Unseen Horizons: Unveiling the Real Capability of LLM Code Generation Beyond the Familiar

Yuanliang Zhang, Yifan Xie, Shanshan Li, Ke Liu, Chong Wang, Zhouyang Jia, Xiangbing Huang,

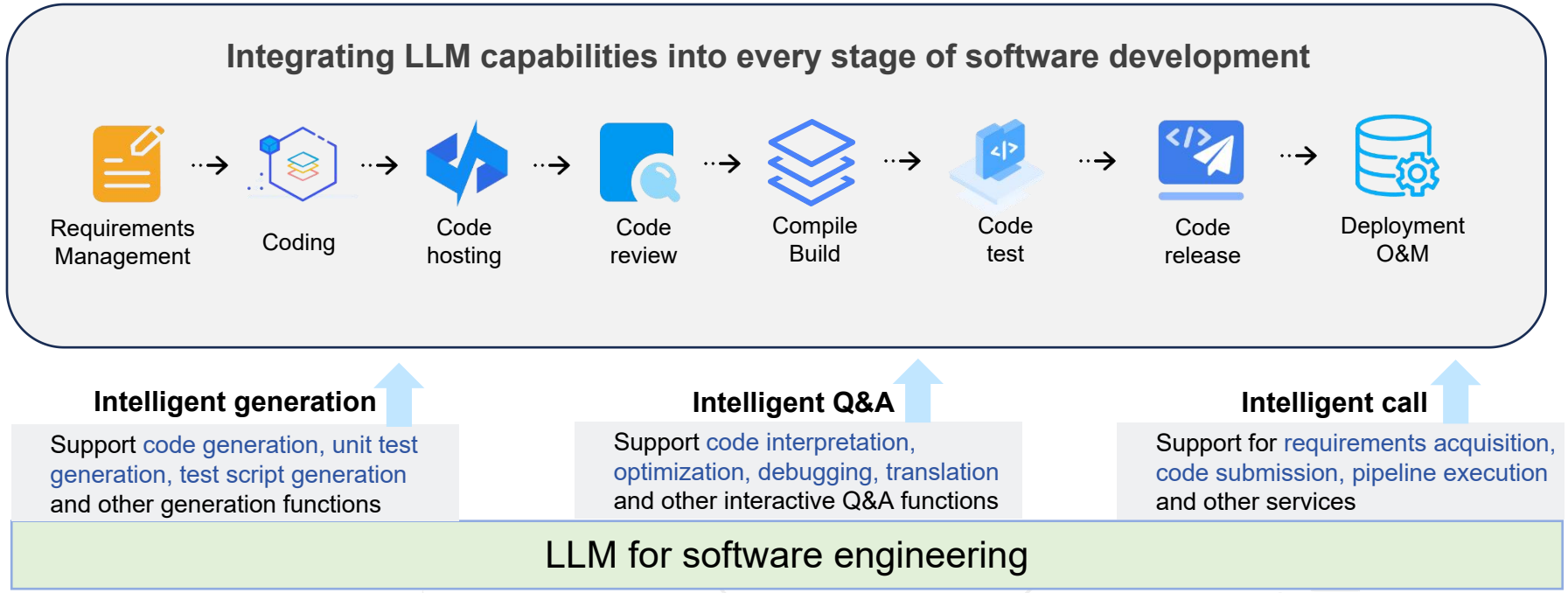
Jie Song, Chaopeng Luo, Zhizheng Zheng, Rulin Xu, Yitong Liu, Si Zheng, Xiangke Liao

College of Computer Science and Technology
National University of Defense Technology
Changsha, China

The 47th IEEE/ACM International Conference on Software Engineering (ICSE 2025)

Background

- **Large language model (LLM) is changing the software development process**



Code generation/ Code completion

Code commenting

Code Optimization/ Bug Fixing

Code testing

…

Motivation

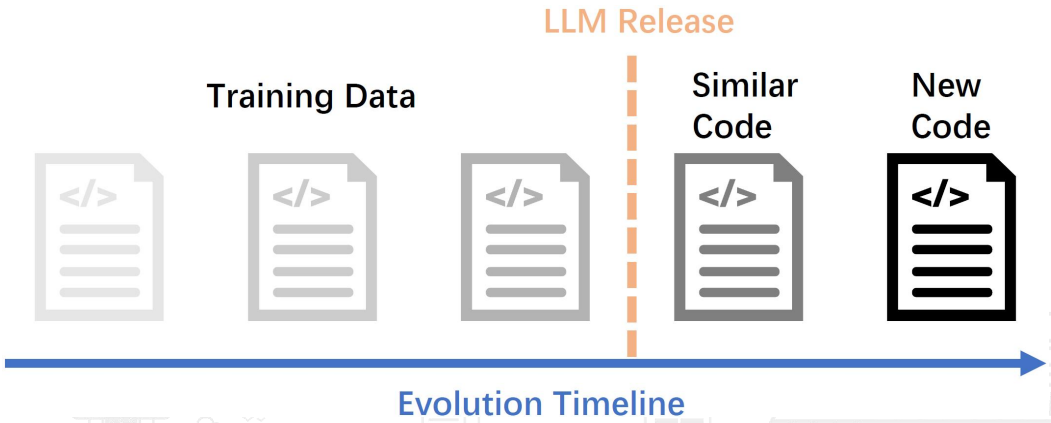
- **Gaps of existing LLM code generation benchmarks**

