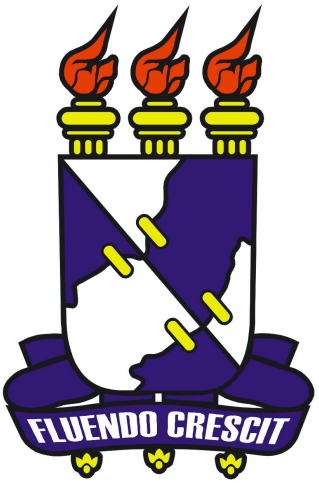


UNIVERSIDADE FEDERAL DE SERGIPE

Evolução de Software

Sanderson Lacy Alves dos Santos e Vinícius Dias Valença

Atividade 1
Refatoração e testes de regressão

São Cristóvão
2025

Sumário

1. Objetivo

Objetivo dessa atividade é classificar issues de um projeto open-source bem sucedido em:

- 1) Refatoração
- 2) Teste de Regressão

Além disso, avaliar o impacto dessas issues na evolução do software.

2. Projeto

Para o desenvolvimento da pesquisa, foi escolhido o projeto open-source Pytorch disponível através do repositório aberto GitHub no endereço https://github.com/pytorch/pytorch. O Pytorch é uma biblioteca desenvolvida para atividades de machine learning em Python, largamente utilizada por pesquisadores da área e amplamente abraçada pela comunidade, prova disso é que cumpriu os requisitos básicos para esta tarefa.
Possui no momento da confecção deste artefato 3666 contribuidores, 14563 issues abertas, 34009 issues fechadas, 96224 pull requests fechados e 1068 pull requests abertos.

3. Repositório no GitHub

Para armazenar os scripts desenvolvidos na atividade foi criado um repositório privado no GitHub disponível através do endereço https://github.com/ViniDias1/COMP0441_20242_Vinicius_Dias_Sanderson_Lacy. Lá é possível encontrar tanto os scripts quanto a versão markdown do presente documento.

4. Modelo de classificação

Para classificar as issues entre Refatoração e Teste de Regressão, foi utilizado um modelo do tipo Zero-Shot-Classification, dado que ele não precisa de nenhum pré-treinamento. O modelo está disponibilizado através do Hugging Face no endereço https://huggingface.co/facebook/bart-large-mnli.

5. Dump de issues e classificação

Utilizando a API do Github, foi feito um dump de mais de 300 issues de forma aleatória. A partir disso, foi utilizado o modelo NLI "Hugging Face Zero-shot-classification facebook/bart-large-mnli" para analisar a issues e classificá-las. Após isso, as issues foram salvas em um banco de dados local (PostgreSQL). Foi utilizado o token do GitHub para fazer as requisições à API visando evitar qualquer limitação de tempo, quantidade ou tamanho das requisições.

GITHUB_TOKEN e **DB_PASSWORD** foram registradas como variáveis de ambiente, por motivos de segurança. Basta usar os dois comandos abaixo no terminal e continuar com a execução.

```
$env:GITHUB_TOKEN="SEU_TOKEN"
$env:DB_PASSWORD="SENHA_DO_DB"
```

Para realizar o dump, a função "fetch_issues()" foi executada:

```
def fetch_issues(page, per_page):
    params = {"page": page, "per_page": per_page, "state": "closed"}
    response = requests.get(GITHUB_API_URL, headers=HEADERS, params=params)
    response.raise_for_status()
    return response.json()
```

Com as issues em mãos, o próximo passo é analisar as issues e classificá-las. Dito isso, as issues do dump são passadas para o modelo Zero-Shot-Classification.

```
from transformers import pipeline

classificador = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

def issue_classification(titulo, descricao):

    categorias = ["Refatoração", "Teste de Regressão"]

    texto = f"{titulo} {descricao}"

    resultado = classificador(texto, candidate_labels=categorias)

    categoria_mais_provavel = resultado["labels"][0]
    pontuacao = resultado["scores"][0]

    limite = 0.6

    if pontuacao >= limite:
        return categoria_mais_provavel
    else:
        return None
```

Após a classificação, temos um conjunto de issues classificadas entre Refatoração e Teste de Regressão salvas em banco.

