计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(1):1−10 [doi: 10.15888/j.cnki.csa.009782] [CSTR: 32024.14.csa.009782]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

大语言模型中提示词工程综述①

王东清, 芦　飞, 张炳会, 李道童, 彭继阳, 王　兵, 姚藩益, 艾山彬

(浪潮电子信息产业股份有限公司, 北京 100095)
通信作者: 王东清, E-mail: wangdongqing01@ieisystem.com

摘　要: 提示词工程在解锁大语言模型潜能上具有重要作用. 该方法通过设计提示指令指导模型响应, 确保响应的相关性、连贯性和准确性. 提示工程无需微调模型参数, 可与下游任务无缝衔接. 因此, 各种提示词工程技术成为近年来研究的热点. 据此, 介绍了创建有效提示词的关键步骤, 总结了基础和高级提示词工程技术方法, 如思维链、思维树, 深入探讨了每种方法的优势和局限性. 同时, 讨论了如何从不同角度和不同方法评估提示方法的有效性. 这些技术的迅速发展使大语言模型在各种应用中取得了成功, 从教育、医疗到代码生成等. 最后, 展望了提示词工程技术的未来研究方向.

关键词: 大语言模型; 提示词工程; 思维链; 思维树

引用格式: 王东清,芦飞,张炳会,李道童,彭继阳,王兵,姚藩益,艾山彬.大语言模型中提示词工程综述.计算机系统应用,2025,34(1):1–10. http://www.c-s-a.org.cn/1003-3254/9782.html

Survey on Prompt Engineering in Large Language Model

WANG Dong-Qing, LU Fei, ZHANG Bing-Hui, LI Dao-Tong, PENG Ji-Yang, WANG Bing, YAO Fan-Yi, AI Shan-Bin

(Inspur Electronic Information Industry Co. Ltd., Beijing 100095, China)

Abstract: Prompt engineering plays a crucial role in unlocking the potential of large language model. This method guides the model's response by designing prompt instructions to ensure the relevance, coherence, and accuracy of the response. Prompt engineering does not require fine-tuning model parameters and can be seamlessly connected with downstream tasks. Therefore, various prompt engineering techniques have become a research hotspot in recent years. Accordingly, this study introduces the key steps for creating effective prompts, summarizes basic and advanced prompt engineering techniques, such as chain of thought and tree of thought, and deeply explores the advantages and limitations of each method. At the same time, it discusses how to evaluate the effectiveness of prompt methods from different perspectives and using different methods. The rapid development of these technologies enables large language models to succeed in a variety of applications, ranging from education and healthcare to code generation. Finally, future research directions of prompt engineering technology are prospected.

Key words: large language model (LLM); prompt engineering; chain of thought (CoT); tree of thought (ToT)

近年来, 大语言模型 (large language model, LLM) 的快速发展成为通用人工智能领域的重要里程碑. LLM 的起源可以追溯到深度学习技术的兴起, 为处理复杂的语言任务提供了新的解决思路. 起初, 循环神经网络 (RNN) 和长短期记忆网络 (LSTM) 等时间序列模型, 在语言类的任务处理上取得了较大进展, 但在处理长距离依赖和并行计算方面受到限制. 2017 年, Vaswani 等人[1]首次提出 Transformer 架构, 在机器翻译、文

① 基金项目: 山东省自然科学基金 (ZR2019LZH006)
收稿时间: 2024-08-19; 修改时间: 2024-09-19; 采用时间: 2024-09-24; csa 在线出版时间: 2024-11-15
CNKI 网络首发时间: 2024-11-18

© 中国科学院软件研究所 http://www.c-s-a.org.cn

本摘要、问答系统等各类语言任务中表现突出. Transformer 模型的核心是使用注意力机制 (attention mechanism) 来捕捉序列中各个元素之间的关系, 其主要特点表现为: 1) 并行计算能力: Transformer 模型可以并行处理序列数据, 使其在计算效率上优于串行计算的 RNN 和 LSTM 模型; 2) 捕捉长距离依赖: 通过自注意力机制, Transformer 能够捕捉序列中任意两个位置之间的依赖关系, 可有效处理长文本序列; 3) 模块化设计: Transformer 由编码器和解码器组成, 每层都包含多头自注意力机制和前馈神经网络, 这种设计使得模型易于扩展和调整. Transformer 架构的出现标志着 LLM 发展的一个转折点.

目前, 主流的 LLM 都采用 Transformer 架构, 如 GPT-3[2]、GPT-4[3]、LLAMA-2[4]、ChatGLM3[5]. 这些模型通过预测句子中缺失的单词或短语, 能够在大量未标记的文本数据上进行自监督学习训练, 从而提取语言的统计规律和语义关系. 特别地, LLM 通过将输入文本编码成具有语义关系的高维空间向量, 经过参数量百万或上亿的模型解码生成响应. 响应的质量会受到多种因素影响, 包括提供给模型的提示词、模型的参数量以及训练数据的多样性.

在实际应用中, 提示词作为已训练模型的输入, 其工程效果会带来显著的输出差异[6]. 提示词工程即通过系统化设计、精炼模型的输入等方式指导模型响应, 确保相关性、连贯性和准确性. 理解如何创建有效的提示词, 可为使用者节省宝贵的时间和资源, 解锁大模型新的潜能, 开发更复杂、有价值的应用, 赋能千行百业. 研究表明, 修改提示词结构 (样例长度、顺序) 和内容 (措辞、样例质量) 会对模型行为明显改善[7]. 文献[8,9]指出, 设计出色的提示词可有效缓解 LLM 的幻觉现象. 通过优化小样本学习中的提示词, 促进构建更有效的聊天机器人、虚拟助手和其他 AI 对话系统[10,11].

1 提示词工程基础

组成提示词的基本元素包括输入指令、上下文和输出指示. 这些元素也会因模型的选择、下游任务的差异需要不断尝试和修正, 以保证 LLM 能够产生高质量的答案. 在本节中, 将讨论提示词设计的基本流程, 以及改善 LLM 输出效果的常用策略.

