Programa de Pós-graduação em Ciência da Computação (PGCC)
Universidade Estadual de Feira de Santana (UEFS)

PGCC - UEFS

Avaliação Empírica da Eficácia de Grandes Modelos de Linguagem na Refatoração de Projetos Python - Um Estudo de Replicação

Discente: Pedro Carneiro de Souza

Orientadora: Profª Drª Larissa Soares (UNEB/PGGC-UEFS)

Coorientador: Profº Drº Eduardo Figueiredo (UFMG)

Roteiro da Apresentação

1. Introdução
2. Objetivo e Questões de Pesquisa
3. Metologia
4. Resultados Parciais
5. Discussão
6. Conclusão

Introdução – Contextualização

- **Desafios na Qualidade de Software** (Pressman 2010)
  - Manutenibilidade e Legibilidade (Martin, 2008).
    - Desafios da Refatoração (Fowler 2018)
      - Manter a integridade da aplicação
      - Métodos longos e complexos
  - Ferramentas de Análise Estática
    - SonarQube, Pylint, Flake8 etc ...

Introdução – Contextualização

- **Surgimento dos Modelos de Linguagem de Grande Escala(LLMs)**
  - **O que são LLMs?**
    - Modelos treinados em grandes volumes de texto e código
    - Capazes de compreender e gerar linguagem natural e programação
  - ChatGPT, Copilot Chat (OpenAI/Microsoft)
  - Gemini (Google), LLaMA (Meta), Mistral, DeepSeek
  - Impacto na Engenharia de Software
- **Estudos Recentes**
  - **Muitos estudos em geração e explicação de código**
  - Poucos estudos sobre refatoração
  - Existência de lacunas

Objetivos

Esta pesquisa tem como objetivo investigar, por meio de um estudo empírico, a eficácia dos Modelos de Linguagem de Grande Escala (LLMs) **na refatoração de código Python**, com foco na correção de **problemas de manutenibilidad**e identificados em cenários reais de desenvolvimento.

Objetivos – Questões de Pesquisa

01

RQ1 – Até que ponto os LLMs são eficazes na **refatoração de códigos Python** com **problemas de manutenibilidade?**

02

RQ2 – Quais tipos de erros os LLMs cometem com mais frequência ao tentar r**efatorar códigos Python** com **problemas de manutenibilidade?**

03

RQ3 – Até que ponto os **desenvolvedores** consideram as soluções refatoradas pelas LLMs mais legíveis?

Metologia

- **Abordagem Quantitativa e Qualitativa**
  - Correção de Problemas de Manutenibilidade
  - Refatoração de Métodos Python
- **Ferramentas**
  - SonarQube
  - VsCode, Pytest, Tox
- **LLMs**
  - Copilot Chat (4o)
  - Llama 3.3 70B Instruct
  - DeepSeek V3

