doi:10.3772/j.issn.2095-915x.2026.04.006

基于检索增强和测试时扩展的知识图谱问答

程环宇[1] 钟秋圆[1] 胡皓[1] 顾彬[1] 相增辉[1] 陈轩[1] 李锦辉[1] 黄祥[2] 胡伟[2]

1. 国网江苏省电力有限公司 南京 210024
2. 南京大学 计算机学院 南京 210023

开放科学（资源服务）标识码（OSID）

摘要：[目的 / 意义] 针对基于 Agent 的知识图谱问答中轨迹数据难构造、样例质量要求高、模型容错率低三个问题，提出改进方法。[方法 / 过程] 提出 RaTS 方法，融合检索增强与测试时扩展。自动将查询转换为轨迹数据，检索最相似样例辅助推理，在每步并行采样多个候选并择优选择。[结果 / 结论] 在 GrailQA 和 GraphQ 数据集上 F1 分别达到 65.8 和 54.3，相比之前最优方法提升 5.3 和 3.5。方法有效提升了复杂推理的准确性与鲁棒性。

关键词：知识图谱问答；大语言模型；检索增强；测试时扩展

中图分类号：G35；TH212

Retrieval-augmented and Test-time Scaling Knowledge Graph Question Answering

CHENG Huanyu[1] ZHONG Qiuyuan[1] HU Hao[1] GU Bin[1)] XIANG Zenghui[1] CHEN Xuan[1] LI Jinhui[1] HUANG Xiang[2] HU Wei[2]

1. State Grid Jiangsu Electric Power Co., Ltd., Nanjing 210024, China
2. School of Computer Science, Nanjing University, Nanjing 210023, China

Abstract: [Objective/Significance] This paper addresses three challenges in agent-based Knowledge Graph Question Answering (KGQA): difficulty in constructing trajectory data, high requirements for example quality, and low model fault tolerance. [Methods/Process] We propose RaTS, which integrates retrieval augmentation and test-time scaling. It automatically converts queries into trajectory data, retrieves the most similar example to assist reasoning, and samples multiple candidates at each step with a

基金项目 国网江苏省电力有限公司科技项目“基于电力信通客服知识图谱的大模型指令微调关键技术研究”（J2025065）。

作者简介 程环宇（1995-），硕士，工程师，主要研究方向为人工智能；钟秋圆（1997-），硕士，工程师，主要研究方向为电力人工智能技术；胡皓（1996-），硕士，工程师，主要研究方向为电力信息通信技术；顾彬（1983-），博士，高级工程师，主要研究方向为电力信息通信技术；相增辉（1990-），硕士，工程师，主要研究方向为电力信息通信技术；陈轩（1992-），硕士，工程师，主要研究方向为电力信息通信技术；李锦辉（1997-），本科，助理工程师，主要研究方向为电力信息通信技术；黄祥（1998-），博士研究生，主要研究方向为知识库问答、大语言模型；胡伟（1982-），通信作者，教授，主要研究方向为知识图谱、数据库、智能软件，E-mail：whu@nju.edu.cn。

引用格式 程环宇 , 钟秋圆 , 胡皓 , 等 . 基于检索增强和测试时扩展的知识图谱问答 [J]. 情报工程 , 2026, 12(4): 65-79.

doi:10.3772/j.issn.2095-915x.2026.04.006

selection mechanism. [Result/Conclusions] RaTS achieves F1 scores of 65.8 and 54.3 on GrailQA and GraphQ, outperforming previous state-of-the-art methods by 5.3 and 3.5 points. The method effectively improves accuracy and robustness for complex KGQA tasks.

Keywords: Knowledge Graph Question Answering; Large Language Model; Retrieval-augmented Generation; Test-time Scaling

引言

知识图谱（Knowledge Graph，KG）是以（主体，关系，客体）三元组形式存储事实的结构化知识库。诸如 Freebase、DBpedia 和 Wikidata 等大规模 KG 已经被构建出来，作为人类知识的数字化结晶，它承载着数十亿实体与关系的结构化网络，为机器理解世界提供了重要的语义基础设施。为了与 KG 中大量的结构化知识进行交互，通常需要使用到 SPARQL（SPARQL Protocol and RDF Query Language）语言。而 SPARQL 存在语法复杂、学习成本较高等问题。因此，如何让普通用户可以轻松利用 KG 中的知识成为一个亟待解决的关键问题。在此需求背景之下，知识图谱问答（Knowledge Graph Question Answering，KGQA）任务应运而生。

虽然 KGQA 具有广泛的应用价值，但也面临诸多挑战。KGQA 作为一个典型的需要进行多步推理的任务，简单地端到端方法（例如端到端微调、上下文学习）并不能很好地应对求解步骤较多的复杂问题。针对此问题，一些工作将 KGQA 建模成智能体（Agent）推理的方式，逐步调用大模型和环境进行交互，实现了可解释的分步求解，尤其适用于处理复杂推理任务。但是这类方法目前还存在一些固有缺陷：

首先，由于提示中需要给出求解的轨迹数据，而现有数据集通常只提供一个 SPARQL 或者 S 表达式的完整查询，没有提供逐步求解的轨迹数据，并且书写轨迹数据难度较大，这给检索增强技术的应用带来了阻碍。

其次，基于 Agent 的方法因为需要多次调用大语言模型，提供的示例也需要为多轮交互的冗长轨迹数据。之前的方法通常只使用一个样例，并且对于所有问题都使用同一个样例，这就对样例的质量、样例与待求解问题的相关性提出了很高的要求。对于一些复杂的场景，Agent 可采用的函数空间可能非常大，一个例子中很难囊括所有可用的函数，对于需要用到罕见函数的例子，模型没有充足的参考信息来学习如何调用。

最后，现有的基于 Agent 的方法通常每个步骤只尝试一次，KGQA 任务环境复杂，推理步骤多，对大语言模型的能力提出了很高的要求。基于 Agent 的方法的求解过程需要经过多个步骤，一旦中间步骤出错，后续步骤的求解就没有意义，造成推理资源的浪费。

综合来看，现有的基于 Agent 的 KGQA 方法在轨迹数据构建、样例选择及推理鲁棒性方面仍存在明显不足。因此，本文重点探讨如何在缺乏轨迹数据的情况下构造可用于多步推理的示例，如何提高示例选择的相关性，以及如何在推理过程中增强模型的容错性与稳定性。这些问题的解决将为实现更高效、更可靠的