1.1 基本概念介绍

输入指令是用户 LLM 提出的具体请求或问题, 可以是直接的, 比如“明天的天气如何?”. 也可以是间接的, 比如“我准备去北京旅游, 有什么建议吗?”. 输入指令的设计对于 LLM 理解用户意图和生成相关回答至关重要.

上下文指与输入指令相关的信息, 可以帮助 LLM 更好地理解指令的含义和用户的需求. 上下文可以是用户与 LLM 的历史对话、相关话题的信息、时间地点等环境因素, 甚至是用户的情感状态. 在处理输入指令时, 考虑上下文可以帮助 AI 生成更准确、更个性化的回答.

输出指示是 LLM 根据输入指令和上下文信息生成的响应. 这个响应应该满足用户的查询需求, 同时遵循系统的输出标准和限制. 输出指示可以是一段文本、一个答案、一组选项或者任何其他形式的反馈, 旨在以最恰当的方式回答用户的指令.

在设计提示词时, 工程师需要精心构造输入指令, 确保它能够清晰地传达用户的意图, 同时提供足够的上下文信息, 以便 LLM 能够生成合适的输出指示. 这个过程需要对语言的细微差别、用户的需求以及 LLM 的能力有深入的理解.

1.2 创建有效提示词的步骤

提示词创建可以分解为 5 个关键步骤.

(a) 明确目标: 在设计提示词之前, 需要明确模型完成的具体任务, 这可能包括问题、生成文本、文本翻译、提供解释或其他任务. 明确目标有助于设计更精确的提示词.

(b) 理解模型能力: 了解你所使用的 LLM 特点和能力. 不同的模型可能在处理不同类型的任务时表现不同. 理解模型的强项和弱点可以帮助你设计更有效的提示词.

(c) 设计初始提示: 根据目标和模型能力, 设计一个初始的提示词. 这个提示词应该尽可能地清晰和具体, 避免使用模糊或多义的词汇. 同时, 它应该提供足够的上下文信息帮助模型理解任务.

(d) 测试和迭代: 使用设计的提示词与模型进行交互, 观察模型的输出. 如果输出不符合预期, 分析可能的原因, 并根据这些发现调整提示词. 该过程可能需要多次迭代, 直到找到最佳的提示词.

(e) 评估和优化: 一旦找到一组有效的提示词, 对其进行评估, 观察在不同的任务和上下文中的表现. 这可能涉及在不同的数据集上测试提示词, 或者收集用户的反馈. 根据评估结果, 进一步优化提示词, 以提高其性能和适用性.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

1.3 初级提示词工程技术

明确性: 是提升 LLM 效果的关键. 一个清晰的提示可以减少模型的歧义理解, 确保模型准确把握用户的意图. 为此, 指令应该直接、简洁, 并且避免使用含糊或多义的词汇[12].

使用关键词: 突出关键词可以帮助模型集中注意力. 在提示中使用加粗、斜体或列表等形式强调关键词, 可以提高模型识别和响应这些关键信息的能力[13].

示例学习: 提供正确或错误的例子可以指导模型理解期望的输出格式. 例如, 如果用户想要模型生成一段代码, 提供一段正确和错误代码的示例可以帮助模型理解用户意图. 根据提供示例的多少, 该技术又分为 zero-shot learning (ZSL) 和 few-shot learning (FSL)[14].

使用正确的语言风格: 根据目标输出调整语言的正式程度和风格. 例如, 如果需要模型生成学术论文, 使用正式和学术的语言风格会更合适[15].

反馈循环: 利用用户提供的反馈来不断优化提示词. 如果用户对模型的输出不满意, 可以根据用户的反馈调整提示, 以进一步提高性能[16].

角色扮演: 指定一个角色或身份给模型, 使其以特定视角回答问题. 例如, 如果用户想知道医生对某个医疗问题的看法, 可以让模型扮演医生的角色[17].

2 高级提示词工程技术

尽管上文中的初级方法可以帮助用户产生令人满意的输出, 但在处理复杂任务时, 如分析或推理, 模型的输出质量仍有提升的空间. 在本节中, 将介绍提示工程中的高级方法. 这些方法增强 LLM 的理解能力, 提高 LLM 解决问题的效率和准确性, 包括提供更加丰富的信息、更加结构化和合理化的提示, 以及利用先验知识和逻辑推理优化模型的响应. 表 1 汇总了要各类提示技术.

表 1 提示词工程技术的描述、应用方向、优点和缺点

提示词工程技术	描述	应用方向	优点	缺点
ZSL/FSL	允许模型在没有或只有少量目标任务样本的情况下, 解决新任务	新任务无训练样本	无需对模型训练和微调, 降低对数据标注依赖	无法适用于高复杂度任务, 决策过程无可解释性
CoT	将复杂问题分解为更易于处理的子问题, 构建一条逻辑推理的链条	推理与逻辑	无需对模型训练和微调, 显著提升在复杂任务上的表现	对模型规模有要求, 且仅在数学问题和常识推理表现出优势
SC	模型解码器采样生成多样化的推理链路, 采用投票方式提高答案的准确性		在不同的采样策略和参数设置下均表现稳定、出色, 有效提升了模型的鲁棒性和解释性	生成多个推理链路并进行投票需要更多的计算资源, 最终结果依赖模型本身的推理能力
ToT	评估多种不同推理路径, 择优执行, 必要时还可以向前看或回溯以做出全局选择		适用于解决各种任务; 对构成ToT各个模块可自由适配	在执行搜索时会产生大量的API调用, 会增加计算成本
GoT	将模型输出视为图结构, 节点代表推理的中间步骤. 边代表步骤间的依赖关系		GoT允许通过反馈循环来反复精炼一个思维; GoT可以同时探索多个独立的推理路径; 可解释性强	需要对任务有深刻的理解并精心设计提示, 以实现最佳结果.
Auto-CoT	自动创建带有问题和推理链示例, 解决人工创建CoT示例既耗时又不太理想问题		Auto-CoT具有自动化、可扩展性强且效果显著的优势	可能会导致事实错误、逻辑错误等幻觉现象; 需要在复杂性和多样性之间找到平衡点
GKP	从LLM中生成知识, 并将这些知识作为额外的输入来回答问题	降低模型幻觉	无需特定任务的知识整合; 无需访问结构化的知识库和微调模型	在执行调用额外的模型, 会增加计算成本; 最终结果依赖生成知识模型本身的推理能力
RAG	检索预先构建的外部知识库, 保证LLM输出内容的实时性和相关性		保证用户隐私数据安全性; 外部信息的引入, 提高了LLM回答的准确性和确定性	若搭建的知识库文档质量较差, 可能会导致生成的回答不准确
APE	将指令视为“程序”, 对LLM生成的候选项进行评估, 从中挑选出得分最高的指令	微调和优化	减少了人工设计提示的需要, 节省了时间和资源	面临着算法复杂性和资源消耗的挑战
ART	利用LLM为任务自动生成中间推理步骤的思维链, 可实现无缝衔接外部专业知识或工具的能力	基于知识推理和生成	用户可以更新任务、为工具库来添加新工具或修复错误, 提高了框架的可扩展性和灵活性	效率和准确性很大程度受限于外部工具的性能, 面对未见过的任务时存在限制
ReAct	允许LLM以交错的方式实现生成推理痕迹和特定任务行动的协同作用		根据环境反馈动态调整行动计划, 提高了任务解决的灵活性, 能够更有效地利用内部知识和外部信息	有效性依赖于外部资源的可用性和准确性; 一旦推理过程中出现错误, 可能会导致行动计划的连锁错误

© 中国科学院软件研究所 http://www.c-s-a.org.cn

2.1 思维链 (chain of thought, CoT)

思维链通过模拟人类解决问题时的思维过程, 通过一系列逻辑推理步骤来引导模型生成答案. 其核心在于将复杂问题分解为多个更小、更易于处理的子问题. 通过逐步解决这些子问题, 模型能够构建一条逻辑推理的链条, 最终得出答案. 这种方法不仅提高了模型解决问题的准确性, 还增强了其可解释性.

零样本或少样本的 CoT 是两种典型的 CoT 代表. zero-shot CoT 不需要提供具体的示例, 而是在问题描述的结尾添加特定的提示, 如“让我们一步步来思考”, 来引导模型生成逐步推理的过程. 这种方法利用了模型的零样本学习能力[18]. few-shot CoT 提供少量的示例. 模型需要根据这些有限的示例生成推理过程. 这种方法结合了示例的引导和模型自身的推理能力, 效果优于初级的 few-shot 方法, 但是提示内容撰写起来也会更加复杂[19].

CoT 适用于各种推理任务, 比如: 数学问题、符号操作、常识推理等[20,21]. CoT 的优势在于不用对模型进行训练和微调, 通过模拟人类解决问题的思维过程, 显著提升了 LLM 在复杂任务上的表现. 然而, 这种技术也有其局限性, 比如需要模型规模足够大才能有效实现, 且应用领域有限, 目前主要在数学问题和常识推理等领域表现出优势. 此外, 结合了 CoT 的 LLM 在处理一些简单的计算问题时仍然可能出现错误, 说明模型并没有真正理解数学逻辑, 而是通过模仿和叠加来生成答案.

2.2 自洽性 (self consistency, SC)

相比于 CoT 中的贪婪解码策略, 自洽性从 LLM 的解码器采样生成多样化的推理链路, 采用多数投票来提高答案的准确性[22]. 值得注意的是, 自洽性可以与大多数采样算法集成, 如温度采样[23]、top-K 采样[24].

自洽性作为一种推理增强技术, 特别适用于那些需要从不同角度进行推理的复杂问题和不确定性问题解答等场景[25]. 它是一种无需额外人工标注、训练微调或模型更改的高级提示词技术, 在不同的采样策略和参数设置下均表现稳定、出色, 有效提升了模型的鲁棒性和解释性[26]. 不足的是, 生成多个推理链路并进行投票需要更多的计算资源, 最终结果也依赖于模型本身的推理能力.

2.3 思维树 (tree of thought, ToT)

Yao 等人[27]扩展了 CoT 方法, 提出思维树框架, 用以解决需要探索、策略性前瞻或初始决策起关键作用的复杂任务. 实验结果表明, ToT 在 24 点游戏中取得了 74% 的成功率, 显著优于 CoT 的 4%. 在单词级任务中, ToT 以 60% 的成功率超过了 CoT 的 16%. 展示了 ToT 的显著效果. ToT 将中间推理步骤结构化为树状形式, 即“思维”, 每个“思维”代表了朝着最终解决方案前进的连贯语言序列, 通过考虑多种不同的推理路径和自我评估选择来执行深思熟虑的决策制定, 必要时还可以向前看或回溯以做出全局选择.

ToT 框架由 4 个关键步骤组成: 1) 思考分解: 将问题解决过程分解为多个中间思考步骤, 每个步骤都足够小, 以便 LLM 可以生成有希望且多样化的样本; 2) 思考生成器: 根据当前的状态生成潜在的思考步骤; 3) 状态评估: 使用启发式方法来评估每个状态, 判断其是否接近解决方案; 4) 搜索算法: 利用搜索算法, 如广度优先搜索 (BFS) 或深度优先搜索 (DFS), 在树状结构中探索不同的思考路径, 并选择最有前景的路径.

ToT 适用于需要复杂规划或搜索的问题解决任务, 如数学推理挑战、策略游戏和填字游戏等[27,28]. ToT 的优势体现如下: 1) 普适性: CoT 和 CoT-SC 方法可以视为 ToT 的特例, 因此, 适用于解决各种任务; 2) 模块化: LLM 以及思考分解、生成、评估和搜索过程都解耦, 每个模块可自由适配; 3) 方便性: 无需额外的训练. 与 CoT 类似, ToT 方法在执行搜索时会产生大量的 API 调用, 会增加计算成本. 此外, ToT 框架的实现相对复杂, 需要精心设计搜索算法和状态评估机制.