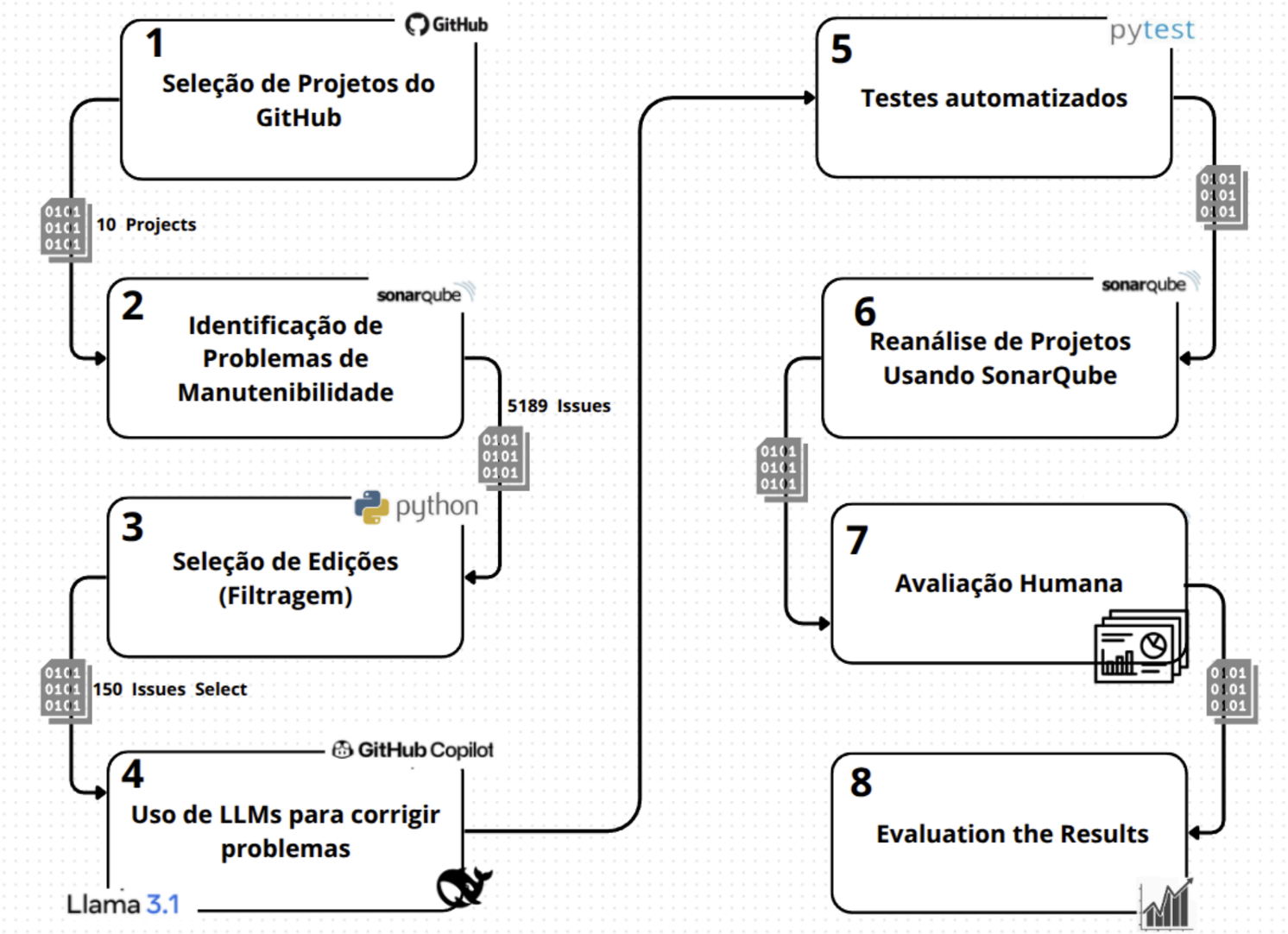
Metologia

Popularidade

Ativo

Testados

Recentes



Testabilidade

Aleatoriedade

Testabilidade

Variabilidade

Etapas do Metodologia

Metologia – Regras do SonarQube

Tabela 3.3: Quantidade de Problema por Projetos

PROJETO	QUANT. DE PROBLEMAS
Django	1.101
Django-Rest	144
Fastapi	152
Mitmproxy	537
Requests	36
Rich	140
Scrapy	240
Sqlalchemy	1670
Pandas	2.735
Poetry	108
Total de Problemas	7.583

Tabela 3.4: Número de Problemas Identificados por Regra e Projetos

SIGLA	REGRAS	PROBLEMAS	RECORRÊNCIA PROJETOS
CCM	Cognitive Complexity of functions should not be too high	30	10
FMP	Functions, methods and lambdas should not have too many parameters	12	6
FNC	Function names should comply with a naming convention	9	5
BCI	Boolean checks should not be inverted	7	4
SLD	String literals should not be duplicated	13	7
LVN	Local variable and function parameter names should comply with a naming convention	6	5
UFP	Unused function parameters should be removed	13	
MIS	Mergeable ”if”statements should be combined	16	8
ULV	Unused local variables should be removed	14	8
SES	’startswith’ or ’endswith’ methods should be used instead of string slicing in condition expressions	5	5
BSV	Two branches in a conditional structure should not have exactly the same implementation	11	5
TUT	Track uses of ”TODO”tags	14	7
	Total	150	

Metologia – Promptings

Few-shot: Treinamento

You are a software engineer specialized in code refactoring to improve maintainability, readability, and quality.
Your goal is to identify and fix maintainability issues in Python code, following best software engineering practices.
The main metric to consider is the cognitive complexity of functions, as defined by the S3776 rule of SonarQube:
"The cognitive complexity of functions should not be too high."
To help you understand the desired refactoring pattern, I will provide examples of code that violate this rule and their respective refactorings.
After that, I will provide you with a piece of code with issues, and you should refactor it following the same principles.
Ex: 1, Ex: 2, Ex: 3, and Ex: 4

Zero-shot

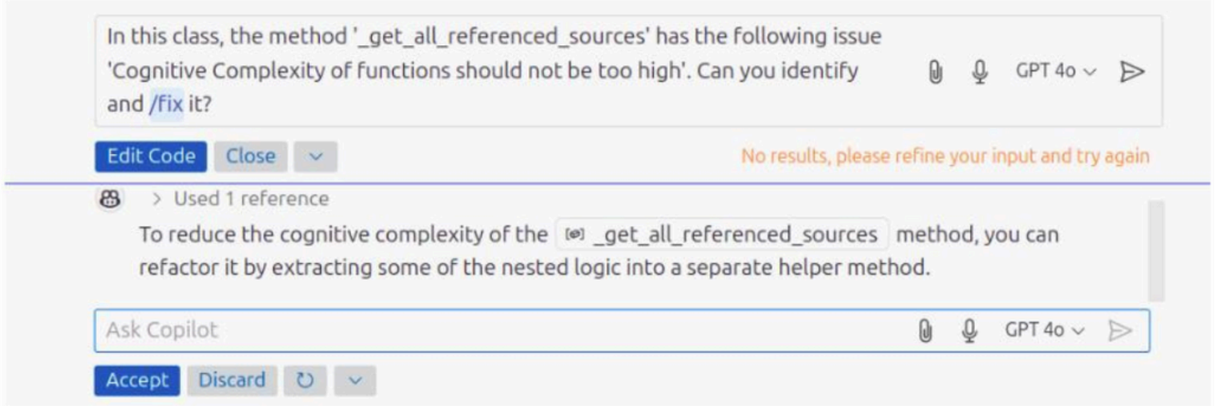
In this class, the method 'def to_internal_value
(self, value):' has the following issue 'Cognitive Complexity of functions should not be too high'. Can you identify and /fix it?