KGQA 系统提供新的思路与方向。

1 相关工作

1.1 大语言模型

大语言模型（LLM）[1] 是近年来自然语言处理领域的核心技术突破，是基于海量文本数据训练、具有千亿级参数的深度神经网络模型。其核心特征在于通过自监督学习机制，从大规模语料库中自动抽取语言规律和世界知识，最终实现文本生成、语义理解、逻辑推理等多任务泛化能力。大语言模型的发展经历了三个阶段的技术跃迁：早期统计语言模型以 n-gram 为核心，依赖局部词频统计但缺乏语义表征能力；2013 年后神经网络语言模型（如 Word2Vec、ELMO）引入分布式表示，实现了词向量空间建模；2017 年 Transformer[2] 架构的提出彻底改变了技术路线，其多头注意力机制突破了序列建模的并行化瓶颈。2018 年 GPT[3] 和 BERT[4] 的问世标志着预训练时代的开启。前者通过自回归生成式预训练展现强大的文本生成能力，后者利用双向掩码预测优化语义理解性能。此后模型规模呈指数级增长，从 GPT-3.5[5] 的 1750 亿参数到 GPT-4[6]，模型性能随参数扩张呈现显著涌现特性。

1.2 检索增强

大语言模型凭借其类人的文本生成和自然语言理解能力，已经在人工智能领域掀起了一场革命。然而，这些模型依赖于静态训练数据，限制了它们应对动态、实时查询的能力，往往导致信息过时或不准确。针对这样的痛点，检索增强生成（Retrieval-Augmented Generation，RAG）[7-8] 应运而生，其从外部知识源获取额外信息辅助模型生成内容。具体而言，RAG 的核心思想可概括为“检索 - 融合 - 生成”三阶段：首先从结构化数据库或非结构化文档库中检索与输入相关的上下文，继而将检索结果与原始提示融合，最终引导语言模型生成具备事实准确性的内容。相较于传统大模型，RAG 具有三大优势：其一，通过实时检索保障知识的时效性，特别适用于电力、金融、医疗等动态领域；其二，显式引用外部证据提升生成结果的可解释性，降低幻觉（Hallucination）风险；其三，支持领域定制化，仅需更新检索库而无需重新训练模型。当前技术挑战集中于检索与生成的协同优化，例如噪声过滤、多跳推理适配，以及检索效率与生成质量的平衡。

1.3 测试时扩展

测试时扩展（Test-Time Scaling，TTS）[9] 作为机器学习领域应对动态环境挑战的新兴研究方向，其核心在于突破传统机器学习模型训练 - 推理分离的固定范式。在 OpenAI o 系列模型 [10] 和 Deepseek-R1[11] 等推理模型推出之前，大部分大模型相关的工作通常会在模型参数和预训练规模（数据集、计算量）上进行扩展以期望获得不断增长的性能。一般来说，预训练投入越大，最终的模型效果就会越好。然而，模型的参数和预训练的规模不可能无限地扩张，在训练成本极高的情况下继续在预训练阶段扩展规模并不是一个经济的做法。增加训练时计算的高昂成本促使人们转而关注另一种方法，即 TTS。与不断增加预训练成本不同，TTS 允

doi:10.3772/j.issn.2095-915x.2026.04.006

许模型在推理过程中增加推理计算开销。近年来，随着 OpenAI o 系列模型和 Deepseek-R1 模型等在各类复杂推理任务上展现出卓越的性能，推理模型逐渐成为大语言模型研究中的一种新范式。不同于直接输出答案，推理模型在回答问题前，首先对问题进行分析，步骤拆解，制定方案，然后逐步进行求解，并在求解结果中进行验证、修改、润色等，迭代优化最终给出一个综合考虑后的答案，尤其适用于数学、代码、创意写作等需要进行多步推理、任务规划的场景。

1.4 基于大模型的KGQA

随着大语言模型在各类自然语言处理任务上取得的惊人进展，使用大语言模型作为 KGQA 的基座逐渐成为一种主流范式。整体来看，基于大模型的 KGQA 主要有两种路线，一种是基于上下文学习的方法，另一种是基于 Agent 的方法。基于上下文学习的方法 [12-14] 给大模型提供若干个（问题，逻辑形式）对，以及待求解的问题，希望大模型能够直接输出最终的结果。这种做法的优点是简单灵活，但这是一种端到端的黑盒方法，缺少可解释性，对于具有复杂结构的问题（例如包含多跳多约束结构，包含复杂函数运算等），期望大模型能够仅根据几个样例就能直接输出最终结果是不太现实的，对基座模型的能力也有很高的要求。基于 Agent 的方法 [15-16] 将 KGQA 建模成逐步求解的形式，将问题拆解成多个步骤，每次在当前基础上向前探索一次或者解决问题中的某一部分。这类方法的处理方式符合直觉，具有可解释性，并且和人类探索 KG、书写查询、执行查询、检验查询的流程类似。逐步求解使得一个复杂的推理问题被拆解为多个相对简单的子问题，降低了模型要处理的任务的难度。考虑到上述因素，本文采用基于 Agent 的方法求解 KGQA 任务，并结合 RAG 和 TTS 进一步提升方法的效果。

2 方法介绍

为了解决引言中提到的核心难点，本文提出一种融合检索增强与测试时扩展机制的知识图谱问答方法（Retrieval-augmented with Test-time Scaling，RaTS）。针对轨迹数据书写困难的难点，本文提出了将查询自动改写为逐步求解的轨迹数据的方式，首先将查询转化成逐步求解的 PyQL（Pythonic Query Language for SPARQL）形式 [18]，然后将每个步骤交互动作转化成自然语言以构造轨迹数据。针对样例质量要求高的难点，本文在 Agent 方法中引入 RAG 的方式，从训练集数据中抽取和待求解问题最相似的例子作为提示中的样例。针对模型容错率低的难点，本文引入了 TTS，在每步推理时给模型多次机会输出多个答案，然后从中选择最可能是正确答案的结果，在增强容错的同时提升得到正确结果的概率。如图 1 所示，RaTS 方法将 KGQA 建模成一个多步问答的形式，以 Agent 范式为整体框架，融入 RAG 和 TTS，以进一步增强性能。