2.4 思维图 (graph of thought, GoT)

思维图模拟人类通过组合不同思想来解决问题的过程, 将 LLM 的输出视为一个图结构, 包含多个节点和边. 每个节点代表一个“思维”, 即模型在推理过程中的一个步骤. 节点之间的边代表这些思维的依赖关系. 这种方法不仅能够合并来自不同推理路径的思维, 还能通过递归和并行探索增强模型的推理能力[29].

GoT 的实现涉及几个关键组件: 1) 提示器: 将图结构转换为 LLM 的提示文本; 2) 解析器: 从 LLM 的响应中提取关键信息以更新图; 3) 评分模块: 评估每个思维的质量; 4) 控制器: 决定扩展图的策略, 并从 LLM 提示下一个思想.

相比于 CoT 和 ToT, GoT 优势表现为: 1) 组合多条推理链: GoT 能够将不同推理路径的思维组合成一

© 中国科学院软件研究所 http://www.c-s-a.org.cn

个更优的解决方案; 2) 递归思维精炼: GoT 允许通过反馈循环来反复精炼一个思维; 3) 并行探索: GoT 可以同时探索多个独立的推理路径; 4) 可解释性: 图模型的推理过程提供了透明度, 有助于解释和调试; 5) 泛化: GoT 泛化了 CoT 和 ToT, 提供了更广泛的应用范围. 虽然 GoT 系统化地提升了 LLM 效果, 但需要对任务有深刻的理解并精心设计提示, 以实现最佳结果.

2.5 自动思维链 (automatic chain of thought, Auto-CoT)

人工创建 CoT 示例既耗时又不太理想, Auto-CoT 是一种自动创建带有问题和推理链示例的过程. Auto-CoT 方法包含两个主要阶段: 首先, 将数据集中的问题进行聚类, 形成若干问题类簇; 接着, 从每个类簇中筛选一个问题作为代表, 并采用 zero-shot CoT 技术和简单启发式规则为其生成推理链. 相比于其他方法, Auto-CoT 具有自动化、可扩展性强且效果显著的优势, 生成的问题示例和推理链路准确且信息量丰富[30].

2.6 生成知识提示 (generated knowledge prompting, GKP)

生成知识提示方法不需要特定任务的知识整合, 也不需要访问结构化的知识库和微调模型, 而是从 LLM 中生成知识, 并将这些知识作为额外的输入来回答问题. Liu 等人[31]的研究表明, 融合了 GKP 的大模型在数值常识、通用常识和科学常识基准测试中均取得了最佳结果.

此外, 一些提示工程插件通过对用户输入进行分析, 根据上下文语境产生相应的输出, 避免用户编写复杂的提示工程, 使 LLM 更有效的理解用户意图. 例如, 名为“Prompt Perfect”的插件[32], 在输入以“perfect”开头的提示, 实现对原始提示内容的增强.

2.7 自动提示工程 (automatic prompt engineering, APE)

受经典程序合成和人类提示工程方法启发, Zhou 等人[33]提出自动提示工程用以自动生成和选择指令. APE 将指令视为“程序”, 通过搜索由 LLM 提出的指令候选池进行优化, 并对 LLM 生成的候选项进行评估, 从中挑选出得分最高的指令.

APE 可以应用于多种自然语言处理任务, 如问答系统、翻译、代码生成等. 通过自动化的方法快速生成和选择有效的提示, 减少了人工设计提示的需要, 节省了时间和资源, 但同时也面临着算法复杂性和资源消耗的挑战. 尽管如此, APE 在提升模型性能方面展现出了巨大的潜力.

2.8 自动多步推理与工具使用 (automatic multi-step reasoning and tool-use, ART)

ART[34]利用 LLM 为任务自动生成中间推理步骤的思维链, 可以依赖外部工具来支持 LLM 功能之外的计算, 如搜索或运行代码, 实现无缝衔接外部专业知识或工具的能力.

ART 框架的实现可分为两个阶段: 任务库选择和任务执行. 在任务库选择阶段, ART 根据特定任务解析中间推理步骤, 并从工具库中筛选合适的工具及其演示范例. 随后, ART 依赖任务执行过程中外部工具调用状态, 决定 LLM 是否暂停生成, 并将工具的输出内容进行整合, 从而借助外部工具解决复杂任务.

ART 作为一种新兴的技术, 在多个基准测试中优于传统的少样本提示和 Auto-CoT, 包括自然语言推理、问答和代码生成等. 此外, 用户可以更新任务、为工具库来添加新工具或修复错误, 提高了框架的可扩展性和灵活性. ART 的效率和准确性很大程度受限于外部工具的性能, 使其面对未见过的任务时存在限制.

2.9 检索增强生成 (retrieval augmented generation, RAG)

LLM 颠覆了文本生成的范式, 但受限于训练数据和模型本身性能, 无法回答最新的信息或在没有答案的情况下提供虚假信息. RAG[35]作为一种结合了检索和 LLM 生成的自然语言处理技术, 通过检索预先构建的外部知识库, 保证 LLM 输出内容的实时性和相关性.

RAG 构建可分为 3 个关键步骤: 1) 搭建私有知识库: 收集与特定领域或任务相关的文档, 文档可以是各类文本文件、网页或数据库; 对收集到的文档进行预处理, 包括去除无关内容、切块; 为切块的文档向量化, 建立索引, 并存储于向量数据库中. 2) 向量检索: 将用户的查询内容向量化, 并从向量数据库中检索最相似的内容作为下一步的输入. 3) 模型输出: 将选定的文档片段与用户的原始查询融合, 将其输入 LLM 中, LLM 会根据增强版的提示给出响应.

RAG 特别适用于需要引用权威知识库或私有数据的场景, 因引入了额外的信息作为输入, 提高了 LLM 回答的准确性和确定性. 若搭建的知识库文档质量较差, 可能会导致生成的回答不准确.