5. Análise da evolução

Em cima da base de dados, consultas SQL foram realizadas para obtermos uma visão geral sobre algumas métricas.

5.1 Número de issues de testes de regressão

SQL: **SELECT COUNT(*) numero_testes_regressao**
FROM issue
WHERE tema_relacionado = 'Teste de Regressão'

	numero_testes_regressao bigint
1	13

5.2 Número de issues de refatoração
SQL: **SELECT COUNT(*) numero_refatoracao**
FROM issue
WHERE tema_relacionado = 'Refatoração'

	numero_refatoracao bigint
1	341

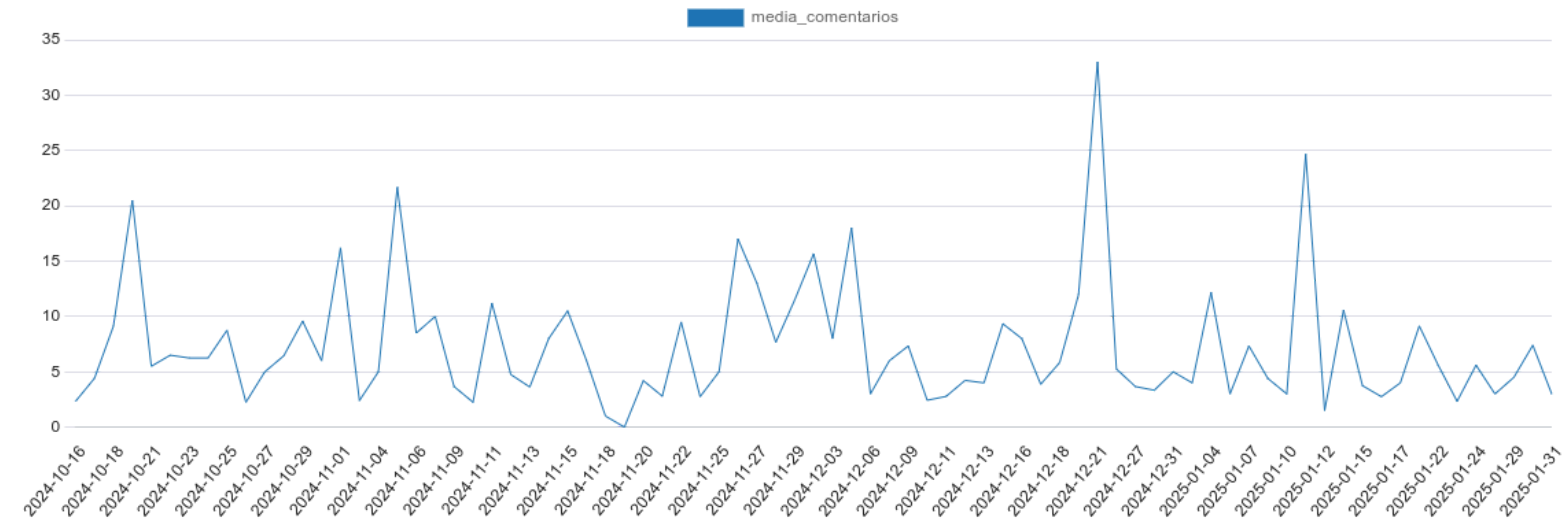
5.3 Média de comentários por tema relacionado

SQL: **SELECT tema_relacionado, AVG(comments_count) media_comentarios**
FROM issue
GROUP BY tema_relacionado

	tema_relacionado character varying	media_comentarios numeric
1	Teste de Regressão	7.1538461538461538
2	Refatoração	6.9149560117302053

