计算机科学
COMPUTER SCIENCE

汽车验证电控系统中的测试用例自动生成方法

李占旗, 吴新维, 张蕾, 刘全周, 谢辉, 熊德意

引用本文

李占旗, 吴新维, 张蕾, 刘全周, 谢辉, 熊德意. 汽车验证电控系统中的测试用例自动生成方法[J]. 计算机科学, 2024, 51(12): 63-70.

LI Zhanqi, WU Xinwei, ZHANG Lei, LIU Quanzhou, XIE Hui, XIONG Deyi. Automatic Test Case Generation Method for Automotive Electronic Control System Verification [J]. Computer Science, 2024, 51(12): 63-70.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240900093

汽车验证电控系统中的测试用例自动生成方法

李占旗[1,3,4]　吴新维[2]　张　蕾[1]　刘全周[1]　谢　辉[3]　熊德意[2]
1 中汽研(天津)汽车工程研究院有限公司　天津 300300
2 天津大学智能与计算学部　天津 300350
3 天津大学机械工程学院　天津 300354
4 中国汽车技术研究中心有限公司　天津 300300
(lizhanqi@catarc.ac.cn)

摘　要　随着“软件定义汽车”的发展,汽车软件功能的复杂性和快速开发需求对电控系统验证提出了更高的要求。当前,电控系统软件功能的测试流程图开发主要依赖人工方式,效率低且存在人为因素影响。文中详细描述了汽车验证电控系统中的测试用例自动生成任务及其面临的挑战,并提出了一种基于大语言模型(LLM)的自动生成测试流程图方法,以提高开发效率并减少人力成本。该方法包括构建领域任务数据集和选择合适场景的大模型应用路线。在实验中探讨了基于传统语言模型微调和大语言模型 API 适配两种技术路线的优劣,并通过实验验证了不同的大模型 API 在测试用例生成任务上的表现,以及提示工程技术对大模型 API 的提升效果。提出了一种高效的自动生成汽车测试流程图的方法,展示了大语言模型在提升汽车软件测试效率中的潜力。

关键词:汽车领域应用;大语言模型;提示工程
中图分类号　TP391

Automatic Test Case Generation Method for Automotive Electronic Control System Verification

LI Zhanqi[1,3,4],WU Xinwei[2],ZHANG Lei[1],LIU Quanzhou[1],XIE Hui[3] and XIONG Deyi[2]
1 CATARC(Tianjin)Automotive Engineering Research Institute Co.,Ltd.,Tianjin 300300,China
2 College of Intelligence and Computing,Tianjin University,Tianjin 300350,China
3 School of Mechanical Engineering,Tianjin University,Tianjin 300354,China
4 China Automotive Technology and Research Center Co.,Ltd.,Tianjin 300300,China

Abstract　With the development of “software-defined vehicles”,the complexity of automotive software functions and the demand for rapid development have imposed higher requirements on the verification of electronic control systems. Currently,the development of test flow charts for electronic control system software functions mainly relies on manual methods,which are inefficient and susceptible to human factors. This paper details the task and challenges of automatic test case generation in automotive electronic control system verification and proposes an automatic test flow chart generation method based on large language models (LLM) to improve development efficiency and reduce labor costs. The method includes constructing domain task datasets and selecting appropriate LLM application routes. The study explores the advantages and disadvantages of two technical routes:traditional language model fine-tuning and LLM API adaptation. Experiments validate the performance of different LLM APIs in test case generation tasks and the effectiveness of prompt engineering techniques in enhancing LLM API performance. In summary,this paper proposes an efficient method for automatically generating automotive test flow charts,demonstrating the potential of LLMs in improving the efficiency of automotive software testing.

Keywords　Automotive applications,Large language models,Prompt engineering

1　引言

大语言模型(Large Language Model,LLM)具有强大的交互与推理能力[1-3],这使得 LLM 的应用范围从基础的自然语言处理(NLP)任务扩展到高级功能,包括视觉问答、自动驾驶、医疗问诊等领域[4-5]。与早期的预训练语言模型[6-7]相比,LLM 的训练方法主要侧重于微调而非预训练[8-10]。通过监督微调和对齐来提升语言模型在特定任务上的指令遵循能力

到稿日期:2024-09-13　返修日期:2024-11-07
基金项目:国家重点研发计划(2021YFB3202204)
This work was supported by the National Key Research and Development Program of China(2021YFB3202204).
通信作者:熊德意(dyxiong@tju.edu.cn)

并让语言模型输出符合人类预期。其中指令数据用以引导模型生成符合格式的输出内容，并通过人类反馈进行强化学习以提高输出质量[11-12]。

随着“软件定义汽车”[13]进程加速，软件和算法将是车企竞争的核心要素，汽车软件功能的复杂性及开发要求的快速性对电控系统验证提出了更高的要求。测试用例是电控系统功能及故障安全策略等需求侧规范映射到验证侧的规范，是电控系统验证实施开展的依据，最终需要以流程图的形式呈现。目前，针对电控系统软件功能的测试流程图开发基本是由人工完成，这种方式效率较低，且开发过程中存在许多人为因素，不能保证用例开发的覆盖率和有效性。汽车软件快速开发及迭代更新对电控系统验证提出了严峻挑战，如何快速、高效地开发测试流程图，是应对挑战亟待解决的关键问题。

