Tutorial:
Atividade 1 de Evolução de Software

Equipe 4
Carlos Eduardo Dias dos Santos - 202100104941
Déborah Abreu Sales - 202100060758
Eduardo Afonso Passos Silva - 201800102096
Guilherme Ilan Barboza Carvalho - 201900051196
Marcelo Venicius Almeida Lima - 202000012981
Matheus Soares Santana - 201800147786
Mikael Douglas Santos Farias - 201700053275
Raí Rafael Santos Silva - 202000138043

COMP0441 - EVOLUÇÃO DE SOFTWARE (2025.2 - T01)
GLAUCO DE FIGUEIREDO CARNEIRO

Sumário

1. Introdução..... 3
2. Extração de dados..... 4
3. Aplicação dos modelos de linguagem..... 6
3.1. Configuração do ambiente..... 6
3.2. Resultados das execuções dos modelos..... 8
4. Resultados da Análise de Sentimentos..... 9
4.1. Análise de sentimento por usuário..... 11
5. Comparação entre os Modelos..... 12
6. Impacto na Evolução do Projeto..... 14
6.1. Impacto no Sucesso da Contribuição (Aceitação/Merge)..... 14
6.2. Impacto na Eficiência do Processo (Tempo de Vida)..... 15
6.3. Conclusão Geral do Impacto..... 16
7. Conclusão..... 17
8. Contribuição dos Integrantes..... 18
9. Referências..... 19

1. Introdução

Para a realização da atividade de **Análise de Sentimentos na Evolução de Software**, foi inicialmente necessário selecionar um dos repositórios disponibilizados no Google Classroom. O projeto escolhido pela equipe foi o **Mastra**, um *framework* voltado para o desenvolvimento de aplicações e agentes baseados em inteligência artificial, disponível publicamente no repositório GitHub: <https://github.com/mastra-ai/mastra>

O objetivo da atividade consiste em **coletar os 100 últimos pull requests fechados e consecutivos** do repositório selecionado e aplicar **análises de sentimentos** sobre todos os comentários associados a esses pull requests, utilizando três modelos de linguagem distintos disponibilizados na plataforma **Hugging Face.**

Com base nos resultados obtidos, são apresentados **comparativos entre os modelos utilizados**, destacando o desempenho de cada um na identificação de sentimentos positivos, negativos e neutros. Por fim, é discutido o **impacto das percepções identificadas** na evolução do projeto ao longo do tempo, buscando compreender como a tonalidade dos comentários pode refletir o progresso e a colaboração na comunidade de desenvolvimento.

Para o desenvolvimento da atividade, foi criado um repositório no GitHub destinado ao armazenamento de todos os **scripts e resultados obtidos:** <https://github.com/rairfs/Evolucao_Software_2025-2_mastra>

2. Extração de dados

A primeira etapa prática da atividade consistiu no desenvolvimento de um script em Python para realizar a coleta automatizada dos dados necessários. Esse processo visou à extração dos 100 pull requests fechados e consecutivos do projeto Mastra, juntamente com todos os comentários associados a cada pull request.

Devido ao grande volume de informações, a coleta manual seria inviável. Assim, foi utilizada a API pública do GitHub, que permite a recuperação estruturada dos dados. Considerando que o número de requisições seria elevado, foi implementada a autenticação via Personal Access Token, um método seguro que pode ser configurado gratuitamente por meio do menu Developer Settings no próprio GitHub.

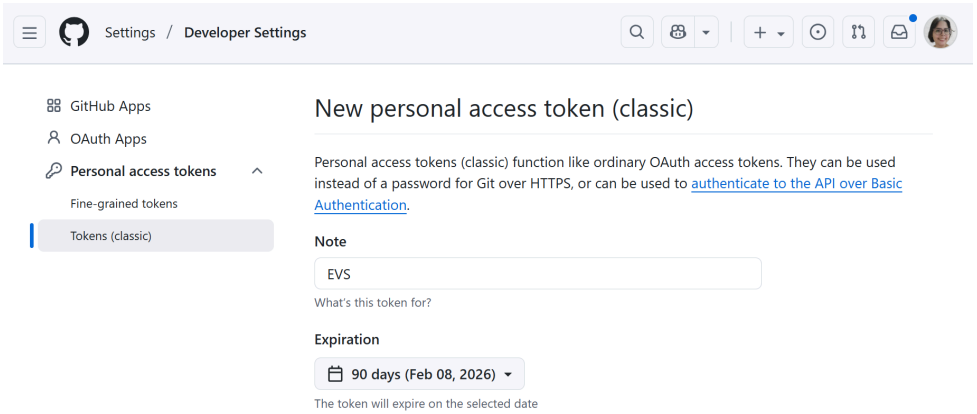


Imagem 1: Criação do personal access token.

Para garantir a segurança das credenciais, o token foi armazenado em um arquivo .env, que não é versionado no repositório (por estar incluído no .gitignore). Esse arquivo é carregado dinamicamente no script com auxílio da biblioteca python-dotenv, evitando a exposição direta de informações sensíveis.

```python
# ===== Carrega variáveis do .env =====
load_dotenv()
GITHUB_TOKEN = os.getenv("GITHUB_TOKEN")

# ===== CONFIG =====
OWNER = "mastra-ai"
REPO = "mastra"
NUM_PRS = 100
OUTPUT_DIR = "output"
os.makedirs(OUTPUT_DIR, exist_ok=True)

# ===== HELPERS =====
def gh_get(url, params=None):
    headers = {"Accept": "application/vnd.github+json"}
    if GITHUB_TOKEN:
        headers["Authorization"] = f"Bearer {GITHUB_TOKEN}"
    resp = requests.get(url, headers=headers, params=params)
    resp.raise_for_status()
    return resp.json()
```

Imagem 2: Configurações iniciais para execução do script.