2.10 推理与行动 (reason and act, ReAct)

常规的 LLM 提示框架将推理和行动分开对待的, ReAct[36]允许 LLM 以交错的方式实现生成推理痕迹和

© 中国科学院软件研究所 http://www.c-s-a.org.cn

特定任务行动的协同作用. 推理痕迹有助于模型归纳、追踪和更新行动计划, 以及处理异常, 而行动使其能够从知识库或环境进行交互, 收集额外信息.

ReAct 的应用场景非常广泛, 包括但不限于问答、事实验证、基于文本的游戏和网页导航. 相比于单独使用推理或行动, 在上述场景中显示出更好的效果. 因其能够根据环境反馈动态调整行动计划, 提高了任务解决的灵活性, 能够更有效地利用内部知识和外部信息. 值得注意的是, ReAct 的有效性依赖于外部资源的可用性和准确性. 此外, 一旦推理过程中出现错误, 可能会导致行动计划的连锁错误.

3 提示词工程的评估方法

评估方法的目标是解决如何判断上述各类提示方法在 LLM 中的有效性问题. 评估方法主要分为自动评估、人工评估、大模型评估和对比评估方法. 自动评估方法通过直接比较正确答案与 LLM 生成结果计算得出, 如准确率、召回率等. 然而, 有些指标无法直接计算, 需人工阅读并按一定的准则打分, 如评估文章的质量时, 需考虑文章的流畅性、逻辑性、新颖性和准确性等因素. 鉴于人工评估方法耗时耗力, 研究人员提出大模型评估方法, 借助大模型强大的自然语言处理和推理能力, 构建合适的提示词指令进行评估. 此外, 以系统化的方式评估不同提示方法的差别, 可采用对比评估方法. 本节将对上述 4 类评估方法分别介绍.

当前的 LLM 在经历强化学习和人工对齐后, 在各种不同任务中均表现出色. 对于经典的自然语言理解任务, 可采用原有的自动评估体系. 在分类任务中, 通常采用精确度 (Precision)、召回率 (Recall)、准确率 (Accuracy)、 PR 曲线等; 在回归任务中, 通常采用均方误差 (mean squared error, MSE)、平均绝对误差 (mean absolute error, MAE)、均方误差根 (root mean squared error, RMSE); 在机器翻译任务中, 通常使用双语评估替换 (bilingual evaluation understudy, BLEU)[37]用于评估 LLM 生成内容和参考翻译句子之间的差异; 在文本摘要任务中, 采用面向召回率的要点 (recall-oriented understudy for gisting evaluation, ROUGE) 评估方法, 根据 LLM 生成的候选摘要和标准摘要之间的匹配度为候选摘要评分[38].

相比于自动评估方法, 人工评估被广泛应用于评估 LLM 生成结果质量和准确性中, 更接近于实际应用场景. 在人工评估中, 可以对 LLM 结果进行整体质量评估, 也可以从语法、语义等细粒度评分. 人工评估通常涉及 4 个方面: 评估人员类型、是否给定参考信息、评估度量指标以及评估者是否提供解释. 对于每条测试数据, 不同类型的评估人员都会参与打分. 因此, 最终打分可采用平均法、中位数等方式给出. 此外, 不同打分者之间的评估一致性也是重要的考量因素, 一致性不仅可以作为模型的反馈机制, 也能够判断评估人员的标注质量. 人工评估的不足也显而易见, 首先, 人工评估需要大量的时间、精力, 成本高, 周期长. 其次, 评估的结果可能受到评估者主观性和认知性差异影响.

大模型评估方法适用于无法提供参考资料、具有创造性和多样性的任务. 大模型评估的过程相对简单, 以文本质量评估为例, 用户准备好任务说明、评测样本以及 LLM 的指令作为输入, LLM 将生成一些输出句子来回答问题, 并解析输出句子以获取评分. 用户可以根据任务的不同, 设定不同的提示词内容生成问题, 以评估 LLM 的效果. 在文献[39]中, 为评估故事生成任务的文本质量, 设计了语法正确性、连贯性、相关性和喜好度 4 种属性. 研究表明, 在开放式故事生成和对抗性攻击两个任务上, 大语言模型取得与人工评估一致的效果.

对比评估的目标是比较不同提示方法或大模型在特定任务上的差异. 越来越多的模型用来对其他模型的输出质量进行评级. 例如 LLM-Eval[40]通过单一提示和统一的框架, 从多维度指标评估开放领域对话质量, 包括内容、语法、相关性和适当性. 为了验证有效性和效率, LLM-Eval 对比了最先进的评估方法在各种基准数据集的表现.

4 提示词工程的应用

由前文中介绍的提示词工程的技术、评估方法可见. 提示词工程提供的增强输出使其更适用于现实世界, 本节将介绍提示词工程的工具及不同场景的应用.

4.1 教育领域

Tang 等人[41]研究了机器学习方法在学生教育中的应用限制, 发现存在教师的技能和知识充分融合入中算法中的问题. 为此, 借助了 LLM 丰富的知识储备和定制化的提示内容, 可实现教师技能高效复刻、学生课程内容个性化定制等功能. 此外, 通过提示工程技巧, 创建自动问题生成和评估系统, 大幅减轻教育工作

© 中国科学院软件研究所 http://www.c-s-a.org.cn

者的负担, 也为学生提供即时反馈[42]. 更进一步, 为了强调提示词工程对于科研工作人员的重要性, 文献[43, 44]介绍了提示词工程的概念、LLM 以及编写提示的技巧和陷阱, 以此帮助科研人员提升写作技巧、探索学术前沿领域.

