodology, tools and selected use cases, 2020: 1-10.
- [21] Demszky D, Yang D, Yeager D S, et al. Using large language models in psychology[J]. Nature Reviews Psychology, 2023, 2(11): 688-701.
- [22] Shi J, Guo Q, Liao Y, et al. Legal-lm: Knowledge graph enhanced large language models for law consulting[C]. International Conference on Intelligent Computing, 2024: 175-186.
- [23] 官赛萍, 靳小龙, 贾岩涛, 等. 面向知识图谱的知识推理研究进展[J]. 软件学报, 2018, 29(10): 2966-2994.
- [24] 来雨轩, 王艺丹, 王立. 基于大语言模型与检索增强的学科试题生成方法[J]. 中文信息学报, 2024, 38(12): 148-158.
- [25] 李佳沂, 黄瑞章, 陈艳平, 等. 结合提示学习和Qwen大语言模型的裁判文书摘要方法[J]. 清华大学学报(自然科学版), 2024, 64(12): 2007-2018.
- [26] 张宁豫, 谢辛, 陈想, 等. 基于知识协同微调的低资源知识图谱补全方法[J]. 软件学报, 2022, 33(10): 3531-3545.
- [27] 柴景贤, 郎许锋, 李红岩, 等. 基于Lora微调的轻量化中医药古籍大语言模型研究[J]. 世界科学技术-中医药现代化, 2025, 27(03): 823-831.
- [28] Chen P, Lu Y, Zheng V W, et al. Knowedu: A system to construct knowledge graph for education[J]. Ieee Access, 2018, 6: 31553-31563.
- [29] Zou X. A survey on application of knowledge graph[C]. Journal of Physics: Conference Series, 2020: 012016.
- [30] 徐正斐, 辛欣. 基于大语言模型的中文实体链接实证研究[J]. 自动化学报, 2025, 51(02): 327-342.
- [31] 陈俊臻, 王淑营, 罗浩然. 融合大模型微调与图神经网络的知识图谱问答[J]. 计算机工程与应用, 2024, 60(24): 166-176.
- [32] 王楚童, 李明达, 孙孟轩, 等. 融合大规模医学事实的跨语言双层知识图谱[J]. 软件学报, 2025, 36(03): 1240-1253.
- [33] 郑炜, 刘程远, 吴潇雪, 等. 基于知识图谱的跨项目安全缺陷报告预测方法[J]. 软件学报, 2024, 35(03): 1257-1279.
- [34] 陈宋生, 王明. 基于大语言模型的财会知识图谱构建及应用展望[J]. 会计之友, 2025(05): 152-161.

- [35] 杨再宝, 张涵, 刘晨晔, 等. 基于推理大模型与知识图谱构建信用债券数智化分析新范式[J]. 债券, 2025(03): 10-14.
- [36] 王永, 秦嘉俊, 黄有锐, 等. 融合知识图谱和大模型的高校科研管理问答系统设计[J]. 计算机科学与探索, 2025, 19(01): 107-117.
- [37] 王佐旭. 知识图谱和大语言模型辅助新工科课程教学资源建设方法[J]. 高等工程教育研究, 2025(01): 40-46+110.
- [38] 乔少杰, 杨国平, 于泳, 等. QA-KGNet:一种语言模型驱动的知识图谱问答模型[J]. 软件学报, 2023, 34(10): 4584-4600.
- [39] 言佳润, 鲜于波. 面向中文网络对话文本的论辩挖掘——基于微调与提示学习的大模型算法[J]. 中文信息学报, 2023, 37(10): 139-148.
- [40] 刘畅, 张琪, 王东波, 等. 基于大语言模型技术的古籍限定域关系抽取及应用研究[J]. 情报学报, 2025, 44(02): 200-219.
- [41] 杨雅娴, 吴金红, 吴彦坤, 等. 融合知识图谱和大语言模型的虚假健康信息识别方法研究[J]. 情报理论与实践, 2025, 48(03): 127-133.
- [42] 漆桂林, 欧阳丹彤, 李涓子. 本体工程与知识图谱专题前言[J]. 软件学报, 2018, 29(10): 2897-2898.

