Desempenho vs. Reconhecimento no OSS Global: Uma Análise sobre o Reconhecimento de Desenvolvedores de Países Emergentes no GitHub

Sophia Mendes[1],
Thiago Andrade Ramalho[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
R. Dom José Gaspar, 500 - Coração Eucarístico, Belo Horizonte - MG, 30535-901

1. Introdução

A participação de desenvolvedores de países emergentes, como Brasil e Índia, em projetos de software de código aberto (OSS) tem recebido crescente atenção em estudos recentes. Pesquisas como [Barbosa et al. 2022] apontam para uma presença cada vez maior desses países no ecossistema global de OSS. Entretanto, grande parte dessas análises enfatiza o volume de contribuições. Permanecem abertas questões fundamentais: em que medida esses profissionais estão envolvidos em funções centrais de coordenação e decisão nos projetos? Ou sua atuação se concentra, predominantemente, em tarefas mais periféricas?

A subvalorização sistemática desses profissionais já é documentada no mercado tradicional. Segundo o [HackerRank 2020], desenvolvedores americanos ganham quase três vezes mais ($109.167/ano) que seus pares indianos ($38.229/ano). Esta disparidade não apenas persiste, mas se expande para novos contextos geográficos. O mercado de trabalho remoto tem testemunhado a emergência de outros países emergentes, como o Brasil, competindo por posições similares às tradicionalmente ocupadas por desenvolvedores indianos, evidenciando a institucionalização de hierarquias salariais baseadas em geografia ao invés de mérito técnico.

O diferencial desta pesquisa reside em fornecer evidências quantitativas e objetivas para investigar se esse padrão de subvalorização se replica no ambiente OSS.

2. Objetivo

Investigar se desenvolvedores de países emergentes, especificamente do Brasil, são subvalorizados em termos de reconhecimento e influência em projetos de software de código aberto internacionais, mesmo quando apresentam desempenho e participação comparáveis ou superiores aos de desenvolvedores de países desenvolvidos, especificamente da Alemanha, caracterizando assim o fenômeno da "mão de obra barata" no ecossistema global de software.

3. Metodologia

3.1. Visão Geral do Pipeline de Coleta

A metodologia de coleta de dados foi estruturada em um pipeline sequencial de cinco scripts principais, projetados para capturar diferentes dimensões de atividade e colaboração em repositórios de código aberto do GitHub. O pipeline segue uma arquitetura de dependências rígida, onde cada script subsequente depende dos dados gerados pelos anteriores, garantindo consistência e integridade dos dados coletados.

3.2. Seleção de Repositórios (script3.py)

Objetivo: Identificar e selecionar repositórios representativos para análise comparativa entre desenvolvedores de países emergentes e desenvolvidos.

Metodologia: O script utiliza a API de busca do GitHub (`/search/repositories`) para identificar os 200 repositórios mais populares (ranqueados por número de estrelas). Para cada repositório selecionado, o sistema busca de forma exaustiva todos os contribuidores através de paginação completa da API (`/repos/{owner}/{repo}/contributors`). Para cada contribuidor identificado, o script coleta informações do perfil (`/users/{login}`) e aplica algoritmos de geocodificação para determinar o país de origem baseado no campo `location` do perfil. O repositório só será incluído na seleção se possuir um ou mais contribuidores dos países alvo.

Países-alvo: Brasil, Índia, Alemanha e Estados Unidos foram selecionados como representantes de economias emergentes (Brasil, Índia) e desenvolvidas (Alemanha, Estados Unidos).

3.3. Identificação Completa de Países (script2.py)

Objetivo: Expandir a identificação geográfica para todos os contribuidores dos repositórios selecionados.

Metodologia: O script processa todos os repositórios identificados no Script 1 e realiza uma coleta exaustiva de todos os contribuidores de cada repositório. A identificação de países utiliza uma abordagem híbrida de três etapas:

1. **Mapeamento direto**: Busca em dicionário de aliases para termos comuns (“Brasil”, “USA”, “”).
2. **Identificação regional**: Reconhecimento de estados/províncias e cidades (ex.: “SP”, “California”, “Mumbai”).
3. **Geocodificação**: Uso da API Nominatim (OpenStreetMap) para localizações complexas, seguido de normalização via biblioteca `pycountry`.

