心於至善


基于回答集编程 视觉问答技术研究与实现 贾梁 东南大学

学校代码: 10286
分 类 号: 000
密 级: 公开
U D C: 000
学 号: 222171


# 工程硕士学位论文

## 基于回答集编程的
## 视觉问答技术研究与实现

(学位论文形式: 应用研究)

研究生姓名: **贾梁**
导 师 姓 名: **张志政 副教授**

申请学位类别 **工程硕士** 学位授予单位 **东 南 大 学**
工程领域名称 **电子信息** 论文答辩日期 **2025 年 5 月 30 日**
研 究 方 向 **计算机技术** 学位授予日期 **2025 年 6 月 20 日**
答辩委员会主席 **翟玉庆** 评 阅 人 **倪庆剑**
**张祥**

2025 年 3 月 22 日

学校代码: 10286
分 类 号: 000
密 级: 公开
U D C: 000
学 号: 222171


# 工程硕士学位论文

## 基于回答集编程的
## 视觉问答技术研究与实现

(学位论文形式: 应用研究)

研究生姓名: **贾梁**

导 师 姓 名: **张志政 副教授**

申请学位类别 **工程硕士** 学位授予单位 **东南大学**

工程领域名称 **电子信息** 论文答辩日期 **2025 年 5 月 30 日**

研 究 方 向 **计算机技术** 学位授予日期 **2025 年 6 月 20 日**

答辩委员会主席 **翟玉庆** 评 阅 人 **倪庆剑 张祥**

2025 年 3 月 22 日

东南大学

# 工程硕士学位论文

## 基于回答集编程的
## 视觉问答技术研究与实现


专 业 名 称: 电 子 信 息

研究生姓名: 贾 梁

导 师 姓 名: 张志政 副教授

# RESEARCH AND IMPLEMENTATION OF VISUAL QUESTION ANSWERING TECHNOLOGY BASED ON ANSWER SET PROGRAMMING

A Thesis submitted to

Southeast University

For the Professional Degree of Master of Engineering

BY

Jia Liang

Supervised by:

Associate Prof. Zhang Zhizheng

School of Computer Science and Engineering

Southeast University

2025/3/22

# 东南大学学位论文独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：_____ 日 期：_____

# 东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

研究生签名：_____ 导师签名：_____ 日 期：_____

# 摘　要

目前，视觉语言模型已在视觉问答、图片描述生成等方面表现出不错的效果，但它们通常在空间推理方面表现不佳，特别是在不完全可见场景下，对空间关系的理解、推理能力较差。回答集编程（Answer Set Programming, ASP）是一种形式化的知识表示和推理方法，已有一些研究表明其在理解空间关系和进行空间推理方面，具有较强的能力。视觉问答的神经符号系统使用深度学习技术进行感知，生成输入图像和问题的逻辑符号表示，然后使用 ASP 进行推理，并做出相应回答。相比单独的视觉语言模型，视觉问答的神经符号系统在空间推理方面更加出色。然而，现有的考察模型的空间推理能力的视觉问答数据集，存在问题设计较为简单、对象类型和属性单一、问题的语言表达缺乏多样性等问题。此外，现有的神经符号系统泛化能力较差，编写、维护提示词的过程过于复杂，开发难度相对较高，且对新场景下的问题需要人为设计额外规则，可拓展性较差。为了解决这些问题，本文构建了一个视觉问答数据集，并提出了一种神经符号框架。本文的主要工作如下：

1. 构造了一个视觉问答数据集，该数据集基于 CLEVR 数据集构建，从推理所需步数、对象类型和属性、问题的语言表达的多样性和模糊性等方面着手，增加了数据集的难度，可以更好地考察模型在解决复杂空间推理任务方面的能力。
2. 设计了一种神经符号框架，实现 ASP 求解器与大语言模型的系统集成。该框架基于 DSPy 开发，融合了 DSPy 自动化提示生成和优化过程的显著优势，架构上包含反馈循环、提示和基于 ASP 的验证等模块，增强了视觉问答系统在解决空间推理问题方面的能力。
3. 基于本文所设计的神经符号框架，设计并实现了一个视觉问答领域的神经符号系统。

通过上述工作，本文提出的神经符号框架在提升大语言模型的空间推理能力方面，尤其是在处理复杂和不完全可见场景的问题时，展现出良好的性能和适应性。

**关键词：** 多模态，回答集编程，视觉问答，空间推理

# Abstract

Currently, large language models (LLMs) have demonstrated exceptional capabilities across various tasks. However, they often perform poorly in spatial reasoning, particularly when dealing with partially visible scenes. Answer Set Programming (ASP) is a formal method for knowledge representation and reasoning that has been shown to significantly enhance the spatial reasoning abilities of LLMs. Neural-symbolic methods utilize deep learning techniques to perceive and generate logical symbolic representations of input images and questions, subsequently employing ASP for reasoning. This approach has proven effective in improving the spatial reasoning capabilities of LLMs. However, existing neural-symbolic methods often suffer from limited generalization, complex prompt engineering and maintenance processes, and relatively high development difficulty. Additionally, they require manual design of extra rules for new scenarios, resulting in poor scalability. To address these issues, this paper proposes a novel neural-symbolic framework. The main contributions are as follows:

1. Construction of a Visual Question Answering (VQA) Dataset: Based on the CLEVR dataset, we expanded the complexity by introducing image occlusion, partial visibility, object interactions, noise interference, and more intricate reasoning tasks. This enhanced dataset aims to assess the model's performance in complex spatial reasoning tasks.
2. Design of a Neural-Symbolic Framework: We developed a system integrating an ASP solver with a large language model. Built upon DSPy, the framework leverages DSPy's advantages in automated prompt generation and optimization. The architecture includes feedback loops, prompts, and ASP-based validation modules, thereby enhancing the VQA system's ability to address spatial reasoning problems.
3. Implementation of a VQA System: Utilizing the proposed neural-symbolic framework, we designed and implemented a VQA system.

Through these efforts, the proposed neural-symbolic framework demonstrates strong performance and adaptability in enhancing the spatial reasoning capabilities of LLMs, especially when handling complex and partially visible scenes.

**Keywords:** Multi-modal, Answer Set Programming, Visual Question Answering, Spatial Reasoning

# 目录

# 插图目录

# 表格目录

# 算法目录

# 第一章　绪论

## 1.1　研究背景

随着计算机视觉（Computer Vision）和自然语言处理（Natural Language Processing）技术的迅猛发展，跨模态智能理解成为人工智能研究的重要方向之一。其中，视觉问答（Visual Question Answering, VQA）[1] 任务因其广泛的应用前景和挑战性，受到了学术界和工业界的广泛关注。

视觉问答是一种多模态任务，它要求计算机能够基于给定的图像内容，理解并回答关于该图像的自然语言问题。例如，给定一幅包含动物的图片，系统需要能够回答“这只动物是什么颜色?”或“图片中有几只猫?”等问题。这一任务的核心在于多模态信息的深度融合，即如何在视觉特征和语言信息之间建立有效的联系。

视觉问答在多个实际应用场景中具有重要价值。例如，在辅助盲人阅读图像信息、自主机器人理解环境、医疗影像分析、教育和娱乐等方面，VQA 技术都可以提供智能化的交互方式，提高系统的可用性和便利性。然而，由于数据的不确定性、语言表达的多样性、以及多模态信息融合的复杂性，VQA 仍然面临诸多挑战，例如开放域问题的泛化能力、推理能力的提升、以及对长尾问题的有效应对等。

随着现实场景愈发复杂多样，VQA 系统需要解决的问题也愈发困难，尤其是涉及物体间相对位置、形状、大小等空间关系的推理任务。空间推理指的是理解物体在某个二维或者三维的场景中的位置关系，并能够根据该场景的信息，以及本身所有的关于空间的常识知识，回答涉及物体排列、方向、距离等方面的问题。例如，在问答任务中，问题可能为“杯子在桌子的哪一侧?”或者“哪个物体位于另一个物体的上方?”这些问题要求 VQA 系统能够理解图像中物体的空间关系，进行推理，并给出准确的答案。

相比传统的 VQA 任务，空间推理更具挑战性。为了正确回答涉及空间推理的问题，VQA 系统需要能够理解图像中的空间布局，并且在多步骤推理中保持逻辑一致性。然而，现有的大语言模型在空间推理方面仍然存在显著局限。虽然已有的模型在目标检测方面表现十分出色，但它们在处理更复杂的空间关系时，尤其是在缺乏常识性空间理解的情况下，容易出现推理错误。因此，如何提升大语言模型在空间推理方面的能力，成为当前研究的重要课题。

为了解决 VQA 系统在空间推理方面的问题，研究者们提出了一些新的方法。近年来，神经符号方法（Neuro-Symbolic Methods）得到了广泛关注。神经符号方法使用深度学习技术进行感知，为输入的图像和问题分别生成符号表示，再使用符号系统进行推理求解，可以有效提升 VQA 系统在空间推理方面的能力。回答集编程（Answer Set Programming，ASP）是一种声明式编程范式，可用于解决复杂的人工智能问题。ASP 起源于对逻辑编程、非单调推理和知识表示的研究。ASP 因其具有表达性声明性的语言和以 Clingo 为代表的一些高效实现而流行起来。ASP 已经在学术界和工业界得到广泛应用，并被证明在人工智能的几个知识密集型应用中能够有效解决问题，如调度、产品配置、机器人、劳动力管理和决策支持等。ASP 能够以简洁和直观的方式来表示知识，允许用户相对容易地去表示复杂问题，而且 ASP 具有分

单调推理的特性，允许不完整信息的表示和默认推理。另外，ASP 对知识的表达能力，使得其支持集成各种类型的知识，包括规则、约束和偏好，有助于灵活解决问题。ASP 的这些特性使得它作为符号系统的杰出代表，被广泛应用于神经符号方法中。

基于上述背景，本文重点关注将神经符号方法在视觉问答中的应用，分析研究现有已使用神经符号方法的 VQA 系统在空间推理方面的优势和不足，，为进一步提升大语言模型在空间推理方面的能力开展研究。

## 1.2 相关研究现状

基于以上背景，本节主要从视觉问答的发展与现状、大语言模型在空间推理中的局限性、神经符号方法在视觉问答中的应用、视觉问答中复杂空间推理的难点等方面进行综述。

### 1.2.1 视觉问答的发展与现状

VQA 任务最早由 Antol[2] 等人在 2015 年提出，标志着 VQA 研究早期阶段的开始。这一阶段的 VQA 系统采用比较简单的架构，主要包括视觉编码器、语言编码器和融合模块。Ren[3] 等人将采用预训练的卷积神经网络（Convolutional Neural Network, CNN）作为视觉编码器提取图像特征，采用循环神经网络（Recurrent Neural Network, RNN）作为语言编码器或提取问题特征。Malinowski[4] 等人在视觉编码器上同样采用 CNN，在语言编码器上则是采用长短期记忆（Long Short-Term Memory , LSTM）来生成问题的答案。融合模块通常采用简单的拼接或注意力机制融合视觉和语言特征。这些系统在一些简单的 VQA 数据集上取得了不错的性能，但在处理复杂的空间推理问题时表现较差。

随着计算机视觉和自然语言处理技术的发展，VQA 系统逐步引入了更高级的特征提取与融合方法。例如，Yao[5] 等人将区域提议网络（RPN）引入到 VQA 模型中，以增强对图像中物体的感知能力。注意力机制也逐渐引入 VQA 模型中，例如 Xu[6] 等人提出堆叠注意力网络（SANs），通过多层次注意力机制，逐步聚焦于图像和问题的关键部分，从而提升了 VQA 模型的推理能力。

在 2019 年前后，得益于基于 Transformer 的预训练语言模型的兴起，VQA 研究取得了显著进展。如 BERT、ViLBERT 和 LXMERT 等模型通过联合训练大规模的视觉和语言数据，学得如何在同一嵌入空间中表示视觉和文本信息，极大程度上提升了 VQA 的准确性和鲁棒性。例如，Tan[7] 等人提出了 LXMERT 模型，将预训练的 BERT 模型与视觉编码器相结合，实现了对图像和问题的联合编码。

如今，VQA 的研究已经进入了全新阶段。随着大规模视觉语言模型（如 GLIP、GPT-4 等）的应用，VQA 模型已经能够处理更加复杂的任务，不仅处理特定问题时效果出色，而且可以进行跨模态推理，通过多种信息源之间的互动生成更加丰富和精准的答案。OpenAI 的研究团队[8] 在 CLIP 模型中，应用了自监督学习的方法，利用海量的图文对数据进行训练，实现了视觉和语言的统一表示，且训练过程无需人工标注，充分体现了自监督学习的优势。Salesforce 的研究团队[9] 在 BLIP 模型中，通过生成式预训练任务，提升了视觉语言模型在理解和生成方面的性能。

### 1.2.2　大语言模型在空间推理中的局限性

当前，大语言模型在空间推理方面的能力仍存在显著局限。Cohn[10] 等人在 RCC-8 框架下对 GPT-4 在定性空间推理任务中的表现进行了评估，发现尽管 GPT-4 能够理解一些简单的空间关系，但在处理复杂的空间关系时，往往无法准确应用 RCC-8 的规则，导致推理精度较低。此外，GPT-4 在不同推理步骤之间表现出明显的不一致性，且其推理过程缺乏明确的策略，往往依赖于经验或直觉做出决策，但这些决策往往缺乏可解释性，难以追踪其推理路径。Bang[11] 等人也指出，GPT 在空间推理任务中面临的挑战尤为突出，特别是在涉及多个空间区域、动态变化的场景或高维空间结构时，模型倾向于依赖已知的简单规则进行推理，但在面对复杂或不常见的空间配置时，难以有效处理。此外，GPT 在基于空间关系进行推理时，往往会出现错误，无法从准确的空间关系中得出合乎逻辑的结论。特别是在涉及常识性空间理解的任务中，GPT 的表现较差，难以理解如物体重叠、邻接、相交等基本的空间常识性假设。

为了解决这一问题，研究者们提出了一些新的方法，如基于神经符号方法的空间推理框架。这些方法通过将大语言模型与符号推理方法相结合，实现了对复杂空间推理问题的有效建模。例如，Wang[12] 等人提出了一种基于大语言模型和 Answer Set Programming（ASP）的神经符号框架，用于解决复杂空间推理问题。该方法通过将大语言模型与 ASP 求解器相结合，实现了对复杂空间推理问题的高效建模。此外，研究者们还提出了一些其他的神经符号方法，如基于知识图谱的推理方法、基于逻辑规划的推理方法等，以增强大语言模型在空间推理方面的能力。

### 1.2.3　ASP 在空间推理中的应用

ASP 作为一种形式化的知识表示和推理方法，已经在空间推理中得到了广泛的应用。ASP 具有表达能力强、推理效率高、易于理解和调试等优点，适用于解决复杂的空间推理问题。有一些学者在这一方面做了一些研究。例如，Wałęga[13] 等人提出 ASPMT(QS)，将非单调空间推理与基于理论的答案集编程相结合，整合了定性和定量的空间信息，解决了传统空间推理方法在处理复杂空间变化和组合约束时的局限性。Baryannis[14] 等人 3 将轨迹建模为由不重叠区域构成的序列，并提出了多种 ASP 编码方案（包括专门优化的编码和通用编码）以利用合成表来保证约束的一致性，展示了 ASP 在空间推理中的应用潜力。这些研究表明，ASP 在空间推理中具有很大的潜力，可以有效解决复杂的空间推理问题。