附录

附录 A

1. 通信标准领域微调数据集 (Communication_standards_dataset):

https://modelscope.cn/datasets/lzz66666/Communication_standards_dataset

2. 通信标准领域的多轮对话助手模型 (Qwen_communication_standards):

https://www.modelscope.cn/models/lzz66666/Qwen_communication_standards

3. 通信标准领域的多轮对话助手 (Assistant_in_the_field_of_communication_standard):

https://modelscope.cn/studios/lzz66666/Assistant_in_the_field_of_communication_standards

4. 通信标准领域的知识图谱 (CE_standards_Knowledge_Graph):

https://modelscope.cn/datasets/lzz66666/CE_standards_Knowledge_Graph

5. 通信标准领域微调模型的 Ollama 部署模型 (Ollma_model):

https://modelscope.cn/models/lzz66666/Ollma_model

6. 本文所用的 ce 与 llamacpp 环境文件压缩包 (ce_llamacpp):

https://modelscope.cn/datasets/lzz66666/ce_llamacpp

7. 本文所有开源内容的 CEstandards-LLM 合集:

<https://modelscope.cn/collections/CEstandards-LLM-f124b3f9124b4c>

附录 B

#	Name	Version	Build	Channel	jinja2	3.1.6	pypi_0	pypi
1	_libgcc_mutex	0.1	main	jsonschema-specifications	4.23.0	0.9.0	pypi_0	pypi
2	_openmp_mutex	5.1	l_gnu	kiwisolver	1.4.8	1.4.2	pypi_0	pypi
3	absl-py	2.2.2	pypi_0	pypi	lark	2.2.2	pypi_0	pypi
4	accelerate	1.5.2	pypi_0	pypi	lazy-loader	0.4	pypi_0	pypi
5	aiofiles	23.2.1	pypi_0	pypi	ld_impl_linux-64	2.40	h12ee557_0	
6	aiohappyeyeballs	2.6.1	pypi_0	pypi	libffi	3.3	he6710b0_2	
7	aiohttp	3.11.16	pypi_0	pypi	libgcc-ng	11.2.0	h1234567_1	
8	aiosignal	1.3.2	pypi_0	pypi	libgomp	11.2.0	h1234567_1	
9	airportsdata	20250224	pypi_0	pypi	librosa	0.11.0	pypi_0	pypi
10	annotated-types	0.7.0	pypi_0	pypi	libstdcxx-ng	11.2.0	h1234567_1	
11	anyio	4.9.0	pypi_0	pypi	libuuid	1.41.5	h5eee18b_0	
12	astor	0.8.1	pypi_0	pypi	llamafactory	0.9.3.dev0	pypi_0	pypi
13	async-timeout	5.0.1	pypi_0	pypi	llguidance	0.7.13	pypi_0	pypi
14	attrs	25.3.0	pypi_0	pypi	llvmlite	0.44.0	pypi_0	pypi
15	audioread	3.0.1	pypi_0	pypi	lm-format-enforcer	0.10.11	pypi_0	pypi
16	av	14.3.0	pypi_0	pypi	markdown	3.7	pypi_0	pypi
17	blake3	1.0.4	pypi_0	pypi	markdown-it-py	3.0.0	pypi_0	pypi
18	blinker	1.9.0	pypi_0	pypi	markupsafe	2.1.5	pypi_0	pypi
19	bzip2	1.0.8	h5eee18b_0		matplotlib	3.10.1	pypi_0	pypi
20	ca-certificates	2025.2.25	h06a4308_0		mdurl	0.1.2	pypi_0	pypi
21	cachetools	5.5.2	pypi_0	pypi	mistral-common	1.5.4	pypi_0	pypi
22	certifi	2025.1.31	pypi_0	pypi	modelscope	1.24.1	pypi_0	pypi
23	cffi	1.17.1	pypi_0	pypi	mpmath	1.3.0	pypi_0	pypi
24	charset-normalizer	3.4.1	pypi_0	pypi	msgpack	1.1.0	pypi_0	pypi
25	click	8.1.8	pypi_0	pypi	msgspec	0.19.0	pypi_0	pypi
26	cloudpickle	3.1.1	pypi_0	pypi	multidict	6.3.2	pypi_0	pypi
27	compressed-tensors	0.9.2	pypi_0	pypi	multiprocess	0.70.16	pypi_0	pypi
28	contourpy	1.3.1	pypi_0	pypi	nanobind	2.6.1	pypi_0	pypi
29	cupy-cudal2x	13.4.1	pypi_0	pypi	ncurses	6.4	h6a678d5_0	
30	cycler	0.12.1	pypi_0	pypi	neo4j	5.28.1	pypi_0	pypi
31	datasets	3.4.1	pypi_0	pypi	nest-asyncio	1.6.0	pypi_0	pypi
32	decorator	5.2.1	pypi_0	pypi	networkx	3.4.2	pypi_0	pypi
33	defpyf	0.18.0	pypi_0	pypi	ninja	1.11.1.4	pypi_0	pypi
34	dill	0.3.8	pypi_0	pypi	nltk	3.9.1	pypi_0	pypi
35	diskcache	5.6.3	pypi_0	pypi	numba	0.61.0	pypi_0	pypi
36	distro	1.9.0	pypi_0	pypi	numpy	1.26.4	pypi_0	pypi