本文聚焦于汽车领域中的测试流程图生成问题，尝试引入 LLM 来替代汽车工程师，从而提高效率并节省大量的人力和时间成本。具体来说，本文首先描述了汽车验证电控系统中的测试用例自动生成任务及其面临的挑战，包括无法直接输出流程图、缺少车辆专业知识、逻辑推理任务困难和样本数量稀少的问题。在此基础上，设计并构建了一种自动生成测试流程图的框架，包括构建领域任务数据集、选择合适的技术路线以及具体的方法构建流程。

在方法上，本文介绍了两种大模型在领域任务上的应用技术路线：基于传统语言模型的微调路线和基于大语言模型 API 的适配路线。在微调路线下，本文介绍了基础的语言模型结构，以及如何通过全量监督微调和 LoRA 微调技术使语言模型适配汽车领域任务。在 API 适配路线下，介绍如何使用大模型 API，以及根据提示工程技术构建汽车领域任务的提示词。

在实验部分，本文使用由汽车领域专家标注的 2 892 条测试用例生成样本作为实验数据集。通过分析实验结果，证明了在语料充足的情况下，微调路线将取得更好的表现；而在小样本场景下，API 适配路线也可以取得不错的生成结果。此外，实验验证了不同大模型 API 的性能，ChatGPT 取得了最佳的表现。本文还进一步探讨了提示词在提升大模型 API 性能中的作用，包括角色扮演提示、思维链提示和小样本提示的影响。

综上所述，本文在研究汽车测试流程图生成任务的过程中，结合大语言模型（LLM）和提示工程技术，提出了一种高效的自动生成方法。本文的主要贡献包括：

1）引入大语言模型用于汽车测试流程图生成。本文首次尝试将大语言模型应用于汽车测试流程图生成任务。利用大语言模型的强大交互与推理能力，替代传统的人工流程图绘制，提高了开发效率并节省了大量的人力和时间成本。

2）构建了大规模的汽车领域任务数据集。由汽车领域专家标注了 2 892 条测试用例生成样本，其中 2 600 条作为训练数据，292 条作为测试数据，为模型训练和评估提供了充足的数据支持。

3）验证不同大模型应用技术路线在测试用例生成任务上的表现。面对传统语言模型微调技术路线与大模型 API 适配技术路线，本文通过实验验证了两种框架的有效性，并总结出不同技术路线的适用场景。

2 问题描述与解析

在汽车电控系统的研究领域，快速且精确地生成测试流程图至关重要。这不仅能显著提高开发效率，还能确保系统的可靠性和安全性。目前，大多数测试流程图仍依赖于人工绘制，这一过程不仅耗时且容易出错。因此，引入先进的大语言模型来自动化此过程显得尤为重要。本章通过详细描述并分析问题，明确任务与挑战。

2.1 问题描述

以下是汽车测试用例生成任务的问题定义与输入输出的数据格式。

1）问题定义

该任务的形式是根据输入文本生成相应的流程图。输入文本为车辆领域电控系统功能需求规范描述，模型需要从中提取出规范描述中的关键信息，最终按照既定的语法规则生成测试目标功能的结构化流程图。整个生成过程涉及多个推理子任务，包括提取车辆初始状态、提取输入输出信号、推理功能逻辑和半结构化数据转换。

2）数据格式

输入数据：面向车辆领域电控系统开发过程中的企业规范、标准规范、经验场景等多维数据源的自然语言输入。表 1 列出了一段描述车身系统位置灯控制功能的规范文本。

表 1 车身系统功能输入文本的示例

Table 1 Example of input for automobile test system

输入文本
当同时满足以下条件时，进入小灯工作模式，小灯点亮： 1. 点火开关在任何档位 2. 小灯开关接通

输出数据：将自然语言表述的设计规范转化为包含初始状态、输入输出信号、功能逻辑判定表达式、期望输出结果等节点在内的结构化流程图表达形式。图 1 给出了车身系统功能测试用例的流程图示例。

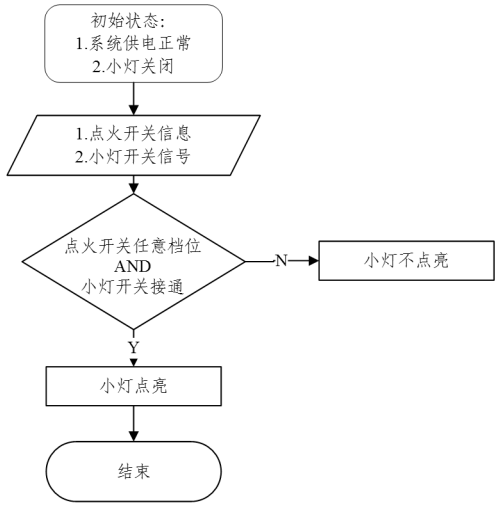

图 1 测试用例流程图的示例

Fig. 1 Example of a test case flow chart

2.2 问题解析

通过大语言模型将汽车功能的自然语言描述转换为对应

功能的流程图面临以下挑战:

1)无法直接输出流程图

语言模型的输入和输出只能处理文本模态的数据,这使得模型无法直接生成流程图。虽然可以通过文本描述流程图的各个元素,但现有的语言模型并不具备直接将这些描述转换为视觉化流程图的能力。即使采用图片生成模型,由于其对格式和结构的严格要求,也难以生成符合流程图规范的结果。因此,无法直接生成流程图成为一个技术难题,制约了模型在此领域的应用。