Few -Shot: Solicitação

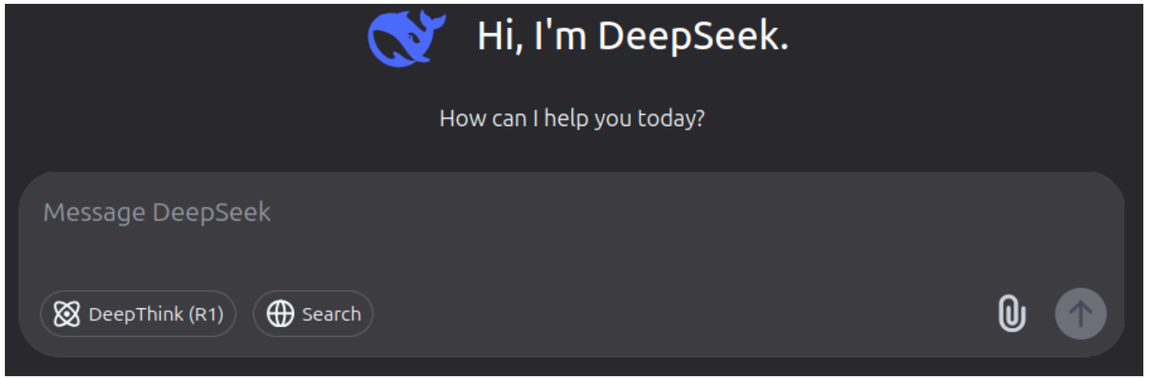
Now that you have seen examples of refactoring, identify and fix the code below to improve its maintainability and readability by issues, keeping the original behavior, without adding new features. Analyze the def inserted_primary_key_rows(self): method, which presents the issue 'String literals should not be duplicated'. Your task is to identify and fix this problem by following good programming practices.
Use only the provided code as a reference, without making assumptions or adding nonexistent details.

