计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(4): 827−853, 2026

基于大语言模型的需求歧义检测方法

高俊涛[1] 刘 芳[1] 杨溢龙[2]
[1](东北石油大学计算机与信息技术学院 黑龙江大庆 163318)
[2](北京航空航天大学软件学院 北京 100080)
(gjt@nepu.edu.cn)

Requirement Ambiguity Detection Method Based on Large Language Model

Gao Juntao[1], Liu Fang[1], and Yang Yilong[2]
[1](*School of Computer and Information Technology*, *Northeast Petroleum University*, *Daqing*, *Heilongjiang* 163318)
[2](*School of Software*, *Beihang University*, *Beijing* 100080)

Abstract In modern software development, requirements ambiguity is a key factor that leads to project failure, cost overruns, and quality issues. Therefore, the automatic detection of requirement ambiguity has been extensively studied. Although these methods reduce the time cost of manual review, their solutions are incomplete for the types of ambiguity detection, and it is difficult to fully cover the six types of ambiguity corresponding to language ambiguity. The traditional method lacks the ability of in-depth semantic understanding, logical relationship recognition and quotation/referential relationship recognition, which limits its semantic processing ability and cannot detect pragmatic ambiguity and linguistic error ambiguity. To this end, we propose a large language model-based method called LMAdetect for automatic detection of requirements ambiguity. For a given requirement, LMAdetect analyzes the requirement according to the heuristic rules and the large language model, and classifies the requirement detection into the corresponding ambiguity type. However, large language models often classify different types of ambiguity, and for the classification results of large language models, LMAdetect uses a rule-based algorithm to assign different types of ambiguity to different experts for detection. Finally, the detection results are summarized according to the confidence level, and the classification results with high confidence are output. Experiments on 1 192 data of 6 ambiguity types show that compared with the traditional rule-based method, the $F1$ score of LMAdetect is increased by 34.4%, and compared with the method based on large language model, the $F1$ score is increased by 31.6%, and the $F1$ score of pragmatic ambiguity and linguistic error ambiguity detection reaches 0.695 0 and 0.888 9, respectively, demonstrating its advantages in requirement ambiguity detection.

Key words requirement ambiguity; detection; large language model; classification; semantic understanding

摘 要 在现代软件开发中,需求歧义是导致项目失败、成本超支和质量问题的关键因素,因此,人们对需求歧义的自动化检测进行了广泛的研究。虽然这些方法降低了人工审查的时间成本,但歧义检测解决方案类型不全,难以全面覆盖语言歧义对应的6种歧义类型。传统方法缺乏深层次的语义理解、逻辑关系识别以及引用/指代关系识别能力,这限制了其对语义的处理能力,无法对语用歧义以及语言错误歧义进行检测。为此,提出了一种基于大语言模型的方法(称为LMAdetect),用于自动检测需求歧义。对于给

收稿日期: 2025-09-21; **修回日期**: 2025-12-29
基金项目: 国家自然科学基金项目(62302029, 62577005); 黑龙江省自然科学基金项目(LH2024F005)
This work was supported by the National Natural Science Foundation of China (62302029, 62577005) and the Natural Science Foundation of Heilongjiang Province (LH2024F005).
通信作者: 杨溢龙(yilongyang@buaa.edu.cn)

定的需求, LMAdetect 根据启发式规则和大语言模型进行分析,将该需求检测分类为对应的歧义类型。然而,大语言模型经常会分类出不同的歧义类型,针对大语言模型的分类结果,LMAdetect 利用基于规则的算法,将不同歧义类型分配给不同专家进行检测。最后对这些检测结果根据置信度进行汇总并输出置信度高的分类结果。对 6 种歧义类型共 1 192 个数据进行了实验,结果表明,相较传统基于规则的方法,LMAdetect 的 $F1$ 分数提升了 34.4 个百分点,相较仅使用基于大语言模型的方法,$F1$ 分数提升了 31.6 个百分点,对于语用歧义和语言错误歧义的检测,$F1$ 分数分别达到了 0.695 0 和 0.888 9,展示了其在需求歧义检测方面的优势。

关键词 需求歧义;检测;大语言模型;分类;语义理解

中图法分类号 TP311.5

DOI: 10.7544/issn1000-1239.202550700 **CSTR:** 32373.14.issn1000-1239.202550700

需求歧义是需求工程中的关键问题,包括语言歧义和软件工程歧义[1]。本文聚焦于语言歧义,具体研究其在自然语言需求中的检测方法,因为语言歧义是需求文档质量下降的主要原因之一,且适合通过自然语言处理(natural language processing, NLP)技术进行自动化分析。

语言歧义是 Kamsties 等人[2]于 2000 年首次系统化定义的概念,它指的是自然语言需求中因语言特性导致的多种可能性解释。如果不及时解决这些问题,就会影响需求文档的清晰性、一致性和可验证性[3-4]。需求歧义源于自然语言的多义性,可能引发开发人员误解[5]、需求冲突或返工。例如,在餐厅优惠纠纷案例中,需求语句“For every Zhongcan consumption reaches RMB 100, receive RMB 30”因“中餐”缩写歧义(可指午餐或中式餐点),导致消费者与商家对晚餐是否适用优惠产生分歧,引发消费纠纷并最终由商家退还 120 元。另一个案例是文件柜收据标点纠纷中,收据语句“27 iron cabinets have been received from the supplier today, the amount of uncollected information cabinet is outstanding”因缺少逗号导致歧义(可解释为全部款项未结或仅未收柜款未结),导致买卖双方对支付义务理解不同,引发诉讼并由法院判决被告支付 12 960 元[6]。这些真实案例凸显了需求歧义的严重影响。一方面,多种歧义类型导致需求表达不清楚,难以被利益相关人员一致理解或验证[4];另一方面,清晰的需求表达可以更好地支持软件的设计和开发。目前已确认的语言歧义类型有词汇歧义、句法歧义、语义歧义、语法歧义、语用歧义和语言错误歧义。值得注意的是,语言歧义不仅是需求质量[7]下降的指标,也是需求澄清和改进的机会指标[8]。需求歧义的检测方法由 Gause 等人[9]提出,是通过分析需求的多方解读来识别潜在的歧义。

鉴于需求歧义可能引发上述严重后果,尽早识别并检测其已成为需求工程的核心任务,这种识别过程被称为需求歧义检测[8-9]。值得注意的是,需求歧义的检测并非易事,而是需要深入理解需求的具体内容,对复杂的语言现象进行分析,还需准确识别需求中的表达不清的语言模式,并确认哪些需求语句可能引发多种解释。为了促进需求歧义的有效检测,研究人员提出了多种解决方案[10-11]。这些解决方案主要分为基于检查表的方法[2,12]和基于 NLP 的方法[10,13]两大类。前者通过预定义的语言模式检查需求文档,并根据检查表结果识别潜在歧义语句。后者通过从需求文本中提取信息以生成语言结构,并根据生成的结构推理出歧义类型。虽然这些方法促进了歧义检测工作,但它们的解决方案难以覆盖复杂的语言歧义情况。语言歧义检测的主要目的是通过消除模糊表达来提高需求文档的清晰性和一致性[10]。目前,许多需求工程工具(如基于 NLP 的分析工具)为检测和消除语言歧义提供了自动或半自动支持。

需求工程工具的提出为语言歧义检测提供了支持,然而现有的语言歧义检测方法仍然存在局限性。我们的研究面临以下挑战。由于语言歧义的多样性与复杂性,且语义歧义和语用歧义高度依赖上下文和领域知识,故单一方法难以全面覆盖词汇、句法、语义、语用等不同类型的歧义。传统的基于规则或词性标注的方法因缺乏深层语义理解能力,难以准确捕捉复杂的语义依赖关系或跨句逻辑关联。现有的语言歧义检测方法因依赖固定规则或局部特征,导致泛化能力不足,并且缺乏动态语义分析和大规模上下文推理能力,故难以适应多样化的语义场景。此外,现有方法在处理歧义分类时使用的精确性和召回率指标存在权衡,难以在检测全面性和准确性

之间取得平衡，特别是在大规模、真实数据集上进行实验的结果表现不佳。近年来，大语言模型渐渐成为各种文本任务，如语言翻译[14-15]、文本分类[16]和文本摘要[17]等任务的有力工具[18]，被用于解决一些需求工程任务，如需求生成[2]、需求验证[10]和自动需求分析[9,19]等。尽管目前未有人尝试使用大语言模型对需求歧义进行检测，但大语言模型擅长自然语言理解[12]的特点，对于检测需求歧义有一定的优势，可见大语言模型具有检测需求文档中语言歧义的潜力。然而自然语言的复杂性使得模型在理解需求文本时可能存在误解，尤其对于包含上下文依赖关系的需求文本歧义检测任务。因此，在需求歧义检测任务中，确保大语言模型能够准确把握语言的含义和上下文是一项重要挑战。

本研究提出了一种结合大语言模型与多智能体协同框架 LMAdetect，通过需求文档的语言特征和上下文数据，准确识别需求文档中的语言歧义。此框架用于自动检测需求文档中的语言歧义，重点针对需求文档中常见的词汇歧义、句法歧义、语法歧义、语义/范围歧义、语用歧义、语言错误歧义进行检测。本文从多篇论文中收集了 1 192 条需求数据，通过数据清洗工作构建了包含多种语言特征需求的数据集。此数据集覆盖了词汇、句法和语义等多个维度。本文所提出的 LMAdetect 框架结合了静态语言特征和动态上下文特征，并将每个语言特征进一步细分为多个子维度，从而识别需求文档中的语言歧义。本文在现有方法的基础上扩充了检测规则，新增了语义/范围、语用和语言错误歧义的条件，通过结构化提示工程增强泛化能力，并通过优先级排序机制进行优化分类。与现有方法的检测不同，本文方法扩充了知识领导大模型进行检测，提供了知识驱动的语义分析，从表层匹配到深度语义结合上下文整合，通过依存句法驱动的结构分析，从语法关系层面验证歧义是否真实存在，而非仅依赖表层词序，从而达到语义检测的深度提升。从孤立词检测到场景化上下文过滤，通过“歧义术语结合量化修饰词”的上下文过滤逻辑，可自动排除这类已量化的歧义词。本研究利用大语言模型构建智能检测机制的方法，不仅揭示了清晰需求与歧义需求之间的差异，而且实现了基于这些特定语言特征的语言歧义自动检测策略。

本文的主要贡献可归纳为 3 点：

1）提出了一种基于大语言模型与规则结合的多智能体协同的检测方法，通过大语言模型提取需求文档的词汇、句法和语义特征，构建多维度语言表示，相比传统工具，该方法能够有效识别复杂的句法和语义歧义。

2）设计了一种基于优先级排序机制的歧义分类机制，能够根据需求语句的上下文特征精准区分词汇歧义、句法歧义、语法歧义等多种歧义类型，并通过注意力权重排序潜在歧义语句的优先级。该机制显著提升了歧义检测的效率和准确性。

3）基于上述技术实现了面向需求工程的智能歧义检测框架 LMAdetect。利用包含 1 192 条需求文档的真实数据集，对 LMAdetect 框架进行了全面评估，并与多种现有方法（如 Paska、POS 标记）进行了对比。实验结果表明，LMAdetect 在已有的需求数据中成功检测出传统方法检测不到的歧义，且在可检测到的歧义类型中的召回率平均提升 35.6%，$F1$ 分数达到 0.906。

1 需求歧义检测相关工作

目前有一系列针对需求歧义检测的相关研究工作，根据它们所采用的检测机制，主要可分为 3 类：基于检查表的方法、基于规则的半自动化方法和基于 NLP 的自动化方法。

基于检查表的方法依靠人类专业知识来识别歧义性，通常使用结构化指南以确保系统分析。Kamsties 等人[2]提出了一种基于检查表的方法来识别语言歧义，针对词汇层面、句法层面以及语义层面，通过预设问题对软件需求规格说明（software requirements specification, SRS）文档进行审查。这种方法适用于小规模项目或初步审查，虽系统化但繁琐耗时，依赖审阅者的专业知识，且难以扩展至大规模文档。Kamsties 等人[20]提出了一种通过检查发现需求文档中歧义的方法，将基于检查表的审查与场景化阅读相结合。检查表方法采用预定义的歧义标识符，供评审人员系统性地扫描文档。场景化阅读则通过构建用例或测试场景，模拟利益相关者的解读来揭示歧义。虽然这种方法能有效调动领域专家参与并实现特定类型歧义的高检出率，但其劳动强度大、需要深厚领域知识且难以扩展至大规模 SRS。此外，由于缺乏自动化处理，该方法容易出现人为错误，并在检测细微语义或语用歧义时存在不一致性。相比之下，LMAdetect 通过使用语言模型自动检测，不仅减少了人工投入，还能覆盖更广泛的歧义类型，包括语用和语言错误类歧义。

基于规则的半自动化方法使用预定义的语言规则或词汇表来检测歧义，减少人工工作量，同时保持

结构化分析。例如 QuARS[21] 这类需求规范质量分析器，通过预设规则识别模糊表达模式，这些工具会标记潜在问题供人工审核，可以减轻人工的工作量，适用于初步筛查，QuARS 以专门定义的质量模型为基础，该模型包含多个质量指标，通过这些指标，QuARS 对自然语言需求文档进行分析，自动标记出存在上述潜在缺陷的句子，为需求工程师提供预警信息，帮助他们识别和修正可能导致歧义的问题。Berry 等人[13] 对自然语言系统需求规格说明书的歧义问题进行了基础性研究，提出了将非正式需求形式化、寻找歧义模式(如含糊表述“only”)、比对利益相关方的解读以及与作者建立反馈机制等技术手段。他们的研究强调人工验证，并使用 NASA 的 ARM 等工具进行模式匹配，但仅限于词汇和句法层面的歧义分析，缺乏语义或语用层面的解析。类似地，Nigam 等人[22] 开发了一种基于简单规则模式匹配的工具，用于检测词汇、句法和语法歧义，工具旨在辅助需求分析人员，在审查规格时突出歧义。尽管计算效率较高，但由于精度不足，该工具在处理复杂句子时存在局限性，仍需人工验证。Tjong[23] 通过分析工业需求语料库提炼指导规则，开发 SREE 工具，通过词素分析和预设规则识别歧义指标，这个实验性工具用于辅助需求分析师检测自然语言需求规范中潜在的歧义实例。当 SREE 发现潜在歧义时，会向用户报告该实例，由用户判断是否确实存在歧义，并可根据需要进行消歧处理。SREE 本质上是一个词法分析器，仅针对其数据库中特定词汇进行实例检索。Rosadini 等人[24] 在铁路领域应用基于规则的自然语言处理模式，成功检测出被动语态和模糊表述等缺陷，实现了超过 80% 的高召回率，但由于误报率过高仍需专家人工审核。这类规则方法虽能有效处理特定类型的歧义，却难以应对语义与语用层面的复杂情况。Gleich 等人[25] 提出了一种自动检测软件需求文档中歧义的工具，旨在不仅检测歧义，还解释歧义来源，以辅助需求分析师提升文档质量。通过分析不同术语的出现频率，从需求文档中提取特定应用领域的术语，其核心是基于预定义的 38 种模式，利用 TreeTagger 和正则表达式高效匹配，通过词法(关键词匹配)和句法(POS 结合正则表达式)分析，虽然无法处理复杂的语义或语用歧义，但其轻量设计和高可靠性使其在工业需求分析中具有显著优势。该方法依赖关键词列表，可以针对词汇歧义、句法歧义、语义/范围歧义进行检测，但其泛化能力有限，没法捕捉到全局的语义依赖关系，对语法、语用以及语言错误歧义无法进行检测。

Sabriye 等人[26] 提出了一种基于 NLP 的自动检测 SRS 中句法歧义和语法歧义的方法。该方法的核心是利用 POS 标注技术，其原理是基于词性标签匹配和规则检查，通过量化歧义百分比和可视化输出支持需求分析。其通过预处理、处理和后处理 3 个主要阶段，将原始文本转换为可处理的结构化数据，使用 POS 标注器为每个句子中的单词分配英语词性标签，如名词(NN)、动词(VB)、形容词(JJ)、副词(RB)等，提供句子的语法和结构信息，利用 POS 标签检测句法歧义和语法歧义，生成可视化结构直观展示检测结果。该方法只能基于局部的词性信息进行分析，无法捕捉到全局的语义依赖关系。所以，不能检测词汇歧义、语义/范围歧义、语用歧义和语言错误歧义，在面对上下文语义关系引起的歧义时，无法准确检测。Ferrari 等人[27] 旨在识别需求获取过程中不同技术领域利益相关者之间的术语歧义，该方法利用领域特定的语言模型和词嵌入技术量化术语在不同域间的使用差异，并通过歧义评分对潜在歧义进行排序。运用词向量通过跨领域语料库对比术语使用情况，对依赖上下文的术语存在局限，仅检测术语级歧义。Gentile 等人[28] 提出的 THAT-dictionary 统计方法，通过语境模式评分技术检测语义词典中的模糊或伪术语，用于识别语义资源(如字典或词典)中相对于特定语料库的歧义或冗余术语，旨在支持信息提取任务中的字典优化。Šenkýř等人[29] 将自然语言处理与领域模型结合，通过 Prolog 推理框架检测结构歧义和指代歧义，但由于过度依赖特定领域的 OCL 规则，导致系统扩展性受限。Liu 等人[30] 运用词嵌入技术检测专业领域中的词汇歧义，但该方法仅关注词汇层面的歧义，忽略了句子整体语境。相比之下，LMAdetect 通过将语言模型与启发式规则相结合，实现了对 6 种歧义类型的全面检测，包括词汇、句法、语义、语用、语法及语言错误。Veizaga 等人[31] 提出基于 Rimay 语言的 Paska 工具，利用 NLP 技术和 Rimay 受控自然语言来检测需求异味(包括与歧义相关的质量问题)。以预处理、分割需求为片段，运用 Tregex 模式[32]、结构模式、规则和词汇表搜索等多种技术，可以检测句法歧义中的协调歧义，无法对词汇歧义、语义/范围歧义、语法歧义、语用歧义、语言错误歧义进行检测。检测协调歧义主要依靠规则来实现。当需求存在 2 个或更多后续条件，且这些条件由协调连词“or”连接时，Paska 工具会触发协调歧义的检测。通过生成解析树来检测协调歧义等，