4.2 医疗领域

提示词工程使 LLM 能够在医疗领域高质量地完成多种任务. 通过定制医疗领域提示信息, 模型可以解决复杂医疗问题, 如疾病诊断、医学图像检测、报告生成等, 并且无需大量的人力、数据等资源. GPT4MIA[45]利用 GPT-3 作为医学图像分析 (MIA) 分类任务的诊断模型, 开发了改进的提示结构设计、样本选择和提示排序 3 种提示工程技术, 以提高 GPT4MIA 在检测预测错误和提高图像分类准确性方面的效率和效果. Napss[46]提出一种“先总结再简化”的策略, 在保留原始叙述流程的同时, 也能识别需要简化的相关内容, 保证医学报告生成的准确、有效. 为了确保简化文本的叙述一致性, 采用了辅助叙述提示指令, 从原始文本的句法分析中提取关键短语. Qin 等人[47]展示了提示词工程如何使预训练的视觉语言模型具备医学图像检测知识.

4.3 数据生成

LLM 拥有强大的上下文学习能力, 通过提示词工程可合成数据集, 用于 LLM 或训练更小的模型. 例如, Aphaca 模型通过 Self-instruct[48]指令技术生成 52k 数据, 在性能表现上与 GPT-3.5 相当. Ding 等人[49]将 GPT-3 与 3 种不同的提示词方法融合, 作为数据标注工具, 探索在不同 NLP 任务中的表现. 实验结果表明, 融合了提示词的 LLM 有潜力以相对较低的成本为不同的任务进行数据标注.

4.4 代码生成

LLM 在代码生成方面表现出令人惊叹的能力. LLM 接受提示词输入, 在 CoT 这种先进的提示技术下生成代码. 然后, CoT 的设计更符合人类思考的逻辑, 在代码生成上存在局限. 文献[50]提出了结构化链式思维 (structured CoTs, SCoTs), 其动机为任何代码由 3 种程序结构组成 (顺序、分支和循环). 直观地说, SCoTs 根据程序结构来构建思维链, LLM 据此生成最终的代码. Nijkamp 等人[51]研究了代码合成的多步骤范式, 代码可以被分解为多个指定子问题的提示, 提出了多轮编程基准 (multi-turn programming benchmark, MTPB). 研究表明, 以多轮方式提供给 LLM 的相同意图显著提高了程序合成的性能

4.5 提示词工具

除了前文介绍的“Prompt Perfect”插件, 更多的提示词工具将被应用于 LLM 中, 提供增强的输入改善 NLP 任务处理能力. Langchain[52]将 LLM 与外部知识来源相结合的程序框架, 可有效缓解 LLM 的幻觉、知识更新不及时等问题. Promptify[53]根据特定任务生成提示词, 让开发者更专注于问题本身.

5 总结与展望

提示词工程技术分类、评估方法和应用如图 1 所示. 创建高质量的提示词是一个需要不断测试、评和优化的过程. 遵循这些步骤和考虑因素, 可设计出更有效的提示词, 从而提升 LLM 的性能.

尽管提示词工程在不同任务和场景中表现出巨大的潜力, 仍然存在一些方向值得进一步探索. 具体如下.

1) 动态提示词生成: 旨在根据用户的实时反馈和上下文动态调整提示词. 例如, 通过在线学习机制, 系统能够在与用户交互的过程中逐步优化提示, 使其更加符合用户需求. 这种自适应能力可以提升用户体验和任务完成的效率.

2) 多模态提示词集成: 探索将文本提示词与图像、音频、视频等多模态信息结合的方式, 以更全面地表达用户意图. 这不仅可以提升 LLM 对复杂场景的理解能力, 还能为创意生成、情感分析等领域带来新的突破.

3) 用户个性化与上下文感知: 研究如何基于用户的历史交互和偏好, 提供个性化的提示词. 结合上下文信息, 系统可以更准确地推测用户意图, 提升交互的自然性和有效性.

4) 提示指令组合和分解: 设计一种由提示词生成器、评估器和选择器构成的动态分解机制, 能够根据任务复杂度和 LLM 当前能力自动调整提示词的组合与分解, 借助预测模型性能和任务依赖关系, 实现最优方案的迭代优化.

5) 社会伦理与偏见检测: 探讨提示词工程在社会伦理和公平性方面的挑战, 开发自动化的评估和自动优化工具, 以定量和定性地分析不同提示词的效果, 确保其在多元文化和社会背景下的适用性, 以降低潜在的偏见和歧视.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

6) 跨领域知识迁移: 借助迁移学习或域自适应技术, 深入探索如何在不同领域之间有效迁移提示词, 通过构建领域间的知识图谱, 识别相似的任务结构, 从而提高新领域中的任务执行能力.