Metologia – Interação com LLMs

Copilot Chat 40

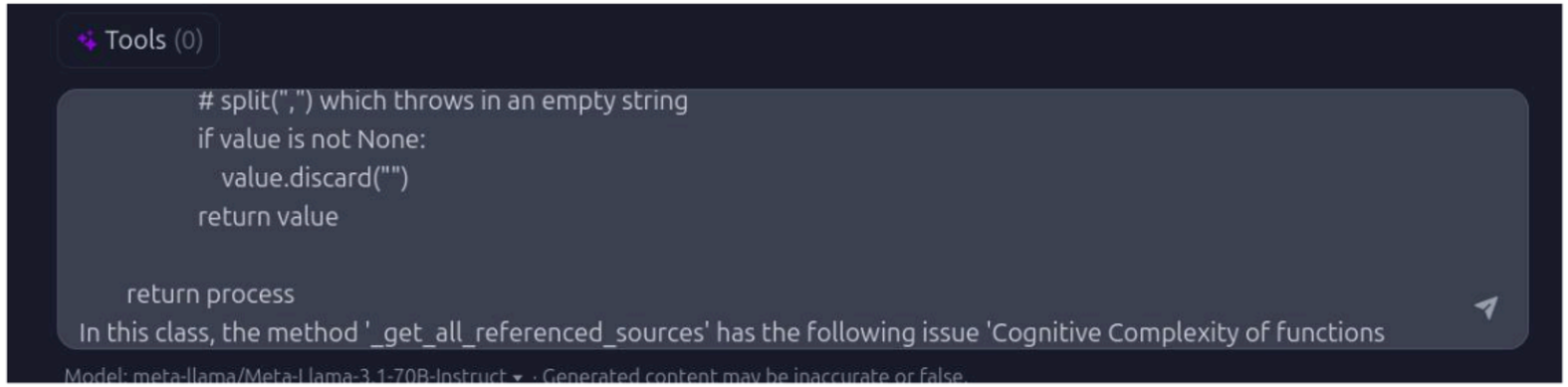


DeepSeek V3



www.huggingface.co

Llama 3.3 70B

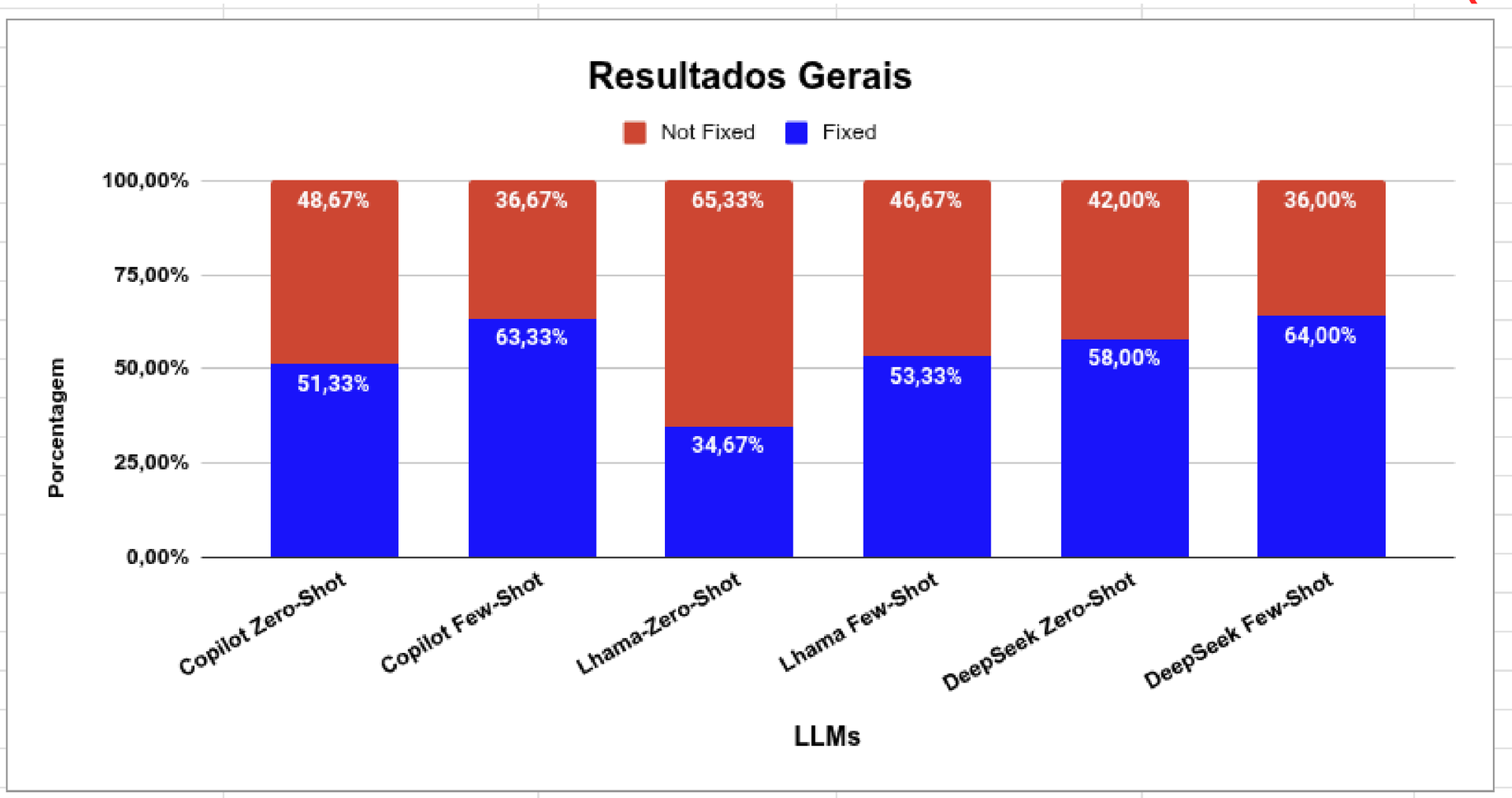


Categorias de Resultados

- Resolvidos
- Não Resolvidos
- Erros de Execução
- Erros de Testes
- Degradados
- Não sugeridos

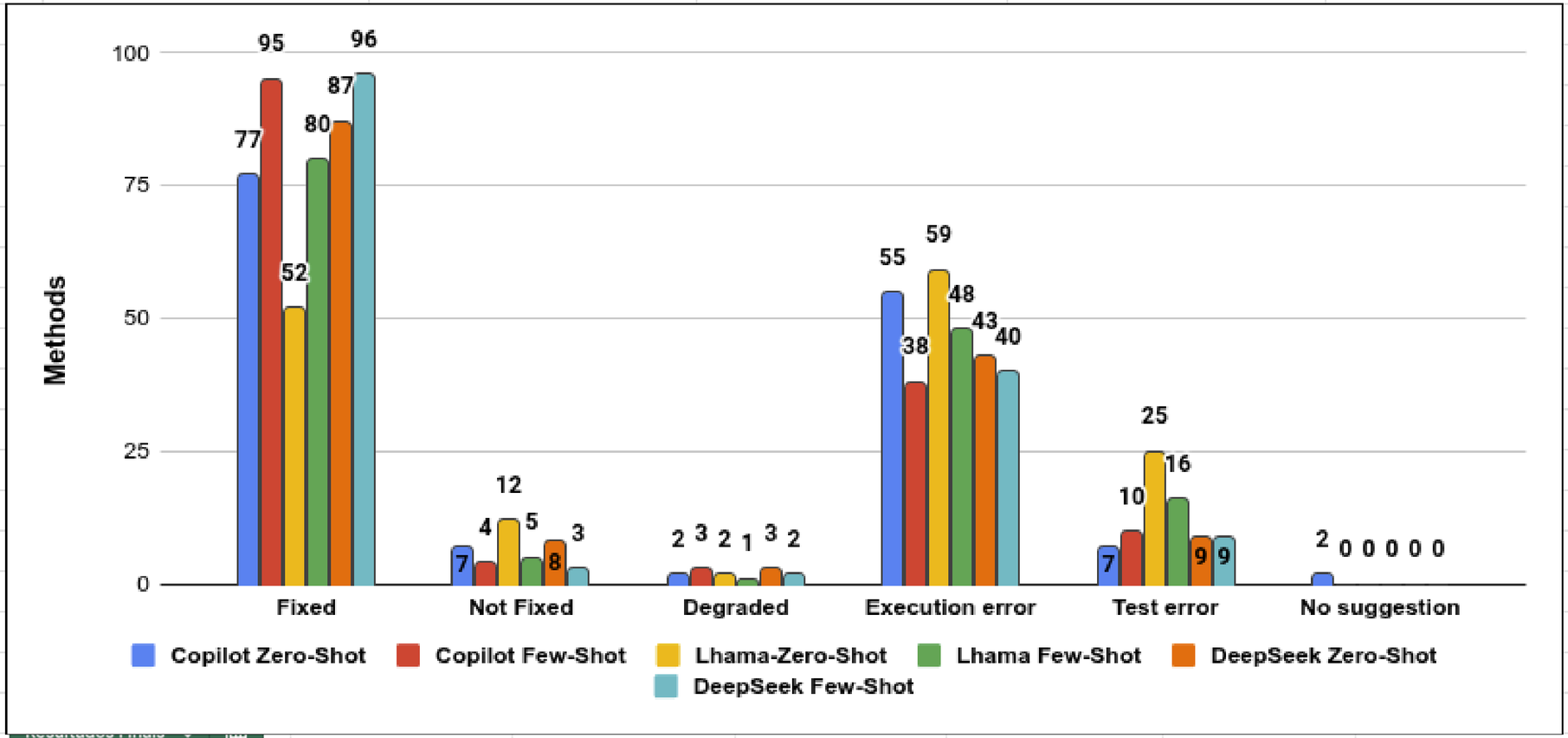
Resultados Preliminares

RQ1 - Até que ponto as LLMs são eficazes na correção de problemas de manutenibilidade em códigos Python?



Resultados Preliminares

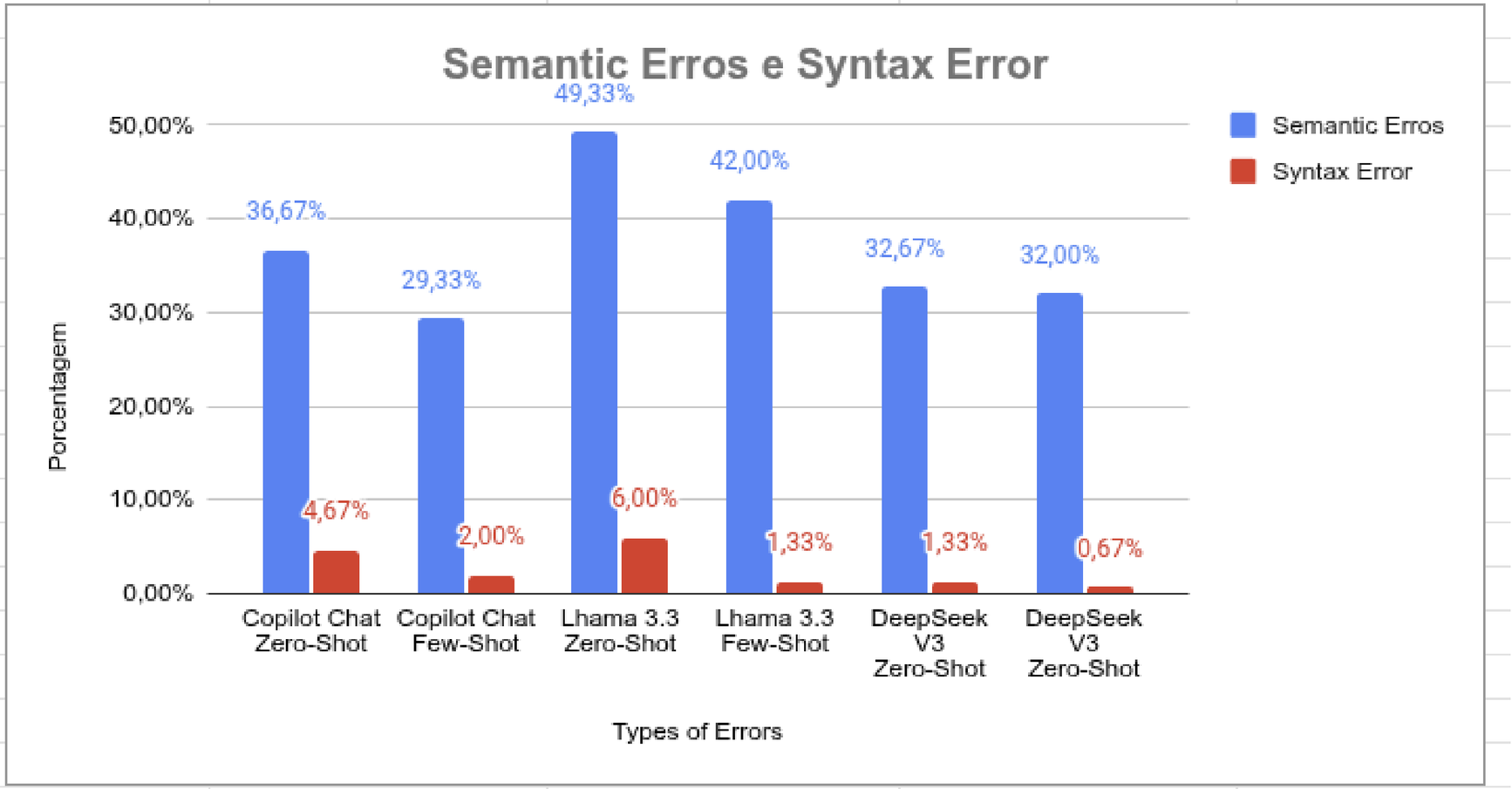
RQ2 - Quais tipos de erros as LLMs cometem com mais frequência ao tentar corrigir problemas de manutenibilidade em código?



Classificação	Copilot Zero-Shot	Copilot Few-Shot	Lhama-Zero-Shot	Lhama Few-Shot	DeepSeek Zero-Shot	DeepSeek Few-Shot
Fixed	51,33%	63,33%	34,67%	53,33%	58,00%	64,00%
Not Fixed	4,67%	2,67%	8,00%	3,33%	5,33%	2,00%
Degraded	1,33%	2,00%	1,33%	0,67%	2,00%	1,33%
Execution error	36,67%	25,33%	39,33%	32,00%	28,67%	26,67%
Test error	4,67%	6,67%	16,67%	10,67%	6,00%	6,00%
No suggestion	1,33%	0,00%	0,00%	0,00%	0,00%	0,00%
	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%

Resultados Preliminares

RQ2 - Quais tipos de erros as LLMs cometem com mais frequência ao tentar corrigir problemas de manutenibilidade em código?



Discussão: Exemplo de Sucesso – EX. 01

Original

```
def lookup_field(name, obj, model_admin=None):
    opts = obj._meta
    try:
        f = _get_non_gfk_field(opts, name)
    except (FieldDoesNotExist, FieldIsAForeignKeyColumnName):
        # For non-regular field values, the value is either a method,
        # property, related field, or returned via a callable.
        if callable(name):
            attr = name
            value = attr(obj)
        elif hasattr(model_admin, name) and name != "__str__":
            attr = getattr(model_admin, name)
            value = attr(obj)
        else:
            sentinel = object()
            attr = getattr(obj, name, sentinel)
            if callable(attr):
                value = attr()
            else:
                if attr is sentinel:
                    attr = obj
                    for part in name.split(LOOKUP_SEP):
                        attr = getattr(attr, part, sentinel)
                        if attr is sentinel:
                            return None, None, None
                value = attr
            if hasattr(model_admin, "model") and hasattr(model_admin.model, name):
                attr = getattr(model_admin.model, name)
        f = None
    else:
        attr = None
        value = getattr(obj, name)
    return f, attr, value
```

S3776 - CCF

Refatorado

```
def lookup_field(name, obj, model_admin=None):
    opts = obj._meta
    try:
        f = _get_non_gfk_field(opts, name)
    except (FieldDoesNotExist, FieldIsAForeignKeyColumnName):
        return _handle_non_regular_field(name, obj, model_admin)
    else:
        attr = None
        value = getattr(obj, name)
    return f, attr, value
```

6

```
def _handle_non_regular_field(name, obj, model_admin):
    if callable(name):
        attr = name
        value = attr(obj)
    elif hasattr(model_admin, name) and name != "__str__":
        attr = getattr(model_admin, name)
        value = attr(obj)
    else:
        attr, value = _get_attr_value(name, obj)
        if hasattr(model_admin, "model") and hasattr(model_admin.model, name):
            attr = getattr(model_admin.model, name)
    return None, attr, value
```

8

```
def _get_attr_value(name, obj):
    sentinel = object()
    attr = getattr(obj, name, sentinel)
    if callable(attr):
        value = attr()
    else:
        if attr is sentinel:
            attr = obj
            for part in name.split(LOOKUP_SEP):
                attr = getattr(attr, part, sentinel)
                if attr is sentinel:
                    return None, None
        value = attr
    return attr, value
```

