

# 【CodeHalu：基于执行验证的大语言模型代码幻觉研究】

—— 【CodeHalu: Investigating Code Hallucinations in LLMs via Execution-based Verification】

## 1 相关资源

pdf: <https://arxiv.org/pdf/2405.00253.pdf>

ppt:

短视频:

数据集:

源码: <https://github.com/yuchen814/CodeHalu>

网站:

【除了网站，其他资源尽量下载】

## 2 论文属性

论文来源: AAAI 2025

【给出具体会议名称和年限，不要仅仅写 ACM, IEEE】

论文类别: Large Language Model

【论文的类别，比如移动计算、轨迹处理、深度学习等】

论文关键字: LLMs, Hallucination Mitigation

推荐程度: 3 (其他说明可标注)

(5 非常棒，建议认真研读、小组讨论和复现；4 好，建议细读，考虑复现；3 可以，部分内容值得注意；2 一般，简单浏览即可；1 没有意义，不建议阅读)

## 3 工作团队

作者: Yuchen Tian, Weixiang Yan, Qian Yang, Xuandong Zhao, Qian Chen, Wen Wang, Ziyang Luo, Lei Ma, Dawn Song

单位:

1. Hong Kong Baptist University
2. University of California, Santa Barbara(Guangzhou)
3. Mila - Qu'ebec AI Institute
4. Universit e de Montr al
5. University of California, Berkeley
6. Alibaba Group
7. The University of Tokyo
8. University of Albert

团队情况描述:

## 4 论文介绍

### (1) 研究目的

**【研究背景是什么？本文工作有什么用？】**

随着大模型在代码生成领域的快速发展，其生成的代码虽然在语法上可能正确，但在执行时可能无法达到预期功能或出现错误，这种现象被称为“代码幻觉”。

本文想要填补代码生成领域中关于幻觉现象的系统性研究空白。代码的最终目的是通过正确执行来解决问题，因此仅依靠语法正确性是不够的，必须通过执行验证来确保其功能正确性。此外，代码幻觉可能导致运行时错误或功能缺陷，阻碍 LLMs 在自动化软件开发中的可靠应用。

因此，本文旨在通过定义代码幻觉的概念，提出一种基于执行验证的分类方法，并开发动态检测算法和基准测试，以系统性地识别、分类和量化 LLMs 生成代码中的幻觉现象，为改进 LLMs 的代码生成能力提供理论支持和实践工具。

### (2) 研究现状

**【当前的最好研究做到什么程度了？存在的问题是什么？这里采信论文的说法，可以给出自己的评点】**

在自然语言处理和多模态学习领域，幻觉现象已经得到了广泛研究。例如，Zhang 等人（2023）将 NLP 中的幻觉分为输入冲突、上下文冲突和事实冲突三类，而 Zhai 等人（2023）在图像到文本的多模态场景中定义了对象存在、属性和关系幻觉。

然而，在代码生成领域，幻觉现象尚未得到系统性探索。现有的代码基准测试（如 HumanEval、APPS 等）主要关注 LLMs 在编程任务中的性能，缺乏有效方法来检测和量化代码生成中的幻觉现象。本文指出，代码幻觉与代码错误不同，代码错误是幻觉的一个子集，而幻觉涵盖更广泛的逻辑和功能问题。因此，现有研究在代码幻觉的系统性定义、分类和量化方面存在明显不足。

### (3) 本文解决的问题

**【一句话概括本文解决的核心问题】**

本文系统性地定义、识别、分类和量化了 LLMs 在代码生成中的幻觉现象，填补了代码生成领域中关于幻觉研究的空白。

### (4) 创新与优势

**【本文的创新之处是什么？新场景？新发现？新视角？新方法？请明确指出】**

**【本文工作的贡献或优点是什么？】**

1. 提出 LLM 中代码幻觉的概念，并基于执行验证方法对其进行定义，填补了代码生成领域幻觉研究的空白
2. 开发动态检测算法 CodeHalu 来识别和量化 LLM 代码生成中的幻觉类型。基于两阶段启发式方法将代码幻觉分为四大类，探讨其理论意义和潜在成因。
3. 提出 CodeHaluE-val 基准系统评估 17 个主流 LLM，揭示跨模型代码幻觉分布规律，为开发更健壮可靠的 LLM 提供见解。