近期的 Agentic RAG 智能体框架通常将大语言模型与外部检索模块结合，通过动态调用外部知识文档来增强生成的事实性与时效性。与 Agentic RAG 不同，本文提出的 RaTS 框架并非侧重知识补全，而是聚焦于推理过程的可解释性与鲁棒性提升。在检索层面，RaTS 检

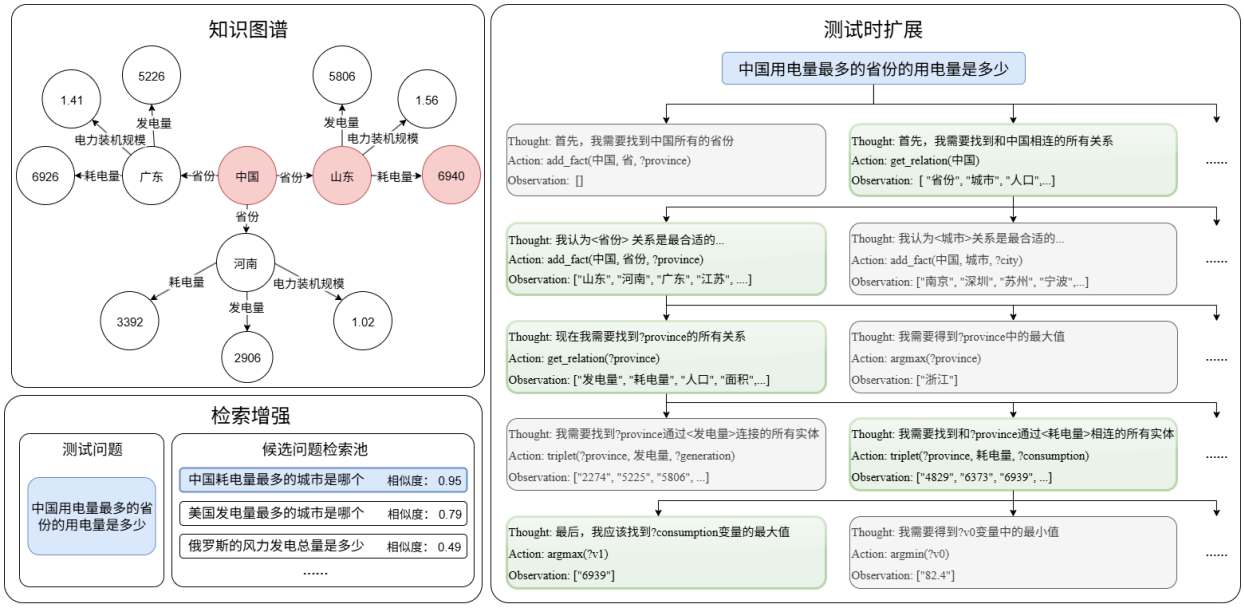

图 1　RaTS 的运行样例

索的对象是具有相似结构和推理模式的轨迹示例，旨在改进多步逻辑推理能力。在推理层面，RaTS 引入了 TTS 机制，在保持推理可解释性的同时提高容错率和最终答案的准确性。因此，RaTS 可视为一种更注重“过程增强”的智能体框架，与强调“知识增强”的 Agentic RAG 形成互补。

2.1　Agent推理

Agent 推理将求解步骤表示成逐步求解的方式，每个步骤分为三个部分，分别是思考（thought）、动作（action）和观察（observation）[17]。对于每个步骤，模型根据历史推理过程，生成对问题求解流程的思考，表示模型认为下一步应该如何继续求解。除了思考之外，模型同时生成一个函数调用，表示下一步应该采用什么动作。动作是一些预定义的操作，通过执行这些函数，模型可以实现和复杂环境的自由交互得到环境的反馈，称为观察。至此，一个步骤就完成了。通过多个步骤的分步求解，每个步骤负责一个简单的子问题，并多次和环境进行交互获取需要的信息，即可实现对复杂问题的求解。

表 1 为本文使用的 PyQL 函数工具。与 KG 交互的工具需要在KG上执行以获得执行结果，并不会向目标查询添加子句。构建查询的工具用于向目标查询添加子句。

KGQA 任务的核心目标是生成一个可被执行的 SPARQL 查询，而 SPARQL 查询本身又可以视作由多个求解步骤组合而成。例如，查找与特定实体通过某种关系相连的其他实体，对某变量进行计数统计，或者求取某变量的最大值等。本文将 KGQA 任务抽象为一个多轮对话过程，每一轮的目标是根据当前对话状态，生成 SPARQL 查询中的部分内容，通过多步推理，最终完成整个查询的构建。

doi:10.3772/j.issn.2095-915x.2026.04.006

表 1　本文使用的 PyQL 函数工具

与 KG 交互的工具	
get_relation(entity_or_variable)	获取实体或变量的一跳关系。参数应为 KB 中的实体(如 m.02xlbx)或变量(?computer)。
execute()	执行当前的 SPARQL 查询，并返回在 KB 上的执行结果。
构建查询的工具	
add_fact(head, relation, tail)	添加一个三元组模式（ head, relation, tail ）。
add_max(max_var)	计算给定变量 max_var 的最大值。
add_min(min_var)	计算给定变量 min_var 的最小值。
add_count(count_var)	计算给定变量 count_var 的出现次数。
add_filter(ob1,op,ob2)	添加一个比较约束。ob1 和 ob2 是两个用于比较的对象。op 可以是 >、<、=、>=、<= 之一。
set_answer(ans_var)	设置 ans_var 为此查询返回的变量。

然而，SPARQL 查询并不是天然地以逐步推理的形式表达的，其整体结构和自然语言的表达顺序存在很大差异。此外，SPARQL 的某些片段若单独拿出来往往无法直接执行，特别是在包含聚合操作时，必须严格遵循语法规范。为了克服这一挑战，使系统能够逐步与 KG 交互并逐步构建目标查询，本文采用 PyQL[18] 实现逐步推理的范式。PyQL 涉及到的函数在表 1 中列出，它将 SPARQL 查询转化为一系列函数调用，每个函数代表问题求解的一个具体步骤，这种方式非常适合用于 Agent 的推理过程。由于最终的 SPARQL 查询可以由一连串 PyQL 函数转译而来，因此 Agent 需要在多轮交互中逐步生成这些 PyQL 函数。这一阶段的提示词主要由四个部分构成，分别是任务说明、可用函数的简要文档、具体的推理示例，以及待解决的新问题。首先，系统会给出对任务的整体介绍以及必须遵守的相关规则。其次，提供所有可用函数的简要说明。然后，展示一个详细的示例问题，演示其完整的推理步骤。最后拼接上要求解的问题，请求系统进行解决。