Discussão: Exemplo de Sucesso – Ex. 02

Original

```
def get_context_data(self, **kwargs):
    view = self.kwargs["view"]
    view_func = self._get_view_func(view)
    if view_func is None:
        raise Http404
    title, body, metadata = utils.parse_docstring(view_func.__doc__)
    title = title and utils.parse_rst(title, "view", _("view:") + view)
    body = body and utils.parse_rst(body, "view", _("view:") + view)
    for key in metadata:
        metadata[key] = utils.parse_rst(metadata[key], "model", _("view:") + view)
    return super().get_context_data(
        **{
            **kwargs,
            "name": view,
            "summary": title,
            "body": body,
            "meta": metadata,
        }
    )
```

S1192 - SLD

Refatorado

```
VIEW_PREFIX = _("view:")

def get_context_data(self, **kwargs):
    view = self.kwargs["view"]
    view_func = self._get_view_func(view)
    if view_func is None:
        raise Http404
    title, body, metadata = utils.parse_docstring(view_func.__doc__)
    title = title and utils.parse_rst(title, "view", self.VIEW_PREFIX + view)
    body = body and utils.parse_rst(body, "view", self.VIEW_PREFIX + view)
    for key in metadata:
        metadata[key] = utils.parse_rst(metadata[key], "model", self.VIEW_PREFIX + view)
    return super().get_context_data(
        **{
            **kwargs,
            "name": view,
            "summary": title,
            "body": body,
            "meta": metadata,
        }
    )
```

Discussão – Alucinação

Original Ex. 01

S117 - LVN

```def dump_stacks(signal=None, frame=None, file=sys.stdout):
    id2name = {th.ident: th.name for th in threading.enumerate()}
    code = []
4   for threadId, stack in sys._current_frames().items():
        code.append("\n# Thread: %s(%d)" % (id2name.get(threadId, ""), threadId))
        for filename, lineno, name, line in traceback.extract_stack(stack):
            code.append('File: "%s", line %d, in %s' % (filename, lineno, name))
            if line:
                code.append("  %s" % (line.strip()))
    print("\n".join(code), file=file)
    if os.getenv("MITMPROXY_DEBUG_EXIT"):  # pragma: no cover
        sys.exit(1)
```

S1172 - UFP

Ex. 02

```# Original
    def prepare_auth(self, auth, url=""):
        """Prepares the given HTTP auth data."""
        ...
-------------------------------------------------
# Refatorado por LLM
    def prepare_auth(self, auth):
        """Prepares the given HTTP auth data."""
        ...
```

Refatorado

```def dump_stacks(signal_parameter=None, frame_parameter=None, output_file=sys.stdout):
    thread_id_to_name = {thread.ident: thread.name for thread in threading.enumerate()}
    stack_code_lines = []
    for thread_id, stack in sys._current_frames().items():
        stack_code_lines.append("\n# Thread: %s(%d)" % (thread_id_to_name.get(thread_id, ""), thread_id))
        for filename, line_number, function_name, line in traceback.extract_stack(stack):
            stack_code_lines.append('File: "%s", line %d, in %s' % (filename, line_number, function_name))
            if line:
                stack_code_lines.append("  %s" % (line.strip()))
    print("\n".join(stack_code_lines), file=file)
    if os.getenv("MITMPROXY_DEBUG_EXIT"):  # pragma: no cover
        sys.exit(1)
```

1

10

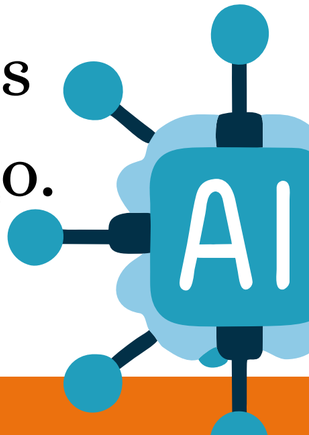
Discussão – Trabalhos Relacionados

- **AlOmar et al. (2024) -** (Refatoração de Código)
  - Refatoração de código
  - Somente 2,3% confiaram totalmente nos modelos
  - Diversidade de interações
- **Bo et al. (2024) -** (Refatoração de código)
  - Gemine(56,2%), GPT(63,6%) , Copilot Chat(53,67%), Llama 3.1(35,33%)
- **Nunes et al (2025) - (** Refatoração de Código **)**
  - **Java**
  - **Avaliação com Humanos - 68,63% (melhorias)**
  - Resultados
    - Ilama (few-shot) = 44,9%
    - Copilot Chat (zero-shot) = 32,9%
    - Llama (zero-shot) = 30%