37	dnspython	2.7.0	pypi_0	pypi	nvidia-cublas-cu12	12.4.5.8	pypi_0	pypi
38	docstring-parser	0.16	pypi_0	pypi	nvidia-cuda-cupti-cu12	12.4.127	pypi_0	pypi
39	einops	0.8.1	pypi_0	pypi	nvidia-cuda-nvrtc-cu12	12.4.127	pypi_0	pypi
40	email-validator	2.2.0	pypi_0	pypi	nvidia-cuda-runtime-cu12	12.4.127	pypi_0	pypi
41	exceptiongroup	1.2.2	pypi_0	pypi	nvidia-cudnn-cu12	9.1.0.70	pypi_0	pypi
42	fastapi	0.115.12	pypi_0	pypi	nvidia-cufft-cu12	11.2.1.3	pypi_0	pypi
43	fastapi-cli	0.0.7	pypi_0	pypi	nvidia-curand-cu12	10.3.5.147	pypi_0	pypi
44	fastrlock	0.8.3	pypi_0	pypi	nvidia-cusolver-cu12	11.6.1.9	pypi_0	pypi
45	ffmpy	0.5.0	pypi_0	pypi	nvidia-cusparse-cu12	12.3.1.170	pypi_0	pypi
46	filelock	3.18.0	pypi_0	pypi	nvidia-cusparselt-cu12	0.6.2	pypi_0	pypi
47	fire	0.7.0	pypi_0	pypi	nvidia-nccl-cu12	2.21.5	pypi_0	pypi
48	flask	3.1.1	pypi_0	pypi	nvidia-nvjitlink-cu12	12.4.127	pypi_0	pypi
49	fonttools	4.57.0	pypi_0	pypi	nvidia-nvtx-cu12	12.4.127	pypi_0	pypi
50	frozenlist	1.5.0	pypi_0	pypi	ollama	0.4.8	pypi_0	pypi
51	fsspec	2024.12.0	pypi_0	pypi	openai	1.71.0	pypi_0	pypi
52	gguf	0.10.0	pypi_0	pypi	opencv-python-headless	4.11.0.86	pypi_0	pypi
53	git-lfs	1.6	pypi_0	pypi	openssl	1.1.1w	h7f8727e_0	
54	gradio	5.21.0	pypi_0	pypi	orjson	3.10.16	pypi_0	pypi
55	gradio-client	1.7.2	pypi_0	pypi	outlines	0.1.11	pypi_0	pypi
56	groovy	0.1.2	pypi_0	pypi	outlines-core	0.1.26	pypi_0	pypi
57	grpcio	1.71.0	pypi_0	pypi	packaging	24.2	pypi_0	pypi
58	h11	0.14.0	pypi_0	pypi	pandas	2.2.3	pypi_0	pypi
59	hf-xet	1.0.2	pypi_0	pypi	partial-json-parser	0.2.1.1.post5	pypi_0	pypi
60	httpcore	1.0.7	pypi_0	pypi	peft	0.15.0	pypi_0	pypi
61	httptools	0.6.4	pypi_0	pypi	pillow	11.1.0	pypi_0	pypi
62	httpx	0.28.1	pypi_0	pypi	pip	25.0	py310h06a4308_0	
63	huggingface-hub	0.30.1	pypi_0	pypi	platformdirs	4.3.7	pypi_0	pypi
64	idna	3.10	pypi_0	pypi	pooch	1.8.2	pypi_0	pypi
65	importlib-metadata	8.6.1	pypi_0	pypi	prometheus-client	0.21.1	pypi_0	pypi
66	interregular	0.3.3	pypi_0	pypi	prometheus-fastapi-instrumentator	7.1.0	pypi_0	pypi
67	itsdangerous	2.2.0	pypi_0	pypi	propcache	0.3.1	pypi_0	pypi
68	jieba	0.42.1	pypi_0	pypi	protobuf	6.30.2	pypi_0	pypi
69	jinja2	3.1.6	pypi_0	pypi	psutil	7.0.0	pypi_0	pypi

```

● 1 10.155.120.146 ✘ +
psutil          7.0.0          pypi_0    pypi
py-cpuinfo     9.0.0          pypi_0    pypi
pyarrow         19.0.1         pypi_0    pypi
pycountry       24.6.1         pypi_0    pypi
pycparser       2.22           pypi_0    pypi
pydantic        2.10.6         pypi_0    pypi
pydantic-core   2.27.2         pypi_0    pypi
pydub           0.25.1         pypi_0    pypi
pygments        2.19.1         pypi_0    pypi
pyparsing       3.2.3          pypi_0    pypi
pypdf           5.5.0          pypi_0    pypi
python          3.10.0         h12debd9_5
python-dateutil 2.9.0.post0   pypi_0    pypi
python-dotenv   1.1.0          pypi_0    pypi
python-json-logger 3.3.0        pypi_0    pypi
python-multipart 0.0.20        pypi_0    pypi
pytz             2025.2         pypi_0    pypi
pyyaml          6.0.2          pypi_0    pypi