2.2　轨迹生成

考虑到轨迹数据通常需要人工书写，而现有数据集中只提供最终查询，本文提出了一种将查询自动转换成轨迹数据的方法，为数据集中的每条数据配备对应的轨迹数据，以便于在 Agent 方法上使用 RAG。

转换过程分为两步，分别生成轨迹数据的动作和思考：1 将查询转换成逐步求解的函数序列（ 动作的生成 ）；2 为函数序列配备对应的思考和观察（ 思考的生成 ）。本文观察到最终的 SPARQL 查询通常是一个整体，无法从中看出一个明显的求解先后顺序，例如带有函数计算（ 例如计数、最高级等 ），在查询中通常放在查询头的位置，但是实际求解时需要在最后一步计算；另外，由于 SPARQL 查询对其中三元组的位置是不敏感的，数据集构造在书写查询的时候无需考虑求解的先后顺序，因此三元组模式也不能直接按照查询中的顺序进行求解。所以转换的第一步是将整体形式表示的查询，转换成有顺序的函数求解序列。本文实现了一种基于规则的方法，将 SPARQL 查询转换成 PyQL，得到轨迹数据中的动作序列。对于思考的生成，首先执行每一步的动作得到每一步的观察，然后提供给大模型一个提示词，里面包含了样例教大模型学习如何根据当前的历史

信息（已经生成的思考、动作和观察）以及当前步骤的动作，生成当前步骤的思考。迭代运用在每个动作上就可以得到每个步骤的思考，从而得到完整的轨迹数据。

2.3 检索增强

当前的基于 Agent 的方法通常从训练集里随机选择一个问题作为样例，对所有测试问题都使用同样的样例，这可能导致很多样例问题和测试问题并不相关，无法从中受到很多指导。对于比较复杂的数据集，所涉及的操作（例如获取关系、加三元组、加过滤器、求最高级、计数等）可能很难在一个例子中全部涵盖，导致有些情况下测试样例并不能从样例问题中得到恰当的参考。另外，由于 Agent 方法通常需要多轮交互，并且多轮交互中需要考虑前面的历史，所以大多数方法都只配备一个样例，导致这个样例的质量对于最终的效果就显得尤为重要。考虑到不同 KG 的表示方式不同，不同问题的查询结构差距也较大，有的包含多跳结构，有的包含多约束结构，有的包含函数的利用，如果能够使用更合适的问题作为样例，将会对模型推理提供更有效的帮助。

为了让大模型能够得到有针对性的指导，减小任务的难度，本文引入了 RAG[7] 的方式。具体而言，给定一个问题，本文去训练集里抽取最相似的问题作为参考。考虑到此任务核心在于检索语义和推理结构相似的问题示例，而非精确的知识匹配，本文参考 KB-Binder[13]、KB-Coder[12]、Pangu[14] 等方法，使用基于向量表示的余弦相似度计算待求解问题和候选问题之间的相似度。检索过程首先使用 BGE-reranker[19] 编码候选问题和测试问题的内容，然后使用余弦相似度计算不同候选问题和测试问题之间的相似度，相似度越高的候选问题被排在前面。记待求解的问题为 q，编码器函数为 $Embedding(q)$，返回一个 emb_q 表示问题的编码表示，在本文中的实现是一个 768 维的向量。记检索池中的所有问题为集合 $D=\{d_1,\ldots,d_n\}$，其中 $|D|=n$ 表示候选池的大小为 n，在本文中 n 是训练集的大小，同样记候选池中问题的编码为 $D_{emb}=\{d_{1_{emb}},d_{2_{emb}},\ldots,d_{n_{emb}}\}$，则 RAG 过程被形式化为：

$$emb_q=Embedding(q) \quad (1)$$

$$emb_{d_i}=Embedding(d_i),1<i\le n \quad (2)$$

$$score_i=cosine_\text{similarity}(emb_q,emb_{d_i}) \quad (3)$$

其中 $score_i$ 表示第 i 个候选 d_i 和待处理问题 q 之间的相似度，通过它们编码的余弦相似度衡量。对所有候选的分数进行排序，选择分数最高的问题作为后续处理要参考的样例，记为 r。

与典型的 RAG 不同，本文的检索目的并非检索外部知识片段，而是检索具有相似语义结构与推理模式的问题示例，即检索的目的是提供一种“类比式”的推理指导，让模型在遇到结构相似的问题时能够参考其他问题的求解思路。例如两个最高级问题可能拥有比较高的相似度（都有类似“最高的 xx 是”这样的表示），两个计数类的问题也拥有比较高的相似度（都有类似“有多少个”这样的模式），而两个多跳问题可能也拥有类似的相似度（都有类似“A 的 B 的 C 是什么”这样的模式），抽取得到的样例仅作为推理过程的参考示例，用于指导模型如何通过工具调用的范式一步一步求解问题，即“思考 – 动作 – 观察”三元轨迹。而对于大语言模型缺乏的知识的获取则是通过工具调用

doi:10.3772/j.issn.2095-915x.2026.04.006

的方式，以检索 KG 的形式获取，并非通过检索得到的例子获得。需要注意的是，尽管检索的样例包含了问题的答案信息，但这只是为了展示一个完整的求解案例，告诉模型可以通过一些终止函数决定什么时候推理过程停止，而非将样例中的答案信息作为外部知识的补充。