2)缺少车辆专业知识

常规语言模型在训练过程中缺乏车辆领域的专业知识,导致在处理相关任务时,模型对专有名词和特定术语的理解存在局限性。当模型遇到车辆电控系统中的复杂术语或特定功能描述时,往往会出现理解偏差,进而导致生成的内容不准确甚至错误。这种专业知识的缺乏,不仅影响了模型的准确性,也限制了其在专业领域的广泛应用。

3)逻辑推理任务困难

生成流程图涉及复杂的逻辑推理任务,这对于现有的语言模型来说是一个巨大挑战。模型需要在理解初始状态、输入输出信号及其关系的基础上,进行多步骤的逻辑推理。这一过程中,任何一个推理环节的错误都可能导致最终生成的流程图不符合实际需求。由于现有模型在处理复杂逻辑推理时的能力有限,因此难以确保所有推理步骤的准确性,进而影响了整个流程图生成的可靠性。

4)样本数量稀少

由于车辆测试用例的版权限制和高昂的人工成本,可用的样本数量非常稀少。这种数据不足的情况使得模型在进行任务学习和性能优化时受到极大限制。样本数量的稀少不仅影响了模型的训练效果,也制约了模型的泛化能力和实际应用。因缺乏足够的数据样本,模型难以充分学习车辆领域的具体任务流程,从而影响了其在实际应用中的表现。

3 方法设计与构建

本章详细介绍了针对汽车验证电控系统中的测试用例自动生成方法的设计与构建过程。首先,从整体流程设计入手,构建了领域任务数据集,并介绍了两种应用技术路线:基于传统语言模型微调和基于大语言模型 API 适配。接着,针对两种技术路线,通过详细阐述所采用的技术方法、模型架构及其实现细节,旨在提供一个高效且可靠的解决方案,以满足流程图生成的精确性和效率要求,适应不断变化的实际应用场景。

3.1 方法流程设计

本节将介绍汽车验证电控系统中的测试用例自动生成方法的流程设计,整体流程如图 2 所示。首先,构建领域任务数据集。然后,设计了两种不同的应用技术路线用于尝试效果,一种是基于传统语言模型微调的应用技术路线,另一种是基于大语言模型 API 适配的应用技术路线,两种路线代表着不同的语言模型应用的思路。在下一章的实验中,本文将展示不同技术路线的优劣。

3.1.1 构建领域任务数据集

为了方便用户理解,需要将识别结果按照既定的语法规则生成半结构化流程图。然而语言模型并不能直接处理流程图数据,需要将流程图转译为 Markdown 语法结构代码。以图 1 的流程图为例,其对应的 Markdown 代码如图 2 所示。

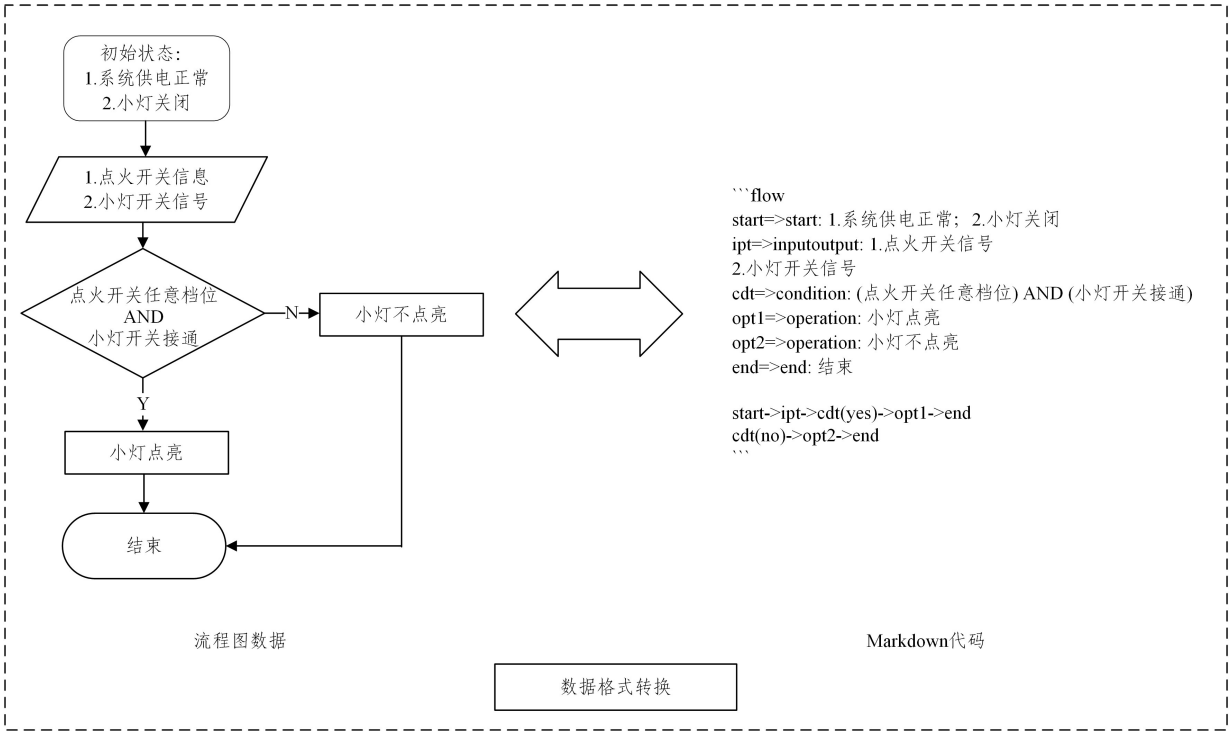

图 2 由流程图转化的 Markdown 代码

Fig. 2 Markdown code converted from flowchart

3.1.2 基于传统语言模型微调的技术路线

如图 3 所示，蓝色框内的是基于传统语言模型的应用技术路线。传统的语言模型采用微调的方式使语言模型获得下游任务的处理能力。首先，需要挑选基本的预训练语言模型。本文采用生成式模型中最常见的两种架构，Encoder-Decoder 结构语言模型和 Decoder-only 结构语言模型。然后，在领域任务数据集上对预训练语言模型进行微调，采用全量微调与 LORA 微调两种下游任务微调方式。