pyzmq           26.4.0         pypi_0    pypi
ray              2.43.0         pypi_0    pypi
readline         8.2            h5eee18b_0
referencing     0.36.2         pypi_0    pypi
regex            2024.11.6     pypi_0    pypi
requests         2.32.3         pypi_0    pypi
rich             14.0.0         pypi_0    pypi
rich-toolkit    0.14.1         pypi_0    pypi
rouge-chinese   1.0.3          pypi_0    pypi
rpds-py          0.24.0         pypi_0    pypi
ruff             0.11.4         pypi_0    pypi
safehttpx       0.1.6          pypi_0    pypi
safetensors      0.5.3          pypi_0    pypi
scikit-learn    1.6.1          pypi_0    pypi
scipy            1.15.2         pypi_0    pypi
semantic-version 2.10.0        pypi_0    pypi
sentencepiece   0.2.0          pypi_0    pypi
setuptools       75.8.0         py310h06a4308_0
shellingham     1.5.4          pypi_0    pypi
shtab            1.7.1          pypi_0    pypi
six              1.17.0         pypi_0    pypi
sniffio          1.3.1          pypi_0    pypi
soundfile        0.13.1         pypi_0    pypi
soxr             0.5.0.post1   h5eee18b_0
sqlite           3.45.3         pypi_0    pypi
sse-starlette   2.2.1          pypi_0    pypi
starlette        0.46.1         pypi_0    pypi
sympy            1.13.1         pypi_0    pypi
tensorboard     2.19.0         pypi_0    pypi
tensorboard-data-server 0.7.2     pypi_0    pypi
termcolor         3.0.1          pypi_0    pypi
threadpoolctl   3.6.0          pypi_0    pypi
tiktoken          0.9.0          pypi_0    pypi
tk                8.6.14         h39e8969_0
tokenizers       0.21.0         pypi_0    pypi
tomlkit          0.13.2         pypi_0    pypi
torch             2.6.0          pypi_0    pypi
torchaudio       2.6.0          pypi_0    pypi
torchvision      0.21.0         pypi_0    pypi
tqdm              4.67.1         pypi_0    pypi
transformers     4.51.0         pypi_0    pypi
triton           3.2.0          pypi_0    pypi
trl               0.9.6          pypi_0    pypi
typer             0.15.2         pypi_0    pypi
typing-extensions 4.13.1        pypi_0    pypi
typing-inspection 0.4.0          pypi_0    pypi
tyro              0.8.14         pypi_0    pypi
tzdata            2025.2         pypi_0    pypi
urllib3           2.3.0          pypi_0    pypi
uvicorn           0.34.0         pypi_0    pypi
uvloop            0.21.0         pypi_0    pypi
vllm              0.8.3          pypi_0    pypi
watchfiles       1.0.4          pypi_0    pypi
websockets        15.0.1         pypi_0    pypi
werkzeug          3.1.3          pypi_0    pypi
wheel              0.45.1         py310h06a4308_0
xformers          0.0.29.post2   pypi_0    pypi
xgrammar          0.1.17         pypi_0    pypi
xxhash            3.5.0          h5eee18b_1
xz                5.6.4          pypi_0    pypi
yarl              1.19.0         pypi_0    pypi
zipp              3.21.0         pypi_0    pypi
zlib              1.2.13         h5eee18b_1
(c) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze$ 