(1) Exposure of target co

- Target code has beer
- Collected case is time-sensitive
  - On OJ website, the solution rate for new questions is significantly lower

(2) Precise dependency availability

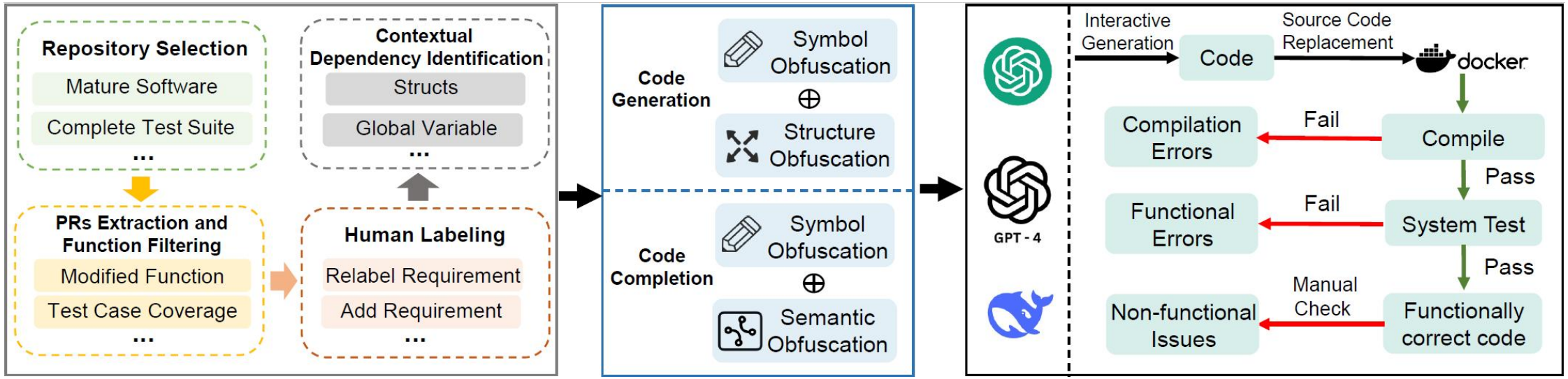
existing evaluation process may suffer from the "Specialist in Familiarity" problem



OJ Website	GPT3.5-turbo (2023.06)		GPT4.0-preview (2023.11)	
	zero-shot	few-shot	zero-shot	few-shot
LeetCode 2023.11-2024.01	19.7%	22.0%	39.4%	40.9%
LeetCode 2018.09-2018.11	95.6%	93.3%	96.7%	95.6%

Approach

- **Evaluating LLM's real code generation capability on "unfamiliar" code**
  - Symbol + Structure + Semantic Obfuscation
  - Strategic dependency obfuscation: Providing right but redundant dependencies
  - Validating syntax/functionality: compilation testing & systematic testing