### (5) 解决思路

【本文是怎样解决问题的？包括方法、技术、模型等，以自己理解的方式表述清楚】

## 代码幻觉定义

代码幻觉指 LLM 生成的代码在语法上正确甚至语义上合理，但最终无法按预期执行或未能满足指定要求的现象。该现象通常源于多种因素，如训练数据中的错误或过时信息、对编程语言语法规则和范式掌握不足，以及模型逻辑处理能力的局限性。

现有研究通常将代码错误和代码幻觉等同或将错误视为幻觉的特定子集，作者认同特定子集的说法。作者认为，错误表现为幻觉的一种形式，但并非所有幻觉都能通过错误充分表达。尽管代码幻觉与代码错误存在轻微重叠，但二者的含义、研究对象和范围差异显著。代码幻觉关注模型为何产生幻觉，而代码错误聚焦代码违反何种语法规则。代码错误构成代码幻觉的真子集，代码幻觉则涵盖更广泛的潜在逻辑与功能问题，是对代码整体质量与功能更细粒度、更全面的评估。

## CodeHalu 算法

作者提出 CodeHalu 算法，用于检测并量化大语言模型在代码生成中的幻觉现象。该算法基于假设 ASS：若特定模式在多个大语言模型生成的代码中频繁出现，则视为常见代码幻觉。这些模式包括错误类型、语法中断、逻辑坍塌或意外执行结果。CodeHalu 具有语言无关性，能根据编程语言动态适配不同编程场景。

算法步骤：

1. 输入：代码生成数据集  $\alpha$  和语言模型集合  $\pi$ 。输出：幻觉类型列表  $\xi$ 。
2. 初始化：将幻觉类型列表初始化为空列表
3. 遍历数据集和模型：对每个数据集中的每个样本，对每个大模型都根据代码生成指令 GI 和问题描述 Q 生成代码解决方案 GC
4. 检查生成代码的状态：如果生成的代码 GC 出现如重复生成、无限循环或乱码的状态之一，则将该状态添加到幻觉类型列表中，否则执行下一步
5. 执行测试用例：对于每个样本中的每个测试用例，使用生成的代码执行这些测试用例获得实际执行结果 ER，若实际执行结果 ER 与预期结果 op 不一致的话则将该状态添加到列表中
6. 统计和聚合：合并列表中相同的幻觉状态，并计算其出现频率

---

### Algorithm 1: CodeHalu Algorithm

---

**Input:** Code Generation Dataset  $\alpha$ , Language models  $\pi$

**Output:** HaluTypes  $\xi$

```
1: Let  $\xi \leftarrow$  empty list
2: for  $\alpha_i$ , where  $i \in \{1, \dots, k\}$  do
3:   for  $\pi_j$ , where  $j \in \{1, \dots, m\}$  do
4:      $GC_j^{\alpha_i} \leftarrow \pi_j(GI_j, Q)$ 
5:     if  $GC_j^{\alpha_i}$  is stuttering, infinite enumeration, or gibberish
6:        $\xi \leftarrow \xi \cup State(GC_j^{\alpha_i})$ 
7:     else
8:       for  $t_n$ , where  $n \in \{1, \dots, N\}$  do
9:         if  $Execute(GC_j^{\alpha_i}(t_n))$ 
10:           $ER_j^{\alpha_i}(t_n) \leftarrow Execute(GC_j^{\alpha_i}(t_n))$ 
11:          if  $ER_j^{\alpha_i}(t_n) \neq op_{t_n}$ 
12:             $\xi \leftarrow \xi \cup State(GC_j^{\alpha_i})$ 
13: Aggregate and count frequencies of unique State( $GC_j^{\alpha_i}$ ) in  $\xi$ 
```

---

图 1: CodeHalu Algorithm.

## 代码幻觉分类

根据编程语言流行度指标 TIOBE 指数，作者主要研究 Python 中的代码幻觉现象。通过在复杂 APPS 数据集和 17 个主流大语言模型上应用 CodeHalu 算法，作者识别并验证了 18 种违反人类预期的代码生成幻觉状态，包括不一致的代码上下文、模糊的逻辑与数据流、冲突的意图等。采用两阶段启发式分类法，根据现象本质将其归类为四大主要类型：映射幻觉、命名幻觉、资源幻觉和逻辑幻觉。