主要依赖预定义的 Rimay 语法，难以全面地考虑句子在整个语境中的复杂语义依赖关系，无法覆盖领域外歧义（如语用歧义）。

Gleich 等人[25]、Sabriye 等人[26] 和 Veizaga 等人[31] 提出的 3 种方法，在自动歧义检测领域实现了重大突破，分别运用正则表达式、词性标注和受控自然语言分析等不同技术手段，为该领域发展注入新活力。为更清晰地对比这些方法在不同类型歧义检测中的表现，表 1 系统梳理了三者的检测范围，重点分析其在词汇、句法、语义/范围、语法、语用及语言错误等维度的识别能力。

Table 1　Detection Scope of Ambiguity Type

表 1　歧义类型检测范围

工具/方法	词汇歧义	句法歧义	语义/范围歧义	语法歧义	语用歧义	语言错误歧义
正则表达式	√	√	√	×	×	×
POS 标记	×	√	×	√	×	×
Paska 工具	×	√	×	×	×	×

尽管这些研究获得了一定成效，但在歧义检测过程中，仍面临着语义语用分析缺失、逻辑关系识别不充分、适用性有限等挑战。本研究将会根据现有技术，进一步分析如何提高需求歧义检测的准确性[33-34] 以及类型的可扩展性。

2　基于大语言模型的需求歧义检测方法

图 1 是需求歧义检测的过程图，主要包括需求数据采集、数据清洗与特征筛选、歧义检测与分类、专家智能体协作、大语言模型推理、结果融合与评估输出等环节，涉及从输入需求文本到输出检测结果的完整过程。获取原始需求数据以提取关键特征，将原始数据转化为结构化特征，同时准备规则代码和大语言模型提示词，预处理触发初步检测执行，先通过门控筛选歧义需求，若检测出无歧义则输出无歧义，若检测出有歧义则触发分类执行。各歧义专家结合大语言模型协同检测具体歧义类型，整合各专家检测结果，通过证据理论计算综合置信度，解决结果冲突，基于评估指标评估结果，判断是否存在未处理需求、决策是否重检、输出无歧义结果，或歧义类型、置信度及消歧建议。本文提出了一种将大语言模型与多智能体协作及启发式规则相结合的需求歧义检测方法 LMAdetect。LMAdetect 利用大语言模型的语义理解能力和多智能体的结构化决策来识别和分类自然语言需求中的歧义。该方法主要针对词汇歧义、句法歧义、语法歧义、语义/范围歧义、语用歧义、语言错误歧义等 6 种类型进行检测。大语言模型的动态语境理解和知识为语义理解和引用/指代识别提供了基础，语用歧义专家和语言错误歧义专家分别通过上下文分析和语言处理工具进行针对性优化，增强了上下文感知能力和逻辑关系识别能力。spaCy 和 benepar 的集成突破了传统规则驱动工具的局限性，进一步支持句法和语义分析，从而能够处理在语境依赖性和语言错误场景中的歧义。图 2 展示了 LMAdetect 多智能体协作框架图，LMAdetect 中的“L”表示大语言模型（LLM），“MA”表示多智能体（multi-agent），“A”表示歧义（ambiguity）。LMAdetect

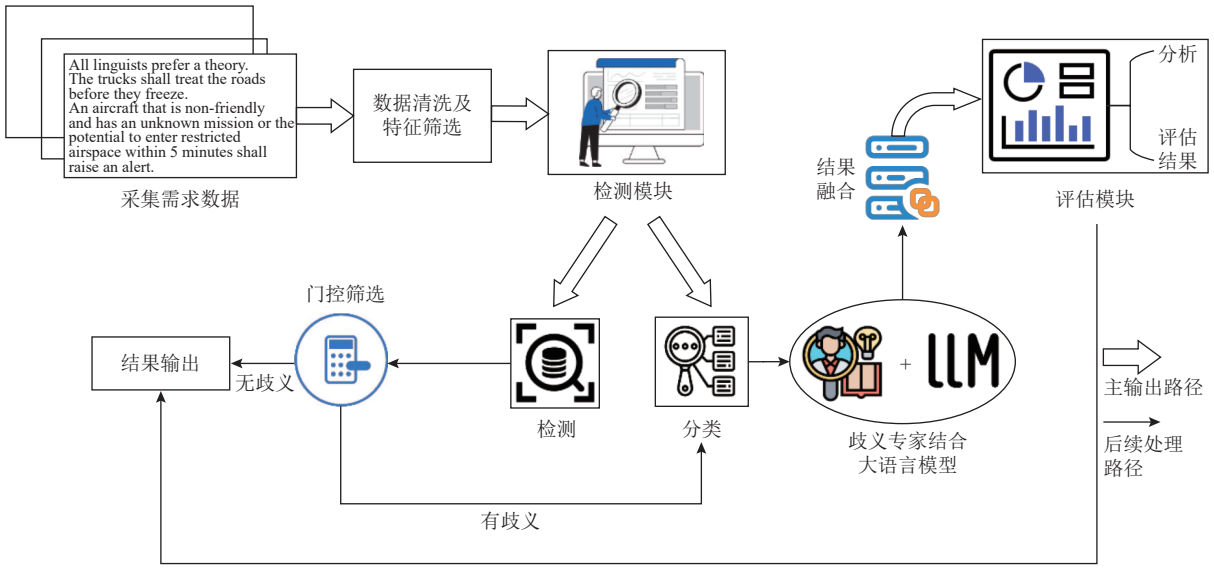

Fig. 1　Detection process diagram of requirements ambiguity

图 1　需求歧义检测过程图

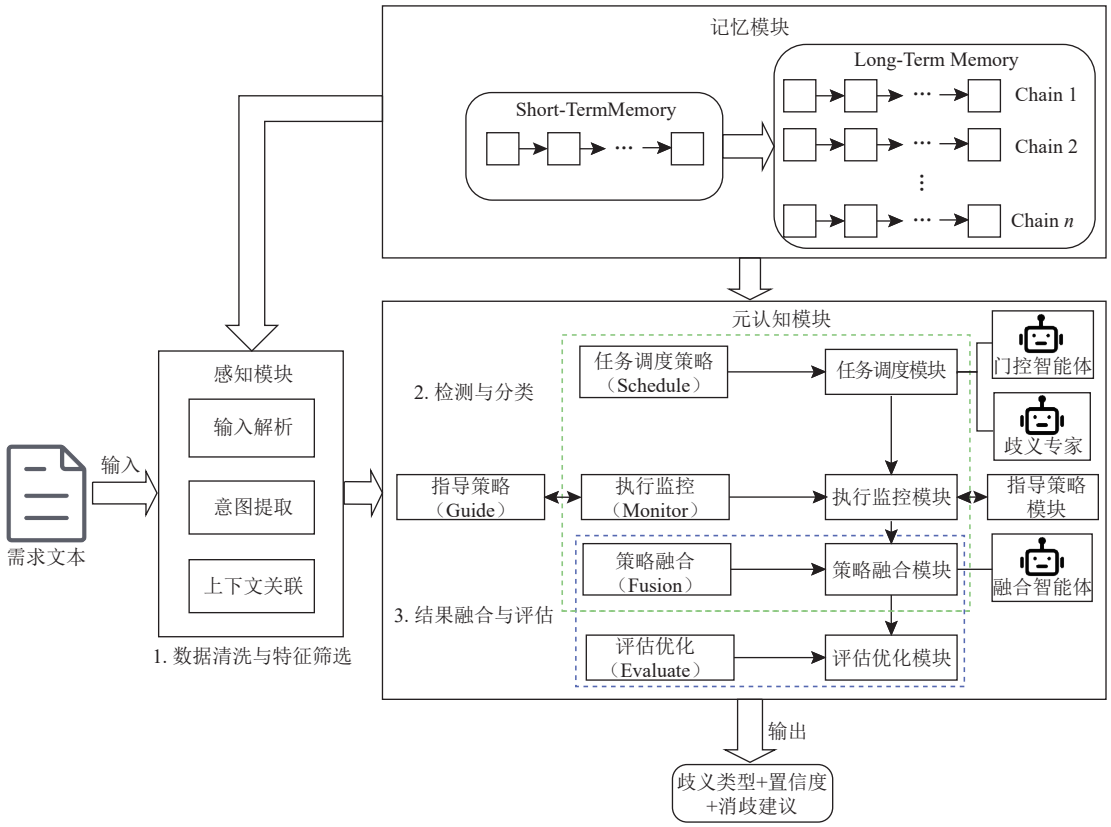

Fig. 2 Multi-agent application collaboration technology framework diagram

图 2 多智能体应用协作技术框架图

的操作分为 3 个阶段：1)基于启发式规则的初步筛选，以提取候选的歧义需求；2)利用结构化提示进行基于大语言模型的歧义分类；3)通过多智能体协作和证据融合，生成稳健的检测结果。

本文采用基于大语言模型与启发式规则结合的方法构建多智能体进行歧义检测。首先，根据需求歧义检测方法的调研结果设计了一些以 NLP 或结构化规则为基础的启发式规则，并提出了一系列可行的歧义检测方案。其次，结合调研结构，我们可以发现，歧义检测工具通常会利用词性标注识别词汇范畴，通过预定义的结构化规则构建检测逻辑，并借助正则表达式匹配文本中的歧义触发模式。而大语言模型更加擅长阅读和理解自然语言，因此它们有可能在给定的需求中根据规则识别出自然语言的多义部分。LMAdetect 会利用大语言模型来分类检测的歧义语句，根据给定的方法，LMAdetect 会按照预定义的模板初始化智能体，并通过 API 调用将生成的提示反馈给大语言模型。大语言模型会将给定的歧义语句进行具体分类，然后我们对大语言模型的结果基于规则进行合并和处理，得出最终结果。最后，针对基于启发式规则和基于大语言模型的一系列需求歧义的检测结果，LMAdetect 会根据设计的评估指标对检测结果进行评估。

LMAdetect 框架的多智能体协作通过优化后的工作流实现高效检测，如图 2 所示。与图 1 的流程框架相比，图 2 更侧重于技术层面的支撑机制，展示了基于多智能体协作的技术架构。具体而言，图 1 中的“数据清洗及特征筛选”对应图 2 中的感知模块，通过输入解析、意图提取、歧义预检测和上下文关联完成对原始需求的特征化表示；图 1 中的“检测”和“分类”对应图 2 中的元认知模块，由任务调度、指导、执行监控和策略融合等子模块驱动不同类型的专家智能体完成具体检测；而图 1 中的“结果融合”和“评估模块”则对应图 2 中的“评估优化”模块，利用证据融合与性能指标分析输出最终检测结果。图 1 和图 2 在逻辑上是一一对应、互为补充的，前者是面向需求歧义检测任务的过程描述，后者是该过程在大语言模型与多智能体协作机制下的技术落地方案。接下来，将依次展开介绍各个环节的具体方法与实现。

任务调度模块初始化工作流，分配 GatingAgent 进行初步筛选；指导策略模块根据输入动态生成优化指令，指导后续专家选择；执行监控模块跟踪 AmbiguousExpert 等智能体的执行，记录中间置信度；

策略融合模块由 FusionAgent 整合结果，应用 D-S 证据理论生成结论；评估优化模块由 EvaluatorAgent 和 RSValidator 验证输出并反馈优化策略，形成闭环。图 2 中的模块与算法 2 的智能体一一对应，记忆模块分为短期记忆和长期记忆。短期记忆存储当前任务的即时上下文（如输入解析后的意图提取结果和歧义预检测的初步证据），允许智能体在执行监控模块的指导下实时访问和更新，以实现动态调整（如在语用歧义检测中共享跨句逻辑关系）；长期记忆则积累历史检测数据和知识（如以往歧义分类的模式），通过任务调度策略提供给策略融合模块，用于优化后续分类的泛化能力。记忆模块存储多智能体协作中关键信息，以避免重要分析过程中遗漏掉这些过程分析，确保分析的正确性和逻辑严密性。

2.1 基于启发式规则的初步筛选

通过对需求歧义检测方法的广泛调研，我们整理出自然语言需求文本中常见的 6 种歧义类型，分别为词汇歧义、句法歧义、语义/范围歧义、语用歧义、语法歧义和语言错误歧义[1,27]。上述歧义类型反映了多种语言挑战，比如歧义术语、多种句法解释、范围不明确、上下文误解、语法错误和拼写错误等语言歧义场景。为了解决上述挑战，LMAdetect 采用了一套启发式规则对需求文本进行预处理并提取潜在的歧义需求，从而生成结构化的特征集，为后续的大语言模型分析提供基础。

在歧义检测的调研中，我们发现有 4 类情形是基于需求文本进行的，其中提取模糊词汇和提取指代不明语句是根据语言语义进行分析的，提取被动语态结构是根据句法信息进行分析的，提取重复表述是根据文本模式进行分析的。基于这些发现，我们针对这 4 类情形设计了相应的启发式规则，以实现更加精确和有效的歧义检测。针对 4 类分析维度，结合 NLTK、Benepar、FastCoref 和 WordNet，生成支持 6 种歧义类型的特征集 F，突破传统方法对单一模式的依赖。通过共指消解和句法分析，LMAdetect 在词汇上下文（明确“bank”含义）和句子上下文（解析“it”的指代）上显著优于传统方法，提升了语义理解和引用/指代识别能力。

2.1.1 歧义类型的检测规则

为了应对更多样的歧义类型，增强歧义检测的正确率，本研究构建启发式规则时，对现有的方法进行了适当的扩展。Gleich 等人[25]提出的规则驱动方法主要是聚焦于词汇和句法歧义的检测方面，比如利用正则表达式去完成关键词和部分词性标注的匹配，以此来识别模糊的术语以及句法结构。但是这些规则局限于静态模式匹配情形，未充分考虑语义依赖性以及上下文复杂性。Sabriye 等人[26]提出的 POS 标记方法尽管可借助词性标签检测语法和句法歧义，但只是局限于局部特征范畴，忽略了全局语义和语用相关的线索信息。LMAdetect 在上述基础之上增添了规则和知识。以原有的词汇匹配为基础，新增了对量化词（如“all”“some”）的语义范围分析规则，并结合上下文长度和修饰语进行一致性判断。当检测到包含量化词但无明确边界定义且句子少于 30 词时，触发歧义检测规则，确保了语义的独立性。该阈值 30 基于对实验数据集的统计分析和经验总结选取。在 1 192 条需求数据中，需求语句的词数平均约为 25（通过对数据集的词频统计得出），选择 30 作为阈值可有效避免过长上下文干扰语义独立性，同时覆盖大多数潜在歧义场景。该选取参考了需求工程实践中对语句简洁性的要求（如 Gleich 等人[25]中对需求语句结构的观察），确保规则的实用性和泛化能力。针对 Sabriye 等人[26]提出方法的不足，本文引入了基于共指消解、结合上下文关联的规则，能检测如“notify users”因没有上下文引发的歧义。检测条件是代词或指示词（如“this”“that”）前后未形成明确的语义连接，且使用部分引用实体出现次数少于 2 的语言情况，以此来捕捉上下文的依赖关系。此外，本文扩展了传统规则的语法检查，增加了自动识别拼写错误以及语法不一致的功能，采用 WordNet 进行词典查询以及语法树解析，若需求语句中的错误词孤立存在，且没有修正上下文的条件时进行检测。而且，本文新引入了基于文本相似度阈值大于 0.8 的优先级排序机制，把高频重复的表述暂定为潜在歧义，采用优先级排序机制优化检测的优先级。上述对现有方法进行的扩展融入了语义角色标注、共指消解等 NLP 技术，并借助大语言模型理解上下文的能力[35]，填补传统规则所受的限制。与原有的静态规则相比较，LMAdetect 的扩充规则不仅覆盖了 6 种歧义类型，还凭借动态特征分析（如句子长度、上下文关联）增强了泛化性能。