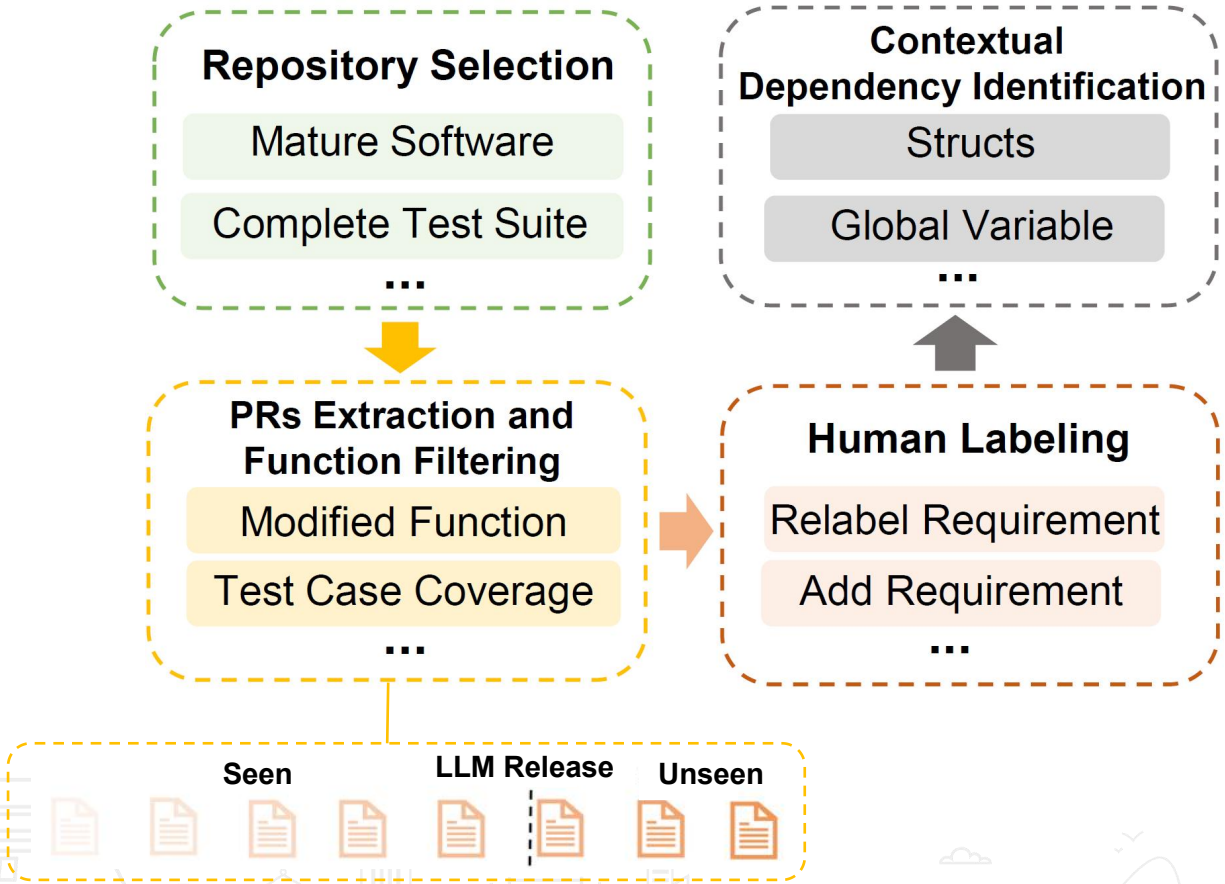


Raw data collection | **Code Obfuscation** | **Systematic Testing and Code Review**

Proposing a novel approach for future LLM evaluation benchmark: prioritize generalizability over dataset-specific optimization

Approach

- **Raw data collection**
  - (1) Repository selection
    - Mature software
    - Complete test suite
  - (2) PRs Extraction and Function Filtering
    - PRs after LLM release
    - Modified and test-covered functions
  - (3) Human labeling
  - (4) Contextual dependency provision
    - Functions
    - Global variables
    - Structs
    - Macros



Approach

- **Code obfuscation**
  - Changing code implementation without changing semantic functionality
  - Symbol + Structure + Semantic Obfuscation

Before

```
double calculate_area(double radius){
    double pi = 3.14159;
    double area = pi * radius * radius;
    return area;
}
```

After

```
double calculate_area(double r){
    double b = 3.14159;
    double c = b * r * r;
    return r;
}
```

Before

```
bool is_even_number(int num) {
    if (num % 2 == 0)
        return true;
    else
        return false;
}
```

After

```
bool is_even_number(int num) {
    return !!(num % 2 == 0);
}
```

We replace the obfuscated code into the code base and run official test to ensure the obfuscation code is syntactic and semantic right.

Approach

Symbol obfuscation

- Identifier extraction: function, class, variables names, etc
- Identifier rewriting

① Project Contexts

```
Struct:
typedef struct client {
    uint64_t id;            /* Client incremental unique ID. */
    ... ...
    char *buf;
} client;
API:
void addReplyNull(client *c)
void addReplyBulk(client *c, robj *obj) /* Add a Object as a bulk reply */
... ...
```

② Requirement

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ Signature

```
void addReplyBulkCString(client *c, const char *s)
```

④ Reference Code

```
{
    if (s == NULL) {
        addReplyNull(c);
    } else {
        addReplyBulkCBuffer(c, s, strlen(s));
    }}
```

Symbol obfuscation

① Project Contexts

```
Struct:
typedef struct ClientInfo {
    uint64_t uniqueId;      /* Incremental unique ID for the client. */
    ... ...
    char *buffer;
} ClientInfo;
API:
void appendResponseNull(ClientInfo *c)
void appendResponseBulk(ClientInfo *c, robject *obj) ... ...
```

② Requirement

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ Signature

```
void appendResponseBulkCString(ClientInfo *c, const char *s)
```

④ Reference Code

```
{
    if (s == NULL) {
        appendResponseNull(c);
    } else {
        appendResponseBulkCBuffer(c, s, strlen(s));
    }}
```

Approach

Structure obfuscation

- Analyzing function call relationships based on LLVM
- Function unfolding & parameter alignment

① Project Contexts

```
Struct:
typedef struct client {
    uint64_t id;            /* Client incremental unique ID. */
    ... ...
    char *buf;
} client;
API:
void addReplyNull(client *c)
void addReplyBulk(client *c, robj *obj) /* Add a Object as a bulk reply */
... ...
```

② Requirement

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ Signature

```
void addReplyBulkCString(client *c, const char *s)
```

④ Reference Code

```
{
    if (s == NULL) {
        addReplyNull(c);
    } else {
        addReplyBulkCBuffer(c, s, strlen(s));
    }}
```

Structure obfuscation

① Project Contexts

```
Struct:
typedef struct client {
    uint64_t id;            /* Client incremental unique ID. */
    ... ...
    char *buf;
} client;
API:
void addReplyProto(client *c, const char *s, size_t len) /*This...objects. */
void addReplyBulk(client *c, robj *obj)... ...
```

② Requirement

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ Signature

```
void addReplyBulkCString(client *c, const char *s)
```

④ Reference Code

```
{
    if (s == NULL) {
        if (c->resp == 2) {
            addReplyProto(c, "$-1\r\n", 5);
        } else {
            addReplyProto(c, "_\r\n", 3);
        }
    } else {
        addReplyLongLongWithPrefix(c, strlen(s), '$');
        ... ...
}
```

Approach

- **Semantic obfuscation**
  - Manual rewriting
  - Same semantic but different implementation

① **Project Contexts**

```
Struct:
typedef struct ClientInfo {
    uint64_t uniqueId;      /* Incremental unique ID for the client. */
    ... ...
    char *buffer;
} ClientInfo;
API:
void  appendResponseNull(ClientInfo  *c)
void  appendResponseBulk(ClientInfo  *c,  robject *obj) ... ...
```

② **Requirement**

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ **Signature**

```
void appendResponseBulkCString(ClientInfo *c, const char *s)
```

④ **Reference Code**

```
{
    if (s == NULL) {
        appendResponseNull(c);
    } else {
        appendResponseBulkCBuffer(c, s, strlen(s));
    }}
```

Semantic obfuscation

① **Project Contexts**

```
Struct:
typedef struct ClientInfo {
    uint64_t uniqueId;      /* Incremental unique ID for the client. */
    ... ...
    char *buffer;
} ClientInfo;
API:
void  appendResponseNull(ClientInfo  *c)
void  appendResponseBulk(ClientInfo  *c,  robject *obj)
... ...
```

② **Requirement**

The function sends a response to a designated client, incorporating the content from the C string parameter s. If s is NULL, it replies with a NULL type; otherwise, it converts s into a C buffer and responds with a binary block (Bulk) type, specifying a length corresponding to strlen(s).

③ **Signature**

```
void appendResponseBulkCString(ClientInfo *c, const char *s)
```

④ **Reference Code**

```
{
    const int isStringNull = (stringPtr == NULL);
    const int responseType = customerPtr->responseType;
    const char *response = (isStringNull && responseType == 2) ? "$-1\r\n" :
                           (isStringNull) ? "_\r\n" : stringPtr;
    const size_t responseLength = (isStringNull && responseType == 2) ? 5 :
                                  (isStringNull) ? 3 :
                                  strlen(stringPtr);
    ... ...
}
```

Approach

- **Benchmark construction**
  - Prompt construction
    - Instruction
    - Context
      - Functions
      - Structs
      - Macros
      - Global variables
    - Function description
  - Code generation scenarios
  - Code completion scenarios

Instruct:
From now on, you play the role of the C code generator. You can generate the corresponding function code according to the function description provided by the user. Please do not return anything other than the target code. Don't return anything other than the function code. The process is as follows:
[prompt-input]
[output]

Context:
Here are some function context details you may need to know when writing objective function for the project:
Functions may be used:
void addReplyNull(client *c)
... ...
Structs may be used:
... ...
Macros may be used:
... ...
Global variables may be used:
... ...

[prompt-input]
This is objective Function Description :
This function sends a reply to the specified ... ... type reply with a length of strlen(s).

This is the declaration of the objective function:
void addReplyBulkCString(client *c, const char *s)

[output]

Example of prompt for code generation scenarios

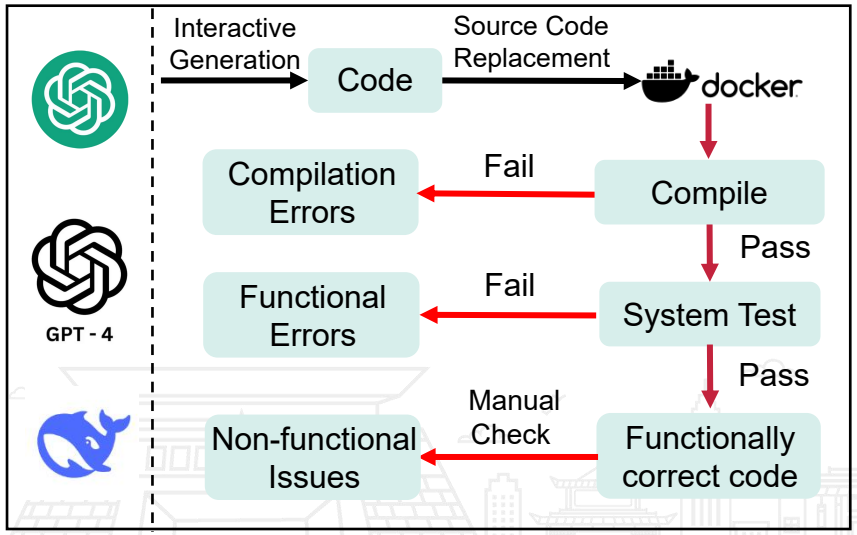
Approach

- **Benchmark construction**
  - 3,926 C code generation/completion tasks from real-world software
  - Available at https://github.com/zhangbuzhang/ObfusEval

Soft.	Original Functions	Symbol Obfuscation Functions	Structure Obfuscation Functions	Semantic Obfuscation Functions	Symbol + Structure Obfuscation Functions	Symbol + Semantic Obfuscation Functions
redis	681	681	215	106	215	106
libvips	203	203	58	17	58	17
lvgl	303	303	115	15	115	15
libgit2	78	78	32	10	32	10
fluent	89	89	30	11	30	11
Total	**1,354**	**1,354**	**450**	**159**	**450**	**159**

Approach

- **Systematic testing and code review**
  - Systematic testing
    - Integrate LLM-generated code into the software
    - Compilation checks
    - Official test suites
  - Manual code review
    - Non-functional issues



Approach

- Testing & Logging

```
    CC script.o
    CC functions.o
    CC function_lua.o
    CC commands.o
    CC strl.o
    CC connection.o
    CC unix.o
    CC logreqres.o
    LINK redis-server
    INSTALL redis-sentinel
    CC redis-cli.o
    CC redisassert.o
    CC cli_common.o
    CC cli_commands.o
    LINK redis-cli
    CC redis-benchmark.o
    LINK redis-benchmark
    INSTALL redis-check-rdb
    INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)
```

Compilation successful

```
25 seconds - unit/cluster/cli
42 seconds - unit/wait
182 seconds - integration/replication
36 seconds - unit/client-eviction
17 seconds - unit/cluster/links
87 seconds - unit/obuf-limits
197 seconds - integration/replication
104 seconds - unit/hyperloglog
147 seconds - unit/tracking
205 seconds - unit/maxmemory
356 seconds - integration/replication
0 seconds - list-large-memory
0 seconds - bitops-large-memory
1 seconds - violations
1 seconds - set-large-memory
276 seconds - defrag

\o/ All tests passed without errors!
```

Testing successful

Test Failure Information

```
*** Preparing to test memory region 7efff2a59000 (
4096 bytes)
.0.0.0.0.0.0.0.0.0.0
Fast memory test PASSED, however your memory can s
till be broken. Please run a memory test for sever
al hours if possible.

----- DUMPING CODE AROUND EIP ------
Symbol: je_malloc_usable_size (base: 0x55aa12ded78
0)
Module: src/redis-server 127.0.0.1:24611 (base 0x5
5aa12c00000)
```

Test Failure Information

```
t_zset.c: In function 'zsetAdd':
t_zset.c:1384:44: error: incompatible type for arg
ument 2 of 'zslFirstInRange'
        znode = zslFirstInRange(zobj->ptr, score,
 ele);
                                            ^~~~~
t_zset.c:333:16: note: expected 'zrangespec * {aka
 struct <anonymous> *}' but argument is of type 'd
ouble'
```

Inconsistent function declarations

```
acl.c: In function 'ACLHashPassword':
acl.c:182:9: error: 'sprintf' is deprecate
d: please avoid use of unsafe C functions.
 prefer use of snprintf instead [-Werror=d
eprecated-declarations]
        sprintf(hex + (i * 2), "%02x", ha
sh[i]);
        ^~~~~~~
In file included from /usr/include/feature
s.h:424:0,
                 from fmacros.h:73,
                 from server.h:33,
                 from acl.c:30:
/usr/include/x86_64-linux-gnu/bits/stdio2.
h:31:1: note: declared here
```

Cache Overflow

EVALUATION

- **Evaluation Setup**
  - Model selection
    - gpt-3.5-turbo-1106、gpt-4-turbo-1106、gpt-4-turbo-0125、DeepSeek-coder-v2
  - Evaluation Metrics
    - Compile Pass Rate (CPR) and Test Pass Rate (TPR)
  - Research questions
    - RQ1：LLM code generation effectiveness
    - RQ2：Code obfuscation effectiveness
    - RQ3：Issues hidden in LLM-generated code

EVALUATION

- RQ1: LLM code generation effectiveness
  - Obfuscated code led to a noticeable decline in the capability of LLMs
  - The average capability decline ranged from 15.3% to 62.5%
  - Symbol + structure obfuscation is effective enough

Software	Model	Original		Symbol		Original (Structure)		Structure		Symbol+Structure	
		CPR	TPR	CPR	TPR	CPR	TPR	CPR	TPR	CPR	TPR
redis	GPT3.5	38.2	11.9	19.0 ⇓	9.7 ↓	44.7	10.7	36.3 ↓	6.5 ⇓	29.3 ⇓	2.9 ⇓
	GPT4-1106	37.0	17.6	31.7 ↓	17.1 ↓	34.4	13.9	36.5 ↑	7.0 ⇓	19.6 ⇓	4.7 ⇓
	GPT4-0125	39.9	20.4	31.8 ↓	17.3 ↓	53.0	13.9	34.9 ⇓	9.8 ⇓	19.5 ⇓	2.8 ⇓
	DeepSeek	39.6	7.2	28.8 ↓	11.5 ⇑	47.4	5.1	32.1 ⇓	5.1 =	22.4 ⇓	4.7 ↓
	Average	38.7	14.3	27.8 ↓28.2%	13.9 ↓2.8%	44.9	10.9	35.0 ↓22.0%	7.1 ⇓34.9%	22.7 ⇓49.4%	3.8 ⇓65.1%
libvips	GPT3.5	58.6	18.2	44.8 ↓	18.7 ↑	70.7	20.7	74.1 ↑	20.7 =	46.6 ⇓	12.1 ⇓
	GPT4-1106	42.9	21.7	32.5 ↓	19.2 ↓	44.8	15.5	44.8 =	20.7 ⇑	27.6 ⇓	13.8 ↓
	GPT4-0125	43.8	25.6	41.4 ↓	22.2 ↓	51.7	27.6	50.0 ↓	24.1 ↓	48.3 ↓	19.4 ↓
	DeepSeek	50.8	29.6	41.3 ↓	24.6 ↓	53.5	25.9	56.9 ↑	22.4 ↓	44.8 ↓	15.5 ⇓
	Average	49.0	23.8	40.0 ↓18.4%	21.2 ↓10.9%	55.2	22.4	56.5 ↑2.4%	22.0 ↓1.8%	41.8 ↓24.3%	15.2 ⇓32.1%
lvgl	GPT3.5	36.7	19.8	35.8 ↓	17.5 ↓	27.3	11.6	27.0 ↓	6.9 ⇓	13.9 ⇓	6.9 ⇓
	GPT4-1106	38.9	26.7	39.7 ↑	24.8 ↓	22.6	13.9	23.5 ↑	12.1 ↓	16.5 ↓	10.4 ↓
	GPT4-0125	46.2	31.0	43.3 ↓	28.1 ↓	28.7	15.7	27.0 ↓	11.3 ↓	20.0 ⇓	11.3 ↓
	DeepSeek	44.2	30.4	42.6 ↓	28.4 ↓	29.6	14.8	28.7 ↓	9.6 ⇓	17.4 ⇓	7.0 ⇓
	Average	41.5	27.0	40.4 ↓2.7%	24.7 ↓8.5%	27.1	14.0	26.6 ↓1.8%	10.0 ↓28.6%	17.0 ⇓37.3%	8.9 ⇓36.4%
libgits	GPT3.5	14.1	14.1	11.5 ↓	7.7 ⇓	13.1	13.1	12.5 ↓	12.5 ↓	6.2 ⇓	3.1 ⇓
	GPT4-1106	38.5	23.1	18.0 ⇓	9.0 ⇓	31.3	25.0	12.5 ⇓	12.5 ⇓	0.0 ⇓	0.0 ⇓
	GPT4-0125	39.7	20.5	12.8 ⇓	6.4 ⇓	34.4	18.8	12.5 ⇓	12.5 ⇓	0.0 ⇓	0.0 ⇓
	DeepSeek	23.1	21.8	14.1 ⇓	12.8 ⇓	18.8	15.6	15.6 ↓	15.6 =	18.6 ↓	9.2 ⇓
	Average	28.9	19.9	14.1 ⇓51.2%	9.0 ⇓54.8%	24.4	18.1	13.3 ⇓45.5%	13.3 ↓26.5%	6.2 ⇓74.6%	3.1 ⇓82.9%
fluent	GPT3.5	27.0	19.1	18.0 ⇓	9.0 ⇓	33.3	30.0	23.3 ⇓	10.0 ⇓	30 ↓	3.3 ⇓
	GPT4-1106	25.3	20.2	12.4 ⇓	10.1 ⇓	30.0	26.7	13.3 ⇓	10.0 ⇓	0.0 ⇓	0.0 ⇓
	GPT4-0125	34.8	29.2	15.7 ⇓	12.4 ⇓	43.0	33.3	20.0 ⇓	10.0 ⇓	16.7 ⇓	6.7 ⇓
	DeepSeek	19.1	14.6	14.6 ↓	9.0 ⇓	13.3	16.7	13.3 =	10.0 ⇓	10.0 ↓	3.3 ⇓
	Average	26.6	20.8	15.2 ⇓42.9%	10.1 ⇓51.4%	29.9	26.7	17.5 ⇓41.5%	10.0 ⇓62.5%	14.2 ⇓52.5%	3.3 ⇓87.6%
Average		36.9	21.1	27.5 ↓25.5%	15.8 ↓25.1%	36.3	18.4	29.7 ↓18.2%	12.5 ⇓32.1%	20.4 ⇓43.8%	6.9 ⇓62.5%