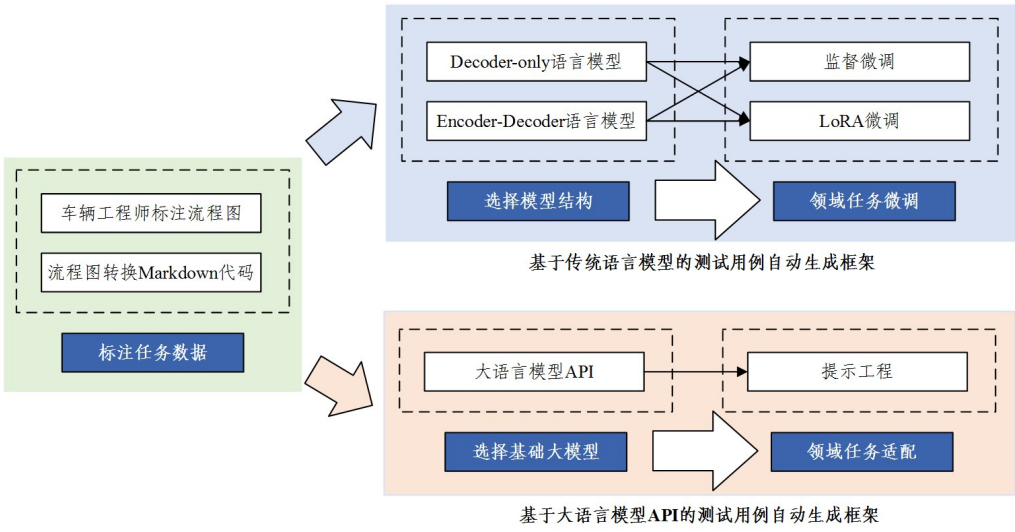

图 3 汽车验证电控系统中的测试用例自动生成方法(电子版为彩图)

Fig. 3 Automatic test case generation method for automotive electronic control system verification

3.1.3 基于大语言模型 API 适配的技术路线

第二种是基于大语言模型 API 的应用技术路线，如图 3 中橙色框图所示。大语言模型的小样本任务表现非常亮眼，通过提示工程等技术可以使大模型快速地获得处理下游任务的能力。首先，本文采用大模型 API 作为语言模型的来源，这样可以避免大模型部署与运行的高昂成本；其次，采用提示工程技术使大模型 API 获取领域任务的处理能力。

3.2 方法构建

在方法流程设计中，针对不同应用技术路线进行了初步探讨。为了具体实现这些路线，需要选择合适的模型结构并进行针对性的微调和优化。本节将详细介绍所选用的技术方法和模型架构，并描述如何利用这些技术构建一个高效、可靠的测试用例自动生成方法。

3.2.1 选择传统模型结构

为了追求更好的流程图代码生成结果，本文采用以下两种常见的语言模型结构进行尝试。

1) Encoder-decoder 语言模型

Encoder-Decoder[10] 结构是一种常用的序列转换架构，也是传统的 transformer 模型结构。该架构包括两个主要部分：编码器(encoder)和解码器(decoder)。编码器接收输入序列并将其转换为固定长度的上下文向量；解码器接收编码器生成的上下文向量，并生成相应的输出序列。将编码器的输出与解码器当前的状态结合起来，为每一步生成提供更丰富的信息。

Encoder-Decoder 语言模型主要应用场景包括机器翻译、文本摘要、问答系统、语音识别、图像描述等。目前基于编码器-解码器架构的语言模型有 BART，GLM，T5，Flan-T5 等。

2) Decoder-only 语言模型

Decoder-Only[11] 语言模型是一种基于 Transformer 架构的深度学习模型，与传统的 Encoder-Decoder 结构不同，其仅由解码器层组成，通过自回归方式生成序列。Decoder-Only 语言模型采用自回归方式生成序列，即每一步生成的输出会作为下一步的输入。

Decoder-Only 语言模型被广泛应用于各种自然语言生成任务中，包括文本生成、对话系统、自动补全文本、代码生成等。在流程图生成任务中，decoder-only 模型可以逐步生成描述各个流程步骤的文本，这些文本随后可以转换为流程图的各个节点和边。Decoder-only 语言模型是当前大型预训练语言模型的主流架构，目前主流的模型为 GPT 系列模型。

其中，Decoder-only 语言模型将上下文序列作为输入，通过自回归方式逐步生成输出序列中的下一个词；Encoder-Decoder 语言模型首先处理完整的输入序列，并基于编码器的输出逐步生成完整的输出序列。在后续实验中，本文采用端到端的任务设计思路，将提示词与车辆功能描述文本作为输入，将车辆测试用例流程图的 Markdown 代码作为输出。

3.2.2 领域任务微调

本文采用以下两种模型微调方法进行尝试。

1) 监督微调

监督微调(Supervised Fine-Tuning，SFT)是提升大型语言模型性能的关键步骤。其主要目的是通过在特定任务或领域数据上进行微调，使模型更好地适应这些任务或领域的特性，从而提高模型在该领域的准确性和效率。SFT 有助于模型捕捉到更细致的语言模式和特定任务的解决策略，使其在实际应用中更加精准和高效。

在 SFT 中，模型的目标是最小化损失函数，该函数衡量模型预测与真实标签之间的差异。常用的损失函数为交叉熵损失，其表达式如下：

$$\boldsymbol{L}(\boldsymbol{\theta})=-\frac{1}{N}\sum\sum y_{ij}\log(z_{ij})$$

其中，θ 是模型的参数，N 是训练样本的数量，C 是类别数，y_{ij} 是第 i 个样本的真实标签，z_{ij} 是模型预测的概率分布中的第 j 个类的概率。通过梯度下降等优化算法来更新模型参数。通过上述方式，SFT 能够有效地提升大型语言模型在特定任务上的表现，使其更加适应实际应用的需求。

2）LoRA 微调

LoRA（Low-Rank Adaptation）[11] 是一种针对大语言模型的微调方法，旨在引入低秩矩阵来高效地调整模型参数，以适应特定任务或领域的需求。LoRA 的主要目的是在不显著增加模型参数量的情况下，实现对预训练模型的有效微调，从而提高模型在特定任务上的性能，同时减少对计算资源的需求。

LoRA 微调的流程和普通的监督微调一致。区别在于在更新模型参数时，预训练模型的整体参数冻结不变，而只更新一个低秩的参数变化矩阵，从而实现更加快速和高效的微调。在 LoRA 微调中，模型的权重矩阵 $\boldsymbol{W}$ 被修改为 $\boldsymbol{W}+\boldsymbol{\Delta W}$，其中 $\boldsymbol{\Delta W}$ 是通过两个低秩矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 相乘得到的，即 $\boldsymbol{\Delta W}=\boldsymbol{A}*\boldsymbol{B}$，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 数被更新而 $\boldsymbol{W}$ 保持不变。通过上述方法和公式，LoRA 微调能够有效地提升大型语言模型在特定任务上的表现，同时减少对计算资源的需求。

3.2.3 大模型 API

大语言模型（OpenAI 的 GPT-4 和百度的文心一言等）因具有强大的自然语言处理能力，在各类任务中表现出色。它们能够理解和生成人类语言，被广泛应用于机器翻译、文本生成、对话系统等领域。由于需要庞大的部署与计算成本，大语言模型难以直接被广大开发者应用。因此大模型公司通常会开放相关模型的 API，这些 API 提供了便捷的接口，使开发者无需训练模型即可利用其强大功能。目前主流的大模型 API 包括 ChatGPT、文心一言、通义千问和星火大模型等。

3.2.4 提示工程

Prompt Engineering 是一种通过精心设计输入提示（Prompts）来引导大型语言模型生成特定类型输出的技术。这种方法不需要更新模型的内部参数，而是通过调整输入文本的结构和内容来影响模型的输出。其主要目的是在不进行模型微调的情况下，实现对模型输出的精确控制，从而提高模型在特定任务上的适用性和效率。

Prompt Engineering 通常涉及以下几个策略：

1）问题定义：设计输入提示时，提供清晰、具体的指令，明确告诉模型需要执行的任务或期望的输出格式。

2）小样本示例：在提示中包含一个或多个示例，展示期望的输出样式，帮助模型理解任务的具体要求。

3）格式约束：通过提供更多的上下文信息，帮助模型更好地理解任务背景，从而生成更相关的输出。

4）角色扮演：在提示中说明希望大模型输出始终采用某种观点或角色，这种限制可以帮助模型专注于提供更专业和准确的回答。

在提示工程方法中，思维链提示是最有效和最常用的方法。思维链（Chain of Thought，CoT）[14] 是一种在 Prompt Engineering 中使用的技术，旨在通过引导模型逐步推理来提高其在复杂任务上的表现。这种方法通过在输入提示中嵌入一系列逻辑步骤或推理链，促使模型模仿人类的思考过程，从而生成更加准确和合理的输出。

通过这些方法，Prompt Engineering 能够在不改变模型参数的情况下，有效地控制和优化模型的输出，使其更加符合特定任务的需求。这种方法的灵活性和易用性使其成为快速部署和应用大型语言模型的重要工具。图 4 给出了为测试用例自动生成任务设计的大模型提示词。

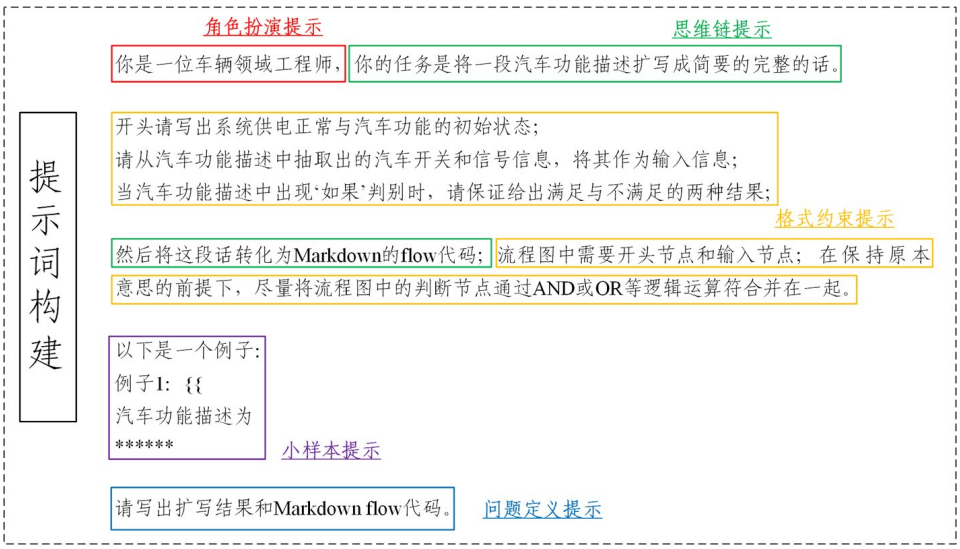

图 4 测试用例自动生成任务的提示词

Fig. 4 Prompts of test case automatic generation task

提示工程中所用对话格式模板如表 2 所列。

表 2 提示工程中所用对话模板

Table 2 Dialogue templates used in prompt engineering

对话模版
你是一位车辆领域工程师，你的任务是将一段汽车功能描述扩写成简要的完整的话。\
开头请写出系统供电正常与汽车功能的初始状态；请从汽车功能描述中抽取出的汽车开关和信号信息，\
将其作为输入信息；当汽车功能描述中出现‘如果’判别时，请保证给出满足与不满足的两种结果；\
然后将这段话转化为 Markdown 的 flow 代码；让我们从系统供电正常与汽车功能的初始状态开始逐步对这段话进行转化；流程图中需要开头节点和输入节点；\
流程图中尽量减少分支。\
流程图 start 变量名为 start，inputoutput 变量名为 ipt，condition 变量名为 cdt，operation 变量名为 opt，end 变量名为 end，格式规范如下面例子所示：\
"flow
start=>start: 文本 1
ipt=>inputoutput: 文本 2
文本 3
cdt=>condition: (条件 1)&&(条件 2)
opt1=>operation: 操作 1
opt2=>operation: 操作 2
end=>end: 结束
start->ipt->cdt(yes)->opt1->end
start->ipt->cdt(no)->opt2->end
"\n
汽车功能描述为""{}"" 请写出 Markdown flow 代码。\

4 方法评估

4.1 实验设置

4.1.1 数据集

依照 2.1.1 节所描述的思路构建数据集，即文本描述->markdown 代码，共包含 2892 条人工标注测试用例生成样本。其中 300 条由 5 名汽车领域专家分别标记，其余部分由标注团队进行标记。在数据标注过程中，每个样本首先由多位标注者独立标注；若标注结果不一致，则由第三位标注者进行仲裁标注。本文使用 2600 条样本作为训练数据，用于路线 1 中的语言模型微调，并在 2600 条数据中随机采样作为路线 2 中提示工程的示例样本。微调数据由原始文本与 Markdown 数据构成，其中，原始文本为输入数据，Markdown 数据为输出数据。本文使用其余的 292 条数据作为测试数据集，在其上进行测试用例生成任务的评估，用于验证不同方法的表现。

4.1.2 评价指标

本文采用了文本生成任务中常用的 4 种指标来评测模型生成效果。

1) ROUGE-1[15]：一元组评价指标，计算生成文本与参考文本之间的一元组重合数。

2) ROUGE-L：最长公共子序列评价指标，计算生成文本与参考文本之间最长公共子序列(LCS)的长度。

3) ROUGE-2：二元组评价指标，计算生成文本与参考文本之间的二元组重合数。

4) BLEU[16]：双语评估替补指标，用于计算生成文本与参考文本之间的相似度。

4.1.3 基线方法

表 3 列出了构建基线方法时实验所使用的软硬件。

为了展现不同语言模型技术路线对测试用例生成任务的应用效果，本文采用了以下基线方法：

1) 基于监督微调的 BART[17] 模型(BART-SFT)：采用开源的 BART-large-Chinese[1] 作为基础模型，该模型参数量达到 407M，在 2600 条训练数据上进行全监督微调。

2) 基于监督微调的 Qwen-7B[18] 模型(Qwen-7B -SFT)：采用开源 Qwen-7B 作为 decoder-only 结构的代表性模型，该模型参数量达到 7.25B[2]，在 2600 条训练数据集上进行全量监督微调。

3) 基于 LORA 微调的 Qwen-7B 模型(Qwen-7B -LoRA)：采用 Qwen-7B 作为基础模型，但在 2600 条训练数据集上进行 LoRA 微调。

4) 基于提示工程的大模型 API(LLMs-API)：采用不同的大模型 API，使用 2.2.4 节构建的提示词来引导模型生成测试用例。采用的 API 基线包括：ChatGPT-3.5(无提示词)、ChatGPT-3.5、文心一言、星火大模型、通义千问。本文实验于 2024 年 2 月前使用上述 API 进行生成任务评估。

表 3 实验软硬件环境

Table 3 Experimental software and hardware environment

环境	参数
操作系统	Ubuntu 18.04
CPU	Intel(R) Xeon(R) Silver 4210 CPU
内存/GB	16
GPU	NVIDIA A100
显存/GB	64
编程语言	Python 4.8
深度学习框架	Pytorch 2.1.0

4.2 不同语言模型技术路线的表现

表 4 列出了不同语言模型技术在测试用例生成任务上的表现。通过对比各个模型的 BLEU，ROUGE-1，ROUGE-2 和 ROUGE-L 指标，可以得出以下几点结论：基于大模型 API 的方法取得了中等表现，虽然在所有指标上不如 Qwen-7B 模型，但由于只需要少量标注数据作为样本，因此该方法在数据缺乏的情况下是一个非常好的选择。BART-SFT 方法在所有指标上表现最差，这说明基础模型的参数量对领域任务的表现有显著影响。Qwen-7B-SFT 的表现显著优于 BART-SFT，说明大模型的基础能力对提升领域任务的表现非常重要。

表 4 不同的基线方法在测试用例生成任务上的表现

Table 4 Performance of different baselines on test case generation tasks

方法	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
BART-SFT	0.362	0.671	0.442	0.660
Qwen-7B-SFT	0.684	0.855	0.724	0.839
Qwen-7B-LoRA	**0.712**	**0.865**	**0.742**	**0.845**
ChatGPT-3.5	0.562	0.801	0.641	0.795

如图 5 所示，参照文献[19]，我们将迭代轮次设置为 1，

1) https://huggingface.co/fnlp/bart-large-chinese

2) https://huggingface.co/Qwen/CodeQwen1.5-7B-Chat

在数据量规模相同且微调的迭代轮次均为 1 轮的前提下，LoRA 微调方法与全量微调方法都已基本拟合。图 5 与表 4 的实验结果表明 Qwen-7B-LoRA 的表现最佳。这与文献[20]中认为全量微调引发了灾难性遗忘，从而使大模型的性能有所下降的结论一致，而 LoRA 微调在保持模型原有能力的同时，进行了高效的微调。

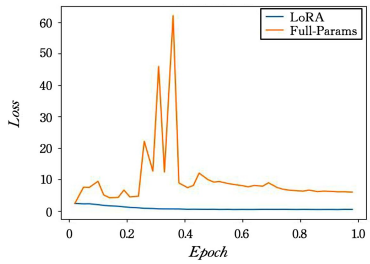

图 5 训练时损失的下降情况

Fig. 5 Loss decline during training

综上所述，基于传统语言模型微调的技术路线适合领域数据量充足的情况，提升基础模型的参数规模可以提升任务性能；而 LoRA 微调能在降低训练开销的情况下，减少大语言模型的灾难性遗忘问题。基于大语言模型 API 的技术路线适合领域数据稀少的情况。后文将讨论提示词对领域任务表现的影响。

4.3 国内外大语言模型 API 的表现

表 5 列出了不同的大语言模型 API 在测试用例生成任务上的表现。

表 5 不同的大模型 API 在测试用例生成任务上的表现

Table 5 Performance of different large model APIs on test case generation tasks

大模型	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
ChatGPT-3.5 无提示词-	0.285	0.351	0.258	0.432
ChatGPT-3.5	**0.562**	**0.801**	**0.641**	**0.795**
文心一言	0.487	0.772	0.613	0.730
星火大模型	0.508	0.765	0.612	0.741
通义千问	0.493	0.719	0.551	0.683

通过对比各个模型的 BLEU，ROUGE-1，ROUGE-2 和 ROUGE-L 指标可以看出：ChatGPT-3.5 在没有提示词的情况下表现最差，这表明在没有引导信息的情况下，模型难以准确生成符合要求的测试用例。ChatGPT-3.5 在使用提示词的情况下，所有指标均有大幅提升，说明提示词对模型性能的提升具有重要作用。文心一言、星火大模型和通义千问在所有指标上均表现良好，其表现虽然略逊于使用提示词的 ChatGPT-3.5，但仍显示出较强的生成能力。

总体来看，使用提示词的 ChatGPT-3.5 在所有指标上表现最佳，显示了其在提示工程的支持下能够生成高质量的测试用例。而无提示词的 ChatGPT-3.5 表现最差，进一步验证了提示词对提升模型生成效果的重要性。文心一言、通义千问和星火大模型表现稍逊，但在生成任务中仍有一定的应用潜力。综上所述，不同的大语言模型 API 在提示词的引导下，能够显著提升测试用例生成的效果，且选择合适的大模型 API 对于特定任务的实现具有重要意义。

4.4 提示词对大模型 API 的影响

为了进一步探讨提示词在大模型 API 中的作用，本文进行了详细的实验，评估了不同类型的提示词对模型性能的影响。这些提示词包括角色扮演提示、思维链提示以及小样本提示。通过对比有无提示词的实验结果，可以更清楚地了解提示词在提高模型生成质量中的作用。

4.4.1 角色扮演提示的影响

表 6 列出了角色扮演提示对大模型 API 的影响。通过表 2 的对话模版，本文要求模型扮演的角色为车辆领域工程师。通过对比可以得出结论，加入角色扮演提示后，模型在所有评价指标上均有所提升，尤其是 ROUGE-1 和 ROUGE-L 指标，分别提升了 0.047 和 0.047。这表明角色扮演提示能够帮助模型更好地理解任务背景，从而生成更高质量的测试用例。

表 6 角色扮演提示的效果

Table 6 Effects of role-playing prompts

提示类型	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
加入角色扮演	**0.562**	**0.801**	**0.641**	**0.795**
无角色扮演	0.518	0.754	0.612	0.748

4.4.2 思维链提示的影响

表 7 列出了思维链提示对大模型 API 的影响。本文通过表 2 中提示工程所采用的对话模板，结合角色设定和任务分解，引导模型扩展汽车功能描述，并根据提取的开关和信号信息生成简化流程图。模型通过处理条件分支来涵盖不同结果，并输出特定格式的 Markdown flow 代码，从而实现思维链的构建。思维链提示对模型性能的提升非常显著，BLEU 得分提升了 0.164，ROUGE-1 得分提升了 0.102，ROUGE-2 得分提升了 0.125，ROUGE-L 得分提升了 0.112。这表明：通过引导模型逐步推理和生成，思维链提示能够显著提高生成结果的质量和连贯性。

