

人工智能算法下的软件开发自动化流程探究

侯 玥, 刘 静

湖北孝感美珈职业学院 湖北孝感

【摘要】传统的软件开发方式存在价格高、效率低、质量难以保证等多方面问题,亟待引进人工智能算法做出改变,推动其实现自动化转型。本文基于此,从需求分析、设计、编码以及测试和维护等软件开发全生命周期的角度出发,阐述使用人工智能算法提高软件开发自动化流程的方法,旨在为软件开发行业智能化转型提供借鉴。

【关键词】人工智能算法; 软件开发; 自动化流程; 软件生命周期

【收稿日期】2025 年 3 月 15 日 【出刊日期】2025 年 4 月 16 日 【DOI】10.12208/j.aics.20250002

Research on automated software development processes based on artificial intelligence algorithms

Yue Hou, Jing Liu

Hubei Xiaogan Meijia Vocational College, Xiaogan, Hubei

【Abstract】Traditional software development methods face challenges such as high costs, low efficiency, and inconsistent quality, necessitating the integration of artificial intelligence (AI) algorithms to drive transformative changes and achieve automation. This paper explores methods to enhance the automation of software development processes through AI algorithms, covering the entire lifecycle—requirement analysis, design, coding, testing, and maintenance. The study aims to provide insights for the intelligent transformation of the software development industry.

【Keywords】Artificial Intelligence Algorithms; Software Development; Automated Processes; Software Lifecycle

在数字化时代发展的背景下,各类型的软件是驱动系统和设备协同、高效运行的关键。但是,传统的软件开发流程面临着耗时、耗力、耗资金的问题,且由于需求分析、设计、编码、测试和维护等阶段严重依赖专业人员支持,从而拉长开发周期。针对于此,借鉴并使用人工智能技术的强大数据分析、模式识别和学习能力,可为解决软件开发存在的问题提供帮助,助力软件开发实现自动化。

1 需求分析阶段

1.1 自然语言处理技术基石

在人工智能算法解析自然语言描述的软件需求进程中,自然语言处理(NLP)技术扮演着关键角色^[1]。NLP 涵盖一系列复杂技术与方法,旨在让计算机理解、分析以及生成人类语言。其基础技术包括词法分析、句法分析和语义分析。

1.1.1 词法分析

词法分析将自然语言文本分割成词汇单元,并标注词性等信息。隐马尔可夫模型(HMM)是词法分析常用算法模型。

HMM 是双重随机过程,状态转移是马尔可夫过程,观测值产生依赖当前状态。假设隐藏状态集合为 $Q = \{q_1, q_2, \dots, q_N\}$,观测值集合为 $V = \{v_1, v_2, \dots, v_M\}$,对于长度为 T 的观测序列 $O = \{o_1, o_2, \dots, o_T\}$ 和状态序列 $I = \{i_1, i_2, \dots, i_T\}$,HMM 概率计算基于以下公式:

状态转移概率: $a_{ij} = P(i_{t+1} = q_j | i_t = q_i)$,表示在时刻 t 处于状态 q_i ,在时刻 $t + 1$ 转移到状态 q_j 的概率。

观测概率: $b_j(k) = P(o_t = v_k | i_t = q_j)$,表示在时刻 t 处于状态 q_j 时,产生观测值 v_k 的概率。

初始状态概率: $\pi_i = P(i_1 = q_i)$,表示初始时刻处于状态 q_i 的概率。

维特比算法求解最可能的状态序列,完成词性标注。HMM 在词法分析中的应用示例表格如下表 1。

1.1.2 句法分析

句法分析聚焦剖析句子语法结构,构建句法树,明确词汇间语法关系。基于图的依存句法分析算法是常用方法^[2]。在基于图的依存句法分析中,将句子看作图,节点为词语,边为依存关系。假设句子中有 n 个词语,依存关系的得分函数定义为: $s(i, j, r) = w_r^T \cdot \phi(i, j)$,其

中 w_r 是依存关系 r 的权重向量, $\phi(i, j)$ 是词语 i 和 j 之间的特征向量。通过寻找最大化得分函数的边连接方式构建句法树。以需求语句“系统需具备高效的数据检索功

能, 满足用户快速查询信息的需求”为例, 通过该算法可确定“系统”是“具备”的主语, “数据检索功能”是“具备”的宾语等依存关系。相关特性如下表 2。

表 1 HMM 在词法分析中的应用

应用环节	描述	优势	局限性
词性标注	利用 HMM 的状态转移和观测概率, 为词汇标注词性	计算效率较高, 能处理大规模文本	对长距离依赖关系捕捉能力弱

表 2 句法树构建特征表

应用环节	描述	优势	局限性
句法树构建	基于图的算法寻找最优依存关系构建句法树	能有效处理复杂句式	对数据标注质量要求高

1.1.3 语义分析

语义分析致力于挖掘文本深层含义, 消除语义歧义。语义角色标注 (SRL) 算法在语义分析中发挥重要作用^[3]。语义角色标注通常基于谓词-论元结构, 假设谓词为 P , 其论元集合为 $A = \{a_1, a_2, \dots, a_n\}$, 通过分类模型判断每个论元的语义角色, 如施事、受事等。例如在句子“用户使用软件进行文件编辑”中, “用户”是“使用”的施事, “软件”是工具, “文件编辑”是目的。语义