映射幻觉指大语言模型在数据操作过程中对数据类型、数值及结构的感知映射出现模糊与混乱的现象。该现象可细分为两个子类：数据合规性幻觉和结构访问幻觉。

- 数据合规性幻觉发生时，大语言模型对操作对象的数据类型及参数值认知模糊，导致生成试图执行类型不匹配或违反规则操作的代码。如将整数与字符串相加、将列表与整数比较等
- 结构访问幻觉发生时，大语言模型误解操作对象的数据结构，导致生成试图访问不存在数组索引或字典键名的代码。

命名幻觉指大语言模型在处理变量、属性和模块的命名、作用域及存在性时表现出的记忆相关问题和事实性错误。该现象可细分为两个子类：身份幻觉和外部源幻觉。

- 身份幻觉发生时，大语言模型存在记忆偏差或对上下文理解不足，导致生成引用未定义变量、访问不存在对象属性或在局部作用域使用未赋值变量的代码。
- 外部源幻觉发生时，大语言模型在外部知识源方面表现出严重记忆问题或明显事实冲突，导致生成试图导入不存在模块或无法正确加载其他路径模块的代码。

资源幻觉指大语言模型对生成代码执行时的资源消耗和控制流缺乏充分感知与预测的现象。该现象可细分为物理约束幻觉和计算边界幻觉。

- 物理约束幻觉发生时，大语言模型低估数据处理操作中的资源消耗，导致代码因超出内存容量、堆栈深度等物理限制而失败。
- 计算边界幻觉发生时，大语言模型对数值计算限界和迭代终点的识别模糊，导致代码因数值溢出或迭代控制失当而失败。

逻辑幻觉指大语言模型生成代码执行后预期结果与实际输出的差异，或产生语义密度低下甚至完全混乱的输出。该现象可细分为逻辑偏离和逻辑崩溃。

- 逻辑偏离发生时，大语言模型生成的代码缺乏充分逻辑考量或与指令意图相悖。虽然执行时可能不报错，但逻辑偏差或混乱导致输出结果不符合预期。
- 逻辑断裂（Logic breakdown）指大语言模型在代码生成过程中难以解析或保持对上下文的持续理解。这表明模型可能在生成代码时迷失方向，难以维持上下文信息的严格一致性。

## 幻觉成因分析

映射幻觉主要源于模型对数据类型与结构的误解。该现象成因包括：

- 模型基于 token 生成代码，缺乏对语句、函数等高层结构的洞察
- 处理长距离数据依赖时（特别是复杂代码块内），模型无法持续跟踪变量结构与状态，过度依赖局部信息而忽视整体上下文重要性
- 代码生成时未显式执行类型检查与结构匹配，缺乏静态检查与纠错机制。

命名幻觉反映模型在信息跟踪与外部知识利用方面的局限。该问题成因包括：

- 基于 token 的特征表示难以准确建模长距离依赖，导致模型对变量作用域、生命周期和可见性产生误判
- 代码生成过程缺乏标识符一致性检查，未对变量定义与使用进行全局追踪

- 外部库知识未能及时有效整合到模型知识体系中，使其难以准确理解库的名称、功能及调用方法

资源幻觉突显模型对代码执行机制与物理约束缺乏深度理解。该问题成因包括：

- 训练数据缺乏资源消耗与性能优化相关信息，使模型难以学习复杂度分析与资源预估
- 概率驱动的代码生成方式缺乏计算模块，无法估算生成代码的资源消耗，难以模拟真实运行环境与资源限制
- 模型训练过程通常聚焦代码功能正确性，常忽略实际执行环境中的复杂度与资源约束

逻辑幻觉揭示了模型在代码语义理解和推理方面的缺陷。该问题由多重因素导致：

- 模型主要依赖模式匹配和统计规则生成代码，缺乏对符号系统的基本理解及程序逻辑的严格验证
- 训练数据通常未经严格准确性验证，可能包含功能高度相似的代码。由于模型有时会模仿记忆先前示例，这会导致模型直接复制代码中的相似逻辑，甚至从一开始就习得错误逻辑
- 当模型生成代码时，行级重复会产生自我强化效应，使模型对其生成的代码愈发自信，可能导致结巴现象