针对每种歧义类型的启发式规则被设计用于识别特定的语言模式，并在表 2 中总结了这些规则及其定义、检测条件和示例。

针对模糊词汇的提取，LMAdetect 将枚举需求语句中的常见模糊词（如“fast”“clear”“possible”），对于每一个模糊词，判断其是否满足以下条件，若满足，则将其加入候选项。

Table 2 Ambiguity Type and Detection Rules

表 2 歧义类型和检测规则

歧义类型	定义	检测条件	示例
词汇歧义	由于存在多种可能的解释，主观术语含义不明确	① 句子长度小于等于 30 词 ② 术语未在词汇表或上下文中定义 ③ 单个主谓结构，无明确修饰语 ④ Word 存在于多义词词典（如 WordNet）中，具有多种含义	“The system shall respond quickly.” （“quickly”不明确）
句法歧义	多种可能的句法结构导致不同的解释	① 分析出的句法树数量超过 1 ② 名词短语修饰语的数量超过 1 ③ 修饰语之间缺乏明确的优先级或关系一致性	“The old men and women ate lunch.” （分组不明确）
语义/范围歧义	句子中的术语范围不明确或含义不一致	① 术语范围存在多种可能的解释 ② 并列结构中分组不一致且可能的解释数量大于 1 ③ 缺乏明确意图或上下文解析	“All files in the system are secure.” （“all”的范围不明确）
语用歧义	由上下文或预期用途引起的模糊性，影响利益相关者的理解	① 可能的解释数量超过 1 ② 与上下文无明确匹配意图 ③ 句子长度小于等于 50 词，以避免过于复杂	“The system will notify users.” （通知方式不明确）
语法歧义	语法结构不正确或不清楚，导致误解	① 生成的语法树存在超过一种可能的结构 ② Word 词性标签显示潜在的语法错误（例如动词时态不匹配） ③ 句子长度不超过 40 词	“The data process daily.” （“process”中的语法错误）
语言错误歧义	语言使用错误导致沟通不畅	① Word 未在词典中找到或词性不匹配 ② 连续词语之间缺乏明确的连接成分 ③ 句子长度小于等于 30 词	“The sistem is fast.” （“system”拼写错误）

1)模糊词所在的需求示例长度不超过 30 词。

2)模糊词后的语句中未明确定义其语义(如无词汇表支持)。

3)模糊词所在的句子最多包含 1 个主谓结构，且无歧义修饰语。

上述条件中，第 1 个要求对需求示例长度的大小范围加以限定，防止过长上下文引发的干扰，后 2 个条件是从语义独立性的角度考虑的，确保模糊词不因上下文澄清而失效。在检测过程中，要特别留意那些即使提供了上下文信息加以澄清，但系统或工具还是不能正确识别或处理这些模糊词的情形。这种情况可能会造成需求理解出现偏差，故而在设计与实现阶段需考虑并进行解决。

针对指代不明语句的提取，LMAdetect 将枚举需求语句中的代词(如“it”“they”)和指示词(如“this”“that”)，并根据其指代部分(前文定义)和使用部分(后文引用)划分成 2 部分，对于每一个部分，判断是否满足以下条件，若满足，则将其加入候选项：

1)指代部分和使用部分之间无明确语义连接。

2)该部分的词数不超过需求示例总词数的 70%。

3)对于指代部分，该部分后的语句中未使用相关实体，且无歧义解。

4)对于使用部分，该部分引用的实体在后续语句中出现次数不超过 1，且最多在句末有单一解释。

以上 4 个条件中，第 1 个条件是基本要求，确保指代关系独立，第 3 个条件针对指代部分，旨在避免后续干扰；第 4 个条件针对使用部分，旨在限制其语义依赖。

针对被动语态结构的提取，则直接根据句法模式，找到包含被动语态的句子结构(例如“be + VBN”)，按照判断语句是否可被标记为歧义的 3 个条件进行判断，若满足，则加入候选项：

1)句子包含被动语态动词(如“is processed”)；

2)主语或施动者未被明确指定；

3)句子词数不超过 40。

针对重复表述的提取，首先我们使用文本相似性对需求语句进行 1 轮比较，标注每个语句的重复模式，随后在相似度超过阈值 0.8 的语句之间进行分析，若 2 个语句仅在具体实体(如“user”vs.“customer”)上不一致且可形成映射关系，则视为重复表述。针对重复表述，按照判断语句是否可标记为歧义的 3 个条件进行判断，若满足，则加入候选项。

对于每一个候选项，我们尝试将对应语句标记为语言歧义，若标记无冲突，则将其作为候选项。若多个候选项重复标记某一句子，则认为它们是相同的，这将在后续优先级排序中多次计入推荐次数。

2.1.2 预处理算法

算法 1. 自然语言需求预处理。

输入：NL requirement (r)；

输出：feature set (F)。

① $SEG \leftarrow SEPARATESEGMENTS(r)$；

② $F \leftarrow \varnothing$；

③ for each $seg \in SEG$ do

④ $W \leftarrow TOKENIZE(seg)$；

⑤ $T \leftarrow POS\_TAGGING(W)$; /*HMM-based*/

⑥ $P \leftarrow PARSE\_TREE(seg)$；

⑦ $CR \leftarrow COREFERENCE\_RESOLUTION(seg)$;

⑧ $GE \leftarrow GRAMMAR\_ERROR\_CHECK(seg)$;

⑨ $F \leftarrow F \cup \{W,T,P,CR,GE,has\_period(seg), is\_passive(seg)\}$;

⑩ end for

⑪ return F。

算法 1 概述了自然语言需求的预处理流程，目标为生成用于歧义检测的特征集 F。预处理阶段把原始需求文本转换成结构化特征集(F)，以支持启发式规则的应用。此流程涉及分段、分词、词性标注、句法分析、共指消解以及语法检查等步骤。

1)分词：将文本拆分为单词和短语。

2)词性标注：使用如 NLTK 这样的工具，为词语分配语法标签(例如名词、动词)。

3)句法分析：利用如 Benepar 这样的解析器生成句法树，以识别结构上的歧义。

4)共指消解：通过 FastCoref 等工具识别代词引用，解决指称歧义问题。

5)词典查询：通过 WordNet 等资源检查单词的有效性和多义性，以解决词汇和语言错误的歧义。

6)上下文分析：评估句子长度和修饰语的一致性，以识别语义/范围和语用歧义。

这种预处理确保了输入需求被转换成结构化的表示，从而能够对候选的歧义需求进行基于规则的有效过滤。

2.2 基于大语言模型的歧义分类

大语言模型擅长阅读和理解自然语言，通过大语言模型的深层语义理解，捕捉词汇上下文和句子上下文，增强语义理解和逻辑关系识别能力。它们更有可能识别出具有内聚性的语言片段，这种内聚性语言片段可能包含相对独立的歧义模式，在需求检测时它们更有可能被分配到同一个歧义类别中。因此，我们可以利用这种内聚性语言片段进行语言歧义检测。基于这一假设和针对语言歧义检测的调研，我们使用大语言模型将给定的需求示例分解为一组小的内聚性语言片段。而大语言模型的性能受提示的影响很大。为此，我们借鉴 Liu 等人[36]的工作进行了提示工程。设计了一个由多维度组成的提示模板，如表 3 所示。

规范化提示词模板即〈Agent Role and Method〉+〈Ambiguity Types List〉+〈Detection Process〉+〈Output Format〉+〈Example Outputs〉由智能体角色和方法、歧义类型、检测过程、输出规范、示例输出这 5 个部分组成。

我们在提示中明确指定了一些约束条件来引导大语言模型进行检测。首先，在提示词中，我们会明确任务，分别让大语言模型直接进行语言歧义检测、基于词汇进行歧义检测、基于短语进行歧义检测。后两者是根据前期调研发现的和语义相关的最常见的 2 种语言歧义类型，我们将单独引导大语言模型针对这 2 类检测进行考虑。其次，我们告诉大语言模型具体的检测流程，因为大语言模型可能会随意执行除歧义检测之外的其他处理，这种意外处理应该避免。接着，明确规定输出格式，以减少小片段或无效片段带来的干扰。最后，要求按照示例进行标准输出，因为我们在后续的处理中，将依赖大语言模型生成的结果来合并这些有内聚性的歧义，即识别出相关的歧义，并将其进行整合处理，以确保在特定上下文下的准确理解。这一过程需要结合语法、语义和上下文信息。

根据表 3 所示，提示词引导大语言模型专注于歧义检测，并减少无关的处理过程，确保输出达到标准化。借助算法 1 来处理特征集 F，每个需求将被划分到明确定义的歧义类型中，或者判定为无歧义。为了弥补语义理解以及引用/指代识别方面的欠缺，提示工程引导模型对句子间的逻辑关系和指代对象进行分析。

2.3 基于智能体对大语言模型歧义检测结果合并

智能体能够感知环境、自主决策并执行动作以实现目标，通常包含感知模块、决策模块、执行模块、记忆模块。大语言模型为智能体提供语言交互与推理能力，智能体通过大语言模型解析用户指令，将自然语言解析为结构化意图，设计结构化提示词，引导大语言模型按特定逻辑推理。大语言模型是智能体实现“认知智能”的核心组件，为其提供语言理解、知识推理能力；而智能体框架则是大语言模型落地现实场景的“桥梁”，赋予其感知环境、执行动作的能力。二者的结合通过“规划—执行—反馈”的闭环设计，推动任务从“被动应答”向“主动解决问题”进化。

LMAdetect 框架借助多智能体协作方法整合大语言模型所得结果，进而强化歧义检测能力。这一过程首先针对各类歧义进行结构优化，如图 3 所示，凭借启发式规则与大语言模型结合的方法，分别对词汇、句法、语义/范围、语用、语法及语言错误等类型的歧义检测进行分析和优化。初步的分类结果根据各类型的语言特征生成。

为了增强检测的鲁棒性，LMAdetect 采用了多智

Table 3 Prompt Words

表 3 提示词

相关要素	提示词
〈Agent Role and Method〉	You are RSDetector, an agent responsible for detecting and classifying ambiguities in requirements statements using〈language understanding or specific method〉
〈Ambiguity Types List〉	Analyze the input sentence based on the following ambiguity types, their definitions, and examples: ①〈Ambiguity Type 1〉:〈Definition of Ambiguity Type 1〉 - Example:〈Example Sentence for Type 1〉(〈Explanation of Ambiguity〉) ②〈Ambiguity Type 2〉:〈Definition of Ambiguity Type 2〉 - Example:〈Example Sentence for Type 2〉(〈Explanation of Ambiguity〉) [… for each ambiguity type]
〈Detection Process〉	①〈Step 1: Analyze Sentence〉 ②〈Step 2: Type-Specific Detection〉 ③〈Step 3: Prioritize or Distinguish Types〉 ④〈Step 4: Classify Ambiguity〉 ⑤ If no ambiguity is detected, output 'No Ambiguity'
〈Output Format〉	- If no ambiguity: 'No Ambiguity' - If ambiguous: 'Ambiguity Detected: [Type]' where [Type] is one of the defined types (e. g., '〈Ambiguity Type 1〉')
〈Example Outputs〉	- Input: '〈Example Input 1〉' → '〈Example Output 1〉' - Input: '〈Example Input 2〉' → '〈Example Output 2〉' [… for each example]
You are RSDetector, an agent responsible for detecting and classifying ambiguities in requirements statements using〈language understanding or specific method〉+ Analyze the input sentence based on the following ambiguity types, their definitions, and examples: ①〈Ambiguity Type 1〉:〈Definition of Ambiguity Type 1〉 - Example:〈Example Sentence for Type 1〉(〈Explanation of Ambiguity〉) ②〈Ambiguity Type 2〉:〈Definition of Ambiguity Type 2〉 - Example:〈Example Sentence for Type 2〉(〈Explanation of Ambiguity〉) [… for each ambiguity type] + Process: ①〈Step 1: Analyze Sentence〉 ②〈Step 2: Type-Specific Detection〉 ③〈Step 3: Prioritize or Distinguish Types〉 ④〈Step 4: Classify Ambiguity〉 ⑤ If no ambiguity is detected, output 'No Ambiguity'+ Output Format: - If no ambiguity: 'No Ambiguity' - If ambiguous: 'Ambiguity Detected: [Type]' where [Type] is one of the defined types (e. g., '〈Ambiguity Type 1〉'). + Example Outputs: - Input: '〈Example Input 1〉' → '〈Example Output 1〉' - Input: '〈Example Input 2〉' → '〈Example Output 2〉' [… for each example]	

能体框架——类型专业化多智能体协作模型。该模型将不同类型的歧义性分配给专门的智能体，并通过 Dempster-Shafer(D-S)证据理论整合这些智能体的输出结果。

如图 4 所示，这种基于混合专家带路由的模式，不仅可以实现单一任务的精准检测，还提供了类型协同优化的创新点。在此基础上形成了一个完整的框架。对第 1 层门控智能体进行判断是否存在歧义，存在歧义的需求传入有歧义专家这个智能体中。对第 2 层门控进行判断具体歧义类型，通过路由将需求分配到具体歧义类型专家进行针对性检测，每个任务是一个小专家，整个模式是一个基于混合专家带路由的形式。由于歧义检测类型的多样性，涉及多个维度(语义逻辑、上下文等)。单一模型处理所有歧义会导致注意力分散、误分类增多、准确性降低。因此我们采用这种智能体分步来实现歧义类型的检测，每种歧义类型需要不同推理深度，例如语义/范围歧义需要逻辑推理而语用歧义需要上下文假设。在大量数据场景下，分布设计通过任务分解降低单次推理复杂度，确保效率和稳定性。在实验过程中，第 1 层门控智能体处理时间约 4 s，第 2 层门控智能体的处理时间约 25 s，各歧义专家的处理时间总共约 1 min，基于第 1 层门控快速筛选可能存在歧义的语句，再将第 2 层门控以及路由分配给多专家智能体进行精准分析，这种方法通过初步筛选节省计算资源，同时通过专业化提升检测的准确性。

类型专业化多智能体协作框架的核心是按歧义类型给智能体精准分工。让智能体的分工更精准专业，让每个智能体当领域专家。通过多智能体分工和内存模块，动态解析句子间的逻辑关系和指代对象，通过专业智能体和大语言模型推理，增强逻辑关系识别和引用/指代识别，通过层次化协作和上下文记忆，提升检测灵活性和覆盖面。

设计增加类型专家多智能体模型的需求歧义检

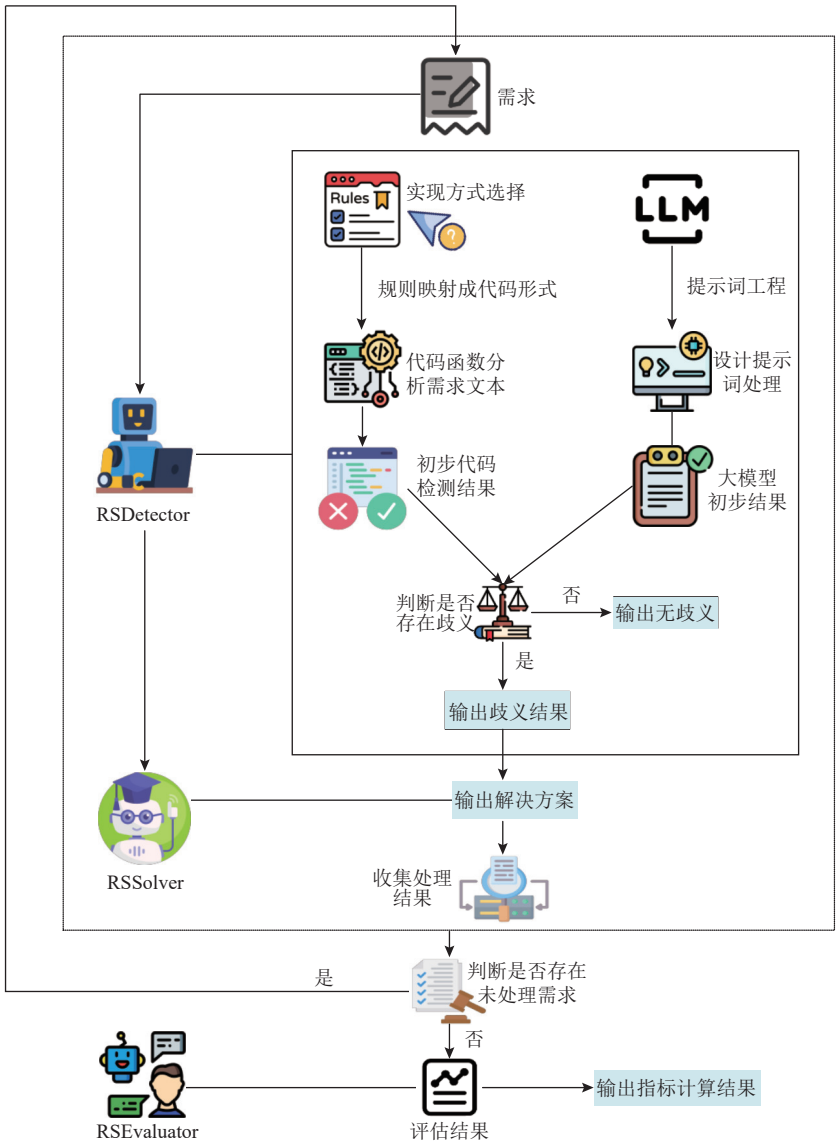

Fig. 3 Method framework diagram of the combination of large language models and rules