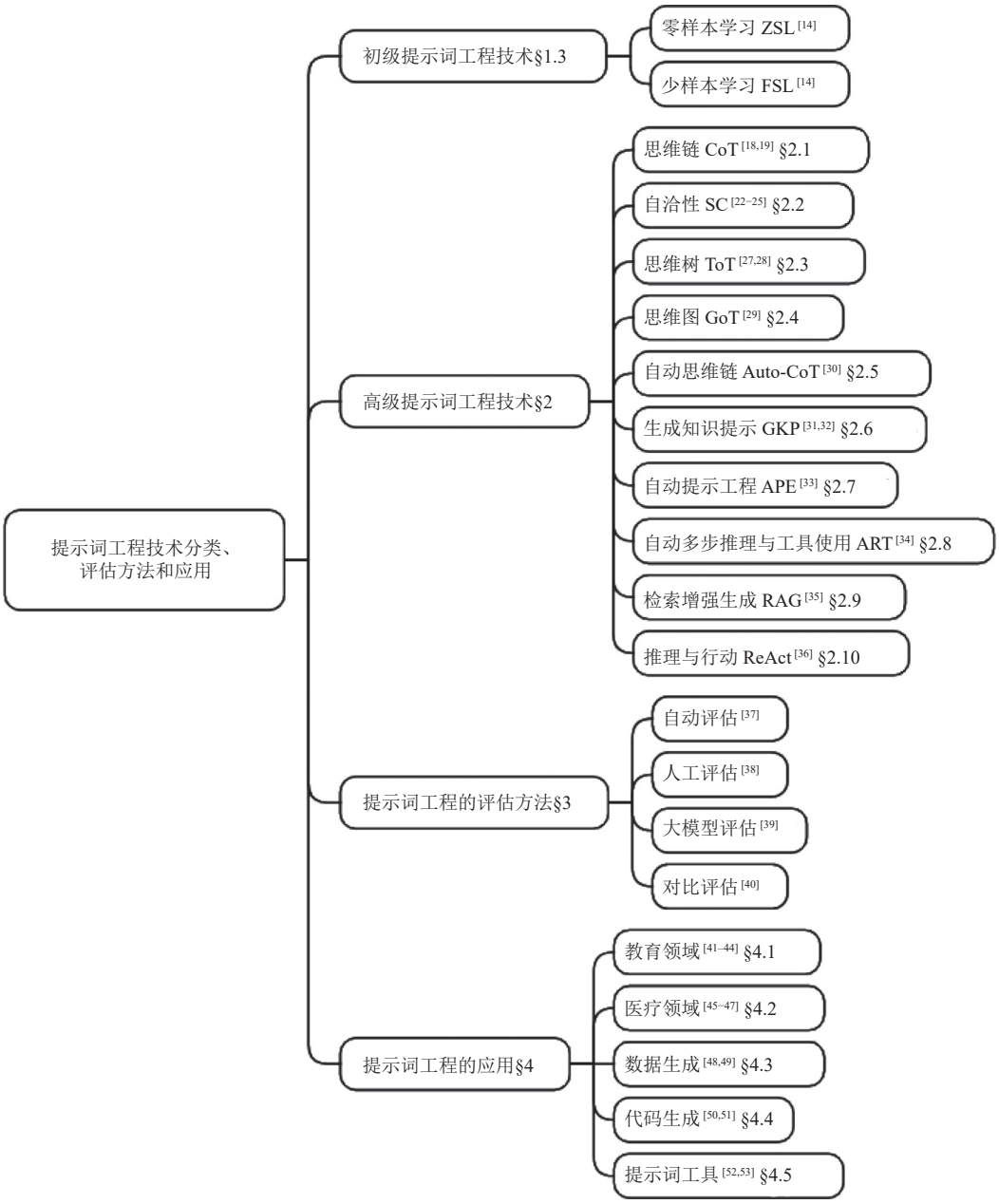

图 1 提示词工程技术分类、评估方法和应用

6 结束语

提示词工程在 LLM 的影响和重要性是不言而喻的, 可以视为 NLP 中另一次重要的技术变革. 因此, 系统性地阐述了提示词工程的基础和高级的技术方法, 剖析了不同方法的优缺点、适用场景, 并介绍了判定这些提示技术性能的评估方法. 此外, 讨论了提示词工程在不同领域中的应用, 如教育、医疗和编程等领域, 对提示词工程的发展方向进行了展望. 最后, 期望这篇综述帮助研究人员更全面地理解提示词工程, 以便在该领域取得新的研究成果.

参考文献

1 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. Proceedings of the 31st International Conference on

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 30.
2 Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 1877–1901.
3 Achiam J, Adler S, Agarwal S, *et al*. GPT-4 technical report. arXiv:2303.08774, 2023.
4 Touvron H, Martin L, Stone K, *et al*. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288, 2023.
5 Zeng AH, Liu X, Du ZX, *et al*. GLM-130B: An open bilingual pre-trained model. arXiv:2210.02414, 2022.
6 Kaddour J, Harris J, Mozes M, *et al*. Challenges and applications of large language models. arXiv:2307.10169, 2023.
7 Lu Y, Bartolo M, Moore A, *et al*. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Dublin: Association for Computational Linguistics, 2022. 8086–8098.
8 Maynez J, Narayan S, Bohnet B, *et al*. On faithfulness and factuality in abstractive summarization. arXiv:2005.00661v1, 2020.
9 Bubeck S, Chandrasekaran V, Eldan R, *et al*. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv:2303.12712, 2023.
10 Short CE, Short JC. The artificially intelligent entrepreneur: ChatGPT, prompt engineering, and entrepreneurial rhetoric creation. Journal of Business Venturing Insights, 2023, 19: e00388. [doi: 10.1016/j.jbvi.2023.e00388]
11 Strobelt H, Webson A, Sanh V, *et al*. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. IEEE Transactions on Visualization and Computer Graphics, 2022, 29(1): 1146–1156.
12 Yang M, Qu Q, Tu WT, *et al*. Exploring human-like reading strategy for abstractive text summarization. Proceedings of the 33th AAAI Conference on Artificial Intelligence. Honolulu: AAAI Press, 2019. 7362–7369.
13 Feng YCJ, Wang XB, Wong KK, *et al*. Promptmagician: Interactive prompt engineering for text-to-image creation. IEEE Transactions on Visualization and Computer Graphics, 2024, 30(1): 295–305.
14 Reynolds L, McDonell K. Prompt programming for large language models: Beyond the few-shot paradigm. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. Yokohama: ACM, 2021. 1–7.
15 Wang L, Chen X, Deng XW, *et al*. Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. npj Digital Medicine, 2024, 7(1): 41. [doi: 10.1038/s41746-024-01029-4]
16 Shah C. From Prompt engineering to prompt science with human in the loop. arXiv:2401.04122, 2024.
17 Zhang Z, Gao J, Dhaliwal RS, *et al*. VISAR: A human-AI argumentative writing assistant with visual programming and rapid draft prototyping. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. San Francisco: ACM, 2023. 5.
18 Kojima T, Gu SS, Reid M, *et al*. Large language models are zero-shot reasoners. Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2024. 1613.
19 Del M, Fishel M. True detective: A deep abductive reasoning benchmark undoable for GPT-3 and challenging for GPT-4. Proceedings of the 12th Joint Conference on Lexical and Computational Semantics. Toronto: Association for Computational Linguistics, 2023. 314–322.
20 Cobbe K, Kosaraju V, Bavarian M, *et al*. Training verifiers to solve math word problems. arXiv:2110.14168, 2021.
21 Huang SH, Dong L, Wang WH, *et al*. Language is not all you need: Aligning perception with language models. Proceedings of the 37th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2024. 3155.
22 Chowdhery A, Narang S, Devlin J, *et al*. PaLM: Scaling language modeling with pathways. The Journal of Machine Learning Research, 2024, 24(1): 240.
23 Ficler J, Goldberg Y. Controlling linguistic style aspects in neural language generation. Proceedings of the 2017 Workshop on Stylistic Variation. Copenhagen: Association for Computational Linguistics, 2017. 94–104.
24 Holtzman A, Buys J, Forbes M, *et al*. Learning to write with cooperative discriminators. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. Melbourne: Association for Computational Linguistics, 2018. 1638–1649.
25 Shum KS, Diao SZ, Zhang T. Automatic prompt augmentation and selection with chain-of-thought from labeled data. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: ACL, 2023. 12113–12139.
26 Wang XZ, Wei J, Schuurmans D, *et al*. Self-consistency improves chain of thought reasoning in language models. arXiv:2203.11171, 2022.
27 Yao SY, Yu D, Zhao J, *et al*. Tree of thoughts: Deliberate problem solving with large language models. Proceedings of the 37th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2024. 517.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