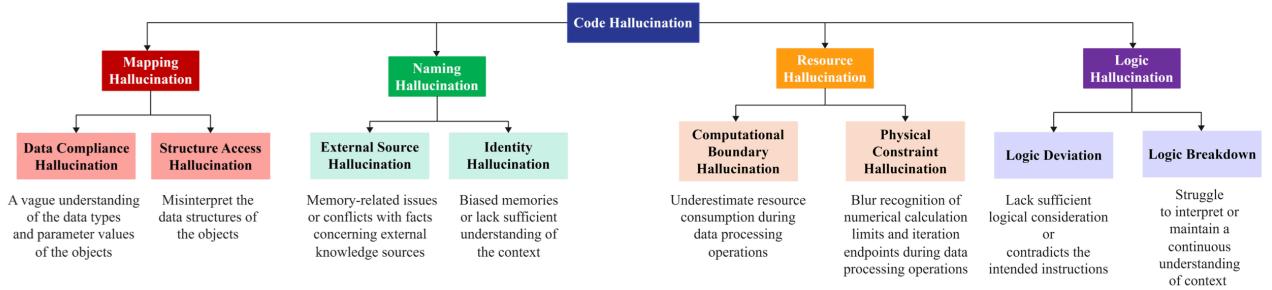


图 2: The definition and classification of code hallucinations, including 4 main categories and 8 subcategories.

### CodeHaluEval 基准数据集及实验

作者构建了 CodeHaluEval 基准数据集，是用于比较不同大语言模型在代码生成中各类幻觉出现类型及频率的统一评估方法。作者基于 APPS 测试集开发该基准，遵循验证-识别-构建结构化流程构建数据集。

鉴于目前对代码幻觉的探索有限，尚无专门用于评估大语言模型（LLM）中该现象的指标。为填补这一空白，作者提出名为“幻觉率”（HR）的评估指标。具体而言，定义为测试集中检测到的幻觉样本占所有样本的百分比。理想情况下，HR 值越低表明 LLM 在代码生成过程中出现幻觉的可能性越低，从而体现更强的鲁棒性和可靠性。

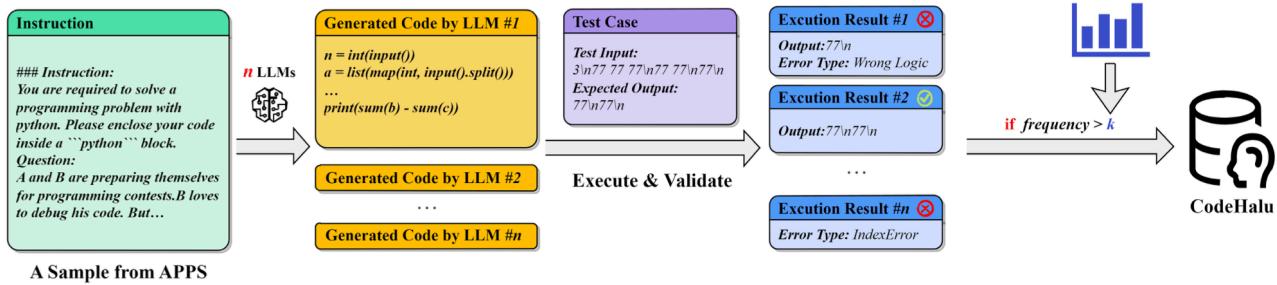


图 3: Collection of CodeHaluEval benchmark based on a verification-identification-construction process.

#### (6) 可改进的地方

**【本文工作的局限性是什么？你觉得可以从哪些方面改进工作？】**

首先，文中选择 Python 作为代码幻觉研究焦点，未将研究扩展至其他语言。

其次，CodeHalu 专注于确保生成代码的正确性，还未考虑识别和防范安全风险等。

最后，文中聚焦于代码生成任务中的代码幻觉问题，不包括代码翻译、代码修复等其他编程任务。

#### (7) 可借鉴的地方

**【你觉得本文哪些方面可以借鉴？比如思路、方法、技术等】**

本文提出的 CodeHalu 算法为检测和量化代码中的幻觉提供了有效工具，同时构建的 CodeHaluEval 数据集也能用于评估模型的代码幻觉问题，其构建过程也值得学习。

#### (8) 其他收获

**【你有什么其他收获吗？比如了解了哪些团队和大牛在某领域做得很好，某类问题通常用什么技术解决，某些技术之间存在什么样的关联，某些会议和期刊在某领域很知名……】**

通过本文，我了解到在代码生成领域，对于代码幻觉现象的研究还处于起步阶段，值得深究。

## 5 评阅人

姓名:

时间: