基于半自动提示工程人工智能任务代码生成方法

张 乐[1] 荆晓远[2] 任 娟[1]
ZHANG Le JING Xiaoyuan REN Juan

摘 要 大语言模型的快速发展极大地推动了人工智能技术的进步，并显著降低了生成人工智能代码的门槛。然而，这也要求用户提供精确的数据，以确保生成高质量的人工智能代码，从而使大模型能够有效地完成复杂的人工智能代码生成任务。提示工程作为与大型模型交互的核心技术，对于保障代码生成的质量和效率起到了决定性作用。文章提出了一个 AI-CODE 框架来替代人工分析的工作，帮助大模型生成特定任务的代码。该框架通过构建专业词汇数据集和半自动提示生成机制，实现了从用户模糊描述到完整项目代码的转换。实验结果表明，AI-CODE 在关键词分类、提示生成和项目代码生成方面具有较高的准确性。与现有框架相比，AI-CODE 在多个方面具有显著优势，为用户提供了一个全新的项目级代码生成解决方案。

关键词 大模型；人工智能；提示工程；代码生成；AI-CODE

doi：10.3969/j.issn.1672-9528.2024.12.046

0 引言

近年来，随着大语言模型的迅速发展，人工智能技术取得了重大进展，尤其是 ChatGPT 类大模型 [1] 的出现，其优秀的推理、理解和交互能力展现出了巨大进步。不再局限于简单的对话交互，而是能够基于用户输入的复杂指令，完成更为精细和多样化的任务，从而极大地推动了人工智能技术的广泛应用和深入发展。实际中完成一个人工智能任务需要经历一系列繁琐的步骤，包括数据准备、模型选择和参数调优，这些过程既需要深厚的专业知识，又消耗大量的时间和精力。因此，在使用这些模型完成具体人工智能任务时，提供准确且完整的提示对于生成高质量项目级别人工智能代码至关重要。

提示工程 [2] 作为一种新兴的技术，通过构造特定的提示词模板来引导模型进行预测，从而在不改变模型结构的情况下提高其在特定任务上的性能。提示已经不仅仅是简单的输入或查询，而是演变成为用户与大型人工智能模型交互的“编程语言”。其通过构建精确且富有指导性的指令，能够有效地引导模型执行复杂的任务。然而，如果用户提供的数据缺乏规范性和明确性，模型在处理任务时可能会产生误解或混淆，进而导致生成的内容无法准确满足用户的实际需求。因此，为了确保大模型能够精确地解读和执行 AI 任务，并且提供明确、详细以及无歧义的提示是生成高质量人工智能项目代码的关键。这不仅有助于模型更好地理解用户需求，还能显著提高代码生成结果的质量和效率。

本文提出了一个名为 AI-CODE 的框架。该框架根据用户输入的 AI 任务描述，将复杂任务分解为若干个子任务，再生成整个人工智能项目的代码。整个 AI-CODE 框架的设计旨在通过信息交互确保代码生成的准确性和系统整体的协调性。

专业词汇数据集：构建了一个专业词汇数据集，包括深度学习领域的各类关键词汇，并对专业名词经常出现在人工智能任务的那个阶段做了人工标注。这个数据集主要用在关键词提取阶段，帮助框架有效地从用户输入中提取关键信息。

AI-CODE 框架：提出了半自动 Prompt 生成机制，该机制能够根据用户输入的关键信息，结合本文的知识补充数据库，自动生成针对具体任务的 Prompt。使 AI-CODE 框架能够根据用户的人工智能任务描述，结合用户输入和知识库，自动生成任务特定代码和元信息，提高代码生成准确性和项目级代码生成效率。为用户提供了一个全新的项目级代码生成。

实验结果表明，AI-CODE 在关键词分类、提示生成和项目代码生成方面具有较高的准确性和效率。与现有框架相比，AI-CODE 在任务分析、输入优化、代码评审和项目文件生成方面具有明显优势。