角色标注模型训练基于最大熵模型, 公式为:

$$P(y|x) = \frac{1}{Z(x)} \exp \left(\sum_{i=1}^n \lambda_i f_i(x, y) \right)$$

其中 y 是语义角色标签, x 是输入文本特征, $Z(x)$ 是归一化因子, λ_i 是特征权重, $f_i(x, y)$ 是特征函数。特点如表 3。

表 3 语义理解特点表

应用环节	描述	优势	局限性
语义理解	标注谓词语义角色, 辅助语义理解	提升语义理解准确性	对语义知识库依赖较大

1.2 深度学习模型的深度应用

1.2.1 循环神经网络 (RNN)

RNN 能够对具有序列特性的需求文本进行有效建模。基本单元计算公式为:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

其中 h_t 是时刻 t 的隐藏状态, x_t 是时刻 t 的输入, W_{hh} 是隐藏层到隐藏层的权重矩阵, W_{xh} 是输入层到隐藏层的权重矩阵, b_h 是偏置项, σ 是激活函数。分析大量用户反馈和需求文档时, RNN 按文本顺序依次处理每个词汇, 捕捉词汇间前后依赖关系, 进而识别高频需求。

如在众多电商软件用户反馈中, 频繁出现“简化购物流程”“提升支付便捷性”等表述, RNN 能精准捕捉这些高频词汇组合, 确定为高频需求^[4]。

1.2.2 长短期记忆网络 (LSTM)

LSTM 解决 RNN 处理长序列时梯度消失或梯度爆炸问题, 更擅长处理长文本需求中的长期依赖关系。

LSTM 单元包含输入门 i_t 、遗忘门 f_t 、输出门 o_t 和记忆单元 c_t , 计算公式如下:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$c_t = f_t \theta c_{t-1} + i_t \theta \tilde{c}_t$$

$$h_t = o_t \theta \tanh(c_t)$$

其中 θ 表示逐元素相乘。通过对过往大量需求文档学习, LSTM 可发现潜在需求。如分析某办公软件需求文档时, 虽未直接提及“多语言支持”, 但通过对相关功能描述和用户使用场景深度挖掘, LSTM 能推断在全球化背景下, 该软件存在多语言支持的潜在需求^[5]。RNN 和 LSTM 在处理软件需求文本时的表现对比如表 4 所示。

表 4 RNN 和 LSTM 在处理软件需求文本时的表现对比表

模型	长序列处理能力	梯度问题	高频需求识别能力	潜在需求挖掘能力
RNN	较弱	易出现梯度消失或爆炸	较好	较弱
LSTM	强	有效解决梯度问题	好	强

2 设计阶段

在软件设计阶段, 人工智能发挥着至关重要的辅助作用, 尤其是在软件架构设计和模块划分方面。遗传算法作为一种经典的优化算法, 在搜索最优软件架构方案中应用广泛。遗传算法基于达尔文的进化论, 模拟自然选择和遗传机制。其基本流程包括初始化种群, 种群中的每个个体代表一种软件架构方案; 计算每个个体的适应度, 适应度函数根据软件的功能需求和性能约束来设计, 例如可以定义为:

$$\text{Fitness} = w_1 \times \text{Functionality} + w_2 \times \text{Performance} + w_3 \times \text{Maintainability}$$

其中, **Functionality**表示软件功能需求的满足程度, **Performance**表示性能指标, 如响应时间、吞吐量等, **Maintainability**表示软件的可维护性, w_1 、 w_2 、 w_3 为权重系数, 根据项目的侧重点进行调整。

通过选择、交叉和变异等遗传操作, 不断迭代更新种群, 逐步逼近最优解。例如, 选择操作可以采用轮盘赌选择法, 个体被选中的概率与其适应度成正比。交叉操作模拟生物遗传中的基因重组, 将两个父代个体的部分基因进行交换, 生成新的子代个体。变异操作则以一定概率随机改变个体的某些基因, 以保持种群的多样性。

同时, 人工智能可以利用深度学习中的神经网络模型, 如多层感知机 (MLP), 学习已有优秀软件设计

模式。通过大量的标注数据训练, 让模型学习不同设计模式与软件功能、性能之间的关联。当面对新的软件项目时, 输入项目的功能需求、性能约束等信息, MLP 模型可以输出合理的设计建议, 包括软件架构类型、模块划分策略等。遗传算法和 MLP 在软件设计阶段的应用对比如下表 5。