2.4 测试时扩展

考虑到KGQA是一个复杂的多步推理任务，每次只给模型一次机会有些过于富有挑战，模型在过程中出现的微小错误都可能导致最终结果出错。为了进一步提升问答系统的性能，本文借鉴了 TTS 的思想，让模型在测试阶段多消耗一些词元（token）以换取更好的性能。具体而言，在每个步骤生成时都会同时采样多个回复（thought, action），在得到多个回复之后，将历史上下文和本轮生成的所有结果一起提供给大模型，让它选择最有可能是正确结果的作为最终结果。本文要求大模型首先输出对于每个结果的评估细节和思考过程，用 <think></think> 标签包裹，便于最终识别和提取。然后，要求大模型综合考虑每个回复和它们的评估结果，给出一个最好的新回复作为最终结果，并将最终的输出用<answer> </answer>标签包裹，便于解析用于后续流程。考虑到模型在较低温度（temperature）的情况下生成结果的多样性受限，导致无法有效搜索到最好的回复，在采样多个回复的阶段将温度调高至 1.0，在这个阶段让模型先比较自由地输出，以搜索更多样的回复可能，增加覆盖到更好结果的可能。记多个生成的候选结果为 $A=\{a_1, a_2, \ldots a_m\}$，其中 m 指要做多大程度的 TTS，即要对多少个生成结果进行分析总结，默认取值为 5，整个过程可以被形式化为：

$$thought_i^t, \sim action_i^t = LLM\left(q, r, his_t\right), 1 \le i \le m \quad (4)$$

$$obs_i^t = Execute\left(action_{final}^i\right), 1 \le i \le m \quad (5)$$

$$c_i^t = \left(action_i^t, thought_i^t, obs_i^t\right), 1 \le i \le m \quad (6)$$

$$C^t = \left\{c_1^t, \ldots, c_m^t\right\} \quad (7)$$

$$thought_{final}^t, \sim action_{final}^t, obs_{final}^t = LLM\left(C^t\right) \quad (8)$$

$$his_{t+1} = his_t \,||\, thought_{final}^t \,||\, action_{final}^t \,||\, obs_{final}^t \quad (9)$$

其中，t 表示当前推理步骤，i 表示候选结果的序号，m 表示每个推理步骤生成的候选结果数量；q 表示待求解的问题，r 表示通过检索增强模块获得的参考样例，his_t 表示步骤 t 之前累积的历史推理信息；$thought_i^t$、$action_i^t$ 和 obs_i^t 分别表示步骤 t 中第 i 个候选结果的思考、动作以及执行该动作后得到的观察结果；c_i^t 表示第 i 个候选结果，C^t 表示由所有候选结果组成的集合；$thought_final_t$、$action_final_t$ 和 obs_final_t 分别表示步骤 t 最终选择的思考、动作及其对应的观察结果；$LLM(\cdot)$ 表示大语言模型的生成或评估过程，$Execute(\cdot)$ 表示执行动作并返回观察结果，“||”表示信息拼接操作。式（4）—（7）表示生成并执行多个候选结果，以构建候选结果集合；式（8）表示大语言模型对候选结果进行综合分析，并确定当前步骤的最终结果；式（9）表示将最终结果加入历史推理信息，供下一步骤推理使用。

3 实验

3.1 实验设置

本文使用F1分数作为KGQA的评估指标，

对比了各种范式的基线方法，包括微调、上下文学习和 Agent 方法。对于同样设置的实验，本文报告它们原始论文中的结果。对于其他设置的实验，本文使用 MIDDLEWARE[20] 和 QueryAgent[21] 提供的复现结果。默认情况下，使用 gpt-4.1 作为实验中的基础模型。

3.1.1 数据集

在四个 KGQA 数据集上进行实验，其统计数据如表 2 所示。由于 GrailQA 没有给出测试集的标准答案，本文仅报告 GrailQA 的验证集性能和其他数据集的测试集性能。

GrailQA[22] 是一个基于 Freebase 的大规模复杂数据集，它首次提出了三种层次的泛化能力的评估，即独立同分布、复合情况和未见情况。其中独立同分布指的是测试集和训练集满足同样的分布。复合情况指的是测试集中见过训练集中的所有实体和关系，但是有一些没有见过的组合结构。未见情况指的是测试集里有训练集里没有见过的实体和关系。

GraphQ[23] 是一个非常具有挑战性的基于 Freebase 的数据集，它的测试集中只有非独立同分布的数据，并且训练集只有两千多个。本文使用 ArcanQA[24] 处理过的版本（即将 KG 从 Freebase 2013-07 版本更新为 2015-08-09 版本）。

KBQA-Agent[20] 是一个混合数据集，共包含 500 个来自四个数据集（即 GrailQA[22]、ComplexWebQuesitons[25]、GraphQ[23] 和 WebQSP[26]）中最困难的问题。

MetaQA[27] 由基于 Wiki-Movies 上的 1 至 3 跳问题组成。Wiki-Movies 是一个比较小的 KG，只有九个关系。本文在最困难的 3 跳子集（记为 MetaQA-3Hop）上进行实验。

表 2　数据集的统计信息

数据集	训练集	开发集	测试集
GrailQA	44337	6763	13231
GraphQ	2381	—	2395
KBQA-Agent	—	—	500
MetaQA-3Hop	114196	14274	14274

3.1.2 基线方法

本文将 RaTS 与微调方法、上下文学习方法和 Agent 方法进行比较。

（1）上下文学习方法包括：

Pangu[14] 提供了微调和少样本两种设置的模型。对于少样本设置，Pangu 采用上下文学习范式，给定若干样例输出目标查询。

KB-Binder[13] 同样是一种基于上下文学习的 KGQA 方法，利用几十个（问题，S 表达式）对作为示例，首先生成一些未绑定到 KG 上的草稿查询，然后绑定到 KG 上得到可执行的查询。

KB-Coder[12] 的流程和 KB-Binder 基本类似，但是将 S 表达式转换为了函数调用序列，从而降低了生成查询的格式错误率。

McL-KBQA[31] 使用基于排名的方法枚举和评分逻辑形式候选，然后使用大语言模型通过上下文学习选择最终的逻辑形式。

StructGPT[32] 是一种基于上下文学习的方法，用于结构化数据（包括表格、数据库和 KG）的复杂推理。它以预定义的顺序预测函数调用序列，具体在数据库上定义了三种操作，在 KG 上定义了两种操作。