1. 广东石油化工学院计算机学院 广东茂名 525000
2. 广东石油化工学院计算机学院省市共建石化装备智能安全广东省重点实验室 广东茂名 525000

[基金项目] 国家自然科学基金项目“基于类不平衡深度特征学习的石化动设备故障信号分类研究”（62176069）

1 相关工作

代码生成技术主从大方向上可以分基于代码特征的代码生成和结合后处理的代码生成。后者依托于前者进行改进，优化模型代码生成效果。

1.1 基于代码特征的代码生成

2016 年定义了高级程序设计语言的智能化代码生成流程，采用序列到序列模型，将自然语言转换为代码片段。尽管对于代码生成有一定的效果，但这种模型未能充分处理代码的结构信息，导致生成代码质量较差。为解决这一问题，研究者提出了一种创新策略，即不直接生成代码，而是预测与代码对应的抽象语法树。赵乐乐等人 [3] 设计了一个能够将自然语言描述转换为目标代码语言抽象语法树的模型，并在之后的工作中不断进行改进，使之能够更好地处理复杂的代码结构和嵌套关系，实现了从序列到序列模型到序列到数模型的转变。研究人员发现，代码生产任务存在严重的长依赖问题，Sun 等人 [4] 又进一步引入了 Transformer 架构。并对模型进行修改提出 TreeGen，使其能够结合代码的结构信息。上述有监督的代码生成模型虽然在任务执行上取得了一定的成效，但它们严重依赖于大量高质量的标注代码数据集，这不仅限制了它们的泛化能力，也影响了它们在不同编程环境和语言中的应用范围。

预训练模型很好地解决了这个问题，从大规模无标注的数据中通过自监督 [5] 的训练策略获取知识。模型整体的架构来说两者并没有太大的区别，且绝大多数预训练模型都是基于 Transformer 架构进行。早期 CuBERT 和 CodeBERT[6] 都是继承 BERT 的架构，利用代码语料进行训练的预训练模型，后续在各个下游任务上被广泛使用。但是在训练过程中仅仅包含文本的语义信息，在预训练模型中也需要融入代码中的结构信息。研究人员做了其它尝试，一方面从在训练中增加结构信息来优化生产效果，将代码的结构信息数据流纳入预训练的过程之中，分别提出了 GraphCodeBERT[7] 和 CodeT5 模型；另一方面，提出 InCoder[8] 结构化的生成代码，打破了先前从左至右的代码生成预训练模型范式，能够填充任意代码区域的大型生成代码模型，这种以双向上下文为条件的能力大大提高了代码生成任务的性能。试图通过编写规范程序的过程描述为用户和系统之间的多轮对话，来使预训练模型可以解决更为复杂的程序问题。用户分多次为系统提供自然语言，进行多轮对话后完成代码生成，分步提供自然语言规范的方式可以将较长且复杂的意图分解为多个简单的意图。

1.2 结合后处理的代码生成

为了进一步解决代码生成中的复杂任务，研究人员开始关注如何利用预训练模型的语言理解能力来优化生成效果。ChatGPT 类大模型的出现为这一领域带来了新的启示，通过引导大语言模型生成一系列中间推理过程来得到最终答案，其强大的语言理解能力使得通过高级目标分解和结合人类工作流程来自动化任务成为可能。Fried 等人 [9] 提出的思维链提示策略，通过引导大语言模型生成一系列中间推理过程来得到最终答案。利用这个特点在代码生成任务中引入思维链技术 [10]，在大模型的输入提示中加入了结构化的思维链约束大模型使用程序结构组织代码生成，引导大模型从程序语言的角度思考如何解决需求，进一步提升了大模型在代码生成上的效果。研究提示的设计对于大模型的性能影响较大，因此如何设计有效的提示内容来使大模型处理更为复杂的任务成为一个焦点。结合了大语言模型和多智能体协作系统 [11]，旨在通过模拟人类工作流程来解决复杂问题。该框架的核心在于将标准化操作程序编码成提示序列，以便在多智能体系统中实现更高效的工作流程和减少错误。提示经过多个智能体角色对输入内容进行多轮补充优化，但是这种补充是直接通过大模型进行自动生成的，其中关键信息、关键内容的准确性是不可控的，并且在一些专业领域，只通过大模型进行知识补充很难满足具体项目代码生成。因此，本文针对人工智能任务提出了 AI-CODE 框架，它能够根据用户的自然语言描述自动生成人工智能项目级代码。通过提示工程显著提升了代码准确性。该系统具有较好的适应性，能够根据用户的具体需求生成相应代码。

2 方法

AI-CODE 框架的工作流程主要分为三个阶段：关键信息提取、半自动任务提示生成、项目代码生成。框架的整体流程如图 1。

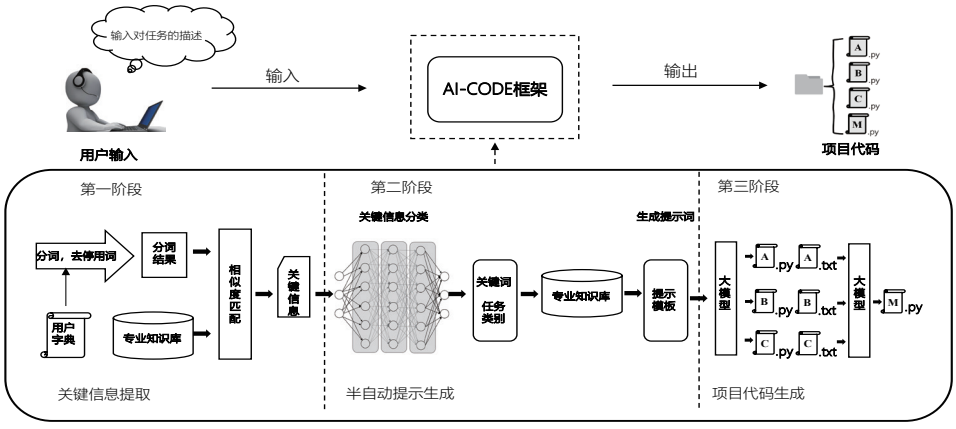

图 1 AI-CODE 框架信息处理流程

其中，关键信息提取阶段使用专业词汇数据集从用户输入中提取关键信息；半自动任务提示生成阶段根据提取的关键信息和任务模板生成提示；项目代码生成阶段根据生成的提示生成针对每个子任务的代码片段，并最终组合成完整的AI 项目代码。

2.1 关键信息提取

在 AI-CODE 的第一阶段，框架接受用户输入，将描述中的关键内容进行提取。为了准确从用户输入中提取出关键内容，本文收集了人工智能相关领域的专业词汇，制作专业词库。根据对各类 AI 任务之间共性的归纳，本文可以发现其中具有固定的环节，因此将 AI 任务分解为一系列子任务，每个子任务都针对 AI 处理过程中的一个关键环节。即使具体任务不同，但是都需要完成数据处理、模型架构、训练优化过和评估指标等子任务，只是在面对不同任务时，数据的特点、模型的设计、算法的选择、优化技巧的应用以及评估标准的具体细节会有所不同，如图 2 所示。

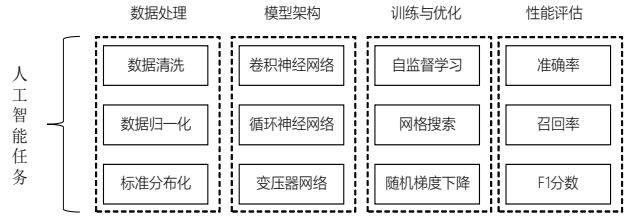

图 2 人工智能任务子任务划分

针对人工智能任务的各个子任务进行了关键词的收集和整理。具体来说，本文收集了 161 个与任务类型相关的关键词，406 个与数据集相关的关键词，378 个与数据处理相关的关键词，298 个与模型架构相关的关键词，308 个与训练优化相关的关键词，以及 189 个与性能评估相关的关键词。这些关键词涵盖了人工智能任务的不同方面，有助于 AI-CODE 框架准确地从用户输入中提取关键信息，并生成针对每个子任务的提示模板。

2.2 半自动提示生成

第一阶段完成后，从输入中提取到了任务的关键信息，进入第二阶段的半自动提示生成。提示词工程作为一个设计、优化和细化输入提示的过程，其目标在于确保用户意图能够有效传递给如 ChatGPT 这样的大语言模型。它通过设计、改进和实施提示或指令的实践，引导模型输出，以助力完成各种人工智能任务为一个复杂的任务。为了帮助大模型更好的理解并完成用户的需求，将一个复杂的人工智能任务分解为多个子任务，根据分析各个子任务之间的共性，编写每个子任务通用部分的提示。如图 3 中数据处理提示模板的内容，展示为数据处理子任务的提示模板。其中文字部分为数据处理子任务提示模板的固定部分，对于同一个子任务固定部分是相同的。“{}”部分的内容来自于用户输入内容，根据用户输入，将提取到的关键信息划分到归属的子任务之中，并对各个子任务的模板进行填充。

