Relatório Técnico: Análise de Governança via Padrões de Commit

Responsável: Irandi (Dupla 3 - Integrante E) **Objeto de Análise:** Histórico de versionamento do `google/langextract`

1. Objetivo

O objetivo desta etapa foi mensurar o nível de disciplina e governança do projeto através da análise semântica e sintática das mensagens de *commit*. A premissa da Engenharia de Software é que projetos maduros utilizam padrões consistentes (como *Conventional Commits* ou *Imperative Mood*) para facilitar a rastreabilidade.

2. Metodologia Implementada

Desenvolvemos um script em Python que combina **Processamento de Linguagem Natural (IA)** com regras de negócio baseadas no estilo de código do Google. Abaixo, detalhamos as funções principais utilizadas na análise.

2.1. Coleta de Dados (Amostragem Estatística)

Para garantir relevância, não analisamos commits isolados. Implementamos uma função para extrair os últimos **100 commits** via API do GitHub, garantindo uma visão histórica do projeto:

Python
```
def get_commits(owner, repo, limit=100):
    # Conecta na API do GitHub para baixar o histórico recente
    url = f"https://api.github.com/repos/{owner}/{repo}/commits?per_page={limit}"
    response = requests.get(url)
    # Processa e limpa as mensagens recebidas
    if response.status_code == 200:
        return [item['commit']['message'].split('\n')[0] for item in response.json()]
    return []
```

2.2. Análise Semântica com IA

Utilizamos o modelo `sentence-transformers/all-MiniLM-L6-v2` do Hugging Face. Este modelo converte textos em vetores matemáticos (*embeddings*), permitindo calcular a similaridade entre o que os desenvolvedores escreveram e o que consideramos "Boas Práticas".

Python
```
# Inicialização do modelo de IA para análise semântica
model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2")
```

```
# Criação dos vetores de referência (Gabarito de Qualidade)
padrao_bom = [
    "feat: add functionality", "fix: resolve bug", # Conventional Commits
    "Add new feature support", "Update dependency" # Imperative Mood (Google Style)
]
embeddings_bom = model.encode(padrao_bom)
```

2.3. Algoritmo de Classificação Híbrida

O diferencial da nossa abordagem foi o refinamento do algoritmo. Inicialmente focado apenas em *Conventional Commits*, ajustamos a lógica para validar também o **Imperative Mood** (verbos de ação), padrão comum em projetos *Open Source* de alta maturidade.

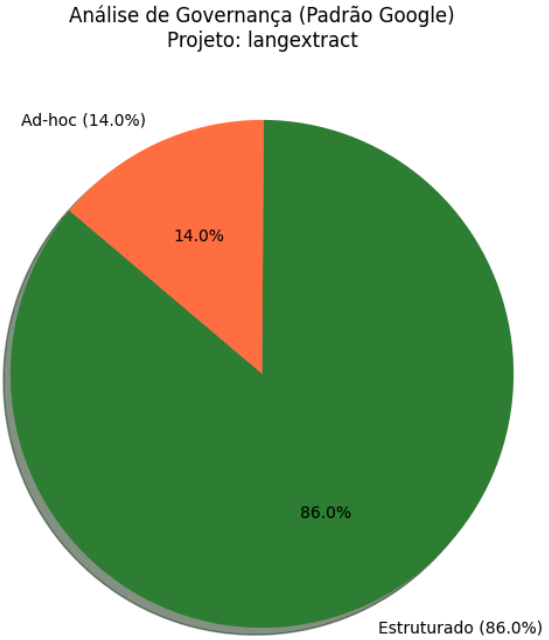
Python

```
# Lógica de Classificação aplicada a cada commit
for commit in commits_reais:
    # 1. Análise Vetorial (IA): Calcula similaridade com boas práticas
    emb_commit = model.encode(commit)
    sim_bom = util.cos_sim(emb_commit, embeddings_bom).max().item()

    # 2. Análise Sintática: Verifica Verbos de Ação (Padrão Google)
    primeira_palavra = commit.split(' ')[0]
    verbos_fortes = ["Add", "Fix", "Update", "Remove", "Refactor", "feat", "fix"]

    # Decisão Final: É padronizado se usar verbo forte OU tiver alta similaridade semântica
    if primeira_palavra in verbos_fortes or sim_bom > 0.30:
        commits_padronizados += 1
    else:
        commits_fora_padrao += 1
```

3. Resultados Obtidos

A execução do algoritmo sobre a amostra (n=100) apresentou os seguintes dados quantitativos:

- **Aderência ao Padrão de Governança: 86%** (86 commits estruturados).
- **Commits Genéricos / Ad-Hoc:** 14% (14 commits).



4. Conclusão

Com um índice de **86% de padronização**, a análise comprova que a governança do projeto *LangExtract* é **ALTA E ESTRUTURADA**.

A presença dominante de mensagens iniciando com verbos imperativos (*Add, Update, Fix*) ou prefixos semânticos (*feat:, chore:*) demonstra que a equipe segue rigorosamente ritos de engenharia de software, o que viabiliza a automação de releases identificada pelas outras frentes de análise do grupo.