28 Long JY. Large language model guided tree-of-thought. arXiv:2305.08291, 2023.
29 Besta M, Blach N, Kubicek A, *et al*. Graph of thoughts: Solving elaborate problems with large language models. Proceedings of the 38th AAAI Conference on Artificial Intelligence. Vancouver: AAAI Press, 2024. 17682–17690.
30 Zhang ZS, Zhang A, Li M, *et al*. Automatic chain of thought prompting in large language models. arXiv:2210.03493, 2022.
31 Liu JC, Liu A, Lu XM, *et al*. Generated knowledge prompting for commonsense reasoning. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Dublin: ACL, 2022. 3154–3169.
32 Prompt perfect plugin for ChatGPT. https://chatonai.org/prompt-perfect-chatgpt-plugin. [2024-08-19].
33 Zhou YC, Muresanu AI, Han ZW, *et al*. Large language models are human-level prompt engineers. arXiv:2211.01910, 2022.
34 Paranjape B, Lundberg S, Singh S, *et al*. ART: Automatic multi-step reasoning and tool-use for large language models. arXiv:2303.09014, 2023.
35 Lewis P, Perez E, Piktus A, *et al*. Retrieval-augmented generation for knowledge-intensive NLP tasks. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 793.
36 Yao SY, Zhao J, Yu D, *et al*. ReAct: Synergizing reasoning and acting in language models. arXiv:2210.03629, 2022.
37 Papineni K, Roukos S, Ward T, *et al*. BLEU: A method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting of the Association for Computational Linguistics. Philadelphia: Association for Computational Linguistics, 2002. 311–318.
38 Lin CY. ROUGE: A package for automatic evaluation of summaries. Proceedings of the 2004 Workshop on Text Summarization Branches Out. Barcelona: Association for Computational Linguistics, 2004. 74–81.
39 Chiang CH, Lee HY. Can large language models be an alternative to human evaluations? Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto: Association for Computational Linguistics, 2023. 9–14.
40 Lin YT, Chen YN. LLM-Eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. Proceedings of the 5th Workshop on NLP for Conversational AI. Toronto: Association for Computational Linguistics, 2023. 47–58.
41 Tang JW, Zhou XF, Wan XY, *et al*. ML4STEM professional development program: Enriching K-12 STEM teaching with machine learning. International Journal of Artificial Intelligence in Education, 2023, 33(1): 185–224. [doi: 10.1007/s40593-022-00292-4]
42 Ariely M, Nazaretsky T, Alexandron G. Machine learning and Hebrew NLP for automated assessment of open-ended questions in biology. International Journal of Artificial Intelligence in Education, 2023, 33(1): 1–34.
43 Giray L. Prompt engineering with ChatGPT: A guide for academic writers. Annals of biomedical engineering, 2023, 51(12): 2629–2633.
44 Lee U, Jung H, Jeon Y, *et al*. Few-shot is enough: Exploring ChatGPT prompt engineering method for automatic question generation in english education. Education and Information Technologies, 2024, 20(9): 11483–11515.
45 Zhang YZ, Chen DZ. GPT4MIA: Utilizing generative pre-trained Transformer (GPT-3) as a plug-and-play transductive model for medical image analysis. Proceedings of the 2023 Workshops of the Medical Image Computing and Computer Assisted Intervention. Cham: Springer, 2023. 151–160.
46 Lu JR, Li JZ, Wallace B, *et al*. NapSS: Paragraph-level medical text simplification via narrative prompting and sentence-matching summarization. Proceedings of the 2023 Findings of the Association for Computational Linguistics: EACL 2023. Dubrovnik: Association for Computational Linguistics, 2023. 1079–1091.
47 Qin ZY, Yi HH, Lao QC, *et al*. Medical image understanding with pretrained vision language models: A comprehensive study. arXiv:2209.15517, 2022.
48 Wang YZ, Kordi Y, Mishra S, *et al*. Self-instruct: Aligning language models with self-generated instructions. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto: ACL, 2023. 13484–13508.
49 Ding BS, Qin CW, Liu LL, *et al*. Is GPT-3 a good data annotator? arXiv:2212.10450v2, 2023.
50 Li J, Li G, Li YM, *et al*. Structured chain-of-thought prompting for code generation. arXiv:2305.06599v3, 2023.
51 Nijkamp E, Pang B, Hayashi H, *et al*. Codegen: An open large language model for code with multi-turn program synthesis. arXiv:2203.13474, 2022.
52 LangChain introduction. https://python.langchain.com/v0.2/docs/introduction/. [2024-08-19].
53 Task specific GPT prompts consumable as APIs. https://www.promptify.ai/. [2024-08-19].

(校对责编: 王欣欣)

© 中国科学院软件研究所 http://www.c-s-a.org.cn