子任务：数据处理提示模板

上下文：	使用**{数据集变量}**，完成**{任务类型变量}**人工智能任务，生成该任务中的数据处理部分代码。
提示内容：	在数据处理的部分使用以下技术**{处理技术变量}**进行处理。
生成格式：	生成数据处理部分代码的python文件，将每个功能分装成函数。并生成独立的txt文件，包括函数名 ，参数信息，和功能。

图 3 数据处理子任务提示模板

通过短文本分类 [12]，将输入中的关键信息标注出具体的子任务类别，是为输入的关键词分配一个单独的标签。将每个 token x_i （词汇）使用词嵌入技术，转换成一个特征向量 $\boldsymbol{X}_i$ 使用 Transformer 模型 [13] 处理 token 的上下文信息。其核心如公式（2）（3）所示。$\boldsymbol{Q}$ 表示查询向量，$\boldsymbol{K}$ 表示链向量，$\boldsymbol{V}$ 表示值向量，d 表示 Transformer 的维度，其中 $\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$ 通过公式（2）来计算。$\boldsymbol{W}_h^{\boldsymbol{Q}}$、$\boldsymbol{W}_h^{\boldsymbol{K}}$ 和 $\boldsymbol{W}_h^{\boldsymbol{V}}$ 是每个注意力头的权重矩阵，$\boldsymbol{X}$ 为输入向量。

$$\boldsymbol{X}i = \text{Embedding}(xi) \tag{1}$$

$$\boldsymbol{Q}_h = \boldsymbol{W}_h^{\boldsymbol{Q}} \cdot \boldsymbol{X},\ \boldsymbol{K}_h = \boldsymbol{W}_h^{\boldsymbol{K}} \cdot \boldsymbol{X},\ \boldsymbol{V}_h = \boldsymbol{W}_h^{\boldsymbol{V}} \cdot \boldsymbol{X} \tag{2}$$

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}) \tag{3}$$

将模型的输出通过全连接层映射到标签空间，并使用 softmax 函数计算每个标签的概率分布。通过交叉熵损失函数来训练模型，衡量预测标签概率和真实标签之间的差异，进行优化模型。

$$L(\boldsymbol{W}_o,b_o;x,y) = -\sum_{i=1}^{n}\sum_{i=1}^{k}1(y_i = L_j)\log(p(y_i = L_j \mid x)) \tag{4}$$

式中：W_o 和 b_o 是模型的权重和偏置；x 是输入数据点；y 是真实标签或期望输出。指导模型调整其参数，使得预测值与真实值之间的差异逐渐减小，从而提高模型的性能。

将输入中的关键信息分好类后，对图 3 中的提示模板镜像填充。但是由于用户输入的不规范性，模板中的关键信息无法完全补全。框架会根据不同任务类型，进行推荐填充。比如，在图像分类任务中缺少数据集的信息，框架会为任务推荐常用的数据集信息。

2.3 项目代码生成

为了确保生成符合用户需求项目级代码，AI-CODE 框架中做了复杂的交互过程如图 4 所示。框架对用户输入进行分词等处理，并提取关键信息。根据分析出来的结果生成各个子任务的规范提示，帮助大模型生成更加符合用户需求的代码和代码文件对应的解释文件。通过 AL-CODE 对各个子任务进行组合，生成项目文件。