Conclusão

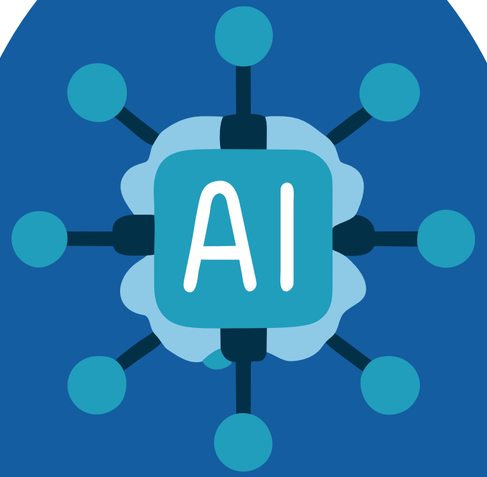
- **Potencial das LLMs**
  - Potencial promissor na refatoração de código

  código Python, com problemas de manutenibilidade
- **Limitações**
  - Introdução de erros e novos problemas de manutenibilidade
  - Necessidade de supervisão humana
- **Contribuições Parciais**
  - Contribuições relevante para Engenharia de Software
  - Catálogo de exemplos de alucinações com análises detalhadas
  - Casos de sucesso ilustrando melhorias na qualidade do código.



Conclusão

- **Limitações e Ameaças**
  - Número de instâncias
  - Número de regras
  - Número de linguagens de programação
  - Execução por um único pesquisador
- **Trabalhos Futuros**
  - Ampliação do conjunto de LLMs
  - Ampliar o número de Instâncias
  - Avaliar e catalogar as técnicas de refatoração
  - Estudo com Humanos (Legibilidade)



Referências

AI, M. (2023). Mistral: Efficiency and scalability in nlp. https://mistral.ai. Acesso em: 30 nov. 2024.

AlOmar, E. A., AlRubaye, H., Mkaouer, M. W., Ouni, A., e Kessentini, M. (2024). How to refactor this code? an exploratory study on developer-chatgpt refactoring conversations. Proceedings of the 21st International Conference on Mining Software Repositories, p´aginas 202–206.

ANTHROPIC (2023). Introducing claude. https://www.anthropic.com/news/introducing-claude. Acesso em: 30 nov. 2024.

Beck, K. e Andres, C. (2004). Extreme Programming Explained: Embrace Change. Addison-Wesley, xp series edi¸c˜ao.

Benko, A. e Sik L´anyi, C. (2009). History of artificial intelligence. Encyclopedia of Information Science and Technology, Second Edition, p´agina 4.

Bo, L., Yanjie, J., Yuxia, Z., Nan, N., Guangjie, L., e Hui, L. (2024). An empirical study on the potential of llms in automated software refactoring. arXiv preprint arXiv:2411.04444v1.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kapoor, A., Gao, J., Shinn, A., Steiner, B., Austin, P., Berner, C., et al. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.1465v4.

Brunette, E. S., Flemmer, R. C., e Flemmer, C. L. (2009). A review of artificial intelligence. 2009 4th International Conference on Autonomous Robots and Agents.

Campbell, A. e Papapetrou, P. (2013). SonarQube in Action. Manning Publications,
1nd edi¸c˜ao.

Ceccon, D. (2024). Anthropic apresenta o claude 3, nova vers˜ao de sua llm. https://iaexpert.academy/2024/03/27/anthropic-apresenta-claude-3-nova-versao-sua-llm/. Acesso em: 30 nov. 2024.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., e et al. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374v2.

Coello, C., Alimam, M., e Kouatly, R. (2023). Effectiveness of chatgpt in coding: A comparative analysis of popular large language models. Rebista Digital, 4:114–125.

DeepMind (2024). Gemini: A new era of ai. https://deepmind.google/technologies/gemini/. Acesso em: 30 nov. 2024.

Duvall, P. M., Matyas, S., e Glover, A. (2007). Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley Professional, 1th edi¸c˜ao.

Feathers, M. (2004). Working Effectively with Legacy Code. Pearson Education, Upper Saddle River, NJ.

Fowler, M. (2018). Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 2nd edi¸c˜ao.

Fowler, M. (2023). D´ıvida t´ecnica: Uma met´afora ´Util para pensar sobre design de software. Blog de Martin Fowler. Acesso em: 01 dez. 2024.

Gemini, G. (2024). Gemini - chat to supercharge your ideas. https://gemini.google/. Acesso em: 30 novembro. 2024.

GitHub (2024). About github copilot individual. https://docs.github.com. Accessed: 2024-09-24.

Hernandez, D. e et al. (2022). Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. arXiv preprint arXiv:2209.07858v2. Version 2, last revised 22 Oct 2022.

Humble, J. e Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Pearson Education.

Kim, G., Behr, K., e Spafford, G. (2018). The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win. IT Revolution Press, 5th anniversary edition edi¸c˜ao.

Marcus, G., Leivada, E., e Murphy, E. (2023). A sentence is worth a thousand pictures: Can large language models understand human language. arXiv preprint arXiv:2308.00109v2.

Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, 1st edi¸c˜ao.

McCorduck, P., Minsky, M., Selfridge, O., e Simon, H. A. (1983). History of artificial intelligence. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).

...