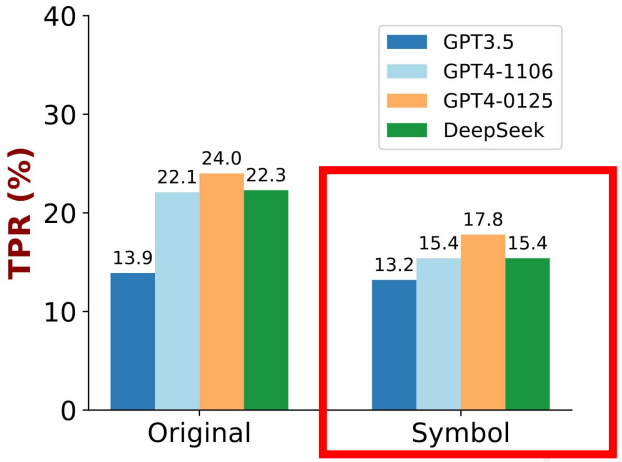
Code generation scenarios

Software	Model	Original		Symbol		Original (Semantic)		Semantic		Symbol+Semantic	
		CPR	TPR	CPR	TPR	CPR	TPR	CPR	TPR	CPR	TPR
redis	GPT3.5	19.0	9.0	30.9 ⇑	11.5 ↑	17.9	13.2	13.2 ↓	7.5 ⇓	36.4 ⇑	11.2 ↓
	GPT4-1106	41.5	25.4	32.9 ↓	14.2 ⇓	51.9	36.8	44.3 ↓	25.5 ⇓	39.6 ↓	20.7 ⇓
	GPT4-0125	39.3	15.4	36.1 ↓	15.9 ↑	43.4	17.9	38.7 ↓	24.5 ⇑	49.1 ↑	25.5 ⇑
	DeepSeek	40.6	24.8	30.9 ↓	4.0 ⇓	54.7	40.6	56.6 ↑	31.1 ↓	53.8 ↓	32.1 ↓
	Average	35.1	18.7	32.7 ↓6.8%	11.4 ⇓39.0%	42.0	27.1	38.2 ↓9.0%	22.2 ↓18.1%	44.7 ↑6.4%	22.4 ↓17.3%
libvips	GPT3.5	19.7	8.4	31.0 ⇑	13.3 ⇑	41.2	11.8	29.4 ↓	5.9 ⇓	41.2 =	10.6 ↓
	GPT4-1106	36.0	22.2	31.5 ↓	18.2 ↓	35.3	23.5	41.1 ↑	23.5 =	35.3 =	17.7 ↓
	GPT4-0125	44.3	27.1	33.5 ↓	22.2 ↓	47.0	29.4	47.0 =	23.5 ↓	29.4 ⇓	29.4 =
	DeepSeek	36.9	25.1	37.0 ↑	21.7 ↓	41.1	23.5	41.1 =	23.5 =	35.3 ↓	23.5 =
	Average	34.2	20.7	33.3 ↓2.6%	18.9 ↓8.7%	41.2	22.1	39.7 ↓3.6%	19.1 ↓13.6%	35.3 ↓14.3%	20.3 ↓8.1%
lvgl	GPT3.5	22.4	18.2	27.2 ↑	20.5 ↑	6.7	0.0	6.7 =	0.0 =	6.7 =	0.0 =
	GPT4-1106	39.6	30.0	31.4 ↓	24.1 ↓	20.0	13.3	20.0 =	13.3 =	13.3 ⇓	13.3 =
	GPT4-0125	44.2	33.00	36.3 ↓	27.1 ↓	20.0	13.3	20.0 =	13.3 =	13.3 ⇓	6.7 ⇓
	DeepSeek	35.2	29.4	28.2 ↓	11.9 ⇓	20.0	13.3	20.0 =	13.3 =	13.3 ⇓	13.3 =
	Average	35.4	27.7	30.8 ↓13.0%	20.9 ↓24.5%	16.7	10.0	16.7 =	10.0 =	11.7 ↓29.9%	8.3 ↓17.0%
libgits	GPT3.5	15.4	12.8	15.4 =	12.8 =	10.0	10.0	10.0 =	10.0 =	20.0 ⇑	10.0 =
	GPT4-1106	10.3	10.3	6.4 ⇓	6.4 ⇓	60.0	30.0	50.0 ↓	30.0 =	20.0 ⇓	20.0 ⇓
	GPT4-0125	12.8	10.3	7.7 ⇓	7.7 ↓	50.0	40.0	50.0 =	30.0 ↓	40.0 ↓	30.0 ↓
	DeepSeek	25.6	21.8	23.0 ↓	19.2 ↓	30.0	30.0	40.0 ⇑	20.0 ⇓	30.0 =	20.0 ⇓
	Average	16.0	13.8	13.1 ↓18.1%	11.5 ↓16.7%	37.5	27.5	37.5 =	22.5 ↓18.2%	27.5 ↓26.7%	20.0 ↓27.3
fluent	GPT3.5	11.2	7.87	16.9 ⇑	11.2 ⇑	0.0	0.0	0.0 =	0.0 =	18.2 ⇑	0.0 =
	GPT4-1106	25.8	23.6	16.9 ⇓	11.2 ⇓	9.1	9.1	18.2 ⇑	9.1 =	9.1 =	9.1 =
	GPT4-0125	30.3	27.0	20.2 ⇓	19.1 ↓	18.2	18.2	18.2 =	9.1 ⇓	18.2 =	9.1 ⇓
	DeepSeek	21.4	18.0	14.6 ⇓	11.2 ⇓	18.2	18.2	18.2 =	18.2 =	9.1 ⇓	9.1 ⇓
	Average	22.2	19.1	17.2 ↓22.5%	13.2 ⇓30.9%	11.4	11.4	13.7 ↑20.2%	9.1 ↓20.2%	13.7 ↑20.2%	6.8 ⇓40.4%
Average		28.6	20.0	25.4 ↓11.2%	15.2 ↓24.0%	29.7	19.6	29.1 ↓2.0%	16.6 ↓15.3%	26.6 ↓10.4%	15.6 ↓20.4%