图 3 大语言模型与规则结合的方法框架图

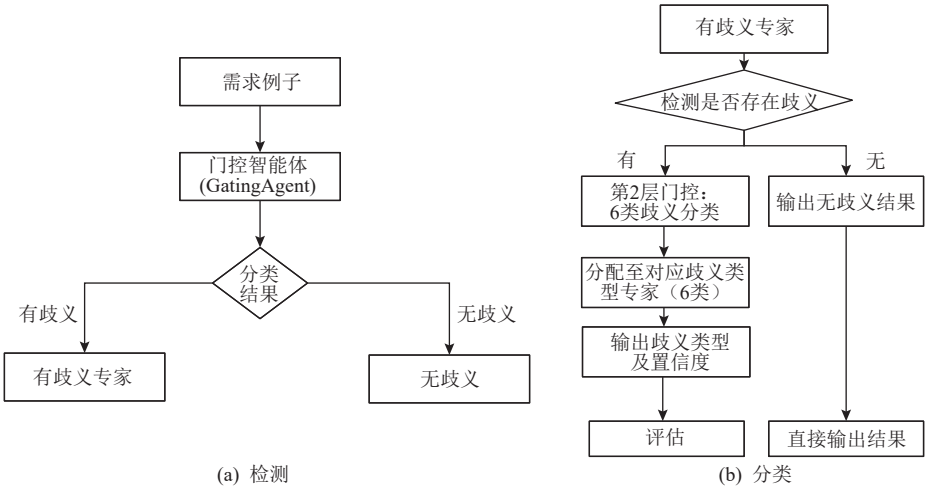

Fig. 4 Mixed expert framework diagram guided by ambiguity detection

图 4 歧义检测引导的混合专家框架图

测工作，目的是借助大语言模型的推理能力和结构化的多智能体协作机制来提升歧义检测效果。如图 5 所示，类型专家多智能体协作模型将智能体组织成一个层次结构，从门控智能体开始，该智能体根据词汇多义性或句法复杂性等句子特征分配检测任务。接下来是专家智能体，词汇专家的工作是检测词汇歧义，句法专家的工作是分析结构方面的歧义，语义专家的工作是识别范围存在的不一致，每个智能体处理来自大语言模型规则优化阶段（如图 4 所示）的初步结果。融合智能体随后整合这些输出，生成一个连贯的歧义判别结果。

算法 2 对智能体的协作过程进行了形式化的说明。借助 GatingAgent 对输入的需求进行筛选，来判断是否存在歧义。AmbiguousExpert 验证潜在的歧义类型。诸如词汇专家、句法专家等特定类型的智能体采用加权投票开展置信度评分，并且运用 Tregex 模式匹配和 WordNet 查询，分别进行句法模式分析以及词汇验证分析。FusionAgent 把上述分析结果进行汇总整合，分配置信度分数并处理冲突，进而判定歧义类型和检测置信度。此算法借助内存模块的上下文存储提升智能体之间的协作，保障高效且准确地检测 6 种歧义类型。

算法 2. 检测需求歧义。

输入：NL requirement (r), feature set (F), Agents {GatingAgent, AmbiguousExpert, Experts, FusionAgent}；

输出：detected ambiguity result (D)。

① $M \leftarrow$ {content: r, POS_analysis: F}；
② $D \leftarrow \varnothing$；
③ /*Step 1: GatingAgent Initial Screening*/
④ $O\_gating \leftarrow RUN\_GATINGAGENT(M)$；
⑤ $(detected\_g, type\_g, conf\_g, expl\_g) \leftarrow PARSE\_OUTPUT(O\_gating)$；
⑥ if not $detected\_gandconf\_g \geqslant 0.9$ then
⑦ $D \leftarrow$ (False, "None", $conf\_g$, $expl\_g$)；
⑧ return D；
⑨ end if
⑩ /*Step 2: AmbiguousExpert Verification*/
⑪ $O\_ambiguous \leftarrow RUN\_AMBIGUOUSEXPERT(M)$；
⑫ $(detected\_a, types\_a, expl\_a) \leftarrow PARSE\_OUTPUT(O\_ambiguous)$；
⑬ $T\_top2 \leftarrow SELECT\_TOP2\_TYPES(types\_a)$；
⑭ if $detected\_aandT\_top2 \neq \varnothing$ then
⑮ $E\_selected \leftarrow$ {Experts matching T_top2}；
⑯ else
⑰ $E\_selected \leftarrow$ all Experts；
⑱ end if
⑲ /*Step 3: Expert Analysis*/
⑳ for each $E\_i \in E\_selected$ do
㉑ $O\_i \leftarrow RUN\_EXPERT(E\_i, M)$；
㉒ $(detected\_i, type\_i, conf\_i, expl\_i) \leftarrow PARSE\_OUTPUT(O\_i)$；

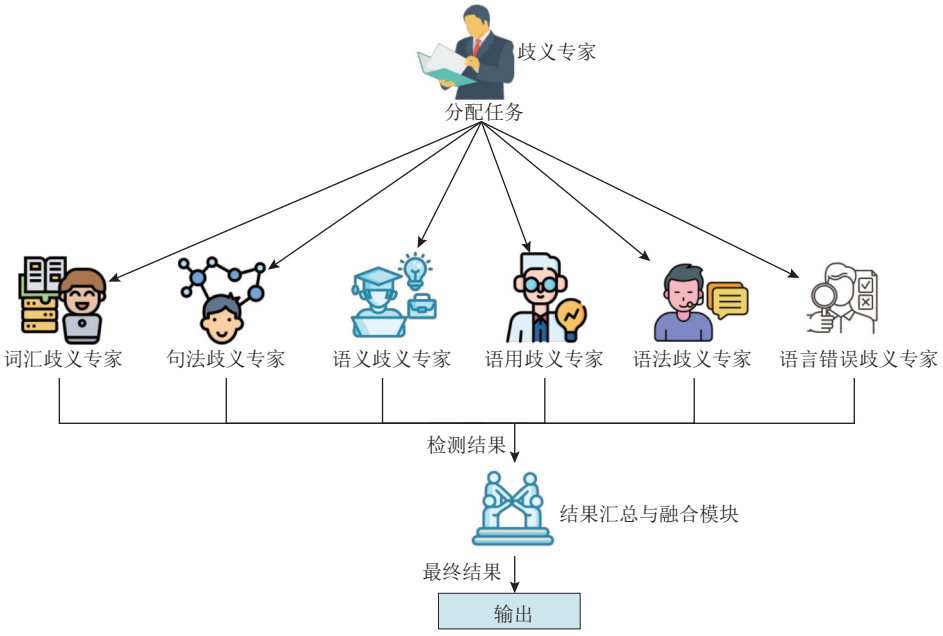

Fig. 5 Type specialized multi-agent collaboration framework diagram

图 5 类型专业化多智能体协作框架图

㉓ $D \leftarrow D \cup \{(detected\_i, type\_i, conf\_i, expl\_i)\}$;
㉔ end for
㉕ /*Step 4: FusionAgent Integration*/
㉖ $I\_fusion \leftarrow$ {GatingAgent:($detected\_g, type\_g, conf\_g, expl\_g$), AmbiguousExpert: ($detected\_a, types\_a, expl\_a$), Experts:$D$};
㉗ $O\_fusion \leftarrow RUN\_FUSIONAGENT(I\_fusion)$;
㉘ $(detected\_f, type\_f, conf\_f, expl\_f) \leftarrow PARSE\_OUTPUT(O\_fusion)$;
㉙ $D \leftarrow (detected\_f, type\_f, conf\_f, expl\_f)$;
㉚ return D。

此模型利用大语言模型强大的语义推理能力，凭借记忆模块存储短期和长期上下文信息，增进各智能体之间的协作效率。记忆模块作为多智能体框架的核心组件，通过存储共享上下文以支持跨智能体协作，即使在智能体间无需频繁实时交互的场景中也能发挥作用。具体使用方式包括：1)初始化阶段，通过感知模块捕获输入需求上下文，并存储到短期记忆中，用于实时任务调整；2)执行过程中，智能体通过 API 调用访问共享数据，如语用歧义专家检索前期句法分析结果；3)融合阶段，长期记忆提供历史知识给策略融合模块以优化分类。在实际场景中，例如处理大型需求文档时，记忆模块存储跨句引用关系，允许异步智能体（如语义专家和语用专家）独立工作，而非实时同步，从而避免交互开销。该设计参考多智能体系统中的异步内存机制[37-39]，澄清了其在非实时交互环境中的实用性，避免潜在误解。模型包含以下关键的智能体部分，GatingAgent 负责任务分配，以词汇多义性和句法复杂性等语句特征为依据，将检测任务分配给专业智能体。LexicalExpert 的职责是检测词汇歧义，查询多义词词典并结合上下文嵌入，从而判断词汇的多义性。SyntacticExpert 负责开展句法歧义的分析工作，采用基于句法树解析的方式识别模糊的句式结构。SemanticExpert 着重处理语义和范围歧义，结合语义角色标注对语句的不一致性进行检测，确保可以精准检测每一种歧义类型。如算法 2 所示，协作逻辑利用加权投票、Tregex 模式匹配和 WordNet 查询对置信度分数进行分配，为后续的融合过程奠定了基础。

为了进一步阐述多智能体协作的详细流程，我们设计了算法 2。该算法概述了从初步筛选到结果整合的歧义检测过程。首先，通过 GatingAgent 进行初始筛选，若置信度大于等于 0.9 且无歧义，则直接返回结果，以优化计算效率。随后，AmbiguousExpert 验证歧义并输出类型列表 types_a。关键步骤在于函数从 types_a（歧义类型与置信度的字典列表）中按置信度降序排序，选取排名前 2 的类型后续分配给对应的专业专家智能体。这一选择的依据在于体现多智能体协作中的“专家优先”效应。通过聚焦置信度最高的歧义类型，避免激活所有专家，减少不必要计算和潜在幻觉干扰，同时提升针对性和鲁棒性。如果 types_a 为空或置信度不足，则回退到激活所有专家，确保全面覆盖。最终，FusionAgent 通过 D-S 证据理论整合结果，生成稳定的检测输出。算法 2 与 TS-MAC 模型相结合，确保了歧义检测的层次化和高效性。

算法 2 描述通过多智能体协作检测需求语句中的歧义类型和置信度。通过多智能体协作检测语句中的歧义，包括 GatingAgent 筛选、AmbiguousExpert 验证、专业智能体分析和 FusionAgent 整合。使用加权投票、Tregex 模式匹配、WordNet 字典查询，最终生成初步的歧义检测结果，包括类型和置信度。

为解决多智能体检测结果的冲突问题，本文方法采用 D-S 证据理论进行结果融合。D-S 理论通过基本概率分配（basic probability assignment, BPA）函数为各智能体的输出分配置信度，并利用组合规则整合结果。生成最终的歧义类型判断如图 6 所示。FusionAgent 根据各智能体检测结果的置信度生成综合判断，输出融合的结果。如果冲突程度超过设定的阈值，系统将触发验证智能体进一步验证检测结果。

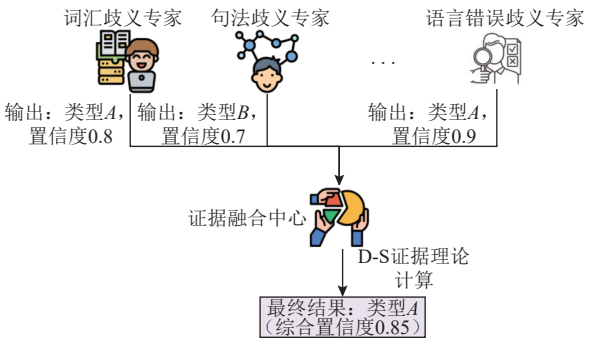

Fig. 6 D-S evidence theory fusion framework diagram

图 6 D-S 证据理论融合框架图

通过结合 D-S 证据理论的应用，使检测结果得到了进一步的保障，能够有效提升检测的准确度，显著提高了检测结果的鲁棒性和一致性。

3 实验分析

本节从不同维度对 LMAdetect 进行实验评估。

首先针对 LMAdetect 在第 1 节提出的挑战与问题,给出关于该歧义检测方法的 4 个研究问题。随后通过评估歧义检测方法的不同方面来解答这 4 个研究问题。为评估 LMAdetect 的检测效果,我们开展了实验进行评估。最后基于评估结果分析了 LMAdetect 的优势与局限性。

3.1 研究问题

为了评估 LMAdetect 是否有效解决了上述挑战,本节提出以下 4 个研究问题:

问题 1: 数据集的单句需求样本是否在语义、场景和逻辑上完全独立?为评估数据集的上下文独立性,我们提出研究问题 1,验证每种歧义类型上实验数据集是否具有独立性,通过梳理论文中对数据集的描述以及基于大语言模型对数据集进行关联性分析,证明其数据集的上下文独立性。

问题 2: LMAdetect 在检测单个类型的需求歧义性能方面是否优于现有的需求歧义检测方法?为对比评估其单独歧义类型检测上的有效性,我们提出研究问题 2,验证 LMAdetect 在每种歧义类型上的检测性能是否优于基线方法,通过准确率、召回率、精确率、*F*1 分数指标计算证明其在单独类型检测上的有效性。

问题 3: LMAdetect 在检测语用歧义和语言错误歧义上的检测效果如何?为了评估现有方法无法检测的 2 种歧义类型(语用歧义和语言错误歧义)的检测性能,我们提出了研究问题 3。验证 LMAdetect 通过大语言模型和多智能体协作,在语义理解、逻辑关系识别和引用/指代识别方面的改进,证明其上下文感知能力的优越性。

问题 4: 在多类歧义检测任务中,不同模型对结果的影响如何?为评估在综合检测多种歧义类型时选择不同模型对 LMAdetect 性能的影响,我们提出了研究问题 4。验证不同模型在处理歧义检测问题上的效果,重点关注通用模型与推理模型的表现差异,通过对比分析各模型在准确率、召回率、精确率和 *F*1 分数等指标上的结果,全面评估其在多样化歧义类型任务中的适用性。

3.2 评价指标及基准方法

在本文中,我们将采用 4 个经典的指标来评估 LMAdetect 的性能,包括准确率[40-41]、召回率、精确率[42]、*F*1 分数[43]。这些指标在歧义检测相关研究中广泛使用。这些性能指标的计算基于标准统计概念,如真阳性(true positive, TP)、假阳性(false positive, FP)、真阴性(true negative, TN)、假阴性(false negative, FN)。我们分别比较每种歧义类型检测的准确率、召回率、精确率、*F*1 分数。

1)准确率。模型正确分类的样本数占总样本数的比例。准确率衡量的是模型整体分类的正确比例。

2)召回率。模型正确检测为正例的样本数占所有实际正例样本数的比例。召回率衡量的是模型识别出所有需求歧义的能力。

3)精确率。模型正确检测为正例的样本数占所有被模型检测为正例的样本数的比例。这个指标反应了模型在所有判定为歧义的需求中,准确识别出需求歧义的能力。

4)*F*1 分数。精确率和召回率的调和平均数,用于综合评估模型的性能。当模型在精确率和召回率方面都表现良好时,*F*1 分数也会较高。

我们将所提方法 LMAdetect 与 Gleich 等人[25]的工作、POS 标记[26]、和 Paska[31] 进行比较,这 3 种方法代表了通过传统规则和句法分析进行歧义检测。

第 1 种方法是基于 NLP 技术的自动化歧义检测与解释方案,第 2 种方法是使用词性标注技术检测 SRS 中语法和句法歧义的方案,第 3 种方法是针对自然语言需求文档的自动化歧义检测与改进推荐方案。

Gleich 等人[25]的工作结合了正则表达式和部分词性标注技术,是一种基于词法和轻量级句法分析的歧义检测工具,检测包括词汇、句法、语义在内的多种歧义类型。相比较而言,LMAdetect 利用大模型的深层语义理解,捕捉量词作用域和逻辑连接词,突破了 POS 标注静态规则的局限。反馈和模型重试机制增强了动态上下文感知能力,而检测和评估的多智能体协作通过误分类分析强化逻辑关系识别,形成了闭环优化框架,结合 6 种歧义类型分类和多样化数据处理,提升了召回率、精确率和 *F*1 分数,超越传统方法依赖人工规则和 POS 标注、缺乏深层语义理解的局限性。