3 编码阶段

代码自动生成技术基于机器学习模型, 极大地提高了编码效率和准确性。以基于 **Transformer** 架构的语言模型为例, 如 GPT 系列, 它在代码生成领域表现出色。**Transformer** 架构采用多头注意力机制, 能够并行地关注输入序列的不同位置, 从而更好地捕捉文本中的语义信息^[6]。

对于代码生成任务, 模型的训练基于大量的代码语料库。在训练过程中, 模型学习代码的语法结构、语义关系以及不同功能代码的模式。当给定软件设计和需求描述作为输入时, 模型通过自注意力机制对输入进行编码, 然后根据学习到的代码模式生成相应的代码。例如, 对于一个简单的需求描述“实现一个计算两个整数之和的函数”, 模型生成如下 Python 代码框架:

```
def add_numbers(a, b):  
    result = a + b  
    return result
```

基于 **Transformer** 架构的语言模型在代码生成方面的特点如下表 6。

表 5 算法模型对比表

算法模型	应用方式	优势	局限性
遗传算法	根据适应度函数搜索最优软件架构方案	全局搜索能力强, 能处理复杂约束	计算量大, 易陷入局部最优
MLP	学习优秀设计模式提供设计建议	学习能力强, 能快速给出建议	对训练数据依赖大, 可解释性差

表 6 代码特点表

特点	描述	优势	局限性
语义理解能力	通过自注意力机制理解需求语义	能根据复杂需求生成代码	生成代码可能存在逻辑错误, 需人工校验
代码风格适应性	学习多种代码风格	可生成符合特定风格的代码	对特定领域代码生成效果受训练数据限制

4 测试阶段

人工智能在软件测试中具有多方面的重要作用。在测试用例优先级排序方面,可采用基于机器学习的分类算法,如支持向量机(SVM)。SVM通过寻找一个最优的超平面,将不同类别的数据分开。在测试用例优先级排序中,将测试用例的各种特征,如执行时间、覆盖代码行数、历史缺陷发现率等作为输入特征,将优先级分为高、中、低等类别。SVM的决策函数可以表示为:

$$f(x) = \text{sign} \left(\sum_{i=1}^n \alpha_i y_i K(x_i, x) + b \right)$$

其中, α_i 是拉格朗日乘子, y_i 是样本的类别标签, $K(x_i, x)$ 是核函数, b 是偏置项。

表7 应用效果表

应用方面	算法模型	优势	局限性
测试用例优先级排序	SVM	能有效区分测试用例优先级	对特征工程要求高, 模型训练时间长
自动生成测试用例	变异测试结合机器学习	提高测试用例针对性和覆盖率	变异操作计算量大, 可能生成冗余测试用例

例如, GNN 中的图卷积网络(GCN)可以通过对AST节点的特征进行卷积操作,提取代码的语义特征,为维护人员提供代码注释和修改建议。在预测软件维护工作量和潜在问题方面,可使用时间序列分析算法,如ARIMA模型。ARIMA模型通过对历史维护数据的分析,预测未来的维护工作量。假设维护工作量时间序列为 y_t , ARIMA(p,d,q)模型的表达式为:

$$\left(1 - \sum_{i=1}^p \varphi_i B^i \right) (1 - B)^d y_t = \mu + \left(1 + \sum_{j=1}^q \theta_j B^j \right) \epsilon_t$$

其中, B 是向后移位算子, φ_i 是自回归系数, θ_j 是移动平均系数, μ 是常数项, ϵ_t 是白噪声序列。

人工智能在软件维护阶段的应用特性如下表8。

表8 人工智能在软件维护阶段的应用特性表

应用方面	算法模型	优势	局限性
代码理解	AST 结合 GNN	准确分析代码结构和功能	对复杂代码理解能力有待提高, 计算资源消耗大
维护工作量和问题预测	ARIMA 模型	基于历史数据进行有效预测	对数据平稳性要求高, 难以处理突发变化

6 结语

在软件开发的全生命周期中, 人工智能算法展现强大赋能作用。但是, 当前人工智能在软件开发自动化流程中面临数据质量、模型可解释性等挑战。但随着技术持续发展与创新, 有望实现更高效、智能的软件开发, 推动软件行业迈向新高度。

参考文献

- [1] 胡远涛.人工智能在计算机应用软件开发中的应用探索[J].信息记录材料,2025,26(02):81-83.
- [2] 张琦.人工智能时代计算机应用技术发展研究[J].数字通

信世界,2024,(12):37-39.

- [3] 田雷.基于人工智能算法的软件开发自动化流程研究[J].家电维修,2024,(01):49-51.
- [4] 周伟娟,钟伟,张礼学.提升人工智能装备软件测试效能研究[J].桂林航天工业学院学报,2022,27(04):463-467.
- [5] 时庆涛,薛泽亮.基于人工智能的模块化嵌入式软件开发研究[J].数字通信世界,2019,(12):111.

版权声明: ©2025 作者与开放获取期刊研究中心(OAJRC)所有。本文章按照知识共享署名许可条款发表。

<http://creativecommons.org/licenses/by/4.0/>



OPEN ACCESS