3.4. Métricas de Repositórios (script1.py)

Objetivo: Coletar métricas quantitativas detalhadas dos repositórios para caracterização do contexto de desenvolvimento.

Metodologia: Para cada repositório selecionado, o script coleta métricas através de múltiplos endpoints da API GitHub:

- Metadados básicos (`/repos/{owner}/{repo}`)
- Contagem de pull requests (`/repos/{owner}/{repo}/pulls`)
- Histórico de commits com paginação completa
- Métricas de atividade (issues, releases, colaboradores)
- Cálculo de métricas derivadas (tempo médio de resposta, tempo até merge)

Métricas coletadas: 20 métricas quantitativas incluindo estrelas, forks, commits, PRs (abertos/mergeados), tempo de resposta em issues, periodicidade de releases, e contagem de mantenedores ativos.

3.5. Métricas Individuais de Desenvolvedores (script4.py)

Objetivo: Coletar métricas detalhadas de performance e comportamento individual de cada desenvolvedor identificado.

Metodologia: Implementação assíncrona utilizando `asyncio` e `aiohttp` para coleta paralela de métricas individuais de desenvolvedores. Para cada desenvolvedor presente no arquivo `users_countries.csv`, o sistema coleta:

- **Métricas de contribuição**: PRs abertos/mergeados, commits totais, issues abertas.
- **Métricas de colaboração**: Reviews submetidos, taxa de aprovação de PRs.
- **Métricas temporais**: Período de contribuição, frequência de atividade.
- **Métricas de influência**: Estrelas em repositórios próprios, nível de permissão.
- **Métricas de rede**: Taxa de solicitação como reviewer.

3.6. Análise de Interações Sociais (script5_*.py)

Objetivo: Mapear interações sociais e colaborativas entre desenvolvedores para análise de redes de colaboração.

Metodologia: O Script 5 foi modularizado em seis sub-scripts especializados, cada um focando em tipos específicos de interação:

- **Pull Requests (script5_prs.py)**: Coleta PRs, reviews e comentários em PRs, capturando interações diretas de revisão de código.
- **Issues (script5_issues.py)**: Mapeia abertura de issues e comentários, representando interações de suporte e discussão técnica.
- **Commits (script5_commits.py)**: Analisa commits individuais e coautorias, identificando colaborações diretas no código.

Metodologia temporal: Todos os sub-scripts aplicam filtro temporal restritivo (2020–2025) para garantir relevância temporal dos dados.

Output: Arquivos separados de edges (interações) e nodes (desenvolvedores) para cada tipo de interação, permitindo análise modular de diferentes aspectos da rede de colaboração.

3.7. Considerações Técnicas

Gerenciamento de Rate Limits: Todos os scripts implementam rotação automática entre tokens de autenticação GitHub e sistema de retry inteligente com backoff exponencial.

Integridade de Dados: Pipeline com verificação de dependências rígida — execução fora da ordem resulta em falhas controladas com mensagens de erro específicas.

Escalabilidade: Uso extensivo de processamento paralelo/assíncrono adaptado às limitações de cada endpoint da API GitHub.

Recuperação de Falhas: Salvamento incremental em todos os scripts de longa duração, permitindo retomada de coleta após interrupções.

4. Dificuldades

A principal dificuldade enfrentada foi a coleta de dados via API REST do GitHub. O volume de dados necessário ultrapassa amplamente o limite de requisições por token, obrigando o desenvolvimento de scripts que operassem com múltiplos tokens simultaneamente (14 no total). Mesmo assim, e trabalhando com amostra reduzida, as limitações da API representaram um obstáculo significativo.

Além disso, o tempo de execução completo dos scripts ultrapassou 78 horas de processamento contínuo.

Outra dificuldade relevante foi dominar a ferramenta Looker Studio para visualização dos dados.

5. Resultados

link do dashboard

References

Barbosa, A., Santana, C. R. B., and Baltes, S. (2022). The geography of open source software: Evidence from GitHub. *Technological Forecasting and Social Change*, 176:121478.

HackerRank (2020). 2020 developer skills report. Technical report, HackerRank.