POS 标记[26]方法聚焦于对 SRS 文档中语法和句法歧义的检测工作,通过预处理、处理以及后处理 3 个组件开展歧义检测工作。与 POS 标记方法相比,LMAdetect 可以更精准地识别 SRS 中的语法与句法歧义,解决了静态标签序列语义缺失的局限问题。引入大语言模型驱动的动态语义理解模块,通过评判被动语态以及施动者的合理性,以“结构 + 语义 + 语境”的多维检测框架全面优化检测过程,进而捕捉需求是否存在歧义情况。检测过程融合 POS 标记与语境推理、评估校验的一致性来进行,形成多智能体协同的闭环反馈机制。该机制增强了语义理解以及

逻辑关系的识别能力，并结合了 6 种歧义类型和工具（如 Benepar 句法树、FastCoref 指代消解）协同检测，进而可实现对非典型结构的覆盖，减少检测结果误判的情况，克服了 POS 标记因语境范围受限、语义内容不足导致的漏检与误判问题，实现了检测结果的召回率、精确率和 $F1$ 分数的提升。

Paska[31] 是一个自动化工具，其结合了多种 NLP 技术。通过分析需求的整体句法结构，匹配并推荐合适的 Rimay 模式来帮助分析师重写需求。与 Paska 相比，LMAdetect 将静态 Tregex 规则集合大模型驱动的语义理解模块，主动捕捉介词短语附着的多义性等句法歧义结构，突破了 Paska 依赖单句结构匹配的局限。此方法形成了“语义解析—结构校验—语境补全”的闭环机制，通过结合 6 种歧义类型分类体系，从而实现对复杂句法结构的自适应检测。相比 Paska 的预定义规则，这种动态协同推理提升了约 8 个百分点~14 个百分点的准确率、召回率和 $F1$ 分数，增强了泛化能力，体现了在歧义和领域泛化上的优势。

在评价过程中，我们采用同一数据集，计算上述指标，将 LMAdetect 和基准方法的检测结果进行比较。此外，我们还对 LMAdetect 在不同歧义类型下的表现进行了分析，以全面比较其有效性与适用性。

3.3 实验数据

我们从 Google Scholar、IEEE Xplore、ACM Digital Library 等学术数据库检索到与特定主题相关联的文献，选取 12 篇相关文献作为分析对象。通过查找这些论文，我们获取了部分数据。此外，我们从相关论文的 GitHub 仓库收集到一些额外的数据资料。这些文献为我们的研究提供了宝贵的研究资料和已经验证的数据，为研究奠定了坚实的基础，是理论研究依据的关键要素，也是重要的数据来源之一。

首先，我们使用关键词（如“需求歧义检测”“需求歧义修复”“大语言模型”等）在多个学术数据库中进行初步检索，以识别与需求歧义相关的研究领域。初始检索结果约 70 篇论文，采用以下标准对文献进行筛选。1）相关性：论文必须聚焦于需求歧义检测或需求工程中的 NLP，优先选择包含真实需求示例的论文；2）质量：选择发表在高影响力期刊或会议上，或引用次数较多的论文（基于 Google Scholar 统计）；3）多样性：确保覆盖多种歧义类型（如词汇、句法、语义等），以提升数据集的泛化能力。首先，按照标题和摘要进行组略筛选，然后按照以上标准，确定文章研究问题的相关性，根据论文研究对象、歧义类别的数量等信息确保相关论文符合标准，排除与需求歧义检测或 NLP 无关的论文，通过阅读全文进一步验证论文的相关性，最终选定 12 篇高质量论文[1,13,20,22-23,25-26,44-48]。重点收集其中涉及实验数据和研究见解的部分，收集过程包括手动提取需求语句、标注歧义类型。我们从这些论文中提取与需求歧义检测相关的具体信息，包括常见的需求歧义类型及其在需求文档中的分布情况。为了获取所需的数据，我们凭借论文所给的 URL 链接访问目标数据库进行数据下载。最后，我们针对收集到的数据进行详细整理与归类，记录每种歧义类型在需求文档中的具体表现及其影响因素，为后续的深入研究和分析提供关键信息支持。

为确保数据集的标注一致性和可重复性，我们参考 Berry 等人[1] 的工作制定了标注准则，如表 4 所示。检查单词是否在上下文中允许多种解释，且这些解释源于词典含义而非句子结构，如果歧义可以通过词典或词源区分，则标注为词汇歧义；句法歧义源于语法结构而非单词含义或语义逻辑，如果通过重新解析树可得出不同含义，则标注为句法歧义；语义歧义源于逻辑形式而非语法，在没有词汇和结构歧义的情况下，如果句子语法唯一但逻辑解读多，则标注为语义歧义；句子在特定语境中出现多种含义，则标注为语用歧义；语法歧义源于句子语法形式缺陷，而非语境、词汇含义或逻辑关系；语言错误歧义源于语言错误而非故意，如果纠正错误可消除歧义，则标注为语言错误歧义。

我们在已获取的数据集上进行实验，在这些数据集中，有歧义的需求数据集占比 73.15%。表 5 给出了数据集所对应的信息。

3.4 实验方法

我们使用收集整理的 1 192 条需求例子作为实验数据集，并执行需求歧义检测工作。所有实验均在配备英特尔酷睿 i9 处理器和 16 GB 内存的 Windows11 机器上进行。为了展示大语言模型的强大性能，我们在实验中的大语言模型部分使用了 deepseek-v3-0324 模型。对于每一种需求歧义类型的检测，我们对应一个歧义类型的检测专家。针对每一类歧义类型专家，我们分别采用启发式方法和大语言模型分别进行检测。最终根据对应专家的检测结果以及置信度值进行汇总得出最终结果。

在本实验中，使用 agentscope 框架构建智能体，结合大语言模型和规则进行歧义检测主要通过模型封装模块中的检测功能实现，这一过程整合了大语言模型的自然语言理解能力与预定义的规则逻辑。

Table 4 Labeling Criteria

表 4 标注准则

歧义类型	标注准则	示例
词汇歧义	① 识别潜在歧义词（如多义词或同形词），使用词典检查多个含义 ② 检查上下文是否允许多种解释 ③ 标注时注明可能的含义，引用词汇表作为参考	“The bank can be slippery when wet.” ① 识别潜在歧义词“bank”，查词典：含义 1（金融机构）、含义 2（河岸） ② 检查上下文（湿滑时），支持含义 2，但 SRS 中可能误为含义 1 ③ 标注为词汇歧义，列出含义，建议用词汇表定义“bank”为“河岸”
句法歧义	① 绘制或想象句子的解析树，检查是否有多个合法结构 ② 测试不同结构是否产生不同含义 ③ 标注时，指定子类型并列出可能结构	“The police shot the rioters with guns.” ① 绘制解析树，结构 1（警察用枪射击暴徒）、结构 2（射击持枪暴徒） ② 测试含义差异，暴力程度不同 ③ 标注为句法歧义，列出结构，（shot（with guns））（rioters（with guns））
语义/范围歧义	① 转换为谓词逻辑，检查量词/否定范围是否允许多种形式 ② 测试上下文是否支持多种逻辑解读 ③ 标注时，列出逻辑形式和原因	“All linguists prefer a theory.” ① 转换为逻辑 ② 测试含义，每个喜欢不同理论、所有喜欢同一理论 ③ 标注为语义/范围歧义，加括号“All linguists prefer（a theory）”指定范围
语用歧义	① 分析上下文（前后句、情境），检查指代/指示是否多义 ② 测试替换先行词是否改变含义 ③ 标注时，指定子类型并列出可能上下文解读	“The trucks shall treat the roads before they freeze.” ① 识别指代词“they”，可能先行词：trucks 或 roads ② 测试上下文：冬季道路维护，含义 1（卡车冻前处理）、含义 2（道路冻前处理） ③ 标注为语用歧义，列出先行词
语法歧义	① 句子末尾无句号或句点，导致句子边界模糊，无法确定语义完整范围 ② 句子符合被动语态的 POS 标签组合且未明确“执行主体”（即未提及“谁执行该动作”），导致责任或操作方不明确	“Student records should be documented.” ① 词性标注 ② 匹配被动模式 ③ 标注为语法歧义
语言错误歧义	① 检查语法或标点错误 ② 测试意图和字面含义 ③ 标注时，注明错误类型和可能误解	“Everybody has their lunch.” ① 检查错误，“Everybody”是单数主语，应配单数代词，但“their”是复数 ② 测试意图 vs.字面：意图（每个人有自己的午餐）vs.字面（可能误为共享午餐或不确定数量） ③ 标注为语言错误歧义，列出误解

Table 5 Experimental Datasets

表 5 实验数据集

样本	歧义类型	样本数量
正例样本	Lexical Ambiguity	135
	Syntactic Ambiguity	127
	Semantic/Scope Ambiguity	353
	Pragmatic Ambiguity	92
	Syntax Ambiguity	80
	Language Error Ambiguity	85
负例样本		320

针对每类歧义类型的检测优化，通过调用 deepseek-v3-0324 模型结合对应检测歧义类型的规则进行针对性检测。在每类歧义类型优化结束后，通过路由分配给对应歧义类型专家进行整合检测，形成多模型多任务检测多种歧义类型。在调用大语言模型的过程中，根据相关工作[49]的经验，温度（temperature）在大语言模型中被用来控制生成文本的随机性。值越小，输出内容越倾向于确定性和稳定；而值越高，大语言模型在处理复杂任务或需要创造性的情况下表现更佳。在综合考虑这 2 种作用之后，我们选择了一个平衡点 temperature 为 0.5 来调用 deepseek-v3-0324 模型。

为确保结果的稳定性和准确性，我们进行了多次运行实验，对每个歧义类型的数据子集进行了 3 次独立运行，使用相同数据计算指标结果的平均值和标准差。为验证参数引入的随机性，我们额外设置 temperature 为 0（完全确定性模式）进行对比实验。如表 6 所示，temperature 为 0.5 的 $F1$ 分数整体略高于 temperature 为 0 的 $F1$ 分数，平均 $F1$ 分数提升 0.2 个百分点到 0.5 个百分点。这表明在需求歧义检测任务中，轻微随机性有助于探索更多边缘案例，提升综合性能，但 temperature 为 0 的稳定性更好。temperature 为 0.5 的随机性在语用歧义中可能导致结果不一致，适合需要创造性理解的场景。在句法歧义和语法歧义等结构化类型上，2 个 temperature 下的指标相似（差异小于 0.5 个百分点），这些类型依赖规则匹配和词性标注，随机性影响小且模型底层表示稳定。而在语用歧义和语义歧义等上下文依赖类型上，temperature 为 0.5 的召回率和 $F1$ 分数更高（提升 1 个百分点到 3 个百分点），随机性有助于捕捉隐含关系；temperature 为 0 的精确率偶尔更高，但召回率较低（因缺少探索遗漏实例）。总体上，准确率和精确率在 2 个 temperature 下最稳定（差异小于 1 个百分点），反映模型的核心分类能力强，受随机性影响小；召回率和 $F1$ 分数对 temperature 更敏感，在 temperature 为 0.5 下更高，因为随机性提升覆盖能力；$F1$ 分数作为

Table 6 Comparison of Stability and Accuracy of Different Temperature Parameters

表 6 不同温度参数稳定性与准确性比较

歧义类型	实验进程	temperature 为 0.5				temperature 为 0			
		准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
词汇歧义	第 1 次	0.844 4	0.870 2	0.966 1	0.915 7	0.859 3	0.878 8	0.974 8	0.924 3
	第 2 次	0.874 1	0.893 9	0.975 2	0.932 8	0.874 1	0.893 9	0.975 2	0.932 8
	第 3 次	0.866 7	0.879 7	0.983 2	0.928 6	0.837 0	0.856 1	0.974 1	0.911 3
	均值	0.861 7	0.881 3	0.974 8	0.925 7	0.856 8	0.876 3	0.974 7	0.922 8
	标准差	0.013 4	0.011 8	0.008 6	0.009 4	0.015 2	0.015 5	0.000 5	0.008 8
句法歧义	第 1 次	0.866 1	0.866 1	1.000 0	0.928 3	0.874 0	0.874 0	1.000 0	0.932 8
	第 2 次	0.889 8	0.889 8	1.000 0	0.941 7	0.881 9	0.881 9	1.000 0	0.937 2
	第 3 次	0.881 9	0.881 9	1.000 0	0.937 2	0.881 9	0.881 9	1.000 0	0.937 2
	均值	0.879 3	0.879 3	1.000 0	0.935 7	0.879 3	0.879 3	1.000 0	0.935 7
	标准差	0.012 1	0.012 1	0	0.006 8	0.004 6	0.004 6	0	0.002 5
语法歧义	第 1 次	0.850 0	0.957 7	0.883 1	0.918 9	0.837 5	0.957 1	0.870 1	0.911 6
	第 2 次	0.837 5	0.957 1	0.870 1	0.911 6	0.837 5	0.957 1	0.870 1	0.911 6
	第 3 次	0.850 0	0.971 4	0.871 8	0.918 9	0.837 5	0.957 1	0.870 1	0.911 6
	均值	0.845 8	0.962 1	0.875 0	0.916 5	0.837 5	0.957 1	0.870 1	0.911 6
	标准差	0.007 2	0.008 1	0.007 1	0.004 2	0	0	0	0
语义/范围歧义	第 1 次	0.716 7	0.816 1	0.854 7	0.835 0	0.711 0	0.809 7	0.853 7	0.831 1
	第 2 次	0.730 9	0.832 3	0.857 1	0.844 5	0.711 0	0.804 5	0.859 6	0.831 1
	第 3 次	0.728 0	0.831 7	0.853 8	0.842 6	0.688 4	0.781 4	0.852 6	0.815 4
	均值	0.725 2	0.826 7	0.855 2	0.840 7	0.703 5	0.798 5	0.855 3	0.825 9
	标准差	0.007 5	0.009 2	0.001 7	0.005 0	0.013 0	0.015 1	0.003 8	0.009 1
语用歧义	第 1 次	0.630 4	1.000 0	0.630 4	0.773 3	0.619 6	0.966 1	0.633 3	0.765 1
	第 2 次	0.608 7	0.982 5	0.615 4	0.756 8	0.597 8	0.964 9	0.611 1	0.748 3
	第 3 次	0.576 1	0.963 6	0.588 9	0.731 0	0.619 6	1.000 0	0.619 6	0.765 1
	均值	0.605 1	0.982 0	0.611 6	0.753 7	0.612 3	0.977 0	0.621 3	0.759 5
	标准差	0.027 3	0.017 7	0.021 0	0.021 3	0.012 6	0.019 3	0.011 2	0.009 7
语言错误歧义	第 1 次	0.811 8	0.971 8	0.831 3	0.896 1	0.811 8	0.971 8	0.831 3	0.896 1
	第 2 次	0.811 8	0.971 8	0.831 3	0.896 1	0.788 2	0.971 0	0.807 2	0.881 6
	第 3 次	0.800 0	0.957 7	0.829 3	0.888 9	0.800 0	0.957 7	0.829 3	0.888 9
	均值	0.807 9	0.967 1	0.830 6	0.893 7	0.800 0	0.966 8	0.822 6	0.888 9
	标准差	0.006 8	0.008 1	0.001 1	0.004 2	0.011 8	0.007 9	0.013 4	0.007 2

平衡指标，体现整体优势，temperature 为 0.5 时在语境类型上更突出。词汇、句法、语法这些结构化类型歧义依赖规则，性能相似（*F*1>92 个百分点）且差异小；语义、语用、语言错误这些语境类型歧义，因随机性有助于语义推理，在 temperature 为 0.5 下更好（*F*1 分数提升 0.5 个百分点到 2 个百分点）。总体，模型对结构化歧义鲁棒性强，对语境歧义 temperature 为 0.5 时优化更好。但稳定性考虑，temperature 为 0 时更适合生产环境。

对于不同 temperature 结果相似问题，在句法歧义的 *F*1 分数上，2 个 temperature 平均为 0.935 7，因为句法歧义主要靠 POS 标注和规则，模型的确定性推理足够，随机性不改变核心语法结构识别。对于不同 temperature 下的结果差异，在语用歧义的召回率上，temperature 为 0 的平均 *F*1 分数 0.759 5 略高于 temperature 为 0.5 时的 *F*1 分数（0.753 7），源于语用歧义依赖上下文指代，temperature 为 0 时的确定性确保一致捕捉逻辑关系，避免随机遗漏；temperature 为 0.5 时的随机性可能在多次运行中忽略边缘上下文，导致轻微召回率下降（temperature 为 0.5 时标准差更高）。

语言错误歧义的 $F1$ 分数在 temperature 为 0.5 下更高，因语言错误需识别异常，temperature 为 0.5 时的随机性帮助捕捉更多变异案例，提升召回率；temperature 为 0 时更精确但覆盖稍低。词汇歧义的精确率在 2 个 temperature 下几乎相同，因为词汇歧义基于词表匹配，模型词嵌入稳定，不受温度影响。整体 $F1$ 分数在语义歧义上差异较大，源于语义歧义涉及逻辑范围，temperature 为 0.5 时的随机性有助于多义解释，减少偏差；temperature 为 0 时确定但可能遗漏复杂场景。总体得出，相似性源于规则主导部分，差异性源于语境依赖部分，temperature 为 0.5 时提升探索但牺牲部分稳定性。对于 temperature 的变化如何影响 LMAdetect 在不同歧义类型上的整体性能稳定性，通过多次实验，temperature 为 0 时提供更高稳定性，标准差平均降低 1%，因为确定性避免波动；temperature 为 0.5 时引入不稳定性，但对词汇、句法歧义影响小（小于 0.5%），适合探索任务。多次运行显示，temperature 为 0.5 时，随机性影响有限，稳定性可以通过多次实验得到保障。

3.5 实验结果与分析

问题 1：数据集的单句需求样本是否在语义、场景和逻辑上完全独立？

为了回答该问题，一方面，我们收集 12 篇论文中对数据集进行独立性判断，判断数据集是否存在关联，分析结果见表 7。另一方面，我们将通过余弦相似度计算、隐含狄利克雷分配（LDA）主题建模、正则表达式结构分析来验证数据集的关联性特征，实验结果见图 7。

通过梳理论文中对数据集的描述，绝大多数论文[1,13,20,22,25-26,44-45,47-48]所使用的数据集本质上是分散的、独立的文本案例或需求语句集合，这些材料来自不同来源、不同领域、不用项目、案例/语句之间没有业务逻辑关联、没有上下文依赖关系、没有共享术语或系统背景，纯粹是为了佐证歧义现象、覆盖多样场景或验证工具泛化性而被整合到一起。仅有其少数论文[23,46]的数据来源相对明确且经过筛选验证，但同样强调领域多样性和样本间的非关联性，以确保研究结论的普适性。数据采集场景独立，所有样本均来源于分散的单一需求场景（如独立的用户查询、单句任务指令、孤立的文本诉求等），未涉及连续篇章或关联任务的串联采集。样本筛选标准明确，收集过程中以单句完整性为核心原则。数据生成无关联约束，样本之间未设置逻辑关联、场景关联或时序关联的采集规则，每个样本的选取均不依赖其他样本的内容或属性，仅因研究需要被放在一起用作歧义示例或测试样本。

Table 7 Description Analysis of Data Set

表 7 数据集的描述分析

代表性文献	数据类型	是否为孤立单句	是否提供文档级上下文
文献 [1]	经典教学与分析用例句	是	否
文献 [13]	从多个项目摘取的“典型歧义句”	是	否
文献 [20]	零散选取的独立样本，来自系统需求、学术案例、人工设计示例	是	否
文献 [22]	从 scribd.com 下载 4 个开源 SRS 纯文本文档，未做任何修改，用于独立验证歧义检测工具的效果，彼此间无任何关联	是	否
文献 [23]	专为检验而收集的独立例句	是	否
文献 [25]	正则匹配后人工确认的 38 类模式句	是	否
文献 [26]	POS 实验中逐句标注	是	否
文献 [44]	铁路、航天、气象等 6 个完全不同领域的独立抽取句	是	否
文献 [45]	咖啡机、电商、图书馆、智能家居 4 个独立系统例句	是	否
文献 [46]	3 类数据集（分类/缺陷/冲突），来源公开库/真实项目/人工构建；3 类间无关联，无共享样本	是	否
文献 [47]	无人机、医疗、缺陷追踪等 6 个独立领域例句	是	否
文献 [48]	“自然语言歧义案例”，来源英语日常/词典/语料库/真实场景	是	否

如图 7 所示，通过余弦相似度计算、LDA 主题建模、正则表达式结构分析，根据大语言模型进行数据关联性分析得出的结果，从 LDA 主题分布来看，数据集被分为 4 个主题，各主题的文本数量分布相对均衡，这表明数据集有一定的主题多样性，但 LDA 模型主要捕捉文档级的主题概率分布，如果存在篇章级关联或跨句依赖，LDA 可能会检测到更强的主题凝聚或子主题结构，但这里主题分布较为均衡且独立，暗示需求语句更多是孤立的而不是形成连贯的篇章叙事，文本的语义方向是分散的，没有围绕“单一/集中的业务场景”展开，并没有同一场景下的跨句依赖关系；从 Top15 关键词词频来看，高频词都是一些通用功能词，并没有特定业务场景的核心词，从而看出数据集都是零散的独立语句，并不存在“同一业务/用户故事下的关联文本”；从文本相似度分布来看，文本对的余弦相似度几乎全部集中在 0.2 以下，且强关联阈值（余弦相似度为 0.6）右侧的文本对数量趋近于 0，表明数据集中的文本之间内容重叠度极低，不存在“同一需求/业务场景下的关联语句”；从主题归属概率来看，各主题的归属概率较为接近，且误差棒（黑色竖线）较长，说明文本的语义归属并不聚焦，没有形成“同一业务场景下的语义聚类”，也就不存

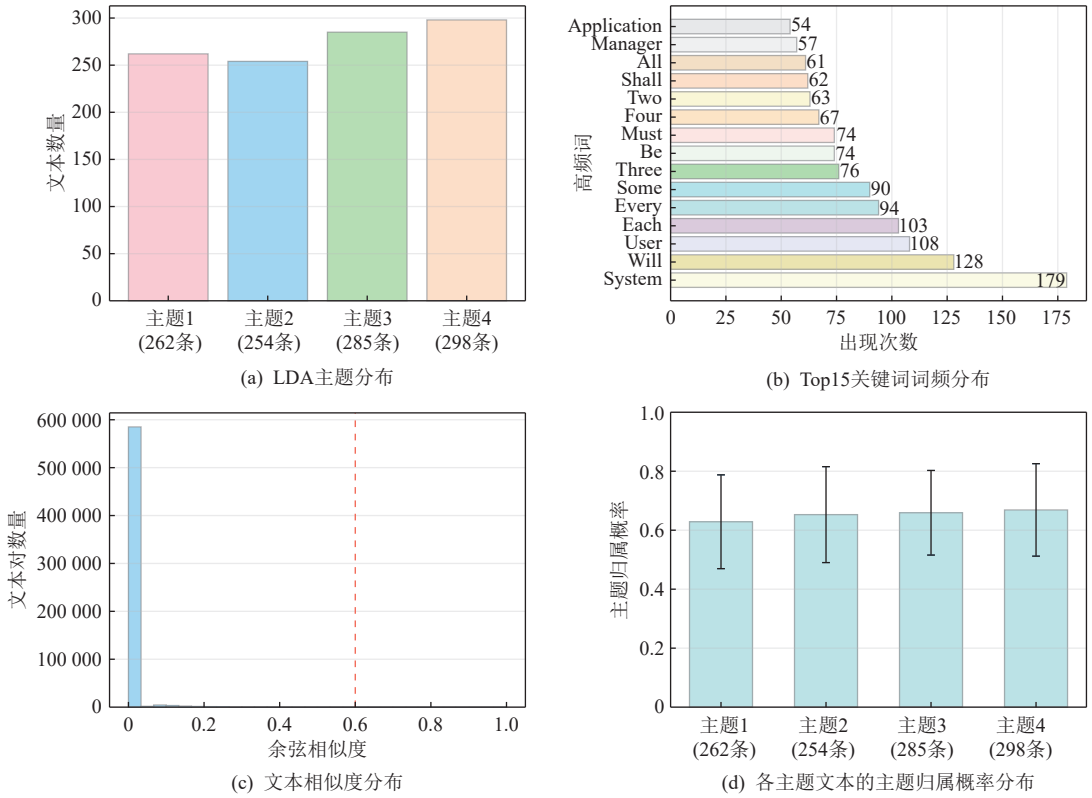

Fig. 7 Data correlation analysis chart

图 7 数据关联性分析图

在跨句的语义依赖。

综上，本研究所使用的数据集以单条需求语句为最小分析单元，数据来自公开的需求歧义检测研究与相关论文中的需求实例。这些数据在原始构建阶段通常已经过句级切分与标注，即每条需求语句均可独立表示完整意义，其上下文关系被有意剔除或未被保留。因此，该数据集并不包含不同需求语句之间的篇章级关联信息，也不涉及同一业务场景或用户故事下的跨句语义依赖。本文的研究范围与数据特性保持一致，集中于句内或局部上下文层面的歧义检测，而不涉及跨句或跨段落的语义一致性问题。从数据标注角度来看，样本均被标注为 6 种语言歧义之一，标注者依据单句语义特征进行判定。因此，本文的方法设计与实验均以单句粒度为假设前提，符合数据集构成特点。若需求文本中存在多句间的引用或逻辑依赖（如用户故事或业务操作链），则这些情形不能在现有数据集中体现。

问题 2：LMAdetect 在检测单个类型的需求歧义性能方面是否优于现有的需求歧义检测方法？

为了验证这个问题的结果，我们将 LMAdetect 与基于正则表达式和 POS 标注方法结合的 Gleich 等人[25] 的工作、基于传统方法实现的原型工具[26] 和基于 NLP 技术和规则结合的 Paska 方法[31] 进行了对比实验，并将分别仅使用规则和大语言模型在数据集上对需求进行歧义检测，计算了对应检测歧义类型的准确率、召回率、精确率、$F1$ 分数，鉴于现有方法无法对语用歧义和语言错误歧义进行检测，所以此实验针对词汇歧义、语义/范围歧义、语法歧义、句法歧义进行检测对比。对比实验的结果分别见表 8~11。

如表 8 所示，将规则和大语言模型相结合的检测效果明显高于仅使用单一策略的情况。这表明，LMAdetect 中各部分的结合，可以有效提高需求歧义检测的能力。相较于仅采用启发式规则的结果，在针对词汇歧义检测方面，LMAdetect 的准确率提升了 0.266 7，召回率提升了 0.279 8，$F1$ 分数提升了 0.177 0；在检测语义/范围歧义方面，LMAdetect 准确率提升了 0.269 1，召回率提升了 0.370 0，$F1$ 分数提升了 0.219 0；在检测语法歧义方面，LMAdetect 准确率提升了 0.012 5，召回率提升了 0.132 1，$F1$ 分数提升了 0.007 5；在检测句法歧义方面，LMAdetect 准确率提升了 0.405 5，精确率提升了 0.500 0，$F1$ 分数提升了 0.283 7。虽然启发式规则在歧义检测方法中发挥了一定作用，但其效果相对有限。启发式规则采用简单逻辑和固

Table 8 Performance Comparison of LMAdetect and Regular Expression Methods (Lexical Ambiguity)

表 8 LMAdetect 与正则表达式方法性能比较(词汇歧义)

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.607 4	0.607 4	1	0.755 8
仅使用大语言模型	0.680 9	0.768 6	0.869 2	0.815 8
LMAdetect	0.874 1	0.887 2	0.983 3	0.932 8
正则表达式	0.890 3	0.237 0	0.627 5	0.344 0

Table 9 Performance Comparison of LMAdetect and Regular Expression Methods (Semantic/Scope Ambiguity)

表 9 LMAdetect 与正则表达式方法性能比较(语义/范围歧义)

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.439 1	0.439 1	1	0.610 2
仅使用大语言模型	0.600 6	0.704 3	0.803 0	0.750 4
LMAdetect	0.708 2	0.809 1	0.850 3	0.829 2
正则表达式	0.765 3	0.651 6	0.625 0	0.638 0

Table 10 Performance Comparison of LMAdetect and POS Tagging Methods (Syntax Ambiguity)

表 10 LMAdetect 与 POS 标记方法性能比较(语法歧义)

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.825 0	0.825 0	1	0.904 1
仅使用大语言模型	0.125 0	0.312 5	0.172 4	0.222 2
LMAdetect	0.837 5	0.957 1	0.870 1	0.911 6
POS 标记	0.893 6	0.818 2	0.675 0	0.740 3

Table 11 Performance Comparison of LMAdetect and Existing Methods (Syntactic Ambiguity)

表 11 LMAdetect 与现有方法性能比较(句法歧义)

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.500 0	1	0.500 0	0.666 7
仅使用大语言模型	0.401 6	1	0.401 6	0.573 0
LMAdetect	0.905 5	0.905 5	1	0.950 4
正则表达式	0.893 9	0.181 1	0.621 6	0.280 7
POS 标记	0.703 1	0.580 2	0.598 4	0.589 2
Paska	0.763 8	0.763 8	1	0.866 4

定模式识别歧义类型,一定程度上可凭借结构化形式检测出歧义类型,但面对复杂的场景时,其能力存在局限。仅使用大语言模型的性能,在基于词汇和语义方面相较于启发式规则有明显的提升。这表明,大语言模型在歧义检测工作中可捕捉到更多细节,对需求语义有相对较深的理解。大语言模型的性能在一些层面上比启发式规则更出色,但在针对结构化的检测上不如启发式规则的检测效果。无论是规则还是大语言模型,和 LMAdetect 的整体效果相比,仍存在差距。与基于正则表达式方法检测的词汇歧义相比,LMAdeetect 的性能有了显著提升。LMAdetect 的召回率和 *F*1 分数分别为传统方法的 3.74 倍和 2.71 倍,准确率和精确率也分别提升了 0.355 8。

同样,如表 9 所示,在检测语义/范围歧义方面,LMAdetect 相比于正则表达式的方法,在召回率、精确率、*F*1 分数提高了 0.2 左右。这表明,正则表达式的方法在处理需求歧义检测任务时存在较大的局限性,而 LMAdetect 有效地克服了这些局限,取得了更好的效果。如表 10 所示,与 POS 标记方法检测的语法歧义相比,LMAdetect 的性能也有所提升,LMAdetect 的召回率、精确率和 *F*1 分数分别提高了 0.138 9、0.195 1 和 0.1713。如表 11 所示,与现有的 3 个方法检测的句法歧义相比,LMAdetect 的性能也相对更好,与正则表达式的方法相比,LMAdetect 的准确率、召回率、精确率、*F*1 分数分别提高了 0.011 6、0.724 4、0.378 4、0.669 7;与 POS 标记方法相比,LMAdetect 的准确率、召回率、精确率、*F*1 分数分别提高了 0.202 4、0.325 3、0.401 6、0.361 2;与 Paska 工具相比,LMAdetect 的准确率、召回率、*F*1 分数也分别提高了 0.1 左右。这得益于我们引入大语言模型的方式,将规则知识与语义理解能力相结合,充分利用了模型的优化效果。

综上,基于大语言模型与规则结合的方法,在对应歧义类型检测上的表现优于传统方法和基线模型。LMAdetect 结合了启发式规则和大语言模型的优势,在需求歧义检测任务中表现出色,与现有方法对比表现优异,而且在面对大语言模型方法时也能保持竞争力,这说明了其在需求歧义检测任务中的有效性与可靠性。

问题 3:LMAdetect 在检测语用歧义和语言错误歧义上的检测效果如何?

为了回答该问题,我们将分别仅使用规则、仅使用大语言模型和增加语义理解、逻辑关系识别、引用/指代识别的上下文感知能力进行检测,并计算了检测语用歧义和语言错误歧义的准确率、召回率、精确率、*F*1 分数,实验的结果见表 12~13。

从实验结果可以看出,将大语言模型与多智能体协作,在增强上下文感知能力的情况下,检测结果会显著高于仅使用规则和仅使用大语言模型的情况,LMAdetect 中各部分的结合能够有效提高需求歧义检测的性能。较仅使用规则的检测结果相比,在检测语用歧义这方面,准确率、精确率和 *F*1 分数分别提高了 0.467 4、0.479 2 和 0.572 6;在检测语言错误歧

Table 12 Performance Comparison of Detect Pragmatic Ambiguity

表 12 检测语用歧义性能比较

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.065 2	1	0.065 2	0.122 4
仅使用大语言模型	0.358 7	0.458 3	0.622 6	0.528 0
LMAdetect	0.532 6	0.960 8	0.544 4	0.695 0

Table 13 Performance Comparison of Detecting Language Error Ambiguity

表 13 检测语言错误歧义性能比较

工具/方法	准确率	召回率	精确率	*F*1 分数
仅使用规则	0.647 1	0.647 1	1	0.785 7
仅使用大语言模型	0.764 7	0.984 8	0.773 8	0.866 7
LMAdetect	0.800 0	0.971 4	0.819 3	0.888 9

义方面，准确率、召回率、*F*1 分数也分别提高了 0.152 9、0.324 3、0.1032。较仅使用大语言模型的检测结果相比，在检测语用歧义这方面，准确率和 *F*1 分数提高了约 0.17，召回率提升了 0.502 5；在检测语言错误歧义方面，准确率、精确率、*F*1 分数也分别提高了 0.035 3、0.045 5、0.022 2。

综上，实验结果表明，LMAdetect 通过增强上下文感知能力，在语义理解、逻辑关系识别和引用/指代识别方面得到了显著提升，从而能够有效检测语用歧义与语言错误歧义。该方法不仅凭借启发式规则的简单高效，还借助了大语言模型的强大识别能力，实现了需求歧义检测效果的显著提升。这一步发现进一步证实了 LMAdetect 设计的有效性。

为评估提出的多智能体协作框架和 D-S 证据理论融合的必要性，进行了基于现有基线的消融分析。消融研究通过比较完整 LMAdetect 框架与简化变体评估关键组件的贡献。“仅使用规则”变体消融了大语言模型和多智能体组件，仅依赖启发式规则检测；“仅使用大语言模型”变体消融了基于规则的算法和专家分配，模拟更简单的单大语言模型多分类方法（如通过 Chain-of-Thought 提示直接处理 6 种歧义类型）。如表 8~13 所示，消融这些组件能显著降低性能。词汇歧义下，完整 LMAdetect *F*1 分数为 0.932 8，较仅使用规则（0.755 8）提升 17.7%，较仅使用大语言模型（0.815 8）提升 11.7%；其他类型类似，平均 *F*1 分数较仅使用规则提升 34.4%，较仅使用大语言模型提升 31.6%。这证明了每个组件的独特贡献：启发式规则提供结构化检测但缺乏深层语义，导致语用和语言错误歧义召回率低；纯大语言模型提供语义理解但无智能体路由和融合，导致分类不一致，精确率下降。多智能体设计为每类型分配专家，而 D-S 融合汇总置信度解决冲突，实现 6 种歧义全面覆盖。虽然我们的框架通过多智能体协作和 D-S 融合引入复杂性，消融比较证明了其相对于简单替代方案的必要性。我们的设计优势增强语义/逻辑识别和指代解析，实现语用和语言错误歧义的优越处理。

问题 4：在多类歧义检测任务中，不同模型对结果的影响如何？

为了评估不同模型在多种歧义类型检测任务中的表现差异，我们针对通用模型和推理模型进行了全面对比分析，详见表 14~19。通过准确率、召回率、精确率、*F*1 分数等性能指标，对词汇歧义、句法歧义、语法歧义、语义/范围歧义、语用歧义和语言错误歧义 6 种类型展开评估。本节重点分析通用模型与推理模型的性能差异，并通过不同模型之间的对比，揭示它们的核心特点以及对歧义检测任务的整体影响。

1）通用模型的效果

通用模型在不同类型的歧义检测任务中表现各异，为了评估通用模型对于检测任务的效果，我们对包括国内的阿里云和 DeepSeek 以及国外的 OpenAI、Claude、Grok、Google 在内的通用模型进行了性能对比。如表 14 所示，在检测词汇歧义上，deepseek-v3-

Table 14 Performance Comparison of General Model Detecting Lexical Ambiguity and Syntactic Ambiguity

表 14 通用模型检测词汇歧义和句法歧义性能比较

通用模型	词汇歧义				句法歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwen2.5-72b-instruct	0.637 0	0.747 8	0.811 3	0.778 3	0.929 1	0.929 1	1	0.963 3
deepseek-v3-0324	0.874 1	0.887 2	0.983 3	0.932 8	0.874 1	0.905 5	1	0.950 4
gpt-4o	0.651 9	0.704 0	0.898 0	0.789 2	0.921 3	0.936 0	0.983 2	0.959 0
claude-3-5-sonnet-20241022	0.681 5	0.948 5	0.707 7	0.810 6	0.954 3	0.984 3	1	0.992 1
grok-4	0.703 7	0.931 4	0.742 2	0.826 1	0.913 4	0.913 4	1	0.954 7
gemini-2.5-pro	0.696 3	0.969 1	0.712 1	0.821 0	0.897 6	0.926 8	0.966 1	0.946 1

Table 15 Performance Comparison of General Model Detecting Syntactic Ambiguity and Semantic/Scope Ambiguity

表 15 通用模型检测语法歧义和语义/范围歧义性能比较

通用模型	语法歧义				语义/范围歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwen2.5-72b-instruct	0.787 5	0.900 0	0.863 0	0.881 1	0.524 1	0.725 5	0.653 7	0.687 7
deepseek-v3-0324	0.837 5	0.957 1	0.870 1	0.911 6	0.837 5	0.809 1	0.850 3	0.829 2
gpt-4o	0.712 5	0.876 9	0.791 7	0.832 1	0.507 1	0.535 9	0.904 0	0.672 9
claude-3-5-sonnet-20241022	0.512 5	1	0.512 5	0.677 7	0.671 4	0.918 6	0.713 9	0.803 4
grok-4	0.500 0	1	0.500 0	0.666 7	0.620 4	0.680 1	1	0.809 6
gemini-2.5-pro	0.625 0	1	0.625 0	0.769 2	0.665 7	0.955 3	0.687 1	0.799 3

Table 16 Performance Comparison of General Model Detecting Pragmatic Ambiguity and Language Error Ambiguity

表 16 通用模型检测语用歧义和语言错误歧义性能比较

通用模型	语用歧义				语言错误歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwen2.5-72b-instruct	0.358 7	0.600 0	0.471 4	0.528 0	0.705 9	0.967 7	0.722 9	0.827 6
deepseek-v3-0324	0.532 6	0.960 8	0.544 4	0.695 0	0.800 0	0.971 4	0.819 3	0.888 9
gpt-4o	0.597 8	0.597 8	1	0.748 3	0.847 1	0.947 4	0.888 9	0.917 2
claude-3-5-sonnet-20241022	0.489 1	1	0.489 1	0.656 9	0.835 3	1	0.835 3	0.910 3
grok-4	0.434 8	0.439 6	1	0.610 7	0.988 2	1	0.988 2	0.994 1
gemini-2.5-pro	0.608 7	0.811 6	0.708 9	0.756 8	0.941 2	1	0.941 2	0.969 7

Table 17 Performance Comparison of Inference Model Detecting Lexical Ambiguity and Syntactic Ambiguity

表 17 推理模型检测词汇歧义和句法歧义性能比较

推理模型	词汇歧义				句法歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwq-max-search	0.933 3	0.961 8	0.969 2	0.965 5	0.905 5	0.920 0	0.982 9	0.950 4
deepseek-reasoner	0.622 2	0.705 9	0.840 0	0.767 1	0.826 8	0.826 8	1	0.905 2
o1-preview	0.696 3	0.770 5	0.878 5	0.821 0	0.834 6	0.834 6	1	0.909 9
claude-3-7-sonnet-thinking	0.851 9	0.935 0	1	0.966 4	0.881 9	0.881 9	1	0.937 1
grok-3-reasoner	0.637 0	0.661 5	0.945 1	0.778 3	0.881 9	0.881 9	1	0.937 1
gemini-2.5-pro-thinking	0.666 7	0.803 6	0.796 5	0.800 0	0.929 1	0.929 1	1	0.963 3

Table 18 Performance Comparison of Inference Model Detecting Syntactic Ambiguity and Semantic/Scope Ambiguity

表 18 推理模型检测语法歧义和语义/范围歧义性能比较

推理模型	语法歧义				语义/范围歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwq-max-search	0.587 5	0.770 5	0.712 1	0.740 2	0.535 4	0.794 1	0.621 7	0.697 4
deepseek-reasoner	0.437 5	0.833 3	0.479 5	0.608 7	0.592 1	0.738 5	0.749 1	0.743 8
o1-preview	0.787 5	0.900 0	0.863 0	0.881 1	0.572 2	0.653 7	0.821 1	0.727 9
claude-3-7-sonnet-thinking	0.650 0	0.712 3	1	0.832 0	0.628 9	0.860 5	0.700 3	0.772 2
grok-3-reasoner	0.375 0	0.422 5	0.769 2	0.545 5	0.300 3	0.320 2	0.828 1	0.461 9
gemini-2.5-pro-thinking	0.650 0	0.866 7	0.722 2	0.787 9	0.501 4	0.688 7	0.648 4	0.667 9

0324 模型的 *F*1 分数最高，表现了强召回率和精确率，体现了它对于词汇这方面的检测能力很强。相比之下，qwen2.5-72b-instruct 和 gpt-4o 模型的性能表现中等，其 *F*1 分数值分别为 0.778 3 和 0.789 2，这表明它

Table 19 Performance Comparison of Inference Model Detecting Pragmatic Ambiguity and Language Error Ambiguity

表 19 推理模型检测语用歧义和语言错误歧义性能比较

推理模型	语用歧义				语言错误歧义			
	准确率	召回率	精确率	*F*1 分数	准确率	召回率	精确率	*F*1 分数
qwq-max-search	0.358 7	0.970 6	0.362 6	0.520 8	0.835 3	0.986 1	0.845 2	0.910 3
deepseek-reasoner	0.337 0	0.815 8	0.364 7	0.504 1	0.952 9	0.975 9	0.975 9	0.975 9
o1-preview	0.423 9	0.722 2	0.506 5	0.595 4	0.682 4	0.906 3	1	0.950 8
claude-3-7-sonnet-thinking	0.510 9	1	0.510 9	0.676 3	0.635 3	0.871 0	1	0.931 0
grok-3-reasoner	0.521 7	0.705 9	0.666 7	0.685 7	0.694 1	0.855 1	1	0.921 9
gemini-2.5-pro-thinking	0.902 2	1	0.902 2	0.948 6	0.835 3	0.876 5	0.946 7	0.910 3

们在处理细微词汇歧义时存在局限性。对于句法歧义，大多数模型都表现出色，其中 claude-3-5-sonnet-20241022 模型的高精确率和高召回率使其 *F*1 分数最高。如表 15 所示，对于语法歧义检测，deepseek-v3-0324 模型的 *F*1 分数为 0.911 6，达到最高，而 grok-4 和 claude-3-5-sonnet-20241022 模型因准确率较低，分别以 0.666 7 和 0.677 7 的 *F*1 分数表现欠佳。对于语义/范围歧义的检测任务，各模型整体表现下滑，deepseek-v3-0324 模型以 0.829 2 的 *F*1 分数取得最高，gpt-4o 模型则以 0.672 9 的 *F*1 分数略逊一筹，显示出其在复杂语义场景下处理歧义的困难。如表 16 所示，在语用歧义检测任务中，deepseek-v3-0324 和 gpt-4o 模型表现尚可，qwen2.5-72b-instruct 模型则表现一般。对于检测语言错误歧义，grok-4 模型的 *F*1 分数最高，达到 0.9941。其次是 gemini-2.5-pro 模型，充分展现了其对语言错误的强大检测能力。

实验结果表明，通用模型在检测句法歧义和语言错误歧义方面表现良好，但由于可能需要语义以及上下文的推理，通用模型对于语用和语义/范围歧义检测方面相对较差。

2）推理模型的效果

为了评估推理模型的检测效果，我们对包括国内的阿里云和 DeepSeek 以及国外的 OpenAI、Claude、Grok、Google 在内的推理模型进行了性能对比。如表 17 所示，对于词汇歧义检测，qwq-max-search 模型的 *F*1 分数最高，超过其对应的通用模型 qwq-max-search 的检测结果。claude-3-7-sonnet-thinking 模型同样表现出色，其 *F*1 分数达到了 0.966 4。在句法歧义检测任务中，多数推理模型表现优异，其中 gemini-2.5-pro-thinking 模型的 *F*1 分数最高，达到 0.963 3，与通用模型表现相当。如表 18 所示，在语法歧义检测任务中，o1-preview 模型 *F*1 分数最高，而 grok-3-reasoner 模型表现稍差一些，这说明推理增强并不能普遍提高性能。对于语义/范围歧义检测任务，qwq-max-search 模型的 *F*1 分数为 0.697 4，而 grok-3-reasoner 模型仅为 0.461 9。如表 19 所示，对于语用歧义的检测，gemini-2.5-pro-thinking 模型的 *F*1 分数达到了 0.948 6，远超其对应的通用模型的检测结果。对于语言错误歧义方面，deepseek-reasoner 的 *F*1 分数达到了 0.975 9，多数模型的检测结果都达到了很好的效果，这表明推理模型在语言错误歧义检测方面效果尤为突出。

实验结果表明，推理模型通常在特定任务中优于其通用模型，尤其是在语用和语言错误歧义方面，能够有效支持上下文理解和逻辑分析。

对比通用模型与推理模型可以发现显著差异。qwq-max-search 模型在词汇歧义和语言错误歧义检测方面明显优于 qwen2.5-72b-instruct 模型，体现了推理模型的改进优势。同样，gemini-2.5-pro-thinking 模型在语用歧义上的表现也优于 gemini-2.5-pro 模型，表明推理模型的推理增强效果在语境任务上起到重要作用。然而，grok-3-reasoner 模型在语义/范围歧义和语法歧义方面表现较差，不如 grok-4 模型，这说明推理模型并不能提升各方面性能，可能会受到模型特定限制或训练偏差的影响。deepseek-v3-0324 模型在通用任务中表现稳定，尤其擅长处理词汇歧义以及句法歧义。而 deepseek-reasoner 模型在语言错误歧义检测方面表现优异，但在检测语用歧义上稍显不佳。OpenAI 的 gpt-4o 模型和 o1-preview 模型则展现出均衡性能，其中 o1-preview 模型在检测语法错误和语义/范围歧义方面有所提升。Claude 模型在句法和语言错误歧义检测方面表现良好，通过推理模型的推理增强在语用任务中显著提升了性能。Grok 模型在检测语言错误歧义方面表现优异，但在语义和语法歧义处理上面存在不足，特别是在推理模型检测中。Google 模型通过推理在语用歧义检测上取得显著进步，使其在上下文任务中具备竞争力。

综上，LMAdetect 在检测多种歧义类型方面具有可行性。通用模型在句法歧义和语言错误歧义检测

任务中表现优异，但在检测语用以及语义/范围歧义方面略显不足。此外，推理模型通常能通过逻辑推理和语境推理相关的语用和语言错误歧义检测任务表现突出。然而，不同模型在特定任务中的推理增强效果参差不齐，部分模型在某些歧义类型检测上表现欠佳。这些发现表明，模型设计与训练策略对处理多样化歧义类型具有重要影响。不同模型的差异反映了它们在架构、训练数据和优化任务方面的不同选择，这些因素共同作用于其在复杂语境中的表现。

为进一步探究不同模型在特定歧义类型下性能差异背后的原因，本文选取了2条具有代表性的需求实例进行分析，并对比了LMAdetect、推理模型与通用大语言模型的输出结果。

实例1：The system shall notify users。该需求语句的正确标注为“语用歧义”，因为其缺乏明确的动作方式（如通知手段、渠道或时机）。在检测结果中，通用大语言模型常将该语句判定为“无歧义”，理由是其语法和语义结构完整；而推理模型能够结合上下文推理得出“notify users”存在多种解释空间，因而更倾向于正确输出“语用歧义”。LMAdetect在此例中输出“语用歧义”，其优势在于结合规则检测中对“未指定执行细节的动作”所定义的启发式条件，从而避免了模型仅凭常识推断而低估歧义风险的问题。该结果表明，推理模型和基于规则的协作框架更擅长处理依赖上下文和多义行为描述的语用歧义。

实例2：All files in the system are secure。该需求语句的正确标注为“语义/范围歧义”，因为“All”是否包含临时文件、日志文件或外部导入文件存在多种可能解释。通用模型在该任务中给出正确判断，但推理模型往往未能识别该潜在问题，而直接判定为“无歧义”。追踪推理模型的输出可发现，它依赖上下文常识（认为“files”应默认为系统内文件集合），因而忽视了边界范围的不明确性。相较之下，LMAdetect能够通过规则定义的“量化词范围检测”模块捕捉到不确定的边界条件，并结合大语言模型的语义分析完成正确分类。该结果表明，推理模型虽在语用类任务中优势明显，但在需严谨逻辑边界推理的语义/范围歧义任务中仍存在不足，而LMAdetect的优势在于规则与模型的互补作用。

通过以上实例分析可以看出，推理模型和多智能体协作框架在处理不同歧义类型时呈现互补性。推理增强提升了语用层面的上下文理解能力，而规则与融合机制则能够弥补其在范围逻辑和精确语义判断方面的不足。这一性质分析与前述量化结果相互印证，进一步验证了本文所提出方法的有效性与适用性。

3.6 讨论与局限性

我们基于实验结果回答了这4个研究问题。问题1数据集以单句形式呈现，在语义、场景和逻辑上是完全独立的。问题2 LMAdetect能够有效的检测单个类型的需求歧义，但在若干方面仍然存在不足，需要持续改进完善。问题3 LMAdetect能够有效检测语用歧义和语言错误歧义，但其针对语用歧义检测方法还有待加强，需持续优化。问题4针对通用模型和推理模型的检测效果分析，通用模型在句法和语言错误检测方面表现出色，而推理模型在语用和语言错误任务中提高了性能，不同模型和歧义类型的效果各不相同。

基于对研究问题答案的分析，我们认为LMAdetect能在一定程度上解决前言部分提到的挑战。同时，LMAdetect为需求歧义提供了有效的检测。这有助于项目开发人员之间和在开发期间减少一定的歧义冲突以及彼此之间的沟通交流。虽然LMAdetect在识别歧义方面是有效的，但对于歧义类型的细粒度分类需要进一步改进，特别是对于复杂的、上下文相关的歧义。未来改进应着重于提升语义和语用歧义情境下的上下文理解能力，从而增强整体检测性能。