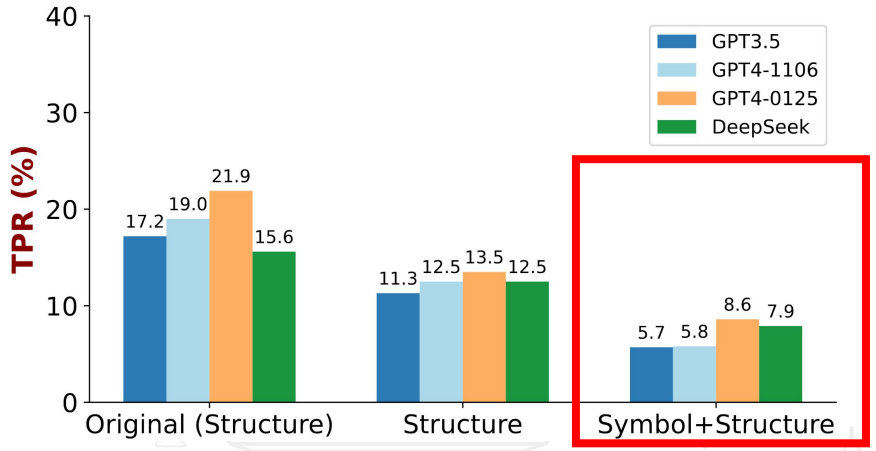
Code completion scenarios

EVALUATION

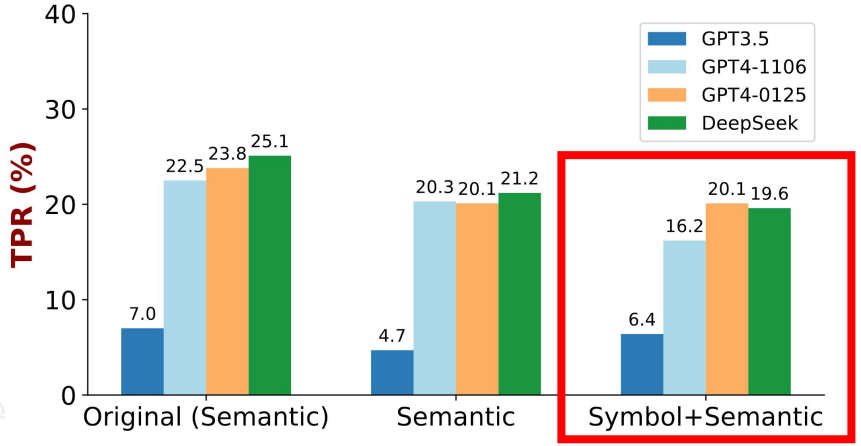
- RQ2：Code obfuscation effectiveness
  - Obfuscated-code-based evaluation more accurately reveals the true capabilities of LLMs
  - Before obfuscation：LLM capabilities vary across different tasks
  - After obfuscation ： GPT4-0125 > DeepSeek v2 > GPT4-1106 > GPT3.5



Symbol obfuscation



Stucture(&symbol) obfuscation



Semantic(&symbol) obfuscation

EVALUATION

- Syntax errors of LLM-generated code

Category	Subcategory	Proportion
Function and Type Declaration Errors	Implicit declaration of function	30.56%
	Type conflict	13.35%
	API parameter count mismatch	1.02%
	Undeclared type	0.31%
Data Structure and Member Access Errors	Non-existent structure member	19.87%
	Misuse of structure pointer	6.60%
	Use → operator to access an integer member	0.23%
Type Conversion and Assignment Errors	Making a pointer from an integer without a cast	8.57%
	Incompatible pointer type	1.25%
	Incompatible type assignment	1.50%
	Redefinition	1.50%
Scope and Definition Errors	Conflict between static and non-static declarations	1.50%
	Incorrect access to structure or union member	0.39%
Other Syntax Errors	Lvalue required as the left operand of assignment	6.60%
	Incorrect use of array, pointer, or vector	1.73%
	Assignment to expression with array type	0.63%
	Incorrect use of parentheses	0.63%
	Invalid binary operands	0.55%
	Expected expression error	0.47%
	Array subscript is not an integer	0.23%
	Subscripted value is pointer to function	0.23%
	Others	2.28%

EVALUATION

- Non-functional code quality issues
  - Resource management、code efficiency、code robustness

```
/* Original code */
Void wtiff_pack2tiff(Wtiff *wtiff, VipsRegion *in,VipsRect *area,
    VipsPel *q){ // Different condition -> Differenet method
    for(int y = area->top;y < RECT_BOTTOM( area );y++) {
       VipsPel *p = (VipsPel *) REGION_ADDR(in,area->left,y);
       if(wtiff->ready->Coding == CODING_LABQ)
            LabQ2LabC( q, p, area->width );
       else if ····· // Omit multiple branches
       else
           memcpy(q,p,area->width *IMAGE_SIZEOF_PEL(wtiff->ready));
           ......

/* LLM-generated code */
Void wtiff_pack2tiff(Wtiff *wtiff, VipsRegion *in,VipsRect *area,
    VipsPel *q){ // Loop through each pixel point
    for(int y = area->top;y < RECT_BOTTOM( area );y++ ) {
        for(int x = area->left;x < area->left + area->width;x++){
            VipsPel *p = (VipsPel *) REGION_ADDR( in, x, y );
            memcpy(q,p,IMAGE_SIZEOF_PEL(wtiff->ready));
              ......
```

low efficiency code

```
/* Original code */
Void clusterUpdateMyselfAnnouncedPorts(void)
{
    if (!myself) // Error Handling
        return;
    deriveAnnouncePorts(&myself->port,&myself->pport,
    &myself->cport);
}


/* LLM-generated code */
Void clusterUpdateMyselfAnnouncedPorts(void)
{
    myself->port = server.cluster_announce_port;
    myself->cport = server.cluster_announce_bus_port;
    myself->pport = server.cluster_announce_tls_port;
}
```

missing error handling

Thanks!

Contact: zhangyuanliang13@nudt.edu.cn