### 1.2.4　神经符号方法在空间推理中的应用

尽管研究者将 ASP 成功引入空间推理问题，取得了一些研究成果，但 ASP 在感知能力方面仍然存在一些不足，如对图像和自然语言的理解能力较弱，难以处理复杂的视觉问答问题。神经符号方法的出现，为解决这一问题提供了新的思路。神经符号方法引入深度学习技术，为 ASP 求解器提供了更加丰富的输入信息，从而提升了空间推理的准确性和效率。Tejas[15] 等人“逻辑透镜”（Lens of Logic, LOL）模型，采用了神经符号方法，利用了问题注意力和逻辑注意力机制，以识别和理解问题汇总的逻辑连接词，并且引入了 Fréchet 兼容性损失（Fréchet-Compatibility Loss），确保组件问题的答案与组合问题的答案在推理过程中保持一致

性。Thomas[16] 等人提出了一种结合神经网络和符号推理的方法，该方法能够有效地进行空间推理和逻辑推理，从而在复杂地视觉问答任务中提高准确性，展示了符号推理与深度学习结合的潜力，推动了神经符号方法在视觉问答中的应用。Pan[17] 等人对神经符号方法进行了改进，引入了一个自我优化模块，利用符号求解器的错误提示信息来修正符号表示，从而提高推理的准确性和可靠性。

### 1.2.5 视觉问答中复杂空间推理的难点

空间推理作为视觉问答系统所需的重点核心能力，涉及对图像中物体间拓扑关系、方位、尺寸等多种空间信息的理解、建模和推理。然而，现有方法在空间推理中仍面临诸多挑战。主要表现在以下几个方面：

#### 复杂空间关系的建模与表示

空间推理需要处理多层次关系，如拓扑关系（包含、相邻）、方位（左/右、前/后）、动态轨迹（移动路径）等。传统方法依赖预定义逻辑规则（如 RCC-8 拓扑模型），但难以适应开放域场景的多样性[18]。例如，自然语言中的“靠近”这个词汇，缺乏量化阈值，难以定量界定两个物体之间的距离在什么场景下为“靠近”这一关系，导致符号逻辑无法精确映射[19]。另外，基于注意力机制的模型虽然能定位目标区域，但难以捕捉长距离或者隐含的空间关联，进而导致错误定位。

#### 多模态对齐与语义鸿沟

视觉与文本模态的语义对齐，是 VQA 模型能够顺利进行空间推理的基础。但是存在跨模态特征映射的困难以及稀疏空间关系的问题。目前，有一些研究者试图解决这些问题。在跨模态特征映射这一问题上，Costanzino[20] 等人提出了一种使用轻量级多层感知器（MLPs）的方法，训练两个映射函数，基于正常样本预测一个模态的特征，通过比较实际特征和预测特征的不一致性来检测工业场景中的异常。在解决稀疏空间关系的问题方面，Zhang[21] 等人提出了 VoT 模型，通过 VoT 提示来提升 LLMs 的空间推理能力，通过生成视觉表示增强了模型对。这些方案在一定程度上解决了多模态对齐和语义鸿沟问题，但仍存在一定局限性。具体而言，多数模型通过全局或局部注意力加权融合多模态特征，但未显式建模空间关系的层次性。另外，一些模型通过子任务分解实现推理，但对空间逻辑的组合泛化能力有限。

#### 动态场景与实时推理的挑战

动态场景（如移动物体的避障路径规划）要求模型实时更新空间状态，但现有方法存在增量式推理不足和数据驱动的固有局限。具体而言，现有方法往往依赖于离线训练的模型，无法实时更新空间状态，导致推理结果复杂空间推理任务滞后，无法满足实时推理的需求。例如，符号推理（如 ASP）需重新求解完整的逻辑程序，导致延迟较高<empty citation>。另外像一些基于监督学习的模型，依赖于静态数据集（如 CLEVR），无法适应动态环境中的连续变化。最新的一些研究成果，如微软提出的 MVoT 框架通过生成可视化中间推理步骤，实现了结合

文本与图像信息背景下，对空间关系表示的动态调整，且在复杂场景中比传统思维链（Chain of Thought, CoT）的稳健性提升 20%。

### 数据集偏见与评估瓶颈

现有 VQA 数据集（如 CLEVR、VQAv2）存在显著偏差，如问题答案分布不均、问题类型单一等，导致模型在特定问题上表现优异，但在真实场景下泛化能力不足。Shrestha[19] 等人指出，像 CLEVR 这一类合成的数据集，虽然能够测试多步逻辑，但其几何简单性无法反映真实世界的复杂性，并且在问题设计上也隐含对特定答案的倾向性，如“左侧”常与特定物体绑定，进而导致模型依赖表面统计规律，而并非真实推理。目前，已有一些研究者使用零样本学习的方法，通过引入从未见过的问题-答案组合，测试模型的泛化能力。例如，Yang[22] 等人提出了一种零样本学习的方法，通过引入从未见过的问题-答案组合，测试模型的泛化能力。此外，新的一些 VQA 数据集如 VSR，通过控制答案分布来减少语言先验，以测试模型的纯视觉推理能力。

### 可解释性与鲁棒性不足

空间推理需要透明化的推理过程，以支持安全验证和决策解释。然而，现有的方法存在黑箱和对抗脆弱性等问题，无法提供可解释性保障。目前，有一些研究者试图在这一方面进行改进，如通过可视化路径的方式，提高模型的可解释性。例如，Li[23] 等人提出了多模态思维可视化框架（MVoT），旨在通过生成推理轨迹的图像可视化，增强多模态大语言模型（MLLMs）在复杂空间推理任务中的表现。该方法通过在自回归 MLLMs 中引入标记差异损失，显著提高了视觉连贯性和保真度。实验结果表明，MVoT 在多个任务中表现出的性能出色，尤其是在思维链（Chain of Thought，CoT）表现很差的场景中，展现出了显著的改进。Shah[24] 等人提出了一种基于循环一致性的训练框架，旨在增强 VQA 模型对语言变化的鲁棒性。该方法通过双向训练和循环一致性，提高了模型对语言变化的适应性，从而提高了模型的鲁棒性和泛化能力。

## 1.3　研究目标与内容

本课题的主要研究目标是研究并设计一种新的神经符号框架，通过实验，证明该框架能够有效提升大语言模型在复杂空间推理问题上的性能。最终将该神经符号框架接入视觉问答系统。

本文的主要研究内容包括以下三个方面：

(1) 在参考借鉴 CLEVR 数据集的基础上，在图像中的物体交互、图像遮挡和部分可见性、图像噪声干扰等方面进行扩展，构建一个新的视觉问答数据集，以更好地评估视觉问答模型在解决复杂空间推理问题上的能力

(2) 研究融合大语言模型及 ASP 的神经符号流水线，并使用 DSPy 来进行实现。通过 DSPy，将 ASP 求解器与大语言模型实现系统集成。在本文构建的视觉问答数据集上，进行对比实验验证。此外，为了评估，还进行了消融实验。

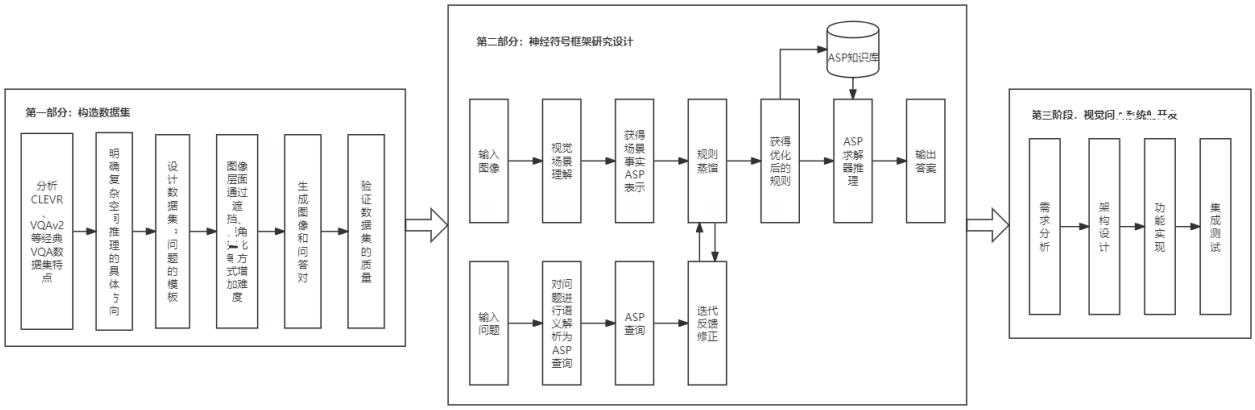



图 1.1: 技术路线图

(3) 以本文中所提出的神经符号框架为核心，设计并实现一个视觉问答系统。

## 1.4 研究方法与技术路线

本文针对以上研究目标和研究内容，综合多种方法进行研究。本文的研究主要涉及三个重点内容：

(1) 对于视觉问答数据集的构建，首先，用案例分析法研究 CLEVR 数据集，分析其特点和不足，为新数据集的构建提供理论依据和设计方向。然后采用文献研究法，调研现有的视觉问答数据集，学习它们在数据集构造过程中的方法和策略。

(2) 对于神经符号流水线的研究，首先，分析现有神经符号方法在空间推理方面的优势和不足。其次，进行模型构建。最后，通过实验验证，进行对比实验和消融实验，评估设计的神经符号框架的性能。

(3) 对于视觉问答系统的设计与实现，首先，进行需求分析，明确系统的功能和性能要求。其次，设计视觉问答系统的架构和模块划分，确定各模块实现所需的技术方案。再次，基于前文的研究成果，实现视觉问答系统的各个模块，并进行系统集成和测试。最后，通过实验验证，评估视觉问答系统的性能和可用性。

具体的技术路线如图1.1所示。

## 1.5 论文结构

本文共分为六个章节，各章节的主要内容具体如下：

第一章为绪论，总体介绍本文的研究背景及意义、相关研究现状与不足、研究目标与研究内容、研究方法与技术路线及本文的结构安排。

第二章为背景知识，对本文涉及到的主要技术进行介绍，具体包括 ASP 程序语法及 ASP 求解器、GLIP 以及 DSPy。

第三章为数据集构建。对本文所用的视觉问答数据集进行详细介绍，包括数据集的构造目的、数据集的研究方向、数据集的设计流程以及对数据集质量的验证。

第四章为神经符号框架的研究设计。对本文设计的神经符号框架进行详细介绍，包括流水线总体架构、视觉场景理解、语义解析、知识蒸馏、迭代反馈和规则修正、ASP 推理等模块的设计。

第五章为实验及结果分析。通过一系列实验，验证本文所提出的神经符号框架对复杂空间推理任务的提升效果，以及其在不同大语言模型架构上的泛化能力。

第六章中对本文工作加以总结，分析本文的创新点和不足之处，并对未来的研究方向进行展望。

# 第二章　背景知识

## 2.1　回答集编程

### 2.1.1　语法

项（terms）是 ASP 程序中最基本的元素，项可以是常量、变量或者函数。常量以符号常量或者数字表示（如 4、a）。变量以字符串来表示，要求首字母必须大写（如 Dog、Person）。函数由函数符号与若干参数构成，例如 $f(t_1,...,t_n),(n \geq 0)$。各参数 $t_i,(i=1,2,...,n)$ 也是项。若项中不存在函数符号与变量，则称该项为实例化项，否则称该项为非实例化项。

原子（atom）由谓词和项共同组成，例如 $q(t_1,...,t_n),n \geq 0$。其中，$q$ 为 $n$ 元谓词的标识符，$t_1,...,t_n$ 为项，$n$ 为 0 时，谓词之后的括号可以省略。原子用于表示不同项之间的关系，例如：$dog(animal)$、$family(jensen,lucy)$、$tomorrow\_sunny$，分别表示狗是动物、jensen 和 lucy 是家人以及明天晴天。如果原子中的每一项都是实例化的，则该原子是实例化原子，否则，该原子是非实例化的原子。例如，$dog(animal)$ 是实例化原子，$family(jensen,lucy)$ 是非实例化原子。原子 $q(t_1,...,t_n)$ 的强否定形式为 $\neg q(t_1,...,t_)n$，其中符号 $\neg$ 是经典逻辑中的否定，$\neg q(t_1,...,t_n)$ 表示各项 $t_i(1 \leq i \leq n)$。不符合 $q$ 所描述的关系，$\neg\neg a=a$，与经典逻辑中的排中律相同。文字（literal）可以是原子 $q(t_1,...,t_n)$，也可以是原子的强否定形式 $\neg q(t_1,...,t_n)$。若文字对应的原子是实例化的，则称该文字是实例化的。

**定义 2.1.1** (规则). 规则 $r$ 是具有如下形式的式子：

$$l_0\, or ...\, or\, l_m \leftarrow l_{m+1},...,l_n\, not\, l_{n+1},...,\ not\, l_p. \tag{2.1}$$

其中 $p \geq n \geq m \geq 0, l_i$ 代表文字，$not$ 为新的逻辑连接符，通常称作缺省否定符（default negation）或者失败即否定（Negation As Failure, NAF），又称为若否定，$not\, l_i$ 被读作“没有理由相信 $l_i$ 为真”，但是这并不表明 $l_i$ 为假。对 ASP 程序而言，$notnota$ 不与 a 等价。规则 r 读作“如果相信 $l_{m+1},...,l_n$ 为真，并且没有理由相信 $l_{m+1},...,l_p$ 为真，则相信 $l_0\, or\, ...\, l_m$ 为真”。将 $notl_i$ 称为缺省文字（default literal），文字与缺省文字共同组成拓展文字（extended literal）。

规则 $r$ 左侧的文字称为头部，可以表示为 $head(r)=\{l_0,...,l_m\}$。规则 $r$ 右侧的文字称为体部，可以表示为 $body(r)=\{l_{m+1},...l_n,not\, l_{n+1},...,not\, l_p\}$，规则的体部可以分为正体部与负体部，正体部表示为 $body^+(r)=\{l_{m+1},...,l_n\}$，负体部表示为 $body^-(r)=\{l_{n+1},...,l_p\}$ 。因此，规则 $r$ 可以表示为：

$$head(r)\ \leftarrow\ body^+(r),\ notbody^-(r) \tag{2.2}$$

一个 ASP 程序是有限条(2.2)所示的规则组成的集合。根据规则各部分所满足的相关条件，可以分别定义事实、约束、正规规则、缺省规则和严格规则，如定义(2.1.2)-定义(2.1.6)所示。

**定义 2.1.2** (事实). 当规则 $r$ 满足 $head(r) \neq \emptyset, body(r)=\emptyset$ 时，被称为事实（*fact*）。

**定义 2.1.3** (约束). 当规则 $r$ 满足 $head(r)=\emptyset, body(r) \neq \emptyset$ 时，被称为约束（*constraint*）。

**定义 2.1.4** (正则规则). 当规则 $r$ 满足 $|head(r)| = 1$ 时，被称为正则规则（*normal rule*）。

**定义 2.1.5** (缺省规则). 当规则 $r$ 满足 $body(r)^- \neq \emptyset$ 时，被称为缺省规则（*default rule*）。

**定义 2.1.6** (严格规则). 当规则 $r$ 满足 $body(r)^- = \emptyset$ 时，被称为严格规则（*definite rule*）。

根据上述各类规则的定义，本文进一步定义简单逻辑程序、缺省逻辑程序和正规逻辑程序如下。

**定义 2.1.7** (简单逻辑程序). 若程序 $P$ 由有限条严格规则组成，则 $P$ 被称为简单逻辑程序。

**定义 2.1.8** (拓展逻辑程序). 若程序 $P$ 由有限条缺省规则组成，则 $P$ 被称为拓展逻辑程序。

**定义 2.1.9** (正则逻辑程序). 若程序 $P$ 除约束外，由有限条正则规则组成，则 $P$ 被称为正则逻辑程序（*Normal Logic Program, NLP*）。

若不做特别说明，本文考虑的程序均为不含约束的正规逻辑程序，即程序中的每条规则头部有且仅有一个文字。

### 2.1.2 语义

#### 文字的可满足性语义解释和程序的模型

作为基础理论，首先给出 Herbrand 域、Herbrand 基的定义，分别如定义 2.10 和 2.11 所示，进一步定义规则与程序的实例化，最终在实例化程序下讨论文字的可满足性语义解释和可满足性。

**定义 2.1.10** (Herbrand 基). $P$ 是一个 *ASP* 程序，将程序 $P$ 中出现的常量和函数形成的所有不含变量的项的集合称为 *Herbrand* 域，使用 $\mathcal{U}_P$ 表示。

**定义 2.1.11** (Herbrand 域). $P$ 是一个 *ASP* 程序，出现在程序 $P$ 的规则中的谓词符号与 $\mathcal{U}_P$ 中的项组成的所有可能的不含变量的原子集合称为程序 $P$ 的 *Herbrand* 基，使用 $\mathcal{A}_P$ 表示，为了保持简洁，通常简写为 $\mathcal{A}$。

**定义 2.1.12** (规则的实例化). 给定 $P$ 中的一个规则 $r$，使用 $ground(r)$ 表示通过使用 $\mathcal{A}_P$ 中的常量对应地替换 $r$ 中变量而获得的规则集合，这一映射被称为规则的实例化。

**定义 2.1.13** (程序的实例化). 程序 $P$ 由一系列规则的集合组成，程序 $P$ 的实例化 $ground(P)$ 表示为式(2.3):

$$ground(P) = \bigcup_{r \in P} ground(r) \tag{2.3}$$

**定义 2.1.14** (文字的可满足性语义解释). 定义文字的可满足性语义解释 $I = \langle I^+, I^- \rangle$, 其中 $I^+ \cup I^- \subseteq \mathcal{A}_P$ （$P$ 是实例化程序），$I^+ \cap I^- = \emptyset$。$I^+$ 表示为已知为真的文字集合，$I^-$ 表示已知为假的文字集合。当 $I^+ \cap I^- = \mathcal{A}_P$ 时，$I$ 被称作完全可满足性语义解释（*complete interpretation*）。若 $I$ 不是完全可满足性语义解释，则 $I$ 中存在真值未确定（*undefined*）的文字。

**定义 2.1.15** (程序的模型). 假设 $P$ 是一个实例化的回答集程序，$I = \langle I^+, I^- \rangle$ 是一个可满足性语义解释。若文字 $l \in I^+$，则称 $I$ 满足 $l$，记作 $I \models l$；对缺省文字 $notl$，若 $l \in I^-$，则称 $I$ 满足 $notl$，记作 $I \models notl$。对于文字集合 $S$，若 $\forall l \in S, I \models l$, 则称 $I$ 满足集合 $S$，记作 $I \models S$，当一个规则 $r$ 满足 $I \not\models body(r)$ 或 $I \models head(r)$ 时，$I$ 满足规则 $r$。当 $I$ 满足程序 $P$ 的所有规则时，称 $I$ 是 $P$ 的一个模型。

## 回答集语义

**定义 2.1.16** (一致性文字集合). 给定一个文字集合 $S$，若 $S$ 中不同时包含 $l$ 和 $\neg l$，则集合 $S$ 是一致性文字集合，其中，$l \in S$ 是集合 $S$ 中的任意文字。

**定义 2.1.17** (可满足性，Satisfiability). 给定一致性文字集合 $S$，文字 $lit$ 与规则 $r$，若 $lit \in S$，则文字 $lit$ 满足集合 $S$。若 $head(r) \cap S \neq \emptyset$，则 $S$ 满足头部。若 $body^+(r) \subseteq S, S \cap body^-(r) = \emptyset$，则集合 $S$ 满足体部。若 $S$ 满足规则 $r$ 的头部或者 $S$ 不满足规则 $r$ 的体部，则 $S$ 满足规则 $r$，记作 $S \models r$。给定一个 *ASP* 程序 $P$，若 $\forall r \in P, S \models r$，则 $S$ 满足程序 $P$。

**定义 2.1.18** (规则的适用与阻塞). 给定一致性文字集合 $S$ 与规则 $r$，$S$ 满足 $body(r)$，则称规则 $r$ 在集合 $S$ 下是适用的（*applicable*），否则称规则 $r$ 在集合 $S$ 下被阻塞（*block*）。

**例 2.1.1.** 给定 *ASP* 程序 $P_3$，该程序包含两条规则：

$$r_1 : p \leftarrow q, s.$$
$$r_2 : s \leftarrow t.$$

下面通过分析说明集合 $S = \{p, q\}$ 对程序 $P$ 的可满足性。

(1) 因为 $p \in S$，所以文字 $p$ 与文字 $q$ 均满足 $S$；
(2) 因为 $S \cap head(r_1) = \{p\} \neq \emptyset$，所以 $S$ 满足 $head(r_1), S \models r_1$；
(3) 因为 $body^+(r_2) \subsetneq S$，所以 $S$ 不满足 $body(r_2)$，所以 $S \models r_2$；
(4) 因为 $S$ 满足程序 $P$ 中的每一条规则，所以 $S$ 满足程序 $P$。

**定义 2.1.19** (Gelfond-Lifshitz 规则（GL 规约）). 假设程序 $P$ 是给定的实例化程序，$S$ 是一致性文字集合，$l$ 是文字，$S \subseteq Lit(P), l \in Lit(P)$, $P$ 关于 $S$ 的 $GL$ 规约结果 $P^S$ 定义为式(2.4)：

$$P^S = \{head(r) \leftarrow body^+(r) \mid r \in P, body^-(r) \cap S = \emptyset\} \tag{2.4}$$

**定义 2.1.20** (简单逻辑程序回答集[<empty citation>]). 假设 $P$ 是简单逻辑程序，程序 $P$ 的回答集是 $S \subseteq Lit(P)$ 满足：

*(1)* 对于程序 $P$ 中的每一条规则 $l_0 \leftarrow l_1, ..., l_m$，如果 $l_1, ..., l_m \in S$，那么 $l_0 \in S$。
*(2)* $S \subseteq Lit(P)$ 且不存在 $S$ 的任何子集也满足程序 $P$ 的每一个规则，即 $S$ 是满足程序 $P$ 的最小集合；
*(3)* 如果 $S$ 包含互补的文字，那么 $S = Lit(P)$；

**定义 2.1.21** (拓展逻辑程序回答集). 假设 $P$ 是包含缺省否定的扩展逻辑程序，$S$ 是实例化文字集合，如果 $S$ 是 $P^S$ 的回答集，则 $S$ 是 $P$ 的回答集。如果程序 $P$ 没有回答集或者只有一个包含互补文字的回答集 $Lit(P)$，那么程序 $P$ 是不一致程序，否则，程序 $P$ 是一致性程序。

可以看出，使用 *GL* 规约，可以将扩展逻辑程序程序转换为简单逻辑程序，从而得到扩展逻辑程序的回答集。下面通过一个例子说明。

**例 2.1.2** (GL 规约与问答集). 给定一个实例化的 *ASP* 程序 $P_4$：

$$r_1 : p(a) \leftarrow notp(b)$$
$$r_2 : p(b) \leftarrow notp(a)$$
$$r_3 :\leftarrow p(b).$$

程序 $P_4$ 可能的回答集有四个：$S_1 = \emptyset, S_2 = \{p(a)\}, S_3 = \{p(b)\}, S_4 = \{p(a), p(b)\}$, 根据回答集的定义依次验证：

*(1)* $P^{S_1} = \{p(a) \leftarrow .\} \cup \{p(b) \leftarrow .\} \cup \{\leftarrow p(b).\}$，事实 $(p(b) \leftarrow .)$ 与约束 $(\leftarrow p(b).)$ 同时在 $P^{S_1}$ 中出现，因此 $S_1$ 不是 $P^{S_1}$ 的回答集，所以 $S_1$ 不是 $P$ 的回答集。

*(2)* $P^{S_2} = \{p(a) \leftarrow .\} \cup \{\leftarrow p(b).\}$，而 $S_2 \models ((p(a) \leftarrow .))$ 且 $S_2 \models (\leftarrow p(b).)$，因此 $S_2 \models P^{S_2}$，且不存在 $S_2' \subset S_2, S_2' \models P^{S_2}$，所以 $S_2$ 是 $P^{S_2}$ 的回答集；

*(3)* $P^{S_3} = \{p(b) \leftarrow .\} \cup \{\leftarrow p(b).\}$，事实 $(p(b) \leftarrow .)$ 与约束 $(\leftarrow p(b).)$ 同时在 $P^{S_3}$ 中出现，因此 $S_3$ 不是 $P^{S_3}$ 的回答集，所以 $S_3$ 不是 $P$ 的回答集；

*(4)* $P^{S_4} = \{\leftarrow p(b).\}$，可知 $P^{S_4}$ 的回答集为 $\emptyset$，因此 $S_4$ 不是 $P^{S_4}$ 的回答集，所以 $S_4$ 不是 $P$ 的回答集。

### well-founded 语义

上世纪 90 年代，良基语义（well-founded semantics）的概念由 Van Gelder A 提出，对于任意的回答集程序，都存在对应的良基模型（well-founded models），并且在多项式时间内可以计算出这个模型[25]。该语义模型是本文对 NAF 文字进行解释和对不一致程序原因进行分类的重要基础，下面介绍这一模型的定义及计算方法。

**定义 2.1.22** (立即结论). $P$ 是一个 *ASP* 程序，$S \subseteq \mathcal{A}_P \square V \subseteq \mathcal{A}_P$，则集合 $S$ 关于 $P$ 和 $V$ 的立即结论（*immediate consequence*）记作 $\mathcal{T}_{P,V}(S)$，定义如式(2.5)所示：

$$T_{P,V}(S) = \{a | \exists r \in P, head(r) = a, body^+(r) \subseteq S, body^-(r) \cap V = \emptyset\} \tag{2.5}$$

考察上式可以发现，当集合 $V$ 确定时，关于集合 $S$ 的函数 $T_{P,V}(S)$ 是单调的。因此，当集合 $V$ 固定时，该函数存在最小不动点（Least Fixed Point, LFP），使用 $lfp(.)$ 表示。

**定义 2.1.23** (良基模型). $P$ 是一个 *ASP* 程序，$P^+$ 是程序 $P$ 中的全部严格规则的集合，序列 $(K_i, U_i)_{i \geq 0}$ 定义如下：

$$K_0 = lfp(T_{P^+}) \qquad K_i = lfp(T_P, U_{i-1})$$
$$U_0 = lfp(T_{P,K_0}) \qquad U_i = lfp(T_{P,K_i})$$

若 $\langle K_j, U_j\rangle = \langle K_{j+1}, U_{j+1}\rangle$，且不存在 $0 \leq k < j$，使得 $\langle K_k, U_k = \langle K_{k+1}, U_{k+1}\rangle$（j,k 是正整数），则 $P$ 的良基模型 $WF_P = \langle W^+, W^-\rangle$，满足 $W^+ = K_j$，$W^- = \mathcal{A}\backslash U_j$

**例 2.1.3** (良基模型). 程序 $P_5$ 包含以下四条规则：

$$r_1 : q \leftarrow, notp. \qquad r_2 : p \leftarrow, notq.$$
$$r_3 : a \leftarrow b. \qquad r_4 : b \leftarrow .$$

首先得到 $P_5^+ = \{a \leftarrow b\} \cup \{b \leftarrow .\}$，而 $lfp(T_{P_5^+}) = a, b$，因此 $K_0 = a, b$；进一步计算得到 $U_0 = lfp(T_{P_5^+, K_0}) = \{a, b, p, q\}$，$K_1 = lfp(T_{P_5^+, U_0}) = \{a, b\} = K_0$，$U_1 = lfp(T_{P_5^+, K_1}) = \{a, b, p, q\} = U_0$。因此 $\langle K_0, U_0\rangle = \langle K_1, U_1\rangle$ 且不存在 $0 \leq k < 1$，$\langle K_k, U_k\rangle = \langle K_{k+1}, U_{k+1}\rangle$。因此程序 $P_5^+$ 的良基模型为 $WF_{P_5^+} = \langle \{a, b\}, \{\}\rangle$，$p$ 和 $q$ 在良基模型中属于未定义（$undefined$）。

### 2.1.3　求解器

求解器是实现 ASP 语义推理的核心工具，其工作原理是：通过搜索逻辑程序的最小模型（即回答集）来解决复杂的组合优化问题。

求解器的工作流程可以划分为两个阶段：基础化（Grounding）和求解（Solving）。基础化是将一阶逻辑程序实例化为命题逻辑形式，而求解则是基于 CDCL 算法搜索满足约束的模型。

截至目前，已经有众多 ASP 求解器。学术界和工业界所用的主流的求解器，可分为两类：单阶段求解器（如 Clasp、DLV）、多阶段求解器（如 Clingo）。两类求解器的主要区别在于，单阶段求解器独立地进行基础化与求解，而多阶段求解器则是集成了基础化与求解的交互式系统。

表2.1对目前的主流求解器的适用场景及核心优势进行了对比。因为 Clingo 求解器具有免费开源、使用简单、运行高效的特点，本文在求解 ASP 程序时，使用 Clingo 进行相关实验。

## 2.2　GLIP

GLIP（Grounded Language-Image Pretraining）是一种融合视觉与语言的预训练模型，旨在提高模型在目标检测、图像理解以及跨模态任务上的能力。GLIP 通过对大规模的图像-文本对数据进行联合训练，使得模型能够理解自然语言查询，并在图像中定位相应的目标，从而实现零样本（zero-shot）或少样本（few-shot）的目标检测任务。其架构如图2.1所示。

GLIP 采用了大规模的图像-文本数据进行预训练，使用基于 Transformer 的架构，并结合视觉和语言的特征进行跨模态学习。其训练过程主要包括：（1）文本引导的目标检测：输入不仅包含图像，还包含文本描述，使得模型能够基于自然语言查询来识别目标。（2）跨模态对齐：通过对比学习和注意力机制，使得视觉特征与语言特征在共享的表示空间中进行对齐。（3）自监督学习：利用大规模无标注数据进行自监督学习，提高模型的鲁棒性和泛化能力。

相比以往的目标检测技术，GLIP 的突出优势在于：（1）零样本检测能力：无需重新训练，即可识别新的类别。（2）泛化性强：由于使用了开放词汇的文本描述，GLIP 在实际应用中更加灵活。（3）跨领域适应性：可以应用于医学影像分析、自动驾驶、机器人视觉等多个领域。

| 求解器 | 核心技术 | 适用场景 | 核心优势 |
| --- | --- | --- | --- |
| Clasp | 多线程 CDCL<br>情性子句学习 | 大规模组合优化<br>工业级应用 | 支持并行搜索<br>性能竞赛冠军 |
| Clingo | 增量式基础化<br>多阶段编程 | 动态规则生成<br>交互式推理 | 集成 Python API<br>支持在线更新程序 |
| DLV | 数据库优化<br>存在线量化推理 | 知识表示型问题<br>语义 Web | 高效处理复杂规则<br>支持非确定域 |
| WASP | 分层弱约束优化<br>冲突驱动剪枝 | 偏好推理<br>多目标优化 | 加权约束高效处理 |
| Smodels | 稳定模型算法<br>部分求值 | 教学与研究<br>基础模型验证 | 算法透明易扩展 |
| LPARSE | 基础化预处理 | 规则实例优化 | 与 Smodels 配合使用 |
| IDP | 基于模型的推理<br>扩展一阶逻辑 | 复杂知识库管理 | 支持类型系统<br>高阶推理 |
| Alpha | 并行求解引擎 | 超大规模问题 | GPU 加速搜索 |

表 2.1: 扩展后的系统对比

目前，GLIP 在多个计算机视觉任务中表现优异，典型的应用包括：开放词汇目标检测（Open Vocabulary Object Detection, OVOD）、跨模态信息检索、复杂视觉问答（Visual Question Answering, VQA）、自动标注和数据增强。

## 2.3 大语言模型

## 2.4 DSPy

DSPy（Declarative Self-improving Language Programs）是由斯坦福大学开发的一种编程框架，旨在通过声明式的方式优化大语言模型（LLMs）的推理能力。DSPy 提供了一种高效的途径，使开发者能够构建自我改进的自然语言处理（NLP）系统，而无需手动调整复杂的超参数或微调大模型。

DSPy 的核心思想是将模型的优化过程抽象为声明式编程，使得用户能够专注于高层任务，而底层优化（如提示工程、少样本学习等）由系统自动完成。其主要特性包括：（1）模块化设计：开发者可以定义不同的语言任务模块，DSPy 会自动优化这些模块的执行方式。（2）自适应优化：系统会根据反馈数据不断调整推理策略，提高模型的准确性和鲁棒性。（3）多

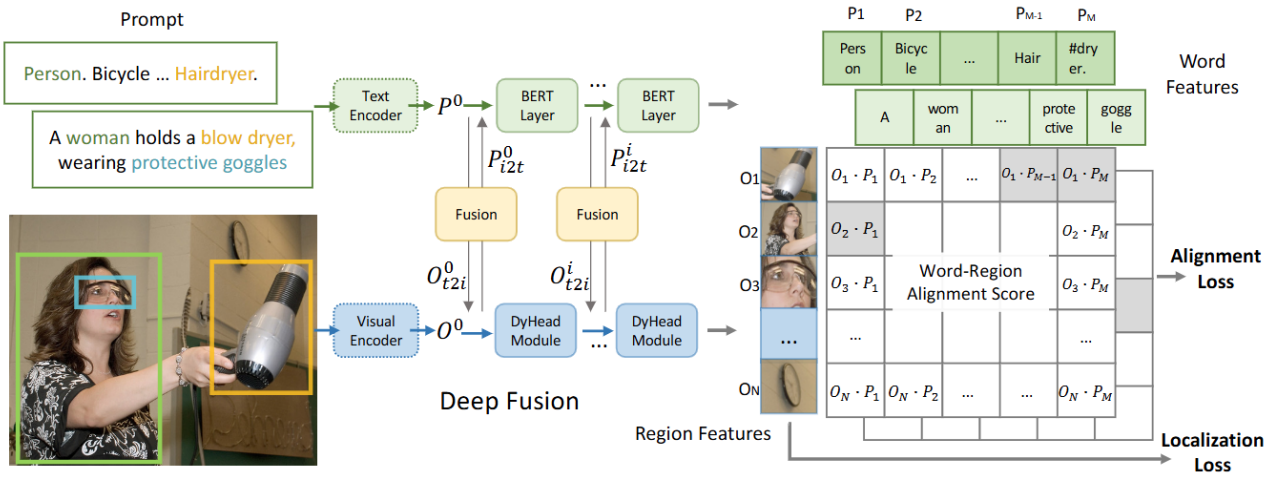



图 2.1: GLIP 结构示意图

模态支持：除了纯文本处理，DSPy 还可以用于多模态任务，如结合视觉和语言进行推理。

DSPy 采用了多种技术来提高大语言模型的推理效率和准确性，包括：（1）自动提示调整（Auto-Prompting）：根据任务需求，动态优化提示词，提高模型的响应质量。（2）少样本学习（Few-shot Learning）：利用少量示例进行任务适配，减少对大规模标注数据的依赖。（3）强化学习（Reinforcement Learning）：通过用户反馈不断优化推理过程，使模型的输出更加符合期望。

使用 DSPy 构建大语言模型应用，具有如下优势：（1）声明式编程范式：减少了对低层次参数调整的需求，使开发者可以更专注于任务逻辑。（2）高效优化：通过自动调整提示词和推理策略，提高大模型的推理能力。（3）可扩展性强：适用于各种 NLP 任务，包括文本摘要、信息抽取、代码生成等。

## 2.5　本章小结

# 第三章　数据集

本章详细描述基于 CLEVR 数据集，构造本文所用的 CLEVR-ASP 数据集的过程，以及数据集的设计思路、ASP 约束编码规则。

## 3.1　物体属性

CLEVR-ASP 数据集的图像中的每个物体，均有形状、尺寸、材质、颜色四种属性。每种属性的可能取值如下：

(1) 形状：圆锥体、球体。
(2) 尺寸：小、中、大。
(3) 材质：橡胶、金属。
(4) 颜色：红色、蓝色、绿色、黄色、灰色、棕色。

除了以上四种属性之外，由于图像中的场景被划分成 4 个区域，故 CLEVR-ASP 数据集也将图像中的物体按照其所在区域进行划分，分别标记为 0、1、2、3。每个物体都恰好位于某一个区域之中。将图像中的场景划分为多个区域，可以在多个级别上分别指定约束。

1. 区域约束：某个区域内的所有物体必须满足特定属性。例如，所有在区域 0 的物体必须是立方体或圆柱体。
2. 夸区域约束：定义了多个区域之间的关系。例如，区域 1 和区域 2 内同一个颜色的物体数量之和不得超过 2。
3. 全局约束：其中包含了一组影响整个场景的规则。例如，场景中至少有 2 个立方体。

以上所有约束，使用 ASP 来进行表示。例如：

```
:- object(X), at(X, 0), not hasProperty(X, shape, cube), not hasProperty
   ↪(X, shape, cylinder).
```

表示如果 X 在区域 0，那么它的形状必须是立方体或圆柱体。

## 3.2　环境表示

CLEVR-ASP 中的一个环境是由一组约束来定义的。每个约束决定了特定环境下物体的属性限制。数据集包含 30 个环境，每个环境中最多包含 15 个不同约束。部分约束的 ASP 编码表示以及对应表示含义见表3.1。

## 3.3 场景表示

CLEVR 数据集以场景图的形式表示场景，其节点表示使用其属性进行注释的对象，边表示对象之间的空间关系（前、后、左、右）。在 CLEVR-ASP 中，除了场景图表示之外，还在 ASP 中表示一个场景。下面以图 1 为例，展示部分场景的 ASP 表示：

```
%Objects in the scene
object(0). object(1). object(2). object(3).

%Attributes of objects
at(0, 2).
hasProperty(0, color, green).
hasProperty(0, size, large).
hasProperty(0, material, rubber).
hasProperty(0, shape, cylinder).
....

%Spatial relations between objects
front(1, 0). right(1, 0). ...
```

谓词 `object` 用于定义不同的物体（所有物体的名称用 0,1 等数字来表示），`hasProperty(Object, Attribute, Value)` 用于将对象的名为 Attribute 的属性的值设置为 Value。对象之间的空间关系用谓词 `left`、`right`、`front`、`behind` 来表示，例如 `left(A, B)` 表示 B 位于 A 的左侧。

## 3.4 图像生成

虽然 CLEVR 中的图像是通过随机采样的场景图生成的，但 CLEVR-ASP 生成的图像是基于已知符合环境约束的场景图。场景图的创建因此成为一个推理问题——在给定环境（基于答案集编程 ASP 的约束条件）和场景中预期物体数量 $n$ 的前提下，需要完成以下任务：将每个物体分配到四个区域之一，并为颜色、尺寸、形状和材质属性赋值，确保这些属性符合环境中的约束条件。

解决这一问题，需要 ASP 求解器来完成。具体而言，ASP 求解器将返回一个解集，集合中的每个答案（即 $n$ 个对象的一个一致属性值分配方案）代表一个场景图，或者说是场景中物体的一种可能配置。由于可能的配置方案众多，因此 CLEVR-ASP 随机采样了一百万个场景图用于后续的图像生成阶段。接下来，使用 Blender3 对场景图进行渲染，最终生成完整场景的图像。而部分场景的图像则是基于部分场景图生成的，具体做法是从实际场景图中随机移除一个物体，从而构造部分场景图。图3.1展示了场景图的构造过程。

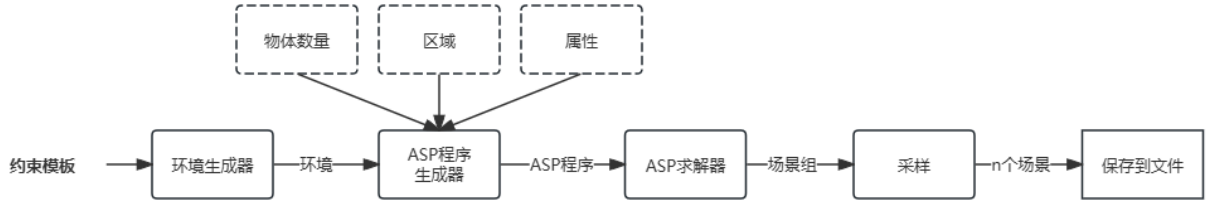



图 3.1: 生成环境以及该环境中的完整场景的流水线

## 3.5　问题表示

CLEVR-ASP 中的问题围绕部分场景中缺失的物体的颜色、大小、形状、材质这四个属性之一。具体的问题实例及 ASP 编码实例如下：

自然语言问题：与中等大小的红色物体的材质相同的，另一个圆柱体的颜色是什么？

```
query(Q) :- hasProperty(X, color, Q),
hasProperty(X, shape, cylinder),

hasProperty(Y, size, medium),
hasProperty(Y, color, red),
same_material(Y, X),
X != Y.
```

如果问题是关于属性 $A, A \in \{color, size, material, shape\}$，那么生成问题时，可能的解集 S 的基数为 $1 \leq |S| \leq |A|$，其中 $|A|$ 是属性 A 的所有可能取值集合的大小。例如，$|size| = 3 = \{large, medium, small\}$ 。如果生成的问题有 $|A|$ 个解，那么可判定该问题无效。例如，对于一个问物体尺寸的问题，答案是“尺寸可以为大或者中或者小”，显然这个答案是无效的，因为对于任意一个尺寸相关的问题，该答案均有效。各种类型的问题在个数上保持平衡。

## 3.6　问题生成

我们以 CLEVR 中的问题作为范本，来生成 CLEVR-ASP 中的问题。生成的问题均关于属性，避免生成是否类问题以及计数类问题。以下为问题模板样例：

```
What shape is the < Z2 > (size) < C2 > (color) < M2 > (material) [that is]
< R > (relation) the < Z > (size) < C > (color) < M > (material) < S > (
   ↪shape) ?
```

问题模板的实例化是基于相关图像的完整场景图完成的。感兴趣的对象始终是从完整场景中移除以生成部分场景图的对象。查询属性的选择需满足问题类型平衡的要求。查询对象的已知属性（填充上述模板中的 <Z2>、<C2> 或 <M2> 位置）是随机选取的。而 <R> 位置（即 left、right、front、behind 之一）的填充值也是随机选择的，但问题中的参考对象是根据完整场景中的空间关系选择的——具体而言，从与查询对象存在 <R> 关系的对象中选取一个。

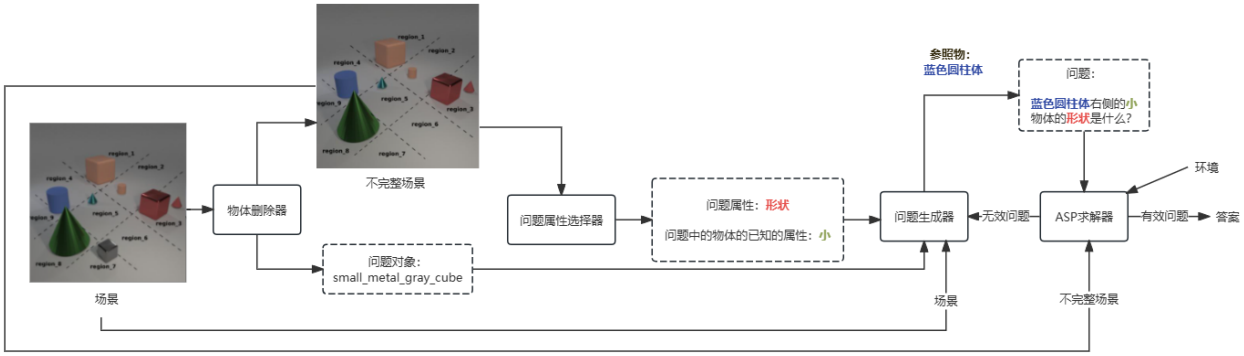



图 3.2: 生成部分场景和问题，并进行标记的流程

问题的 ASP 表示、部分场景以及环境中的约束都会被提供给 ASP 求解器，以确定问题的可能解。图3.2展示了问题生成的流水线过程。

## 3.7 本章小结

本章介绍了 CLEVR-ASP 数据集的构造过程，重点描述了如何基于完整场景图生成部分场景图，并通过 ASP（Answer Set Programming）实例化问题模板以确保问题的多样性和合理性。

首先，本章阐述了对象移除的原则，即如何选择查询对象以及如何保证其属性在问题类型上的均衡性。随后，详细说明了查询模板的填充策略，包括查询属性的选取、参考对象的确定以及空间关系的合理性，以确保生成的问题符合实际场景。最后，本章介绍了 ASP 求解器在问题生成过程中的作用，即利用 ASP 规则对部分场景进行推理，以确定查询属性的可能取值范围，从而生成符合逻辑约束的高质量视觉问答数据。

通过上述方法，CLEVR-ASP 数据集不仅保证了问题的可解释性，还增强了对复杂空间关系的推理能力，为视觉问答任务提供了更具挑战性的数据支持。

| 模板 | 描述 |
| --- | --- |
| **模板 1（取值约束）** | `:- object(X), at(X, R), not hasProperty(X, P1, V1).`<br>解释: 对区域 R 中的所有物体，它们 P1 属性的取值均为 V1。<br>具体实现: : $-object(X), at(X, 0), nothasProperty(X, color, red)$. |
| **模板 2（否定约束）** | `:- object(X), at(X, R), hasProperty(X, P1, V1).`<br>解释：对区域 R 中的所有物体，它们的 P1 属性的取值，均不能为 V1。<br>具体实现：:- object(X), at(X, 0), hasProperty(X, material, metal). |
| **模板 3（恰有 N 个约束）** | `:- #count{X: hasProperty(X, P1, V1), object(X), at(X, R)} != N.`<br>**解释**：在区域 R 中，恰好有 N 个物体的 P1 属性的取值为 V1。<br>具体实现：:- #count{X: hasProperty(X, size, small), object(X), at(X, R')} != 2. |
| **模板 4（至少有 N 个约束）** | `:- #count{X1, X2: sameProperty(X1, X2, P1), object(X1), object(X2), at(X1, R1), at(X2, R2)} < N.`<br>解释：在区域 R1 和区域 R2 中，至少有 N 对物体，它们的 P1 属性的取值都是 V1。<br>具体实现：:- #count{X1, X2: sameProperty(X1, X2, shape), object(X1), object(X2), at(X1, 1), at(X2, 2)} < 1. |
| **模板 5（或约束）** | `:- object(X), at(X, R), not hasProperty(X, P1, V1), not hasProperty(X, P1, V2).`<br>解释：区域 R 中的所有对象都具有属性 P1 的 V1 值或属性 P2 的 V2 值。<br>具体实现：:- object(X), at(X, 1), not hasProperty(X, color, yellow), not hasProperty(X, color, blue). |

表 3.1: 约束模板

# 第四章　神经符号框架的架构设计与实现

## 4.1　引言

本章对神经符号框架的架构进行详细的介绍，包括流水线总体架构、视觉场景理解、语义解析、知识蒸馏、迭代反馈和规则修正、ASP 推理等模块的设计和实现。框架的目标如下：

(1) 增强 VQA 系统在复杂空间关系推理方面的能力。
(2) 使用 Dspy 来完成对 LLM 的提示、优化，以降低神经符号方法的开发难度，并增强其可扩展性。

## 4.2　框架总体架构

框架的总体架构如图所示。

LLM 在整个框架中的作用有以下两点：

(1) 在语义解析模块中，对自然语言问题进行语义解析，将问题以 ASP 程序的形式表示出来。
(2) 在迭代反馈模块中，对 ASP 表示进行多次迭代优化，其中包括：添加规则，对规则进行一致性检查以及整合错误信息的反馈。

本文在语义解析模块和迭代反馈模块中，分别采用不同的 LLM，各自进行微调，以更好地满足不同任务的需求。

## 4.3　视觉场景理解

视觉场景理解是整个流水线的重要任务之一，其目标是从输入图像中提取结构化的信息，包括物体的属性（形状、颜色、大小等）、位置以及物体之间的空间关系。这些信息随后以 ASP 程序来表示，为使用 ASP 求解器进行推理提供基础。本文选择使用 GLIP 模型来完成目标检测任务，以实现高效且准确的目标检测和定位。下面本文

### 4.3.1　目标检测

目标检测是视觉场景理解的最基础的任务，其目标是从图像中识别出所有相关物体，并标注其边界框、类别和属性。本文选择 GLIP 模型作为目标检测的实现工具，原因在于其独特的语言-视觉预训练特性。基于大规模的图文对数据，GLIP 进行大量预训练，能够根据自然语言描述（如“红色立方体”）直接定位图像中的对应物体。这种能力特别适合 VQA 任务，使问题中的语言信息与图像内容高效对齐成为可能。

具体到实现中，本文首先对输入图像进行预处理，将其调整为 GLIP 模型的输入分辨率（例如，800×1333 像素）。随后，构造一组文本提示，以覆盖图像中可能出现的物体。考虑到本文第三章构造的数据集，本文使用以下短语集合：

(1) “大物体”、“小物体”等，描述物体的大小。
(2) “红色立方体”、“蓝色球体”、“绿色圆柱体”等，涵盖所有颜色和形状的组合。

GLIP 接收这些文本提示和图像作为输入，输出每个物体的边界框及其对应的类别标签。例如，对于一张包含“红色立方体”和“蓝色球体”的图像，GLIP 的输出可能如下：

1. 物体 1：类别 =“红色立方体”，边界框 =（$x_1, y_1, x_2, y_2$）。

2. 物体 2：类别 =“蓝色球体”，边界框 =（$x_3, y_3, x_4, y_4$）。

为了进一步提取物体的属性（如颜色、形状和大小），本文对 GLIP 的类别标签进行解析，将其分解为单独的属性值。例如，“红色立方体”被分解成 color=red、shape=cube。此外，通过边界框的面积（即 $(x_2 - x_1) \times (y_2 - y_1)$），本文进一步推断物体的大小属性，设定阈值以区分“大”和“小”物体。

### 4.3.2 空间位置提取

在完成目标检测后，需要确定每个物体的空间位置，以便为空间关系推理提供依据。GLIP 提供的边界框信息为位置提取提供了基础。本文使用以下方法计算物体的空间位置：

(1) 中心点坐标：对于每个物体，计算其边界框的中心点坐标 $(x_c, y_c)$，其中：

$$x_c = \frac{x_1 + x_2}{2}, y_c \frac{y_1 + y_2}{2}$$

该坐标表示物体在二维图像中的位置。

(2) 深度信息：通过图像的深度信息进一步推断物体的三维位置（$x$,$y$,$z$）。由于本文生成的数据集时使用的为 Blender 渲染引擎，故可直接从渲染引擎中获取深度信息。

最终，每个物体的位置信息被表示为 ASP 事实，例如：position(obj1, x1c, y1c, z1) 表示物体 1 的中心点位置。

### 4.3.3 空间关系提取

空间关系是复杂空间推理的核心，例如“左边”、“前面”、“遮挡”等关系。本文通过以下步骤从图像中提取这些关系：

(1) 二维空间关系：取二维平面中，物体的中心点坐标。对于两物体之间的距离，通过欧几里得距离公式进行计算，用于判断“靠近”等关系。对于“左右”关系，若物体 A 的 x_c 坐标小于物体 B 的 x_c 坐标，则认为 A 在 B 的左边，记作 left(objA, objB)。同理，对于“上下”关系，若物体 A 的 y_c 坐标小于物体 B 的 y_c 坐标，则认为 A 在 B 的上方，记作 above(objA, objB)。

(2) 三位空间关系：对三维空间中的物体 A 和物体 B，若 A 的 z 值小于 B 的 z 值，则认为 A 在 B 的前面，记作 in_front_of(objA, objB)。

(3) 遮挡关系：遮挡关系的判定需要边界框信息以及深度信息。若物体 A 的边界框与物体 B 的边界框重叠，并且 A 的 z 值小于 B 的 z 值，则 A 遮挡 B，记作 occludes(objA, objB)。

最终，所有提取的空间关系都被转化为 ASP 事实，例如：left(obj1, obj2) 表示物体 1 在物体 2 的左边，in_front_of(obj1, obj2) 表示物体 1 在物体 2 的前面；occludes(obj1, obj2) 表示物体 1 遮挡了物体 2。

### 4.3.4　场景图生成

为了将提取的物体属性、位置和空间关系整合为一个统一的结构化表示，本文生成场景图。场景图是一种图结构，其中：

1. 节点表示物体，带有属性标签（如 color=red，shape=cube）。

2. 边表示物体之间的空间关系（如 left、in_front_of）。

场景图的生成过程中，首先为每个检测到的物体创建一个节点，并标注其属性和位置。随后，根据提取的空间关系，在节点之间添加有向边，例如从 obj1 到 obj2 添加边 left_of。场景图随后被转化为 ASP 事实，以供后续环节使用。

### 4.3.5　实现细节与实例

为展示视觉场景理解模块的实际效果，本文提供一个具体示例。假设输入图像包含以下场景：

1. 一个红色大立方体位于图像左侧；

2. 一个蓝色小球体位于图像右侧，且在红色立方体前面。

通过 GLIP，检测结果如下：

1. 物体 1：类别 =“红色立方体”，边界框 =(50, 100, 150, 200)；

2. 物体 2：类别 =“蓝色球体”，边界框 =(250, 50, 350, 150)。

中心点计算：

1. 物体 1：(x_c, y_c) = (100, 150)；

2. 物体 2：(x_c, y_c) = (300, 100)。

假设深度信息显示物体 2 的 z 值 =75，小于物体 1 的 z 值 =50，则空间关系提取如下：

1. left_of(obj1, obj2)（因为 100 < 300）；

2. above(obj2, obj1)（因为 100 < 150）；

3. in_front_of(obj2, obj1)（因为 75 < 50）。

最终生成的 ASP 事实为：

```
color(obj1, red).
shape(obj1, cube).
size(obj1, large).
position(obj1, 100, 150, 50).

color(obj2, blue).
shape(obj2, sphere).
size(obj2, small).
position(obj2, 300, 100, 75).

left_of(obj1, obj2).
above(obj2, obj1).
in_front_of(obj2, obj1).
```

### 4.3.6 技术挑战与解决方案

在实现过程中，遇到了以下几点技术上的难题，本文提出了相应的解决方案：

(1) GLIP 的准确性：GLIP 可能对某些复杂场景（如物体密集或遮挡严重）产生误检。为解决此问题，本文在 GLIP 的基础上引入后处理步骤，通过非极大值抑制（NMS）去除重复检测，并结合深度信息过滤误检。
(2) 空间关系的鲁棒性：二维空间关系的提取可能因视角变化而失效。为此，本文优先利用深度信息，并在缺乏深度信息时使用几何约束（如边界框重叠面积）提高鲁棒性。
(3) 计算效率：GLIP 的推理速度可能限制流水线的实时性。本文通过批处理图像和优化提示设计，显著减少了推理时间。

### 4.3.7 小结

通过 GLIP 进行目标检测，并结合空间关系提取和场景图生成，成功将输入图像转化为结构化的 ASP 事实。这一模块为后续的语义解析和神经符号推理提供了可靠的基础数据。下一节将介绍如何将自然语言问题解析为 ASP 查询，从而与视觉场景理解的输出无缝衔接。

## 4.4 语义解析

语义解析的主要任务是，通过 LLM，使用上下文学习的方法，将自然语言问题转为用 ASP 进行表示，以便与视觉场景理解提取的场景事实结合进行逻辑推理。

直观上，LLM 可能很难直接解决复杂的推理问题。然而，LLM 已经在理解文本输入并将其转化为形式化程序方面取得了巨大成功，例如程序代码[26] 和数学方程[27]。接下来，本文将介绍通过微调 LLM，根据自然语言问题，生成正确的 ASP 程序。

### 4.4.1　ASP 模板设计

根据第三章构造的数据集，本文设计了一组 ASP 模板，以覆盖数据集中可能出现的问题类型。以下将针对各问题类型介绍相应的 ASP 模板设计。

#### 基础存在性问题

基础存在性问题是最简单的问题类型，其形式为“是否存在一个满足条件的物体”。本文设计了以下 ASP 模板：

```
模板：是否存在一个[颜色][材质][形状]的物体？
提示：是否存在红色金属立方体？
编码：exists :- object(ID, red, metal, cube, _, _, _).
```

#### 三维空间关系问题

```
模板：物体A（[属性]）是否在物体B（[属性]）的[方位]方，且两者在Z轴上[关系]？
提示：红色球是否在蓝色立方体的左上方？
编码：left_above(O1, O2) :- object(O1, red, _, ball, X1, Y1, Z1), object(O2,
    ↪ blue, _, cube, X2, Y2, Z2), X1 < X2, Z1 > Z2 + 10.
exists :- left_above(O1, O2).
```

#### 多跳推理问题

多跳推理的跳数的取值范围为 2-5，主要考察模型的推理能力。本文对此设计了以下 ASP 模板：

```
模板：若[物体A属性]在[物体B属性]的[方位1]，且[物体B属性]在[物体C属性]的[方位2]
    ↪，那么[物体A属性]相对于[物体C属性]的位置是什么？
提示：若红色球在蓝色立方体左边，且蓝色立方体在绿色圆柱体前面，那么红色球相对于
    ↪绿色圆柱体的位置？
编码：transitive_left_front(O1, O3) :- left(O1, O2), front(O2, O3).
final_relation(O1, O3) :- transitive_left_front(O1, O3), object(O1, red, _,
    ↪ball, _, _, _), object(O3, green, _, cylinder, _, _, _).
```

多参考系问题

```
模板：以[物体属性]为参照物，[目标物体属性]位于其哪个方向？
提示：以蓝色立方体为参照物，红色球是否在其右后方？
编码：local_right_behind(Target, Ref) :- object(Ref, blue, _, cube, Xr, Yr,
    ↪Zr), object(Target, red, _, ball, Xt, Yt, Zt), Xt > Xr, Zt < Zr.
exists :- local_right_behind(Target, Ref).
```

动态反事实问题

```
模板：如果移除[物体属性]，那么[某条件]是否成立？
提示：如果移除所有红色物体，是否还存在比蓝色立方体大的球？
编码：hypothetical_world(ID) :- object(ID, _, _, _, _, _, _), not (object(ID
    ↪, red, _, _, _, _, _), removed(ID)).
hypothetical_condition :- hypothetical_world(ID1), object(ID1, _, _, ball, _
    ↪, _, S1), object(ID2, blue, _, cube, _, _, S2), S1 > S2.
```

对抗性样本问题

```
模板：图中是否有[数量]个[属性]物体？注意[干扰条件描述]。
提示：是否有3个红色球？注意反光物体可能是玻璃材质而非球体。
编码：valid_ball(ID) :- object(ID, red, glass, ball, _, _, _), not material(
    ↪ID, metallic).
count(N) :- N = #count{ ID : valid_ball(ID) }.
```

### 4.4.2 ASP 查询生成

为了对语义解析模块的专用 LLM 进行微调，以更好地生成 ASP 查询，本文首先根据上述设计的 ASP 模板，生成训练数据，再对该 LLM 进行训练。生成的数据集共包括 10 万条训练数据，各类型占比见表。数据生成完成后，采用 Clingo 求解器。

目前，工业界和学术界已有的 LLM 有很多，如 GPT、Copilot、Gemini 等。为了选择最适合的 LLM，本文首先对主流 LLM 在生成 ASP 查询这一方面进行评估。LLM 的参数大小、架构及训练数据源见表4.1。从表4.1中，不难看出 Gemma 2B 是在这一组 LLM 中最小的模型。

最终获得的数据集包括 10 万条训练数据，将数据集按照 8:2 的比例划分成训练集和验证集，并保持每个问题类别的数据分布比例保持一致。

为了能够进行高效微调，最终选择参数量最小的 Gemma 2B 作为语义解析模块的 LLM。Gemma 2B 模型采用了分组查询注意力（GQA）机制，将查询头划分为 2 组共享键值投影，相比传统多头注意力（MHA）减少 25% 内存占用。同时引入局部滑动窗口注意力（窗口大小 4096）与全局注意力交替层，平衡长程依赖建模与计算效率。

| 模型 | 参数量 | 架构 | 训练数据源 |
|---|---|---|---|
| ChatGPT 3.5 | 175B | 编码器-解码器架构 | 网络数据 |
| Copilot | 1.5T | 编码器-解码器架构 | Github 仓库 |
| Gemini | 1T | 混合专家模型 | 文档，书籍，代码 |
| Gemma | 2B–7B | 纯解码器 | 文档，数学，代码 |
| LLaMa2 | 7B–13B | 纯解码器 | 网络数据 |
| LLaMa3 | 7B–13B | 纯解码器 + 分组查询注意力机制 | 网络数据 |
| Mistral | 7B–141B | 纯解码器 + 分组查询注意力机制 | 网络数据 |

表 4.1: LLM 对比详细信息，其中参数量以十亿（B）或者万亿（T）为单位。

为了验证 LLM 生成 ASP 查询的正确性，此处通过 Python API 调用 ASP 求解器 Clingo。Clingo，从而定义一个函数 $f(P) = AS(P)$，其中 $AS(P)$ 表示程序 P 的所有回答集（可能为空）。对于 LLM 根据提示 $x$ 生成的 ASP 程序 $y$ $P_L(|x)$，以及与 $x$ 对应的能够真实表示问题的 ASP 程序 $y^*$，按照以下步骤来进行验证：

1. 事实集合构建：构造一组事实，用集合 $F_{y^*}$ 表示，其代表了问题 $x$。

2. 程序合成。将 $F_{y^*}$ 与 $y$ 合并，得到新的 ASP 程序 $P = y \cup F_{y^*}$。同理，将 $y^*$ 与 $F_{y^*}$ 合并，得到新的 ASP 程序 $P^* = y^* \cup F_{y^*}$。

3. 语法命中：调用 Clingo 计算 $f(P)$，若未发生解析错误，则判定为语法命中。

4. 语义验证：进一步计算 $f(P)$ 与 $f(P^*)$，分别得到 $AS(P)$ 与 $AS(P^*)$。若 $AS(P)$ 与 $AS(P^*)$ 完全匹配，则判定为语义命中。

此后，基于 DSPy 框架发起对 LLM 的调用，将自然语言问题转化为 ASP 查询。此处使用 DSPy 的 Template 组件定义提示模板，指导 LLM 如何生成指定格式的 ASP 程序。其中，也使用了 Example 组件，用于构建优化过程中使用的示例数据，封装输入和输出的对应关系，以帮助 LLM 进行学习。

### 4.4.3　实验与结果分析

为检测语义解析模块的性能，本文使用上节构造的数据集进行实验。考核指标选取上一节定义的语法命中率和语义命中率。将未经过训练的 LLM 与本文经微调后的 Gemma 2B 模型进行对比。

实验结果如表及图所示。语法正确并不能保证语义正确。例如，Gemma 7B 实现了 45% 的语法正确率，但是语义正确率明显偏低。根据实验结果分析，没有任何模型能够做到全方面正确，轻量级模型的综合表现最差。此外，GPT-4.0 turbo、Copilot 实现了 100% 的语法正确率。同时也注意到，虽然 Gemini 和 LLaMa3 70B 的参数规模和前述模型相当，但是准确率却明显较低。本文的经微调后的 Gemma 2B 模型在所有模型中的语义准确率最高。根据实验结

果，也能够看出，当生成的 ASP 程序在语法上正确时，其语义正确的可能性也会更大。所有模型在每种类型的问题上的语法正确率和语义正确率的对比见表4.2。

| 模型 | 基础存在性 | | 三维空间关系 | | 多跳推理 | | 多参考系 | | 动态反事实 | | 对抗性样本 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 语法 | 语义 | 语法 | 语义 | 语法 | 语义 | 语法 | 语义 | 语法 | 语义 | 语法 | 语义 |
| GPT-4.0 turbo | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Copilot | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Gemini | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Gemma 2B | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| LLama3 turbo | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **微调模型** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

表 4.2: 不同模型在各问题类型上的语法正确率和语义正确率的对比

通过对比综合表现最佳的几个模型所生成的 ASP 程序，发现所有上述模型均未对多跳推理问题进行正确编码。

## 4.5 迭代反馈与规则修正

迭代反馈模块是整个管道的最核心部分。具体而言，Clingo 求解器执行输入的 ASP 程序，并输出信息。如果在执行过程中出现错误，Clingo 将会把这些错误信息输出。而 LLM 则对这些错误信息进行分析，并对 ASP 程序进行优化。优化后的 ASP 程序再次输入 Clingo 求解器，如此循环至多 3 次，最终获得优化修正后的 ASP 程序，以供进行正式推理。不同的错误信息对应的问题不同，所需要的对 ASP 程序进行修正的方式也不同。指导 LLM 对不同问题进行修正的提示，也会因为修正方式的不同而不同。针对不同的错误信息，本文设计了不同的提示模板，以指导 LLM 进行修正。

根据 LLM 的反馈并对提示模板进行修改，再重新提示 LLM，以最终得到较优的提示模板。该过程中，需要多次和 LLM 进行交互，流程比较复杂。本文使用 Dspy 来对该过程进行管理。Dspy 的模块化特性增强了模块之间的记忆保留能力，方便进行参数自适应调整和优化。此外，Dspy 支持自动对 LLM 提示词和参数进行优化，极大降低了人工修改提示模板的工作量。为了方便调试，所有模块的输出日志均记录错误信息与状态提示。

DSPy 提供的三种优化器及人工提示工程的对比结果见表4.3。本文最终选择了 BootstrapFewShow 优化器，主要理由是：人工标注 ASP 事实的成本较高，平均每个样本的标注时间为 3.5 分钟。而 BootstrapFewShot 专门为少样本场景设计，仅需 15-20 个标注样本就可启动优化。基于本文有限的标注样本，选择了 BootstrapFewShot 优化器，以提高优化效率。此外，BootstrapFewShot 优化器能够进行多阶段分层次优化。具体而言，其将优化过程拆分为教室优化和学生训练两个阶段，前者生成高质量的示例，后者完成知识蒸馏，以确保优化的效果。BootstrapFewShot 优化器也支持限制最大错误次数和迭代轮数，避免无限循环和资源浪费。

表 4.3: Dspy 框架支持的优化器比较

| 优化器名称 | 主要功能 | 适用场景 |
| --- | --- | --- |
| BootstrapFewShot | 通过在提示中自动生成并包含优化示例来扩展签名。 | 少样本学习 |
| BootstrapFewShot-WithRandomSearch | 在 BootstrapFewShot 的基础上，对生成的示例进行多次随机搜索，选择优化后的最佳程序。 | 少样本学习，需要更高精度 |
| MIPRO | 在每个步骤中生成指令和少量示例，使用贝叶斯优化来有效地搜索模块中的生成指令和示例空间。 | 需要复杂指令和示例优化的场景 |
| BootstrapFinetune | 将基于提示的 DSPy 程序提炼为较小语言模型的权重更新，微调底层大型语言模型以提高效率。 | 需要微调模型以提高效率的场景 |

## 4.6 ASP 推理

ASP 推理阶段中，ASP 求解器接收经过优化后的 ASP 查询语句、视觉场景理解模块提取的 ASP 事实以及已有常识的 ASP 表示，进行逻辑推理，最终获得答案。本文使用 Clingo 求解器来进行求解，其工作过程分为基础化（grounding）阶段和求解（solving）阶段。

基础化阶段将 ASP 程序中的变量替换为常量，生成一个新的 ASP 程序。Clingo 通过使用内置的基础化器分析程序，生成所有可能的规则。例如，对规则 $a(X) : -b(X), notc(X).$，其将被展开为所有可能的值为 X 的实例。

求解阶段中，Clingo 采用类似 SAT 求解器的冲突驱动答案集求解（CDNL）方法。CDNL 方法通过迭代地添加约束，直到找到一个满足所有约束的解，或者证明无解。具体而言，Clingo 基于如下步骤进行求解：（1）初始化。从空分配开始（没有原子被标记为真）；（2）选择与分配。选择一个未分配的原子，猜测其真值为真或假；（3）传播。根据当前分配和规则，推导其他原子的真值。例如，若规则 $a : -b, notc.$ 满足，b 为真且 c 不在答案集中，则 a 必须为真；（4）冲突检测。若分配导致规则冲突（如与已有的约束条件相抵触），记录冲突原因；（5）回溯与学习。若发生冲突，回溯到之前的选择点，学习冲突子句以避免未来类似错误；（6）验证。当所有原子分配完成后，检查是否为答案集，即确保它是程序的模型，且最小化（无子集也能满足规则）。

相比其它的 ASP 求解器，Clingo 在以下几个方面进行了优化：（1）从冲突中学习新的规则，为将来进一步的搜索提供指导；（2）使用多线程实现并行化，同时搜索多个路径，加速求解过程；（3）使用启发式方法决定分配顺序，例如优先选择高影响力的原子；（4）预测可能发生的冲突，选择可能导致冲突的原子优先分配，减少搜索空间。

## 4.7 实验与结果分析

### 4.7.1 对比试验

为全面评估神经符号框架的有效性以及泛化能力，本文选取了目前主流的三种 LLM：DeepSeek R1、Llama3 和 GPT-4 turbo，在其上应用神经符号框架，进行对比。以上既包含 DeepSeek 这类轻量级专用模型，也涵盖 GPT-4 turbo 这类通用型先进系统，而 Llama3 性能和效率之间取得了平衡，属于居中水平的模型。通过在多个基座上进行实验，证明神经符号框架对不同 LLM 的泛化能力。

比较基准一选取直接提示 VLM 的方式，即直接将自然语言问题和图像输入到 VLM 中，并不给予任何的额外提示。直接提示 VLM 的方式虽然简单，却是评估模型的关键基准，因为直接提示方法能够反映模型在没有任何外部推理辅助机制的情况下，自身对空间问题的处理能力。

比较基准二采用“事实 + 规则”的提示方法。该方法的核心思路是：指示 LLM 使用预先定义的谓词，将输入的自然语言问题转换为结构化事实，然后 LLM 应用相关逻辑规则，通过自然语言推理答案。“事实 + 规则”的提示方法满足了将原始自然语言问题转为结构化符号表示，以配合视觉场景理解所得的 ASP 事实以及原有常识，共同进行推理的核心需求。同时，该方法使用具有精确参数结构的谓词对 LLM 进行提示，使得 LLM 可以创建一致的中间态的知识表示，为问题解答提供便利。综合来看，“事实 + 规则”的提示方法作为一种简化流程，既保留了形式化推理的优势，同时能够避免在生成 ASP 程序时和 LLM 进行多次交互，进而降低对计算资源的要求，降低了成本，也减少了对外部求解器的依赖。

实验结果见表4.4，本文提出的神经符号框架在 DeepSeek R1、LLama3 和 GPT-4.0 turbo 上均超过了两种比较基准方法，证明大语言模型与逻辑推理结合对于解决复杂空间推理问题的有效性。

| 模型及方法 | 基础存在性问题 | 三维空间关系问题 | 多跳推理问题 | 多参考系问题 | 对抗性样本问题 | 总体 |
|---|---|---|---|---|---|---|
| **DeepSeek R1** | | | | | | |
| 直接提问 | 1 | 1 | 1 | 1 | 1 | 1 |
| 事实 + 规则提示方法 | 1 | 1 | 1 | 1 | 1 | 1 |
| 神经符号框架方法 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Llama3** | | | | | | |
| 直接提问 | 1 | 1 | 1 | 1 | 1 | 1 |
| 事实 + 规则提示方法 | 1 | 1 | 1 | 1 | 1 | 1 |
| 神经符号框架方法 | 1 | 1 | 1 | 1 | 1 | 1 |
| **GPT-4.0 turbo** | | | | | | |
| 直接提问 | 1 | 1 | 1 | 1 | 1 | 1 |
| 事实 + 规则提示方法 | 1 | 1 | 1 | 1 | 1 | 1 |
| 神经符号框架方法 | 1 | 1 | 1 | 1 | 1 | 1 |

表 4.4: 不同模型及方法在各问题类型上的表现

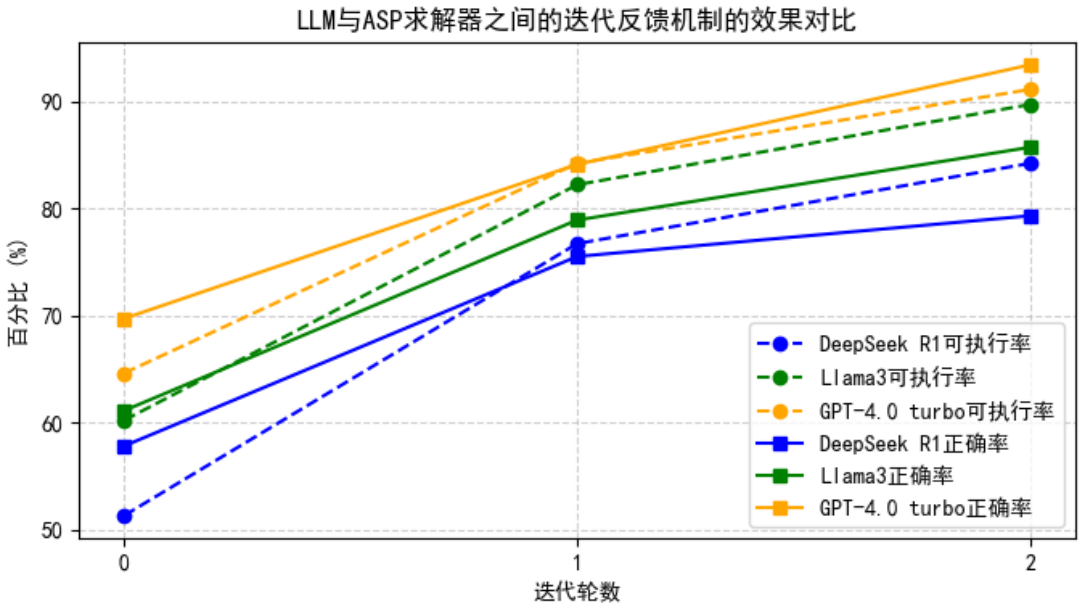



图 4.1: LLM 与 ASP 求解器之间的迭代反馈机制的效果对比

### 4.7.2　消融实验

尽管 LLM 在语义解析任务中表现出色，但其生成的 ASP 程序的正确率仍有较大的提升空间。Feng[28] 等人的研究表明，将自然语言直接转为逻辑规则的成功率一般较低。而神经符号框架通过在 LLM 和外部 ASP 求解器之间设立反馈循环机制，使 LLM 能够根据 ASP 求解器在试图求解 ASP 程序之前的语法检查结果，对生成的 ASP 程序中的错误进行修正，极大提高了将自然语言问题转为 ASP 程序的准确率。

为进一步探讨神经符号框架的迭代反馈机制的作用，本文重点关注 ASP 求解器执行过程中常发生的三类主要错误：解析错误、实例化失败以及求解阶段失败。另外，即便生成的 ASP 程序执行成功，也有很大可能由于自然语言问题与 LLM 生成的 ASP 程序之间不一致，而产生与真实答案偏离的结果。

在测试中，迭代反馈机制显著提升了所有模型的性能。如图4.1所示，经过两轮迭代反馈之后，DeepSeek R1 的可执行率从 51.3% 提升到了 84.2%，LLama3 的可执行率从 60.2% 提升到了 89.7%，GPT-4.0 turbo 的可执行率从 64.6% 提升到了 91.1%。在准确率方面，DeepSeek R1 的准确率从 57.8% 提升到了 79.3%，LLama3 的准确率从 61.1% 提升到了 85.7%，GPT-4.0 turbo 的准确率从 69.7% 提升到了 93.4%。以上结果表明，LLM 与 ASP 求解器之间的迭代反馈机制能有效解决自然语言到逻辑程序的转换过程中面临的问题。对准确率和可执行率的提升，主要集中在第一轮迭代反馈中，此后继续迭代的效果呈现边际递减趋势。由于计算资源有限，本次消融实验仅进行了三轮。实验结果有力证明，迭代反馈机制在提升 ASP 程序的可执行率和正确率方面具有明显效果，充分验证了神经符号方法的有效性。

## 4.8　本章小结

# 第五章　问答系统的构建

## 5.1　需求分析

本系统旨在基于本文提出的神经符号框架，构建一个视觉问答系统，既能利用深度学习技术对图像进行特征提取，用 LLM 对自然语言进行语义解析，又能通过 ASP 求解器实现推理，同时为用户提供友好、直观的前端交互界面。

### 5.1.1　功能需求

核心功能上，需要实现以下几个功能：（1）用户输入与对话处理。系统需要能够支持文本输入、图像输入，未来远期可以考虑加入语音输入。此外，提供自动补全、拼写检查等输入辅助功能，以改善用户的输入体验。（2）实时对话回复。系统能够通过 Dspy 的 SDK，发起对本地 LLM 的调用，而 LLM 能够实时处理用户的问题，返回高质量的回答。另外，要能够支持多轮对话，能够保持上下文关联，具有记忆功能。（3）对话历史记录。系统需要能够自动保存用户每次的对话记录，以供用户查看。（4）帐户与认证。系统需要支持用户注册、登录、账号设置等功能。（5）数据统计与反馈。系统需要收集交互数据（如访问量、对话时长、常见问题等），供后续模型优化使用。另外，要提供用户反馈入口，支持用户评价回复质量。

辅助功能方面，主要围绕以下几点：（1）响应式设计。前端能够自适应桌面、平板和手机等不同设备。（2）辅助说明。提供 FAQ、帮助文档和使用指南，方便用户进行查阅。（3）安全与异常处理。能够对恶意输入、不符合规范的输入进行拦截和提示。

### 5.1.2　非功能需求

非功能需求主要聚焦于以下几个方面：（1）性能需求。系统需要能够在较短时间内完成对用户问题的回答，如果确实因为某些原因导致无法及时回答，需要能够给出合理的提示，例如“正在思考...”等。另外，各个微服务应支持高并发访问，保证系统能够承受一定规模的大流量访问。（2）可拓展性。对系统要按照主要功能来进行拆分，形成若干个微服务，并且各个微服务之间要能够相互独立，实现松耦合。当系统需要进行拓展时，只需要增加新的微服务，而不需要对原有的微服务进行修改。（3）可维护性。代码和文档需要保持清晰，易于理解和修改，特别是对于接口文档，要使用 Swagger 等工具进行规范化管理，以便于后续的维护和拓展。另外，要提供清晰的日志记录、监控报警机制，使用一些监控工具，如 Prometheus、Grafana 等，对系统的运行状态进行监控，及时发现问题并解决。（4）安全性。用户的数据传输和存储需要进行加密，对于用户的隐私数据，例如手机号、密码等，需要进行脱敏处理，以保证用户的隐私安全。此外，对登录、注册等操作，需要进行身份验证和权限控制，防止恶意攻击。（5）可用性和用户体验。在整体外观上，系统需要提供简洁友好的用户图形界面，降低使用门槛。

## 5.2 架构设计

### 5.2.1 总体架构

对本系统的总体架构设计如图5.1所示。

后端部分采用微服务架构设计，包括视觉场景理解服务、语义解析服务、符号推理服务、迭代反馈服务、答案生成服务、用户交互服务、日志与监控服务、数据存储服务等。各微服务之间通过 API 网关进行通信，API 网关负责请求的路由、负载均衡、安全认证、限流等功能。

具体到各个微服务的主要职能如下：（1）视觉场景理解服务负责对用户上传的图片进行目标检测，并使用预定义的谓词，使用 ASP 程序来表示场景中的物体的相关信息；（2）语义解析服务负责将用户的自然语言问题转化为 ASP 规则，以便于后续的推理；（3）符号推理服务负责对 ASP 规则进行推理，得到问题的答案；（4）迭代反馈服务负责对 ASP 规则进行修正；（5）答案生成服务负责将符号推理服务输出的以 ASP 谓词表示的答案转化成自然语言；（6）用户交互服务负责提供用户接口，接收用户的输入，调用其他服务，返回结果；（7）日志与监控服务负责记录系统的运行日志，监控系统的运行状态，及时发现问题并解决；（8）数据存储服务负责存储系统的数据，包括用户的信息、对话记录、ASP 规则等。

前端部分，对 UI 界面的开发，选择 Element UI 框架，其具有丰富多样的按钮、输入框等搭建网页所需的元素，开发快捷方便。同时，使用 Axios 向后端发送请求，实现前后端的数据交互。使用 Vue 构建组件化的前端页面，方便代码的维护和拓展。Vue Router 用于实现前端路由，实现页面的跳转。Vuex 用于实现状态管理，方便组件之间的数据传递。

在中间件上，系统引入了 Sentinel 进行流量控制，对系统的流量进行监控，实现限流、熔断等功能。RocketMQ 用于实现消息队列，实现微服务之间的异步通信。Redisson 用于实现分布式锁，保证系统的并发安全。

### 5.2.2 接口设计

接口设计总体按照服务划分，每个服务提供一个接口供其它服务调用。接口之间的数据传输采用 JSON 格式，采用 RESTful API 设计风格。具体的接口设计方案见表**??**。

### 5.2.3 安全设计

安全设计主要聚焦数据存储加密和传输安全，以确保系统的安全性和数据的机密性，防止数据泄露和未授权访问。

在存储加密上，针对数据库中存储了用户密码以及身份证号、手机号等高度敏感信息，根据字段的分类，采取不同措施，予以保护。对于用户密码，采用 SHA-256 算法对明文密码进行加密后存储。对于除密码外的其它高度敏感字段（如身份证号、银行卡号），使用 AES 加密函数，搭配密钥管理。

在传输安全上，采用 HTTPS 加密传输的方式，以防止中间人攻击。同时，使用 JWT 进行身份验证。为了防止数据在传输过程中被篡改，使用 HMAC-SHA256 进行数据完整性校验。

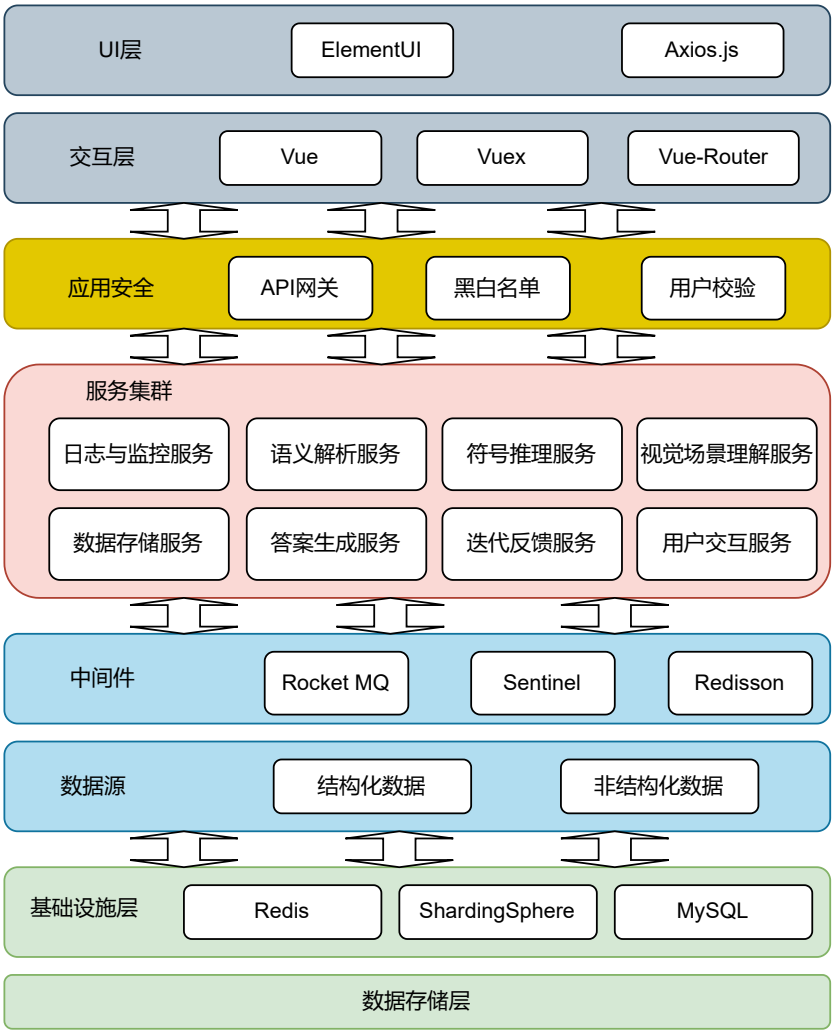

UI层
ElementUI
Axios.js
交互层
Vue
Vuex
Vue-Router
应用安全
API网关
黑白名单
用户校验
服务集群
日志与监控服务
语义解析服务
符号推理服务
视觉场景理解服务
数据存储服务
答案生成服务
迭代反馈服务
用户交互服务
中间件
Rocket MQ
Sentinel
Redisson
数据源
结构化数据
非结构化数据
基础设施层
Redis
ShardingSphere
MySQL
数据存储层

图 5.1: 系统总体架构

### 5.2.4 数据库设计

设置 User、Rule、Log、Dialog 以上四个数据库。下面，对这四个数据库的职责及内设表进行介绍。

User 数据库主要用于存储用户、角色、权限、会话信息等。内部包含如下表：（1）用户表 users，用于存储用户信息；（2）用户组表 roles，用于存储用户组信息；（3）用户组关系表 user_roles 用于存储用户与用户组之间关系；（4）用户状态表 user_session，用于存储会话或者登录状态记录。

Rule 数据库专门用于存储 ASP 规则及其相关信息。内部包含 rules 表和 rule_conditions 表。rules 表为主规则表，rule_conditions 表为规则条件表。具体表字段见表5.1。

| 字段名称 | 数据类型 | 说明 |
|---|---|---|
| rule_id | INT AUTO_INCREMENT PRIMARY KEY | 规则唯一标识 |
| rule_name | VARCHAR(255) | 规则名称 |
| rule_head | VARCHAR(255) | 规则若论部分 |
| priority | INT | 优先级 |
| status | ENUM('active', 'inactive') | 状态标识 |
| description | TEXT | 描述或备注 |
| created_at | DATETIME DEFAULT CURRENT_TIMESTAMP | 创建时间 |
| updated_at | DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP | 更新时间 |

表 5.1: rule 表字段设置及说明

| 字段名称 | 数据类型 | 说明 |
|---|---|---|
| condition_id | INT AUTO_INCREMENT PRIMARY KEY | 条件唯一标识 |
| rule_id | INT | 外键，关联到主规则表 |
| condition_text | VARCHAR(255) | 单个条件表达式，例如"X > 10" |
| sequence | INT | 条件顺序号，用于保持条件排列 |
| created_at | DATETIME DEFAULT CURRENT_TIMESTAMP | 创建时间 |
| updated_at | DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP | 更新时间 |

表 5.2: rule_condition 表字段设置及说明

Log 数据库存储系统运行日志、错误日志、操作记录等信息。其内部包含如下表：（1）系统日志表 system_logs，用于记录系统一般日志；（2）错误日志表 error_logs，用于记录错误信息；（3）审计日志表 audit_logs，用于记录关键操作审计日志。

Dialog 数据库存储所有用户与系统的对话记录。其内部包含如下表：（1）对话主表 dialogs，用于记录每个对话的基本信息；（2）对话消息表 dialog_messages，用于存储对话中用户发出的消息与 LLM 给出的每条回复。

## 5.3　缓存设计

为系统使用缓存的目标包含以下两点：（1）提高响应速度。在整个 VQA 系统中，需要 LLM 参与的语义解析、迭代反馈以及视觉场景理解环节，所需时间较长，目前较难以大幅度优化。为尽可能减少模型调用次数，以提高响应速度，改善用户体验故为系统引入缓存；（2）降低后端负载。如果访问量超出系统的承载能力，可能会导致部分服务宕机。以 Redis 为代表的缓存，单机 QPS 最高可达 10 万，远超数据库等请求链路上的其它环节的承载能力。引入缓存，可一定程度上减少对数据库和 LLM 的直接访问次数，缓解 VQA 系统在高并发场景下的压力。

为了更好地做到成本与性能之间的平衡，本文采用了多层次缓存架构，共分两层。

第一层是本地内存缓存，用于缓存短生命周期、访问频率非常高的数据，如最近对话状态、会话上下文的部分数据。在实现方式上，使用 Java 的 ConcurrentHashMap 配合 LRU 算法对缓存进行管理。

第二层是分布式缓存，本系统选用 Redis，用于缓存跨多个服务器共享的数据，如会话历史、用户信息、通用问答的缓存结果等。分布式缓存可以保证在集群环境下数据的一致性和高可用性，并支持数据持久化和数据过期策略。至于选用 Redis 作为分布式缓存的理由，是因为其支持丰富的数据结构、持久化存储和分布式部署，同时具有高性能。

## 5.4　功能展示

开发的前端页面如图5.2所示，采用了模仿 ChatGPT 的网页样式。用户可在其中自行选择模型，包括本文中微调后的模型、OpenAI 的 GPT 系列模型、Google 的 Gemini 系列模型、Meta 的 Llama 系列模型以及阿里巴巴的 QWen 系列模型，默认选用本文的微调后模型。用户可以在下方的菜单栏中，点击第二个按钮，以上传图片。同时，历史对话也会在左侧栏中予以显示。在输入框旁，也添加了语音输入按钮，点击后系统将会调用百度的语音识别 API，为用户提供语音输入服务。

上传图片并提出问题、系统给予答复后的效果如图5.3所示。用户发送的文本和图片将会在同一个消息框内显示。用户要求系统重新生成回复，或者引用系统的某条回复，以让系统能够更加贴合用户的语境。

## 5.5　集成测试

### 5.5.1　测试环境

系统的部署运行环境的硬件配置如下：（1）CPU 为 Intel Core i9-13900K；（2）内存为 128GB；（3）显卡为 3 张 NVIDIA GeForce RTX 3090 并联，显存均为 24GB；（4）操作系统为 Ubuntu 20.04 LTS；开发工具为 IntelliJ IDEA、IntelliJ Pycharm 和 Microsoft Visual Studio Code。

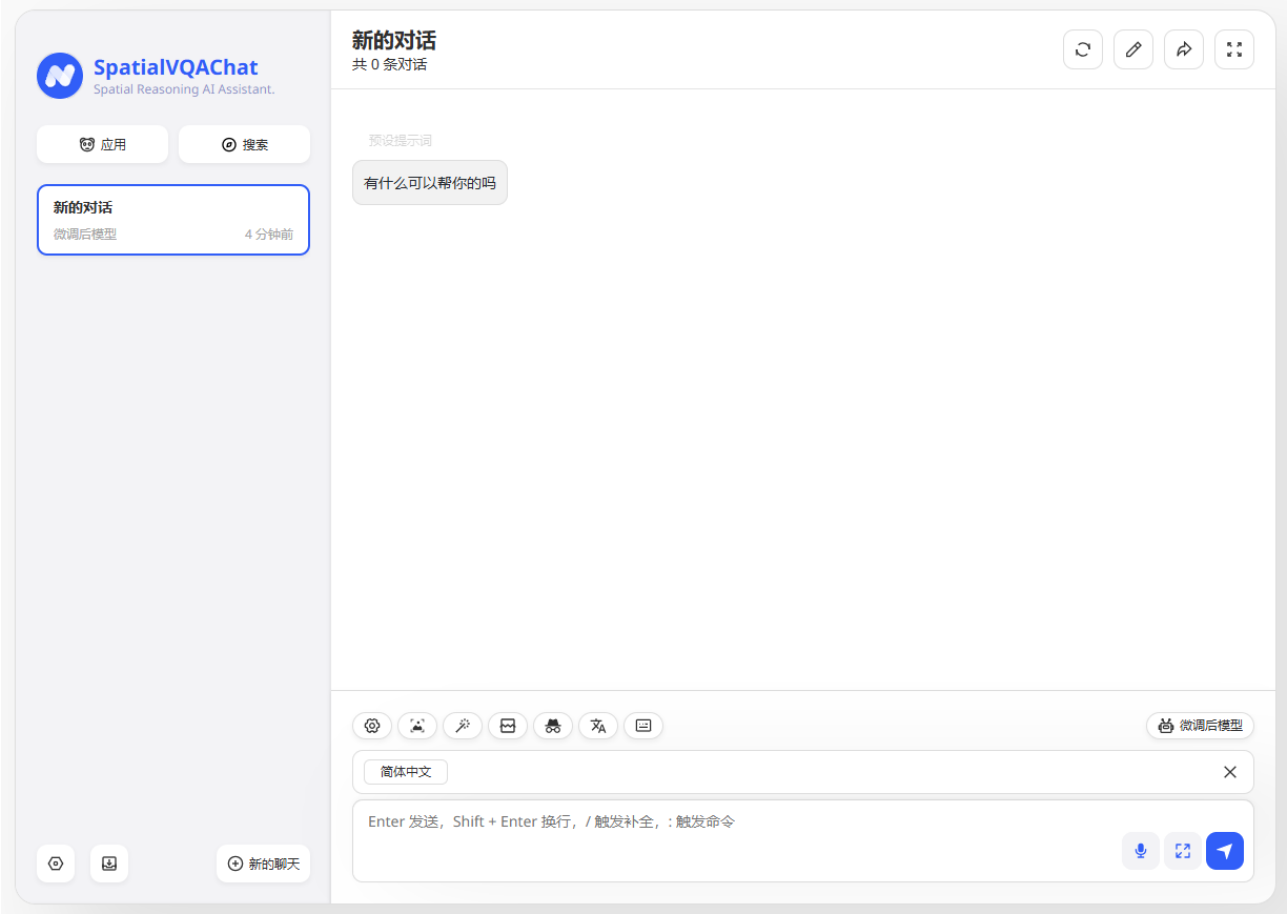

SpatialVQAChat
Spatial Reasoning AI Assistant.
应用
搜索
新的对话
微调后模型
4 分钟前
新的聊天
新的对话
共 0 条对话
预设提示词
有什么可以帮你的吗
微调后模型
简体中文
Enter 发送，Shift + Enter 换行，/ 触发补全，: 触发命令

图 5.2: 新对话页面

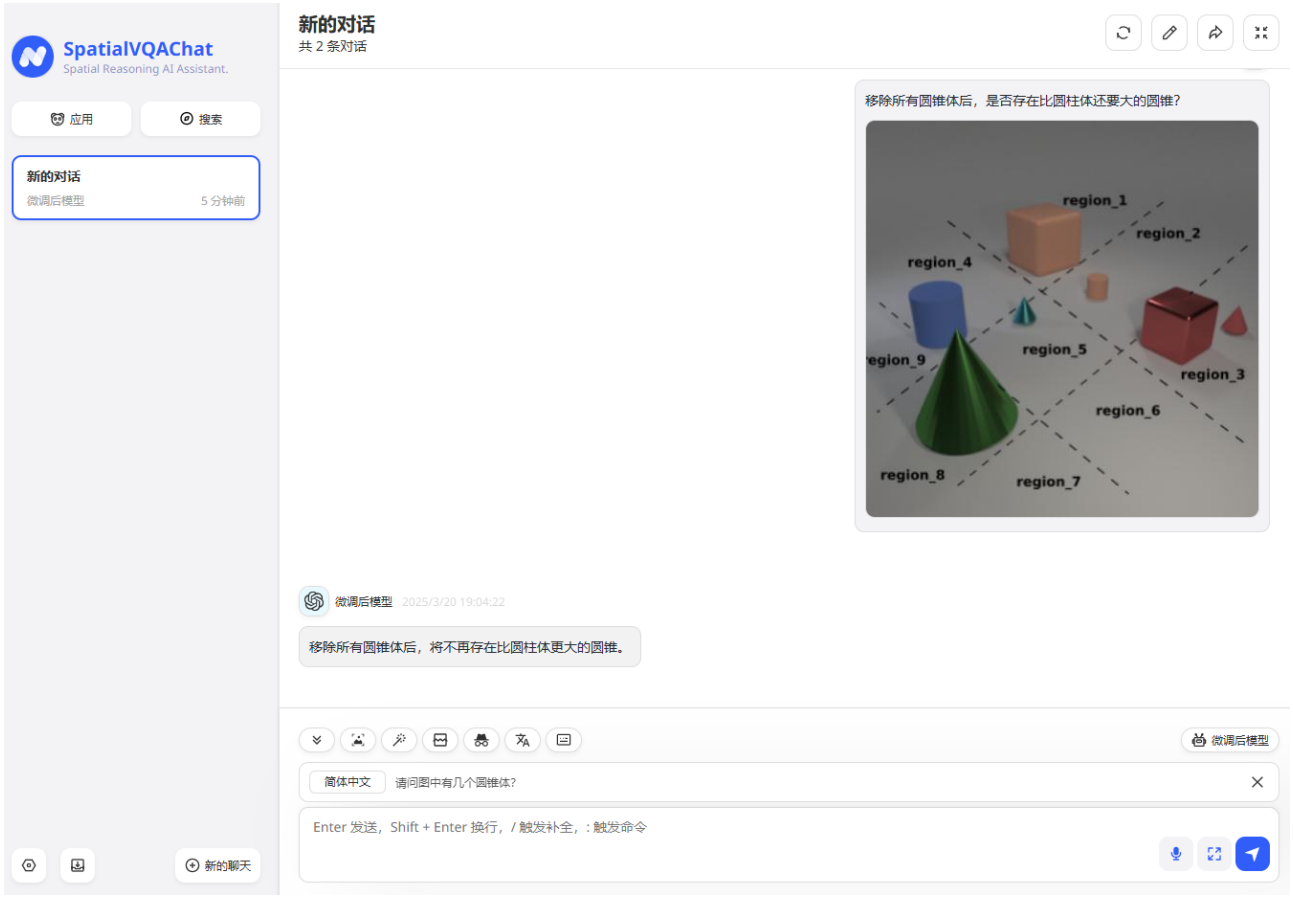



图 5.3: 上传图片并提问后的效果

### 5.5.2　功能测试

### 5.5.3　非功能测试

非功能性需求中，需要通过测试以量化评估的主要为性能需求。因此，本文通过压力测试，对系统的性能进行评估。本文使用 JMeter 工具进行压力测试。JMeter 是一种开源的压力测试工具，它可以用于对 Web 应用程序、FTP 服务器、数据库等进行性能测试。JMeter 可以模拟多个用户同时访问服务器，以此来测试服务器的性能。此外，使用 SkyWalking 对整个应用进行性能监控。图5.4为 SkyWalking 对本系统中语义解析服务的监控面板，在其中可以直观监测服务的平均响应时间、成功率、每分钟调用数等信息。

本文使用第三章中构造的数据集作为测试数据，测试的接口为用户交互接口。在开发各个接口时，已经在各个接口上设置了监控埋点。在 SkyWalking 中，能够看到各个接口的调用次数，以及接口的响应时间等信息。即便是只对某一个接口发起模拟请求，进行压力测试，也可以通过 SkyWalking 来查看其它所有相关接口的相关参数。用户交互接口处于整个调用链路的起点，对其进行压测，能够观测到整个 VQA 系统在执行一次任务时，所有相关接口的情况，故对其进行压测。

首先，确定压测目标。本系统作为一个原型系统，拟定的用户群体规模上限在 20 人左右，则可以假设并发用户数目标为 15 人。通过对 ChatGPT、DeepSeek、豆包等现有的聊天机器人的使用的观察，发现目前普遍限制同一用户在同一时刻只能进行一个对话，若在原有对话仍

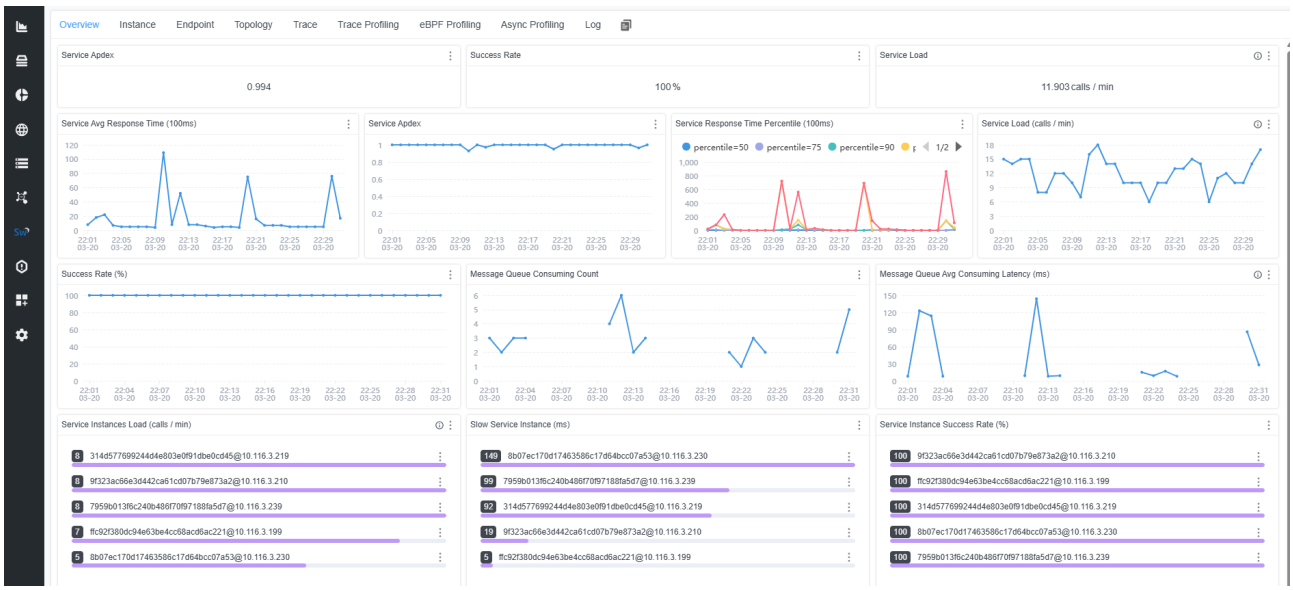



图 5.4: 上传图片并提问后的效果

在生成答案的同时，切换到新对话，将会导致原有对话的生成中断。所以，本原型系统在压测目标上，设置每个用户的线程数都为 1。对以上各个接口，压测时长设置为 30 分钟。各用户线程加载，选用阶梯式加载的方式，随着时间推移，逐步增加并发用户数，以便观察系统在压力逐步增大时的性能变化。

其次，执行压测。在压测过程中，实时监控接口的性能指标，包括响应时间、吞吐量和错误率，同时监控系统的 CPU 使用率、内存使用率。

最终测试结果见表??。在 15 个并发用户下，系统响应整体较快，能满足大多数用户对视觉问答的需要。TPS 表现整体稳定，错误率也比较低。总体而言，压测结果表明，该系统能够满足其作为原型系统的要求，符合设计预期。

| 参数 | 指标 | 参数 | 指标 |
|---|---|---|---|
| 平均响应时间 | 320ms | 90% 响应时间 | 300ms |
| **参数** | **指标** | **参数** | **指标** |
| TPS | 12 | 错误率 | 0.5% |
| **参数** | **指标** | **参数** | **指标** |
| CPU 使用率 | 60% | 内存使用率 | 55% |

## 5.6 本章小结

# 第六章　结论与展望

## 6.1　本文工作总结

本文的主要工作如下：

1. 构建旨在考察模型复杂空间推理能力的数据集。
2. 设计了一种神经符号框架。
3. 设计实现了一个视觉问答系统。

## 6.2　未来工作展望

本文设计的面向空间推理的神经符号框架在视觉问答系统中，对增强其空间推理能力起到了积极作用，但仍有许多方面有待完善。未来的工作可以从以下几个方面展开：

# 参考文献


[1] Goyal Y, Khot T, Summers-Stay D, et al. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering[C]. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 6904-6913.

[2] Antol A, Agrawal A, Lu J, et al. VQA: Visual Question Answering[J]. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015: 2425-2433.

[3] Ren M, Kiros R, Zemel R S. Exploring Models and Data for Image Question Answering[C]. in: Proceedings of the IEEE International Conference on Computer Vision. 2015: 2953-2961.

[4] Malinowski M, Fritz M. Neural-Image-Question Embedding for Visual Question Answering[C]. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 2015-2023.

[5] Lu Y, Wang J, Gao J, et al. Look, Read and Answer: Towards Universal Visual Question Answering Models[C]. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 10577-10586.

[6] Xu K, Ba J, Kiros R, et al. Stacked Attention Networks for Image Question Answering[C]. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2011-2019.

[7] Tan H, Bansal M. LXMERT: Learning Cross-Modality Encoder Representations from Transformers[C]. in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2019: 5100-5111.

[8] Radford A, Kim J W, Hallacy C, et al. Learning Transferable Visual Models From Natural Language Supervision[J]. ArXiv preprint arXiv:2103.00020, 2021.

[9] Li J, Li D, Xiong C, et al. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation[J]. ArXiv preprint arXiv:2201.12086, 2022.

[10] Cohn A, et al. Evaluation of GPT-4 on Qualitative Spatial Reasoning Tasks[J]. Journal of Artificial Intelligence Research, 2023, 70: 1-20.

[11] Bang Y, Cahyawijaya S, Lee N, et al. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity[C]. in: Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). 2023: 675-718.

[12] Ishay A, Yang Z, Lee J. Leveraging Large Language Models to Generate Answer Set Programs[J]. ArXiv preprint arXiv:2307.07699, 2023. arXiv: 2307.07699 [cs.AI].

[13] Wałęga P A, Bhatt M, Schultz C. ASPMT(QS): Non-Monotonic Spatial Reasoning With Answer Set Programming Modulo Theories[Z]. 2015. arXiv: 1506.04929 [cs.AI].

[14] Baryannis G, Tachmazidis I, Batsakis S, et al. A Trajectory Calculus for Qualitative Spatial Reasoning Using Answer Set Programming[J]., 2018. arXiv: 1804.07088 [cs.AI].

[15] Gokhale A, Banerjee S, Baral C. Mutual Information Maximization for Robust Multimodal Representations[C]. in: Proceedings of the Neural Information Processing Systems (NeurIPS). 2020.

[16] Eiter T, Higuera N, Oetsch J, et al. A Neuro-Symbolic ASP Pipeline for Visual Question Answering[J]. Theory and Practice of Logic Programming, 2022, 22(5): 739-754.

[17] Pan L, Albalak A, Wang X, et al. LOGIC-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning[C]. in: Findings of the Association for Computational Linguistics: EMNLP 2023. 2023: 4000-4012.

[18] 李智涛, 周之平, 叶琴. 基于空间注意力推理机制的视觉问答算法研究[J]. 计算机应用研究, 2021, 38(3): 952-955.

[19] Shrestha R, Kafle K, Kanan C. Answer Them All! Toward Universal Visual Question Answering Models[C]. in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 6559-6568.

[20] Costanzino A, Ramirez P Z, Lisanti G, et al. Multimodal Industrial Anomaly Detection by Crossmodal Feature Mapping[C]. in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024: 12345-12354.

[21] Wu W, Mao S, Zhang Y, et al. Mind's Eye of LLMs: Visualization-of-Thought Elicits Spatial Reasoning in Large Language Models[C]. in: Adv/lances in Neural Information Processing Systems (NeurIPS). 2024.

[22] Yang P, Wang J, Gan R, et al. Zero-Shot Learners for Natural Language Understanding via a Unified Multiple Choice Perspective[C]. in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022: 7010-7022.

[23] Li Z, Zhou Z, Ye Q. Imagine while Reasoning in Space: Multimodal Visualization-of-Thought[J]. ArXiv preprint arXiv:2501.07542, 2025.

[24] Shah M, Chen X, Rohrbach M, et al. Cycle-Consistency for Robust Visual Question Answering[C]. in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 6647-6656.

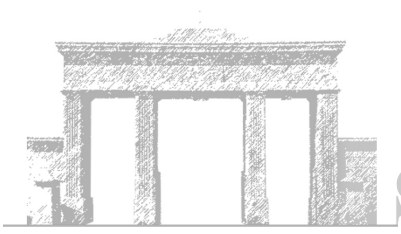
[25] Van Gelder A, Ross K A, Schlipf J S. The Well-Founded Semantics for General Logic Programs[J]. Journal of the ACM, 1991, 38(3): 619-649.

[26] Gao L, Madaan A, Zhou S, et al. PAL: Program-Aided Language Models[C]. in: Proceedings of the 40th International Conference on Machine Learning (ICML): vol. 202. 2023: 10764-10799.

[27] He-Yueya J, Poesia G, Wang R E, et al. Solving Math Word Problems by Combining Language Models with Symbolic Solvers[J]. ArXiv preprint arXiv:2304.09102, 2023.

[28] Feng J, Xu R, Hao J, et al. Language Models Can Be Deductive Solvers[C]. in: Findings of the Association for Computational Linguistics: NAACL 2024. 2024: 4026-4042.

心求至善



SOUTHEAST UNIVERSITY