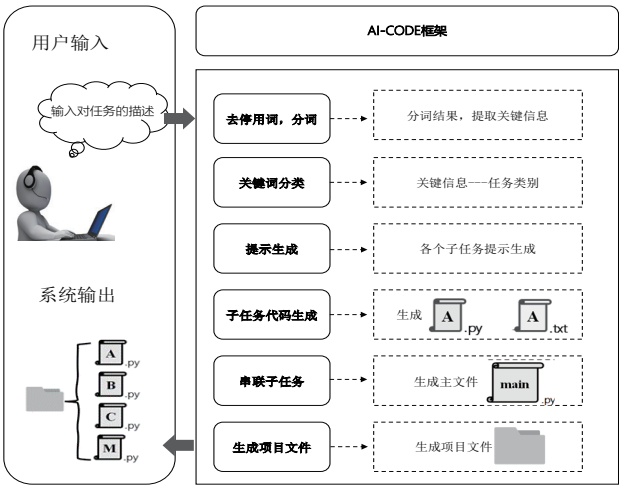

图 4 用户与 AL-Code 框架交互生成项目文件生成

其中关键过程分为三个部分：

（1）提示生成：系统根据用户输入生成初步提示，包括通用部分和特定任务部分。如果用户输入不完整，系统将使用知识补充数据库来填充缺失的信息，生成最终的提示。

（2）子任务代码生成：系统根据提示生成针对每个子任务的代码片段，并产出相应的元信息，包括函数名称、参数列表等。将所有子任务的元信息整合成一个说明文档，详细列出每个函数的功能描述和参数说明。并调用各个子任务的函数，并串联各个功能模块。完成整个 AI 项目，输出项目所在位置。

（3）系统交互：通过这些信息交互步骤，系统确保了生成的代码片段能够针对特定子任务有效协作，并且在主函数中准确调用，从而提高了代码生成的准确性和系统整体的协调性。

3 实验

3.1 关键词分类

本文收集了 1720 个人工智能任务常用的关键词，这些关键词被分为六类标签：任务类型、数据集、数据处理、模型架构、训练优化和性能评估。为了训练一个能够有效分类这些关键词的短文本分类网络，将 1740 个关键词分为训练集和测试集，其中训练集和测试集的比例为 7:3，具体划分情况如表 1 所示。

表 1 关键词数据集训练集和测试集划分

标签类型	总数据量	训练集	测试集
任务类型	161	112	49
数据集	406	284	122
数据处理	378	264	114
模型架构	298	208	90
训练优化	308	215	93
性能评估	189	132	57

对短文本分类的网络选取 Precision、Recall 和 F_1-score 指标来评估模型的性能。训练结果显示，所构建的短文本分类网络在深度学习专业词汇的分类任务上表现优异，平均 F_1-score 达到 0.922 5，显示出模型的高准确性和可靠性。特别是模型架构类别的表现最为突出，F_1-score 高达 0.966 6，而数据集、数据处理、训练优化和性能评估等类别也均展现出较高的分类能力。任务类型类别的性能相对较低，但整体模型具有良好的泛化能力，能够有效服务于深度学习领域的关键词分类需求，具体实验结果数据如表 2 所示。

表 2 短文本分类训练结果

标签类别	Precision	Recall	F_1-score
任务类型	0.826 9	0.877 5	0.851 4
数据集	0.940 1	0.901 6	0.920 5
数据处理	0.914 5	0.938 5	0.926 4
模型架构	0.966 6	0.966 6	0.966 6
训练优化	0.945 6	0.935 4	0.940 5
性能评估	0.929 8	0.929 8	0.929 8
平均	0.920 6	0.924 9	0.922 5

3.2 提示生成

半自动提示工程根据用户输入将关键词填充的各个子任务的提示模板中，并对用户输入中没有给出的内容进行填充。通过 AI-CODE 优化用户输入内容，帮助大模型生成更加符合用的人工智能任务的代码。填充过程如图 5 所示，通过之前的关键信息提取和 AI-CODE 对“{}”中内容进行填充，优化输入提示。