```

附录 C

```
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze$ conda list
# packages in environment at /home/h3c/anaconda3/envs/llamacpp:
#
# Name           Version    Build  Channel
_libgcc_mutex   0.1        main   main
_openmp_mutex   5.1        main   l_gnu
aiohttp         3.9.5      pypi_0 pypi
aiosignal       1.3.2      pypi_0 pypi
annotated-types 0.7.0      pypi_0 pypi
anyio           4.9.0      pypi_0 pypi
async-timeout   4.0.3      pypi_0 pypi
attrs            25.3.0     pypi_0 pypi
bzip2            1.0.8      h5eee18b_6
ca-certificates 2025.2.25  h06a4308_0
certifi          2025.4.26  pypi_0 pypi
charset-normalizer 3.4.2      pypi_0 pypi
click             8.1.8      pypi_0 pypi
contourpy        1.3.2      pypi_0 pypi
cycler            0.12.1     pypi_0 pypi
distro            1.9.0      pypi_0 pypi
exceptiongroup  1.3.0      pypi_0 pypi
filelock          3.18.0     pypi_0 pypi
fonttools         4.58.0     pypi_0 pypi
frozenlist        1.6.0      pypi_0 pypi
fsspec            2025.5.0   pypi_0 pypi
gguf              0.16.3     pypi_0 pypi
h1l               0.16.0      pypi_0 pypi
httpcore          1.0.9      pypi_0 pypi
httpx              0.28.1     pypi_0 pypi
huggingface-hub  0.23.5     pypi_0 pypi
idna              3.10       pypi_0 pypi
iniconfig         2.1.0       pypi_0 pypi
jinja2             3.1.6       pypi_0 pypi
jiter              0.10.0     pypi_0 pypi
kiwisolver        1.4.8       pypi_0 pypi
ld_impl_linux-64  2.40       h12ee557_0
libffi             3.4.4       h6a678d5_1
libgcc-ng          11.2.0     h1234567_1
libgomp            11.2.0     h1234567_1
libstdcxx-ng       11.2.0     h1234567_1
libuuid            1.41.5     h5eee18b_0
markdown-it-py     3.0.0       pypi_0 pypi
markupsafe         3.0.2       pypi_0 pypi
matplotlib        3.10.3     pypi_0 pypi
mdurl              0.1.2       pypi_0 pypi
mpmath              1.3.0       pypi_0 pypi
multidict          6.4.4       pypi_0 pypi
ncurses             6.4        h6a678d5_0
networkx            3.4.2     pypi_0 pypi
numpy              1.26.4     pypi_0 pypi
openai              1.55.3     pypi_0 pypi
openssl             3.0.16     h5eee18b_0
packaging            25.0      pypi_0 pypi
pandas              2.2.3       pypi_0 pypi
pillow              11.2.1     pypi_0 pypi
pip                 25.1       pyhc872135_2
pluggy              1.6.0       pypi_0 pypi
prometheus-client  0.20.0     pypi_0 pypi
propcache            0.3.1     pypi_0 pypi
protobuf             4.25.7     pypi_0 pypi
pydantic            2.11.5     pypi_0 pypi
pydantic-core       2.33.2     pypi_0 pypi
pygments            2.19.1     pypi_0 pypi
pyparsing            3.2.3     pypi_0 pypi
pytest              8.3.5       pypi_0 pypi
python              3.10.16    he870216_1
python-dateutil     2.9.0.post0  pypi_0 pypi
pytz                2025.2     pypi_0 pypi
pyyaml              6.0.2       h5eee18b_0
readline             8.2        h5eee18b_0
regex                2024.11.6  pypi_0 pypi
requests             2.32.3     pypi_0 pypi
rich                 14.0.0     pypi_0 pypi
safetensors          0.5.3       pypi_0 pypi
seaborn              0.13.2     pypi_0 pypi
sentencepiece        0.2.0       pypi_0 pypi
setuptools           78.1.1     py310h06a4308_0
shellingham          1.5.4       pypi_0 pypi
six                  1.17.0     pypi_0 pypi
sniffio              1.3.1       h5eee18b_0
sqlite              3.45.3      pypi_0 pypi
sympy                1.14.0     h39e8969_0
tk                   8.6.14      pypi_0 pypi
tokenizers           0.20.3     pypi_0 pypi
tomli                2.2.1       pypi_0 pypi
torch                2.2.2+cpu   pypi_0 pypi
tqdm                 4.67.1     pypi_0 pypi
transformers         4.46.3      pypi_0 pypi
typer                 0.15.4     pypi_0 pypi
typing-extensions   4.13.2      pypi_0 pypi
typing-inspection  0.4.1       pypi_0 pypi
tzdata                2025.2     pypi_0 pypi
urllib3              2.4.0       pypi_0 pypi
wget                 3.2        pypi_0 pypi
wheel                0.45.1     py310h06a4308_0
xz                   5.6.4       h5eee18b_1
yaml                 1.20.0     pypi_0 pypi
zlib                 1.2.13     h5eee18b_1
(llamacpp) h3c@h3c-H3C-UniServer-R4900-G501:~/luozhongze$
```

致谢

感谢王健老师对这篇毕业论文的指导。

终于要毕业了。时光荏苒，回首四年，发生了许多不曾预想的精彩故事。

四年前，懵懂地来到这里，当时还是在材料学院，经常奔走去校园最西北边的化资实验楼做化学实验，可如今已经三年没去过那里了。很难相信四年间发生了这么多的轨迹转变，当时有一位名为李春英的化资学院老师是我的分析化学实验老师，当时这门加权占比不大的实验课最后出分 80 分，我还在图书馆厕所里和老师打电话询问，仍记得当时她非常细心地告诉我问题在哪，好像是问答题答的都很好，不过选择题失分了许多，当时我告诉她我想转专业到信息学院，她给我的许多祝愿现在也记忆深刻。

很感谢自己当时做出的许多选择，也感谢父母一直以来的支持，也许现在遇到考差的分数不会去询问老师了，但是当时的热忱与执着确实改变了很多事。现在回想，也许当时转专业失败了就没有这么多然后了，可笑的是当时好像也没有这么大压力。

大一的时候依然是我觉得最惬意的时光，当时虽然有了转专业的目标，但是更多的是刚踏入大学时的新鲜感，无意中我做出了大学中最不后悔的决定，也就是进入了新媒体中心，很多时候轨迹的转变也都是不经意的。很难相信在这个复杂的大学社会中，会在这个学生组织中注入这么多的笑与泪。这里有致谢写不完的感动，与编辑部，与 F6，与我们的熬夜，与我们的不回寝，与我们的灵魂契合。

之后是大二，大二开始就转专业了。现在还能感同身受当时经历课表满课的自己，因为自己有拖延症，许多个考前的夜晚都是在新媒体办公室通宵爆肝直至清晨。这时另一个转变是那时候申请的大创项目，搞笑的是当时一知半解的自己还是用传统图像的 OpenCV 预立项的，最后正式立项却还是用的 YOLO，更惊喜的是竟然是为数不多的国家级，也许现在回想很多都是运气和偶然，但很感谢当时指导的董光辉老师，让我有机会在 1008 实验室和研究生实验室学习，在这些地方度过了许多欢快的磨炼。

大三的时光总是过得很快。很多个晚上都是赶在晚上 11 点跑回寝室，回寝室之后借宿舍舍友的电脑远程操控在实验室的电脑争取多跑几组实验，很多个早上起床打开舍友电脑都很失望啊，但值得庆幸的是后来我发表了一篇 SCI 论文，也算是一点回报。后来又有机会和孙敬云老师合作，发表了几篇不错的会议论文，自己莫名其妙地也就积累了一些科研的经验了，就像很多电影故事的发展，也没有按照预期。

后来申请去了一些南方的夏令营，那些地方也教会我很多，走过广州充满垃圾和电线的巷子，走在整晚通明的贯穿式学校，这些所见所闻也转变了我的所思所想。也许又是运气使然，在发了 30 多封邮件后终于有老师回复，并让我去实验室学习，这也就是后来在深圳的故事了，许许多多的巧合构成了现在的我，这些经历都是我感谢的对象。

值得庆幸的是，在即将离别东林的时候，曾经的照片留下了这里的美好，宛如林场里的春风，把那些瞬间吹到了我的脑海中，脑海中的东林是阳光倾洒的。

学位论文原创性声明

本人郑重声明：所提交的毕业论文是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等学术不端行为。

本人愿为此声明承担法律责任。

作者签名： 日期： 年 月 日

学位论文版权授权书

本学位论文作者同意授予东北林业大学对本论文全部内容的信息网络传播权、汇编权、发行权、转授权与复制权，其中包括但不限于将学位论文上传到有关数据库，影印、缩印或扫描等手段保存、查阅、借阅、汇编等。
(保密的学位论文在解密后适用本授权书)

作者签名： 日期： 年 月 日