表 7 思维链提示的效果

Table 7 Effect of chain-of-thought prompts

提示类型	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
加入思维链	**0.562**	**0.801**	**0.641**	**0.795**
无思维链	0.398	0.699	0.516	0.683

4.4.3 小样本提示的影响

表 8 列出了小样本提示对大模型 API 的影响。为确保实验的客观性与一致性，小样本提示的数据均从原始数据集中按照问题类型随机筛选获得。可以看到，随着提示样本数量的增加，模型性能逐步提升。从 0-shot 到 2-shot，BLEU 得分提升了 0.081，ROUGE-1 得分提升了 0.176，ROUGE-2 得分提升了 0.115，ROUGE-L 得分提升了 0.183。这表明小样本提示能够有效地提升模型的生成质量，即使是少量示例也能显著改善模型在特定任务上的表现。

表 8 小样本提示的效果

Table 8 Effect of few-shot prompts

提示类型	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
0-shot	0.481	0.625	0.526	0.612
1-shot	0.528	0.799	0.629	0.773
2-shot	**0.562**	**0.801**	**0.641**	**0.795**

结束语 本文研究了在汽车电控系统中自动生成测试流程图的方法，并提出了一种基于大语言模型(LLM)的高效解决方案，旨在提高开发效率并减少人力成本。实验结果表明，在数据充足或小样本的情况下，本文提出的方法都能显著提升生成效果。通过验证不同的大模型 API 的性能，证明了本文构建的提示词在提升模型输出质量方面的重要作用。本研究不仅展示了 LLMs 在汽车测试领域的潜力，也为其他复杂工程任务的自动化提供了新的思路和方法。未来的研究可以进一步优化提示词设计框架，并探索更多应用场景下的 LLMs 自动生成技术，以推动大语言模型在实际工程中的广泛应用。

参 考 文 献

[1] LI Y J, LI X P, ZHANG W G. Survey on vision-based 3D object detection methods[J]. Computer Engineering and Applications, 2020, 56(1): 11-24.

[2] PEEBLES P Z. Probability, random variable, and random signal principles[M]. 4th ed. New York: McGraw Hill, 2001: 100-110.

[3] WEINSTEIN L, SWERTZ M N. Pathogenic properties of invading microorganism[M] // Pathologic Physiology: Mechanisms of Disease. Philadephia: Saunders, 1974: 745-772.

[4] MONREALE A, PINELLI F, TRASARTI R, et al. WhereNext: a location predictor on trajectory pattern mining[C] // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, Jun 28 — Jul 1, 2009. New York: ACM, 2009: 637-646.

[5] LUO W, WANG H F. Evaluating Large Language Models: A Survey of Research Progress. [J] Journal of Chinese Information Processing. 2024, 38(1): 1-23.

[6] MORZY M. Prediction of moving object location based on frequent trajectories[C] // Proceedings of the 21st International Symposium on Computer and Information Sciences, Istanbul, Nov. 1—3, 2006. Berlin, Heidelberg: Springer, 2006: 583-592.

[7] WANG L. Research of fuzzy clustering algorithm for incomplete data based on the improved ACO with interval supervision[D]. Shenyang: Liaoning University, 2016.

[8] Online Computer Library Center, Inc. History of OCLC [EB/OL]. (2000-01-08) [2019-12-23]. http://www.oclc.org/about/history/default.htm.

[9] ZHANG F X, YU X R, HE W F, et al. Face recognition with improved loss function and multiple-norm[J]. Computer Engineering and Applications, 2020, 54(6): 114-120.

[10] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[J]. arXiv: 1409.3215, 2014.

[11] RADFORD A, WU J, CHILD R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[12] HU E J, SHEN Y, WALLIS P, et al. Lora: Low-rank adaptation of large language models[J]. arXiv: 2106.09685, 2021.

[13] LIU H Q. In the Era of Software-defined Vehicles, AUTOSAR is Helping to Transform China's Automotive Industry[J]. Transport Energy Conservation & Environmental Protection, 2024, 20(3): 74-77.

[14] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in Neural Information Processing Systems, 2022, 35: 24824-24837.

[15] LIN C Y. Rouge: A package for automatic evaluation of summaries[C] // Text Summarization Branches Out. 2004: 74-81.

[16] PAPINENI K, ROUKOS S, WARD T, et al. Bleu: a method for automatic evaluation of machine translation[C] // Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002: 311-318.

[17] LEWIS M. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[J]. arXiv: 1910.13461, 2019.

[18] BAI J, BAI S, CHU Y, et al. Qwen technical report[J]. arXiv: 2309.16609, 2023.

[19] MENG X, DAI D, LUO W, et al. Periodiclora: Breaking the low-rank bottleneck in lora optimization[J]. arXiv: 2402.16141, 2024.

[20] BIDERMAN D, ORTIZ J G, PORTES J, et al. Lora learns less and forgets less[J]. arXiv: 2405.09673, 2024.

LI Zhanqi, born in 1985, postgraduate, senior engineer. His main research interests include simulation development and system validation of automotive electronic control systems.

XIONG Deyi, born in 1979, Ph.D, professor, Ph.D supervisor, is a member of CCF(No. 57174S). His main research interests include natural language processing, large language model and AI alignment.

(责任编辑：何杨)