针对LMAdetect进行的实验，使用deepseek-v3-0324模型在1 192条数据集上取得了一定的效果，但研究期间也暴露出诸多局限性和可改进之处。一方面关于模型的选择，该模型可能并非最佳选择。即使模型具备一定的语义分析能力，但既未选用参数规模更大的最新模型，也缺乏针对特定任务的深度微调优势。这限制了LMAdetect的整体性能上限，尤其是在需要进行语境理解或知识库知识的复杂语用和语义歧义识别时表现欠佳。此外，实验设置本身也存在局限性，固定配置的实验环境可能无法充分反映在不同硬件环境或超参数调优条件下的性能变化，这是源于缺乏多样化的计算资源配置以及交叉验证设计。在数据集范围方面，尽管1 192条需求具有一定多样性，但其主要源自学术文献和GitHub仓库，这也许会带来选择性偏差。若数据集覆盖范围不足或分布不均，可能引起规则对新数据集的泛化能力受限。LMAdetect不涉及自定义数据集的预训练，而是依赖现成大语言模型（如通过中间商访问DeepSeek）的实时推理，因此本质上避免了训练阶段的直接数据泄露风险。但是大语言模型本身也存在

一定的局限性。LMAdetect借助大语言模型进行语义理解和分类，而大语言模型可能受到训练数据偏差或“幻觉”现象的干扰，尤其是在处理领域特定的术语及复杂任务时，可能产生不准确的分类结果。此外，LMAdetect基于多智能体协作框架，虽然提高了检测鲁棒性，但其计算复杂性较高。当处理大规模需求文档时，这可能导致检测时间显著增加，从而限制其在实时应用场景中的实用性。

3.7 有效性威胁

LMAdetect方法的有效性可能面临以下威胁。首先，内部有效性威胁，主要体现在实验数据的清洗与预处理过程中可能出现的偏差。尽管通过多种学术数据库和GitHub仓库收集到1 192条需求数据，但数据选择和清洗过程也许会引入人为选择偏差，进而影响实验结果的代表性。此外，deepseek-v3-0324模型的选择及参数设置（如temperature为0.5）可能对检测结果产生影响，若未充分研究其他模型与参数的组合形式，可能低估了方法性能的上限。其次，外部有效性威胁源于数据集的局限性。现有的实验仅依据特定学术文献和公开数据集，未能涵盖到各个行业或领域的需求文档类型，限制了方法在不同上下文中的普适性。此外，实验环境可能与实际工业环境存在差异，如资源限制或硬件差异，这可能影响方法的可扩展性。最后，构造有效性威胁可能源于对歧义类型的定义和检测条件简化的结果。尽管表1列出了6种歧义类型及其规则，但实际需求中的歧义可能涉及更复杂的交互或未定义的模式，现有的分类框架可能未能完全捕捉这些细微差别，使得检测指标可能高估了方法的实际能力。然而，LMAdetect需要通过提示词提供用户数据（如需求语句）进行大语言模型分析，尽管我们的设计最小化了此类暴露，但数据可能在操作途径中存在潜在泄露。针对工具本身，我们的框架采用严格的无存储策略，用户输入仅在内存中临时处理，分析后立即丢弃，无日志或数据库持久化。这确保了工具本身无泄露，因为数据从未在推理会话之外本地留存。针对中间服务商Chatfire，在实验途中，调用的是中间商Chatfire，由于缺乏透明度，ChatFire的数据处理存在不确定性。作为中间商，它可能转发用户提示和数据到后端大语言模型（如DeepSeek），但没有明确承诺不存储、加密或保护数据，这增加了潜在泄露风险。针对大语言模型提供商DeepSeek，在调用完Chatfire后把请求给DeepSeek，DeepSeek在公开文档中明确声明，用户输入的数据不会用于再训练，且遵循行业标准的隐私保护策略，仅在必要情况下保留短期日志用于服务监控，这在原则上降低了数据泄露的可能性。但需要注意，具体实现上的透明度有限，用户仍需依赖提供商的承诺。因此，尽管DeepSeek提供了书面保证，但其是否完全消除泄露风险的能力仍有待实践验证。

4 总　　结

SRS中的歧义是自然语言需求中常见的质量问题，可能导致误解和项目失败。为此，我们在本文中提出了一种基于大语言模型协调多智能体的需求歧义检测方法LMAdetect。该方法结合了启发式规则和大语言模型，针对SRS中的语法和句法歧义进行自动化检测和分析。根据前期对实际需求文档中歧义问题的调研结果，LMAdetect通过NLP技术和启发式规则，从需求文本中提取关键结构特征生成候选歧义区域，并利用大语言模型的语义理解能力对歧义进行精准识别和分类，从而提供有效的检测结果。

虽然LMAdetect的设计不针对特定领域的需求文档，但其原型实现目前主要处理英语编写的SRS文档。LMAdetect的文本分析部分（包括分词、词性标注和句法分析）依赖于英语的语言特性，因此若未来需要扩展到其他语言（如中文或德文），则需重新实现相应的NLP模块。未来的研究将考虑收集包含上下文依赖关系的数据集，如用户故事级需求集合或跨句关联的需求片段，以进一步研究篇章级语义一致性下的歧义检测问题。例如，一个用户故事往往对应多个功能性需求，这些需求在语义和逻辑上存在明显关联。若能综合分析此类关联，LMAdetect的上下文感知能力可从局部扩展到篇章级，从而更贴近真实软件开发场景。在此基础上探索更多的指导机制，未来可利用本文提出的关键结构分析方法，扩展到其他需求质量问题的检测，如不完整条件或不精确术语，以进一步提升需求工程的自动化水平。研究本文方法旨在提高在其他多智能体应用领域的有效性，同时探索大型语言模型在需求歧义修复中的实际应用。

作者贡献声明：高俊涛提出研究思路，指导论文，对论文整体框架与核心内容提出修改意见；刘芳设计研究方案，完成实验设计并撰写论文；杨溢龙对研究方案进行针对性指导，参与论文审阅并提出修改意见。

参 考 文 献

[1] Berry D M, Kamsties E, Krieger M M. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity[M]. Waterloo: University of Waterloo, 2003

[2] Kamsties E, Paech B. Taming ambiguity in natural language requirements[C]//Proc of the 13th Int Conf on Software and Systems Engineering and Applications. Paris, France: Center for Mastering Systems & Software (CMSL) of CNAM, 2000: 1−8

[3] Doe J. Recommended practice for software requirements specifications (IEEE)[S]. New York: Midori, 2011-12-29

[4] Jackson M. Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices[M]. New York: ACM Press/Addison-Wesley, 1995

[5] Beer A, Junker M, Femmer H, Felderer M. Initial investigations on the influence of requirement smells on test-case design[C]//Proc of 2017 IEEE 25th Int Requirements Engineering Conf Workshops (REW). Piscataway, NJ: IEEE, 2017: 323−326

[6] Shang Yanran, Jin Shui. On the ambiguity caused by the language structure in the legal case[C]//Proc of the 2018 3rd Int Conf on Education, E-learning and Management Technology (EEMT' 2018). Paris, France: Atlantis, 2018: 499−504

[7] Wang Chunhui, Jin Zhi, Zhao Haiyan, et al. An approach for improving the requirements quality of user stories[J]. Journal of Computer Research and Development, 2021, 58(4): 731−748 (in Chinese)
(王春晖, 金芝, 赵海燕, 等. 一种用户故事需求质量提升方法[J]. 计算机研究与发展, 2021, 58(4): 731−748)

[8] Rupp C, Goetz R. Linguistic methods of requirements-engineering (NLP)[C]//Proc of the European Software Process Improvement Conf (EuroSPI). Copenhagen Denmark: Copenhagen Business School, 2000: 7−11

[9] Gause D C, Weinberg G M. Exploring Requirements: Quality Before Design[M]. New York: Dorset House, 1989

[10] Heitmeyer C L, Jeffords R D, Labaw B G. Automated consistency checking of requirements specifications[J]. ACM Transactions on Software Engineering and Methodology, 1996, 5(3): 231−261

[11] Boehm B W. Software engineering - as it is[C]//Proc of the Int Conf on Software Engineering. Piscataway, NJ: IEEE, 1979: 11−21

[12] Robertson S, Robertson J. Mastering the Requirements Process: Getting Requirements Right[M]. Boston: Addison-Wesley, 2012

[13] Berry D M, Kamsties E. Ambiguity in Requirements Specification[M]//Perspectives on Software Requirements. Boston, MA: Springer US, 2004: 7−44

[14] Xu H R, Kim Y J, Sharaf A, et al. A paradigm shift in machine translation: Boosting translation performance of large language models[C]//Proc of the 12th Int Conf on Learning Representations. OpenReview. net, 2024: 1−21

[15] Yao B W, Jiang M, Bobinac T, et al. Benchmarking machine translation with cultural awareness[C]//Proc of the Findings of the Association for Computational Linguistics: EMNLP 2024. Miami: ACL, 2024: 13078−13096

[16] Shi Y C, Ma H H, Zhong W L, et al. ChatGraph: Interpretable text classification by converting ChatGPT knowledge to graphs[C]//Proc of the 2023 IEEE Int Conf on Data Mining Workshops. Piscataway, NJ: IEEE, 2023: 515−520

[17] Kurisinkel L J, Chen N F. LLM based multi-document summarization exploiting main-event biased monotone submodular content extraction[J]. arXiv preprint arXiv: 2310.03414, 2023

[18] Xu Zimao, Jiang Yanjie, Zhang Yuxia, et al. Large-language-model-based decomposition of long methods[J]. Journal of Software, 2025, 36(6): 2501−2514 (in Chinese)
(徐子懋, 姜艳杰, 张宇霞, 等. 基于大语言模型的长方法分解[J]. 软件学报, 2025, 36(6): 2501−2514)

[19] Vinay S, Aithal S, Desai P. An approach towards automation of requirements analysis[C]//Proc of the Int MultiConf of Engineers and Computer Scientists 2009. Hong Kong: IMECS, 2009: 1080−1085

[20] Kamsties E, Berry D M, Paech B, et al. Detecting ambiguities in requirements documents using inspections[C]//Proc of the 1st Workshop on Inspection in Software Engineering (WISE'01). Hamilton, Canada: Software Quality Research Lab, McMaster University, 2001: 68−80

[21] Fabbrini F, Fusani M, Gnesi S, et al. An automatic quality evaluation for natural language requirements[C]//Proc of the 7th Int Workshop on Requirements Engineering: Foundation for Software Quality REFSQ. Interlaken, Switzerland: REFSQ Workshop, 2001: 4−5

[22] Nigam A, Arya N, Nigam B, et al. Tool for automatic discovery of ambiguity in requirements[J]. International Journal of Computer Science Issues, 2012, 9(5): 350−356

[23] Tjong S F. Avoiding Ambiguity in Requirements Specifications[D]. Semenyih, Selangor Darul Ehsan, Malaysia: University of Nottingham, 2008

[24] Rosadini B, Ferrari A, Gori G, et al. Using NLP to detect requirements defects: An industrial experience in the railway domain[C]//Proc of the Int Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ). Cham: Springer, 2017: 344−360

[25] Gleich B, Creighton O, Kof L. Ambiguity detection: Towards a tool explaining ambiguity sources[C]//Proc of the 18th IEEE Int Requirements Engineering Conf (RE'10). Berlin: Springer, 2010: 218−232

[26] Sabriye A O J, Wan Z W M N. An approach for detecting syntax and syntactic ambiguity in software requirement specification[J]. Journal of Theoretical and Applied Information Technology, 2018, 96(8): 2275−2284

[27] Ferrari A, Esuli A. An NLP approach for cross-domain ambiguity detection in requirements engineering[J]. Automated Software Engineering, 2019, 26(3): 559−598

[28] Gentile A L, Coden A R, Lourentzou I, et al. Identifying ambiguity in semantic resources: U. S. Patent 11, 379, 669[P]. 2022-7-5

[29] Šenkýř D, Kroha P. Patterns of ambiguity in textual requirements specification[C]//Proc of the World Conf on Information Systems and Technologies (WorldCIST'19). Cham: Springer, 2019: 886−895

[30] Liu Y, Medlar A, Głowacka D. Lexical ambiguity detection in professional discourse[J]. Information Processing & Management,

2022, 59(5): 103000

[31] Veizaga A, Shin S Y, Briand L C. Automated smell detection and recommendation in natural language requirements[J]. IEEE Transactions on Software Engineering, 2024, 50(4): 695−720

[32] Levy R, Andrew G. Tregex and tsurgeon: Tools for querying and manipulating tree data structures[C]//Proc of the 5th Int Conf on Language Resources and Evaluation. Genoa, Italy: European Language Resources Association, 2006: 2231−2234

[33] Bang Y, Cahyawijaya S, Lee N, et al. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity[J]. arXiv preprint arXiv: 2302.04023, 2023

[34] Liang P, Bommasani R, Lee T, et al. Holistic evaluation of language models[J]. arXiv preprint arXiv: 2211.09110, 2022

[35] Chen Xuanting, Ye Junjie, Zu Can, et al. Robustness of GPT large language models on natural language processing tasks[J]. Journal of Computer Research and Development, 2024, 61(5): 1128−1142 (in Chinese)
(陈炫婷, 叶俊杰, 祖璨, 等. GPT 系列大语言模型在自然语言处理任务中的鲁棒性[J]. 计算机研究与发展, 2024, 61(5): 1128−1142)

[36] Liu Chao, Bao Xuanlin, Zhang Hongyu, et al. Improving ChatGPT prompt for code generation[J]. arXiv preprint arXiv: 2305.08360, 2023

[37] Toy J, Tabor P, MacAdam J. Metacognition is all you need? Using introspection in generative agents to improve goal-directed behavior[J]. arXiv preprint arXiv: 2401.10910, 2024

[38] Jiang Xun, Li Feng, Zhao Han, et al. Long term memory: The foundation of AI self-evolution[J]. arXiv preprint arXiv: 2410.15665, 2024

[39] Qian Chen, Liu Wei, Liu Hongzhang, et al. Chatdev: Communicative agents for software development[C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Bangkok: ACL, 2024: 15174−15186

[40] Wang Boxin, Chen Weixin, Pei Hengzhi, et al. Decodingtrust: A comprehensive assessment of trustworthiness in GPT models[C]// Proc of the 37th Annual Conf on Neural Information Processing Systems. New Orleans, LA, USA, Neural Information Processing Systems Foundation, 2023:1−95

[41] Zhang Yue, Li Yafu, Cui Leyang, et al. Siren's song in the AI ocean: A survey on hallucination in large language models[J]. Computational Linguistics, 5114: 1−46

[42] Chang Yupeng, Wang Xu, Wang Jindong, et al. A survey on evaluation of large language models[J]. ACM Transactions on Intelligent Systems and Technology, 2024, 15(3): 1−45

[43] Guo Biyang, Zhang Xin, Wang Ziyuan, et al. How close is ChatGPT to human experts? Comparison corpus, evaluation, and detection[J]. arXiv preprint arXiv: 2301.07597, 2023

[44] Abualhaija S, Fucci D, Dalpiaz F, et al. Preface: 3rd workshop on natural language processing for requirements engineering (NLP4RE'20)[C]//Joint Proc of REFSQ−2020 Workshops, Doctoral Symp, Live Studies Track, and Poster Track. Pisa, Italy: CEUR-WS. org, 2020: 1−3

[45] Fantechi A, Gnesi S, Semini L. Rule-based NLP vs ChatGPT in ambiguity detection, a preliminary study[C/OL]//Proc of REFSQ Workshops. 2023: 1−10[2026-02-15]. https://ceur-ws.org/Vol3378/NLP4RE-paper1.pdf

[46] Fazelnia M, Koscinski V, Herzog S, et al. Lessons from the use of natural language inference (NLI) in requirements engineering tasks[C]//Proc of 2024 IEEE 32nd Int Requirements Engineering Conf (RE). Picataway, NJ: IEEE, 2024: 103−115

[47] Saleem S, Asim M N, Dengel A. PassionNet: An innovative framework for duplicate and conflicting requirements identification[J]. Expert Systems with Applications, 293: 128684

[48] Wasow T, Perfors A, Beaver D. The puzzle of ambiguity[J]. Morphology and the web of grammar: Essays in memory of Steven G. Lapointe. Stanford, CA: CSLI Publications, 2005: 265−282. https://web.stanford.edu/~wasow/Lapointe.pdf

[49] Pomian D, Bellur A, Dilhara M, et al. Together we go further: LLMs and IDE static analysis for extract method refactoring[J]. arXiv preprint arXiv: 2401.15298, 2024

Gao Juntao, born in 1979. PhD, associate professor. Member of CCF. His main research interests include data analysis, process mining, business process management, and software engineering.

高俊涛, 1979 年生。博士, 副教授。CCF 会员。主要研究方向为数据分析、过程挖掘、业务过程管理、软件工程。

Liu Fang, born in 1999. Master. Her main research interests include static analysis of requirements, and ambiguity detection and resolution.

刘　芳, 1999 年生。硕士。主要研究方向为需求的静态分析、歧义检测与修复。

Yang Yilong, born in 1988. PhD, associate professor. Member of CCF. His main research interests include intelligent requirements engineering, intelligent software engineering, and intelligent system engineering.

杨溢龙, 1988 年生。博士, 副教授。CCF 会员。主要研究方向为智能需求工程、智能软件工程、智能系统工程。