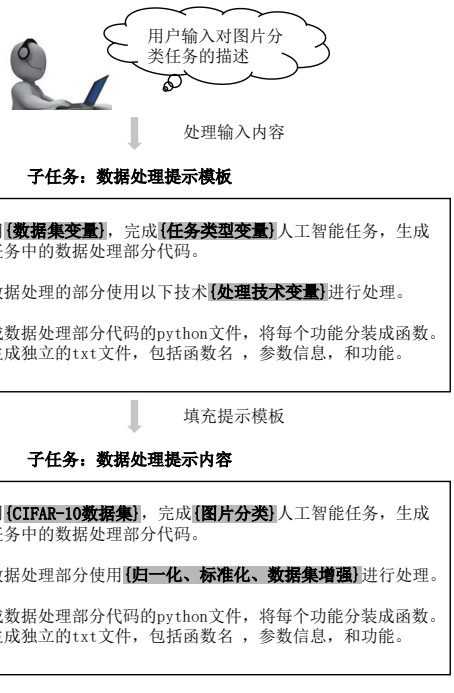

图 5 提示填充效果

为了测试 AI-CODE 半自动提示生成的效果，本文选取了 10 种常见的任务智能任务，对用户输入的任务描述进行模糊处理。一个人工智能任务提示模板中变量个数为 23 个，AI-CODE 会根据用户输入进行补充优化。本文对 10 个任务描述分别在缺失 4 个、8 个、12 个和 16 个关键信息的情况下进行测试，计算半自动提示机制对输入补充的准确率，将上述过程重复 5 次求平均，测试结果如表 3。从表 3 中可以看出，AI-CODE 的半自动提示机制在处理常见人工智能任务时显示出较高的准确性，尤其在信息缺失较少的情况下，能够完全准确地补充任务描述。系统仍能在大约三分之二内容缺失的情况下提供相对完整的任务提示。这表明 AI-CODE 具有一定的智能和适应能力。整体而言，AI-CODE 在实际应用中对用户输入不完整的情况具有较好的容错性，有助于提升用户体验。

表 3 常见人工智能任务提示填充准确率

人工智能任务	正确填充准确率			
	缺失 4	缺失 8	缺失 12	缺失 16
机器翻译	1	1	1	1
文本分类	1	1	1	1
情感分析	1	1	1	0.916 6
语音识别	1	1	1	1
语言合成	1	1	0.913 0	0.791 6
图像识别	1	1	1	1
物体检测	1	1	1	0.869 5
图像分割	1	1	0.833 3	0.791 6
图像生成	1	1	1	1
视频分析	1	1	1	1

3.3 项目代码生成

在本文中对 AI-CODE 框架与几个常见的框架进行了能力对比，这些框架包括 GLM-3.5、GLM-4.0、GPT-4.0、AutoGPT 和 AgentVerse。我们主要从任务分析、输入优化、代码生成、代码评审和项目级别文件生成五个方面进行了对比。表 4 为常见框架的能力对比，其中“×”代表无法完成，“√”表示具有其功能。

表 4 常见框架能力对比

框架能力	GLM-3.5	GLM-4.0	GPT-4.0	AutoGPT	AgentVerse	AI-CODE
任务分析	×	×	×	×	×	√
输入优化	×	×	×	×	×	√
代码生成	√	√	√	√	√	√
代码评审	×	×	×	×	×	√
项目文件生成	×	×	×	×	√	√

结果显示，AI-CODE 框架在任务分析、输入优化、代码生成和项目文件生成方面具有显著优势，而其他框架则无法完成或提供这种优化。在代码评审方面，AI-CODE 框架同样表现出色，能够对生成的代码进行评审，确保代码的质量和准确性。AI-CODE 框架为用户提供了一个全新的项目级代码生成解决方案，具有明显的优势。

本文选用了 chatglm4.0 API 作为 AI-CODE 框架中的大模型部分，该 API 基于深度学习技术，能够理解和生成自然语言。AI-CODE 框架通过半自动提示工程，利用 chatglm4.0 API 的强大能力，实现了从用户输入到完整项目级代码的自动生成。这一过程主要包括数据处理、模型构建、训练优化和模型评估等子任务，每个子任务都有其特定的代码生成需求。

具体来说，数据处理子任务涉及数据的读取、预处理和加载，模型构建子任务则需要设计网络结构，训练优化子任务包括选择合适的优化器和损失函数，以及调整超参数，而模型评估子任务则需要对模型的性能进行评估和分析。这些子任务在 AI-CODE 框架中通过半自动提示工程和 chatglm4.0 API 的结合，得以高效地生成相应的代码。最终，在主函数中，这些子任务的代码片段被组织调用，形成了一个完整的项目文件。用户只需运行项目文件中的主文件，即可实现整个项目的运行。

如图 6 所示，AI-CODE 框架在代码生成方面的效果显著，它能够从用户的模糊描述出发，通过半自动提示工程，生成一个结构清晰、功能完整的代码项目。这极大地简化了人工智能项目的开发流程，提高了开发效率。

4 结语

本文提出的 AI-CODE 框架，通过构建专业词汇数据集和半自动 prompt 生成机制，实现了从用户输入到完整项目级代码的自动生成。该框架在关键词分类、提示生成和项目代码生成方面表现出较高的准确性和效率，与现有框架相比在多个方面具有显著优势。AI-CODE 通过自动化流程显著提升了代码准确性，为人工智能项目代码生成提供了有效的支持，为用户提供了一个全新的项目级代码生成解决方案。尽管 AI-CODE 在生成项目级人工智能代码上取得了积极成果，但仍存在一些挑战。一方面，依赖于专业的数据集。另一方面，可以进一步设计更加灵活和通用的提示模板，以适应不同的任务和用户输入。针对这些挑战，未来的研究方向可以考虑结合预训练模型与领域专业知识进行微调提升人工智能代码生成质量与应用效果。

图 6 AI-CODE 代码生成效果图

参考文献：

[1] BROWN T B, MANN B, RYDER N, et al. Language models are few-shot learners[DB/OL]. (2020-07-22)[2024-07-16]. https://doi.org/10.48550/arXiv.2005.14165.

[2] LIU P F, YUAN W Z, FU J L, et al. Pre-train, Prompt, and Predict: a systematic survey of prompting methods in natural language processing[J]. ACM computing surveys, 2023, 55(9): 1–35.

[3] 赵乐乐 , 张丽萍 , 赵凤荣 . 基于注意力机制的 Tree2Seq 代码注释自动生成 [J]. 计算机工程与科学 , 2023, 45(4): 638-645.

[4] SUN Z Y, ZHU Q H, XIONG Y F, et al. TreeGen: a tree-based transformer architecture for code generation[DB/OL]. (2019-11-28)[2024-05-10]. https://doi.org/10.48550/arXiv.1911.09983.

[5] 刘壮 , 宋祥瑞 , 赵斯桓 , 等 . 进化网络模型 : 无先验知识的自适应自监督持续学习 [J]. 电子与信息学报 , 2024, 46(8): 3256-3266.

[6] 杨泽洲 , 陈思榕 , 高翠芸 , 等 . 基于深度学习的代码生成方法研究进展 [J]. 软件学报 , 2024,35(2): 604-628.

[7] WANG R C, XU S L, TIAN Y, et al. SCL-CVD: Supervised contrastive learning for code vulnerability detection via GraphCodeBERT[J]. Computers & security, 2024, 145(10): 103994.

[8] WEI J, WANG X Z, XIA F, et al. Chain-of-thought prompting elicits reasoning in large language models[DB/OL].(2023-01-10)[2024-05-17].https://doi.org/10.48550/arXiv.2201.11903.

[9] FRIED D, AGHAJANYAN A, LIN Y, et al. Incoder: a generative model for code infilling and synthesis[DB/OL]. (2023-04-09)[2024-07-01]. https://doi.org/10.48550/arXiv.2204.05999.

[10] 林博 , 王尚文 , 毛晓光 . 基于思维链的软件漏洞自动修复技术研究 [J/OL]. 软件学报 , 2024:1-21[2024-07-16].https://www.jos.org.cn/jos/article/abstract/7205.

[11] 马悦 , 吴琳 , 许霄 . 基于多智能体强化学习的协同目标分配 [J]. 系统工程与电子技术 , 2023, 45(9): 2793-2801.

[12] 淦亚婷 , 安建业 , 徐雪 . 基于深度学习的短文本分类方法研究综述 [J]. 计算机工程与应用 , 2023, 59(4): 43-53.

[13] 王明申 , 牛斌 , 马利 . 一种基于词级权重的 Transformer 模型改进方法 [J]. 小型微型计算机系统 , 2019,40(4): 744-748.

【作者简介】

张乐（1999—），通信作者（email: zhangle_2023@126.com），女，陕西西安人，硕士研究生，研究方向：自然语言处理、大预言模型研究、故障诊断。

荆晓远（1971—），男，江苏南京人，博士（工程学），教授、博士生导师，研究方向：模式识别、计算机视觉、故障诊断。

任娟（1999—），女，四川南充人，硕士研究生，研究方向：自然语言处理、大语言模型研究、故障诊断。

（收稿日期：2024-09-12）