（2）Agent 方法包括：

AgentBench[16] 通过将 KGQA 建模为多轮开放式生成任务，提出了一种基于 Agent 的基线方法。

QueryAgent[21] 也是一个基于 Agent 的方法，它引入了一种基于环境反馈的自纠错机制缓解

doi:10.3772/j.issn.2095-915x.2026.04.006

误差传递现象。

MIDDLEWARE[20] 设计了作为大语言模型和复杂环境之间中间件的定制工具，并结合了解耦生成和错误反馈以提高性能。

3.2 实验效果

本文在四个 KGQA 评测集上与多个基线方法进行了对比，实验结果如表 3 所示。

本文提出的方法在四个数据集上的表现均优于之前的对比方法。相比上下文学习方法，本文的方法在 GrailQA 和 GraphQ 上分别比最好的上下文学习方法高 4.5 和 18.5 分。本文方法的优势主要来源于基于 Agent 的求解方式，GrailQA 和 GraphQ 上的问题基本上都是复杂问题，更适合用逐步求解的方式解决。特别地，GraphQ 上的问题全部都属于复合类型的问题，有很多全新的组合方式，非常适合通过多步探索的方式求解，因此 Agent 类方法在 GraphQ 上有明显的性能优势。

相比其他基于 Agent 的方法，本文方法也有明显的优势，相比这类方法中效果最好的 QueryAgent，在 GrailQA 和 GraphQ 上分别提升 5.3 和 3.5。这得益于 RAG 和 TTS 的引入，RAG 提高了样例的质量，降低任务的难度；而 TTS 探索了更多可能的候选，增加了找到正确结果的概率。

表 3 为 GrailQA、GraphQ、KBQA-Agent、MetaQA-3Hop 数据集上的总体结果。所有结果均通过 F1 分数进行评估。表中所有先前方法的结果均取自其原论文报告，其中“-”表示对应方法在原论文中未报告该数据集的结果。对于上下文学习的方法，Pangu 对所有数据集使用 100 个例子作为示例。KB-Binder 和 KB-Coder 对 GrailQA 使用 40 个示例，对 GraphQ 使用 100 个示例。KB-Binder 对 KBQA-Agent 使用 5 个示例，对 MetaQA-3Hop 使用 5 个示例。McL-KBQA 对 GrailQA 使用 221 个示例。StructGPT 对 MetaQA-3Hop 使用 32 个示例。AgentBench、QueryAgent、Middleware、RaTS 对所有数据集仅使用 1 个示例。

表 3 GrailQA、GraphQ、KBQA-Agent、MetaQA-3Hop 数据集上的总体结果

类别	方法	GrailQA	GraphQ	KBQA-Agent	MetaQA-3Hop
上下文学习方法	Pangu[14]	53.5	35.4	18.1	-
	KB-Binder[13]	58.5	34.5	4.2	96.5
	KB-Coder[12]	61.3	35.8	-	-
	McL-KBQA[31]	54.8	-	-	-
	StructGPT[32]	-	-	-	87.0
Agent 方法	AgentBench[16]	30.5	25.1	25.9	-
	QueryAgent[21]	60.5	50.8	-	98.5
	Middleware[20]	-	-	34.3	-
	RaTS（本文方法）	65.8	54.3	40.3	99.5

3.3 实验分析

3.3.1 消融实验

为了分析各个模块对于问答性能的贡献，以及模块间的协同效应，本文开展了完整的消融实验。实验结果如表 4 所示，其中 R 代表 RAG 模块，即抽取训练集中相似问题的推理指

导作为参考样例；C 代表思维链模块，即要求模型在输出认为正确的动作前需先输出思考过程；T 代表 TTS 模块，即每步生成多个候选回答用于综合决策，“-”表示去除该模块。

从结果可以观察到，在分别去掉 3 个模块后，模型整体的表现都有明显的下滑。以 GrailQA 为例，去掉 RAG 导致 F1 下降了 1.1，去掉思维链导致 F1 下降了 2.4，去除 TTS 导致 F1 下降了 2.8，验证了每项改进对于性能提升都有明显帮助。此外，3 种单模块设置（仅 R、仅 C、仅 T）的性能均低于任意双模块组合，表明各模块本身虽具备一定独立贡献，但单独使用时无法有效解决复杂的多步推理问题。对比不同双模块组合可以发现：R+C（63.0）显著优于仅 R（59.9）和仅 C（61.5），说明 RAG 在显式思维链推理的配合下，能够更有效地引导模型复用相似结构的推理轨迹，二者存在明显正向协同；C+T（64.7）相比仅 C（61.5）提升更为显著，表明 TTS 依赖于清晰稳定的中间推理轨迹来进行多候选筛选，思维链为 TTS 提供了必要的“可比较中间状态”；R+T（63.4）的提升幅度小于 C+T，说明当缺少思维链约束时，TTS 虽然可以在一定程度上弥补推理错误，但其纠错能力仍受限于推理路径的稳定性。

表 4　消融实验

方法	GrailQA	GraphQ	KBQA-Agent
RaTS (R+C+T)	65.8	54.3	40.3
- R (C+T)	64.7	53.9	39.6
- C (R+T)	63.4	52.4	37.1
- T (R+C)	63.0	51.4	36.8
- R - C (仅 T)	60.8	48.6	34.2
- R - T (仅 C)	61.5	49.1	34.9
- C - T (仅 R)	59.9	47.8	33.6

3.3.2　不同泛化层级的效果

本文也对比了不同泛化层级下模型的效果，以验证方法的泛化能力。以 GrailQA 为例，它包含了 25% 的独立同分布数据、25% 的复合数据，以及 50% 的未见数据。如表 5 所示，本文提出的方法在三个泛化层级上相比之前的方法都有不同程度的优势。本文发现独立同分布的数据相对于另一个基于 Agent 的方法 QueryAgent 提升较大，这主要受益于 RAG 提高了提示中样例的质量。虽然弱于 KB-Binder，但是 KB-Binder 是一个上下文学习的方法，能够在输入中包含上百个样例，更有可能覆盖到相似的数据，因此这方面优势较大，但是这类方法以端到端的方式求解问题，对于具有复杂的结构复合类问题就表现出明显的劣势。对于复合类型的测试数据，本文方法相比 QueryAgent 的提升也比较明显，因为复合类型的数据通常涉及没有见过的实体与关系组合，引入 TTS 可以让模型多考虑几种情况再做最终的决定，容错率更高更有机会覆盖到正确答案，非常适合用于探索拥有复杂结构和新的组合的问题。对于未见类型的数据，RaTS 同样表现出更稳定的泛化能力。未见类问题同时包含训练集中未出现过的实体和关系，对模型的结构泛化能力要求更高。尽管 RAG 在该场景下难以命中完全一致的示例，但仍可在推理结构层面提供有效参考，且由于 RaTS 采用了基于 Agent 的逐步推理范式，可通过多轮与 KG 的交互逐步定位到正确的推理路径。因此在未见场景下仍能保持较稳定的性能。对未见样本的错误分析表明，主要错误来自未见关系组合与低频实体绑定偏差，两者均会在多步推理中被放大。相比不具备 TTS 的 Agent 方法，RaTS 在多数情

doi:10.3772/j.issn.2095-915x.2026.04.006

况下能够通过多候选筛选机制规避单路径错误，从而提升未见场景下的鲁棒性。该结果表明，RaTS 的泛化能力主要来源于结构级推理能力而非对具体实体与关系的记忆。

表 5　不同泛化层级的效果

方法	独立同分布	复合	未见	平均
KB-Binder	80.9	53.6	50.7	58.5
QueryAgent	60.9	58.7	60.8	60.5
RaTS	66.3	63.8	65.4	65.8

3.3.3　不同TTS设置的效果

本文还分析了不同成本下，TTS 所带来的性能变化。本文测试了生成 1 个候选到 9 个候选时的性能变化。实验结果如图 2 所示，随着候选数量的增加，模型生成的回复质量明显增加，在候选数量为 5 个后达到一个逐渐稳定的水平但仍有增强趋势，这体现了 TTS 在计算资源充分时的强大的可扩展性和应用潜力，只要增加尝试次数和成本，就有机会探索到更好的回复，可以灵活地根据可接受的成本范围调整 TTS 的规模，实现性能和成本之间的平衡。

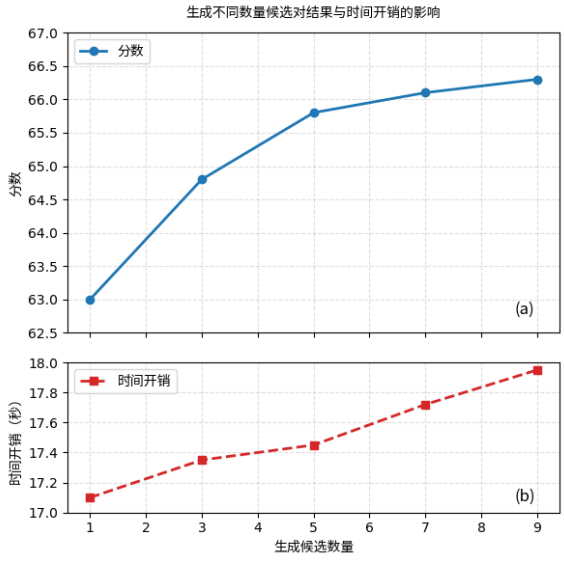

图 2　不同 TTS 设置下在 GrailQA 上的实验效果

此外，TTS 并不会显著增加生成时间开销。在工程实现中，本文使用的大语言模型接口支持通过修改参数同时生成多条候选结果，相比只返回一个结果，同时返回多个结果几乎不增加额外的时间成本，只有执行得到的多个动作阶段时间开销会有所不同，但相比请求大语言模型接口而言，KG 的查询速度是非常快的，吞吐量完全可以承受并行执行 10 次查询。实际实验中，当生成候选数从 1 增加至 9 时，平均推理时间仅增加约 5%（增加了 0.85 秒），因此该方法在性能提升与推理效率间也取得了良好平衡。

3.3.4　不同嵌入模型的效果

为了验证不同的嵌入模型会对性能产生怎样的变化，除了默认的 bge-base-zh-v1.5 之外，本文另外选用两个其他嵌入模型，分别是 Sentence-Transformer[33] 和 OpenAI text-embedding-ada-002 进行实验：

bge-base-zh-v1.5 是智源推出的 BERT 类通用类（中英文领域）语义向量模型，于 2023 年 2 月推出，可以被用于抽取、聚类、重排序、摘要、分类等多个任务。它使用了一种三阶段的训练策略，从预训练到通用能力微调再到任务特定微调；前两个阶段是保证通用性的基石，最后一个阶段则在保持通用的基础上，进一步精进下游任务的效果。

Sentence-Transformer 是另一个开源的嵌入模型代表，是一种专门用于计算句子、段落或更长文本片段语义向量表示的深度学习模型架构。它主要利用“孪生网络”（Siamese Network）或“三元组网络”（Triplet Network）结构，把句子分别输入同一个 Transformer（如

BERT），得到各自的句向量，然后通过余弦相似度等方法，直接衡量句子之间的语义距离。

OpenAI text-embedding-ada-002 是 OpenAI 发布的一个高效、通用的文本向量嵌入模型，常用于将文本（如句子、段落、文档等）转换为高维稠密向量表示。这种向量可以用来进行文本相似度计算、语义搜索、聚类、推荐、去重等各种自然语言处理任务。

实验结果如表 6 所示，作为目前开源嵌入模型的代表，本文默认采用的 BGE 嵌入已经取得良好效果，相比传统的 Sentence-Transformer，提升 1.3 分。和更强的闭源模型 OpenAI text-embedding 相比，只落后了 1.1 分，但是可以节省成本，对于大规模应用场景意义更大，因此本文默认采用了开源模型中表现较好的 bge 作为嵌入模型。

表 6　使用不同嵌入模型在 GrailQA 上的实验效果

方法	GrailQA
（默认）bge-base-zh-v1.5	65.8
Sentence-Transformer	64.5
OpenAI text-embedding	66.9

3.4　可扩展性与实时性讨论

可扩展性方面，RaTS 的主要计算开销来自多轮 Agent 推理过程中的 KG 查询与 TTS 带来的多候选生成。当 KG 规模增大时，获取实体一跳关系与执行 SPARQL 查询等操作的响应时间将成为影响整体效率的主要瓶颈。但需要指出的是，RaTS 的推理过程仅涉及对局部子图的逐步访问，每一步只查询当前已绑定实体的一跳或有限多跳邻域，其计算复杂度主要与局部子图规模相关，而与整个 KG 的全局规模弱相关。因此在采用常见工业级图存储方案时，即使在十亿级三元组规模的 KG 上，局部查询仍可在可接受时间内完成。RAG 模块仅依赖问题语义向量计算，与 KG 规模无关，因此不会引入额外的图查询负担。TTS 在默认设置下会带来少量额外推理与查询开销，且由于各候选之间可以并行执行，其时间增长在实际工程中是可控的。在实时性方面，RaTS 更适用于对响应时间要求为秒级而非毫秒级的复杂问答场景。当候选数量进一步增大或推理步数较多时，整体延迟仍可能上升。后续工作可通过缓存中间查询结果、并行化图查询、构建高效图索引等工程手段进一步提升系统吞吐能力。

4　结论

本文致力于缓解 KGQA 任务中的三个难点：轨迹数据难以生成、样例数据质量要求高，以及模型容错率低。针对上述问题，本文提出了一种融合 RAG 与 TTS 的基于 Agent 的 KGQA 方法 RaTS，并设计了一种自动将查询转换为轨迹数据的构造流程。通过为模型提供更相关的例子作为样例，并给模型更多机会尝试更多种路径，RaTS 有效地提升了方法的效果，在多个 KGQA 数据集上取得了良好的性能。

参 考 文 献

[1] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[C]// Advances in Neural Information Processing Systems. Curran Associates, Inc, 2020: 1877-1901.

[2] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]// Advances in Neural Information Processing Systems: volume 30. Curran Associates, Inc., 2017: 6000-6010.

doi:10.3772/j.issn.2095-915x.2026.04.006

[3] RADFORD A, NARASIMHAN K, SALIMANS T, et al. Improving language understanding by generative pre-training[EB/OL]. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[4] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[C]// Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 4171-4186.

[5] SHAFIK W. Introduction to ChatGPT[C]// Advanced applications of generative AI and natural language processing models. IGI Global Scientific Publishing, 2024: 1-25.

[6] ACHIAM J, ADLER S, AGARWAL S, et al. GPT-4 technical report[J]. arXiv preprint arXiv:2303.08774, 2023.

[7] LEWIS P, PEREZ E, PIKTUS A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks[C]// Advances in Neural information processing systems, Curran Associates, Inc., 2020, 33: 9459-9474.

[8] KARPUKHIN V, OGUZ B, MIN S, et al. Dense Passage Retrieval for Open-Domain Question Answering[C]// Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020: 6769-6781.

[9] SNELL C, LEE J, XU K, et al. Scaling LLM test-time compute optimally can be more effective than scaling model parameters[J]. arXiv preprint arXiv:2408.03314, 2024.

[10] JAECH A, KALAI A, LERER A, et al. Openai o1 system card[J]. arXiv preprint arXiv:2412.16720, 2024.

[11] GUO D, YANG D, ZHANG H, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning[J]. arXiv preprint arXiv:2501.12948, 2025.

[12] NIE Z, ZHANG R, WANG Z, et al. Code-style in-context learning for knowledge-based question answering[C]// Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(17): 18833-18841.

[13] LI T, MA X, ZHUANG A, et al. Few-shot in-context learning on knowledge base question answering[C]// Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023: 6966-6980.

[14] GU Y, DENG X, SU Y. Don't generate, discriminate: A proposal for grounding language models to real-world environments[C]// Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023: 4928-4949.

[15] LAN Y, JIANG J. Query graph generation for answering multi-hop complex questions from knowledge bases[C]// Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020: 969-974.

[16] LIU X, YU H, ZHANG H, et al. AgentBench: Evaluating LLMs as agents[C]// The Twelfth International Conference on Learning Representations. 2024.

[17] YAO S, ZHAO J, YU D, et al. React: Synergizing reasoning and acting in language models[C]// The Eleventh International Conference on Learning Representations. 2023.

[18] HUANG X, CHENG S, BAO Y, et al. MarkQA: A large scale KBQA dataset with numerical reasoning[C]// Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023: 10241-10259.

[19] CHEN J, XIAO S, ZHANG P, et al. M3-embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation[C]// Findings of the Association for Computational Linguistics: ACL 2024. Bangkok, Thailand: Association for Computational Linguistics, 2024: 2318-2335.

[20] GU Y, SHU Y, YU H, et al. Middleware for LLMs: Tools are instrumental for language agents in complex environments [C]// Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Miami, Florida, USA: Association for Computational Linguistics, 2024: 7646-7663.

[21] HUANG X, CHENG S, HUANG S, et al.

QueryAgent: A reliable and efficient reasoning framework with environmental feedback based self-correction[C]// Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Bangkok, Thailand: Association for Computational Linguistics, 2024: 5014-5035.

[22] GU Y, KASE S, VANNI M, et al. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases [C]// WWW '21: Proceedings of the Web Conference 2021. New York, NY, USA: Association for Computing Machinery, 2021: 3477-3488.

[23] SU Y, SUN H, SADLER B, et al. On generating characteristic-rich question sets for QA evaluation[C]// Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Empirical Methods in Natural Language Processing (EMNLP). Austin, Texas, USA, 2016: 562-572.

[24] GU Y, SU Y. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering[C]// Proceedings of the 29th International Conference on Computational Linguistics. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, 2022: 1718-1731.

[25] TALMOR A, BERANT J. The web as a knowledge-base for answering complex questions[C]// Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018: 641-651.

[26] YIH W T, RICHARDSON M, MEEK C, et al. The value of semantic parse labeling for knowledge base question answering[C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2016: 201-206.

[27] ZHANG Y, DAI H, KOZAREVA Z, et al. Variational reasoning for question answering with knowledge graph[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[28] YE X, YAVUZ S, HASHIMOTO K, et al. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering [C]// Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Dublin, Ireland: Association for Computational Linguistics, 2022: 6032-6043.

[29] SHU Y, YU Z, LI Y, et al. TIARA: Multi-grained retrieval for robust question answering over large knowledge base[C]// Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022: 8108-8121.

[30] YU D, ZHANG S, NG P, et al. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases[C]// The Eleventh International Conference on Learning Representations. 2023.

[31] TAN C, CHEN Y, SHAO W, et al. Make a choice! knowledge base question answering with in-context learning[J]. arXiv preprint arXiv:2305.13972, 2023.

[32] JIANG J, ZHOU K, DONG Z, et al. StructGPT: A general framework for large language model to reason over structured data[C]// Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023: 9237-9251.

[33] REIMERS N, GUREVYCH I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks[C]// Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 3982-3992.

[34] ZHONG V, XIONG C, SOCHER R. Seq2sql: Generating structured queries from natural language using reinforcement learning[J]. arXiv preprint arXiv:1709.00103, 2017.

（责任编辑：高雄）