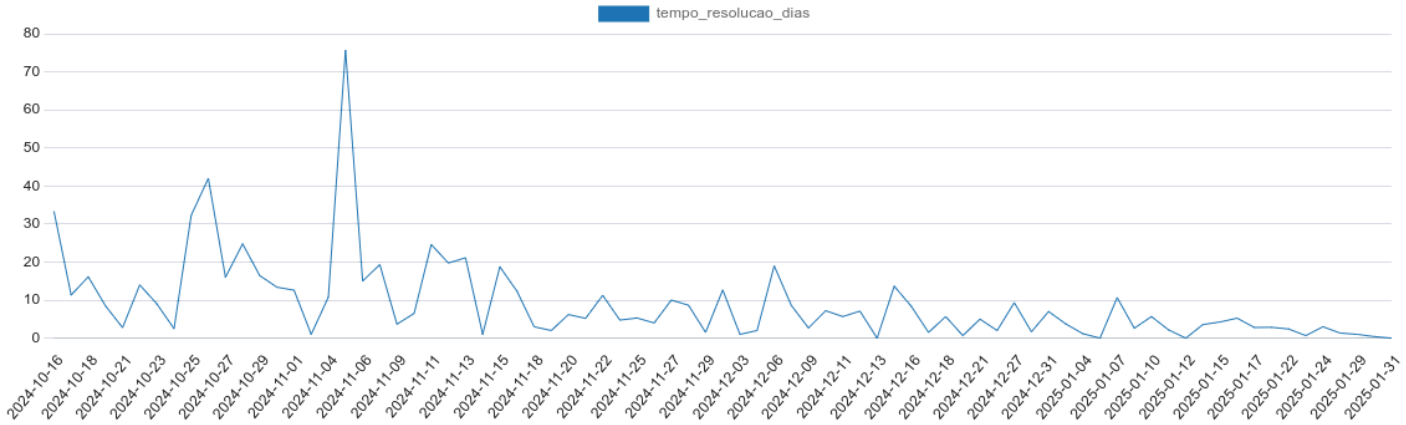
5.4 Média de comentários ao longo do tempo

Podemos ver com a seguinte consulta a evolução no número de comentários ao longo do tempo.
SQL: **SELECT created_at::DATE momento_criacao, AVG(comments_count) media_comentarios**
FROM issue
GROUP BY created_at::DATE
ORDER BY momento_criacao



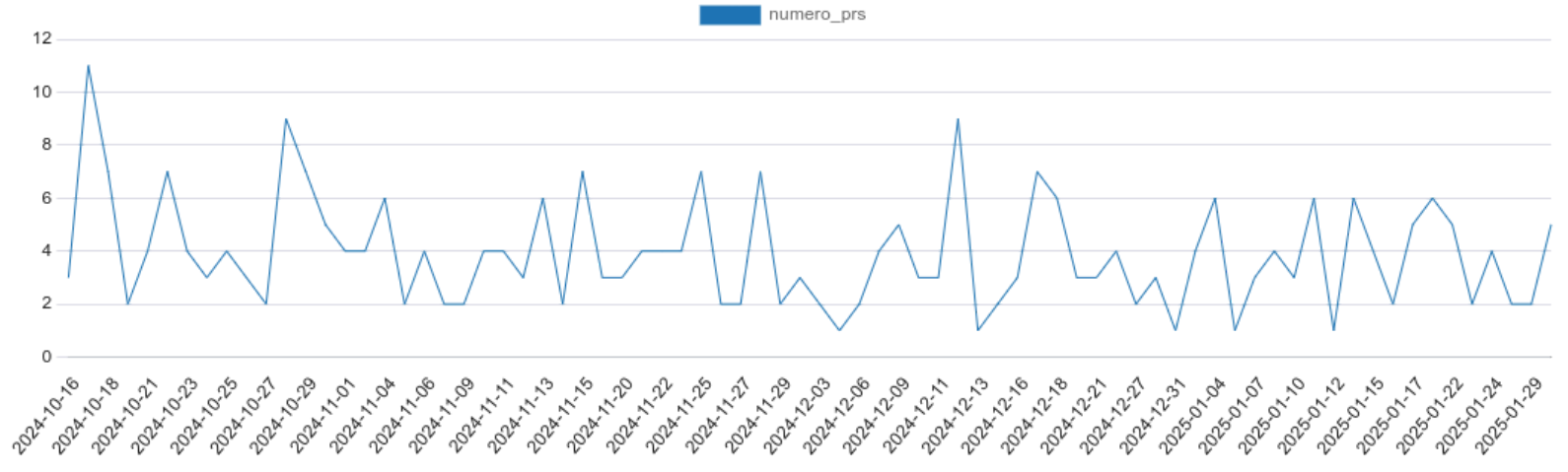
5.5 Tempo de resolução em dias ao longo do tempo

SQL**: SELECT created_at::DATE momento_criacao, AVG(tempo_resolucao_dias) tempo_resolucao_dias**
FROM issue
GROUP BY created_at::DATE
ORDER BY momento_criacao



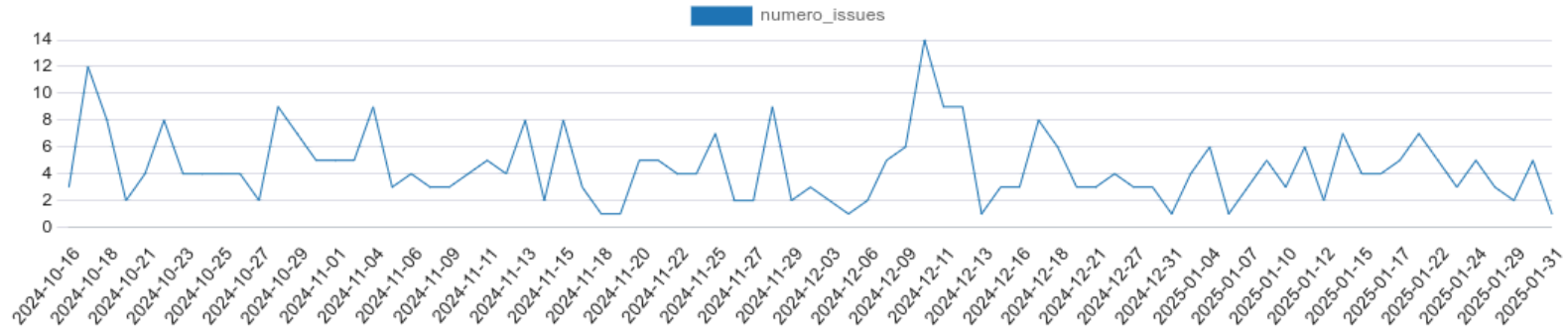
5.6 Números de pull requests ao longo do tempo

SQL: **SELECT created_at::DATE momento_criacao, COUNT(*) numero_prs**
FROM issue
WHERE html_url LIKE '%pull%'
GROUP BY created_at::DATE
ORDER BY momento_criacao



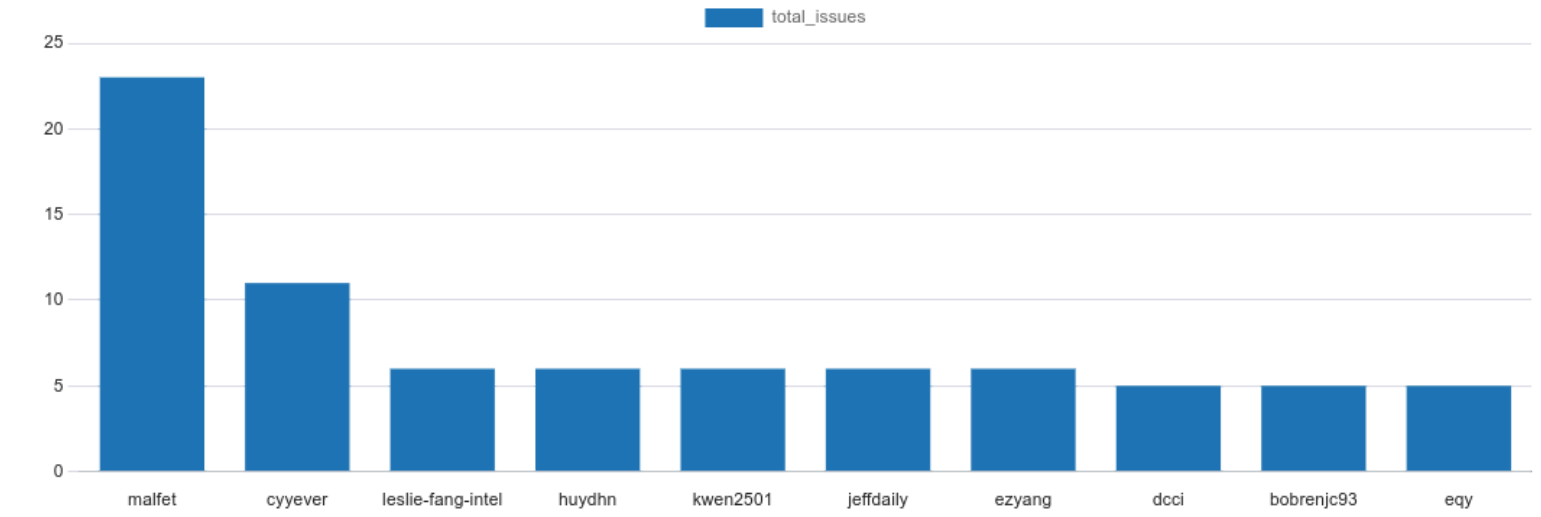
5.7 Quantidade de issues criadas ao longo do tempo

SQL: **SELECT created_at::DATE momento_criacao, COUNT(*) numero_prs**
FROM issue
WHERE html_url LIKE '%pull%'
GROUP BY created_at::DATE
ORDER BY momento_criacao



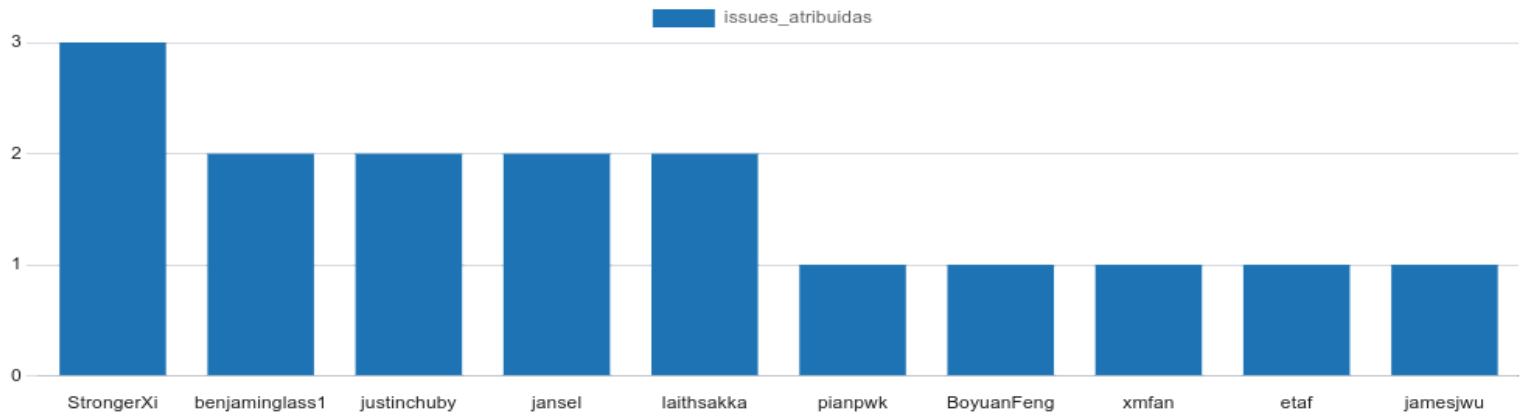
5.8 Top 10 contribuidores com mais issues que viraram pull requests

SQL: **SELECT user_login user, COUNT(*) AS total_issues**
FROM issue
WHERE html_url LIKE '%pull%'
GROUP BY user_login
ORDER BY total_issues DESC
LIMIT 10



5.9 Top 10 contribuidores com mais issues atribuídas

SQL: **SELECT assignee, COUNT(*) issues_atribuidas FROM issue**
WHERE assignee IS NOT NULL
GROUP BY assignee
ORDER BY issues_atribuidas DESC
FETCH FIRST 10 ROWS ONLY



Podemos ver que a quantidade de issues associadas a testes de regressão é consideravelmente menor que a de issues de refatoração. Isso indica que o projeto não sofre mudanças abruptas, de modo que código adicionado não altera significativamente o funcionamento do que já existe. Também pode significar que a cobertura dos testes é boa o suficiente para que poucas demandas de teste sejam criadas.

Outra métrica interessante para analisarmos é a quantidade de comentários: um maior número de comentários numa issue pode nos dar algumas pistas da evolução do projeto como uma maior complexidade na demanda, ou a demanda gera um impacto maior no projeto. Os temas de testes de regressão de refatoração possuem uma média de 7 comentários por issue, sendo que a média de comentários para issues de testes de regressão são ligeiramente maiores que para issues de refatoração.

Outra métrica a ser analisada é o número de issues ao longo do tempo: um número maior de demandas pode indicar a existência de um número grande de bugs ou muitas features sendo implementadas. O número de issues x tempo varia bastante em nossa amostra, tendo uma leve inclinação à queda.

O tempo de resolução é um indicativo relevante para compreendermos a qualidade técnica da equipe de desenvolvimento, e o nível dos problemas em aberto no projeto. Em nossa amostra, percebemos queda no tempo de resolução.

O número de pull requests ao longo do tempo indica se há um desenvolvimento contínuo, e que manutenções devem estar sendo realizadas nos artefatos. A variação de pull requests em nossa amostra não mudou.

Em última instância, podemos analisar a participação dos desenvolvedores no projeto. Obtivemos os contribuidores com mais issues que se tornaram pull requests, e com mais issues atribuídas. Em ambas as análises, percebemos que certos desenvolvedores destacam-se frente aos outros em relação aos números, mas que de forma alguma monopolizam o desenvolvimento.

6. Observações adicionais

Como foi utilizado um modelo de classificação sem treinamento prévio, é esperado que certas falhas de classificação ocorram em nossa análise. Buscando minimizar este fato, foi feita uma análise manual e superficial das classificações obtidas. Nenhum resultado incorreto foi encontrado usando esta abordagem. Além disso, o projeto não utiliza de milestones ou prioridades, por isso essas métricas foram desconsideradas nessa análise. O número de assignees também foi aquém.

7. Conclusão

A partir das issues classificadas e do entendimento trazido pelas consultas SQL, podemos concluir que existem diversos fatores que contribuem para entendimento da evolução do software. Em projetos grandes e consolidados como o Pytorch, é possível ver como a comunidade é ativa, empenhada e organizada para entregar um software de alta qualidade. Conseguimos, dessa forma, compreender sua evolução, pois revela desafios enfrentados, melhorias implementadas e prioridades da comunidade. Esse processo ajuda a identificar padrões de desenvolvimento, entender decisões sobre a arquitetura e acompanhar a introdução ou remoção de funcionalidades. Além disso, permite observar como a colaboração entre contribuidores impulsiona a inovação e a maturidade do projeto.