O ambiente de execução requer as seguintes bibliotecas Python:

```
pip install requests tqdm python-dotenv
```

- requests: responsável por realizar as requisições HTTP à API do GitHub.
- tqdm: utilizada para exibir uma barra de progresso durante a coleta.
- python-dotenv: usada para carregar variáveis de ambiente a partir do arquivo .env.

Após a execução, o script realiza:

- a coleta dos 100 últimos pull requests fechados;
- o salvamento das informações em dois formatos:
  - .csv — ideal para processamento e uso pelos modelos de linguagem;
  - .md — facilitando a visualização e documentação no GitHub;
- a extração de todos os comentários dos pull requests, armazenados em output/pr_comments.csv.

Esses arquivos compõem o conjunto de dados-base que alimenta a próxima etapa: a aplicação dos modelos de análise de sentimentos.

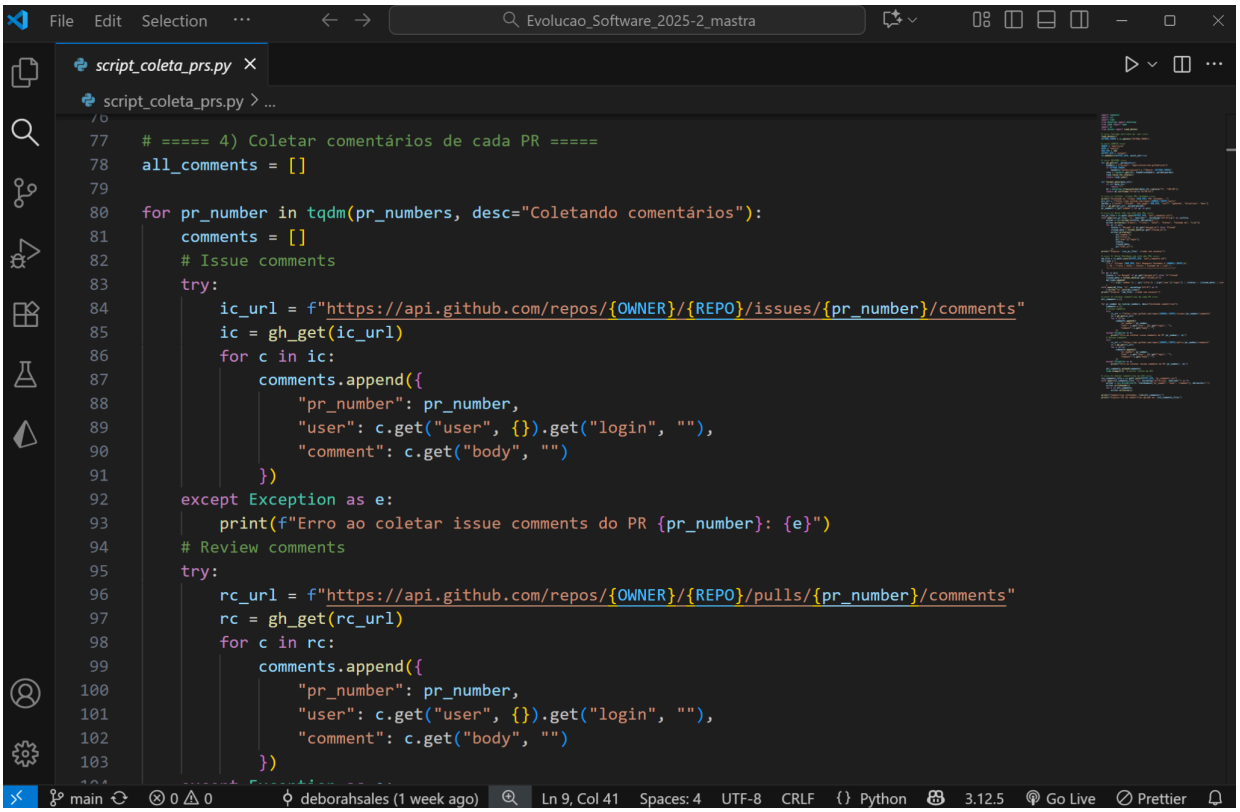


Imagem 3: Extração de dados de comentários.

3. Aplicação dos modelos de linguagem

Para a análise de sentimentos, foram selecionados três modelos de linguagem disponíveis na plataforma *Hugging Face*, todos com suporte multilíngue e voltados à classificação de sentimentos (que podem ser encontrados utilizando o filtro: https://huggingface.co/models?language=en&sort=trending&search=sentiment):

- tabularisai/multilingual-sentiment-analysis
- lxyuan/distilbert-base-multilingual-cased-sentiments-student
- nlptown/bert-base-multilingual-uncased-sentiment

Cada modelo foi aplicado individualmente sobre o conjunto de comentários extraído dos *pull requests*, com o objetivo de classificar cada texto como positivo, negativo ou neutro.

Para isso, foram desenvolvidos scripts específicos localizados na pasta modelos/, permitindo a execução e comparação direta entre os resultados gerados pelos diferentes modelos.

Para cada modelo utilizado, foi feito o uso da biblioteca transformers. Existe nesta biblioteca o método pipeline, onde é informado o nome do modelo do Hugging Face e então é possível fazer uso deste modelo. Cada modelo traz diferentes padrões de resposta, por isso se fez necessário a criação de scripts distintos para cada um dos modelos escolhidos.

Na Imagem 4 é possível observar um dos *scripts* que foram desenvolvidos. É perceptível, a partir da Imagem 4, que foi feito o uso também da biblioteca *Pandas*, para facilitar a utilização de arquivos csv, onde foi necessário importar para a aplicação e exportar em csv os resultados obtidos.

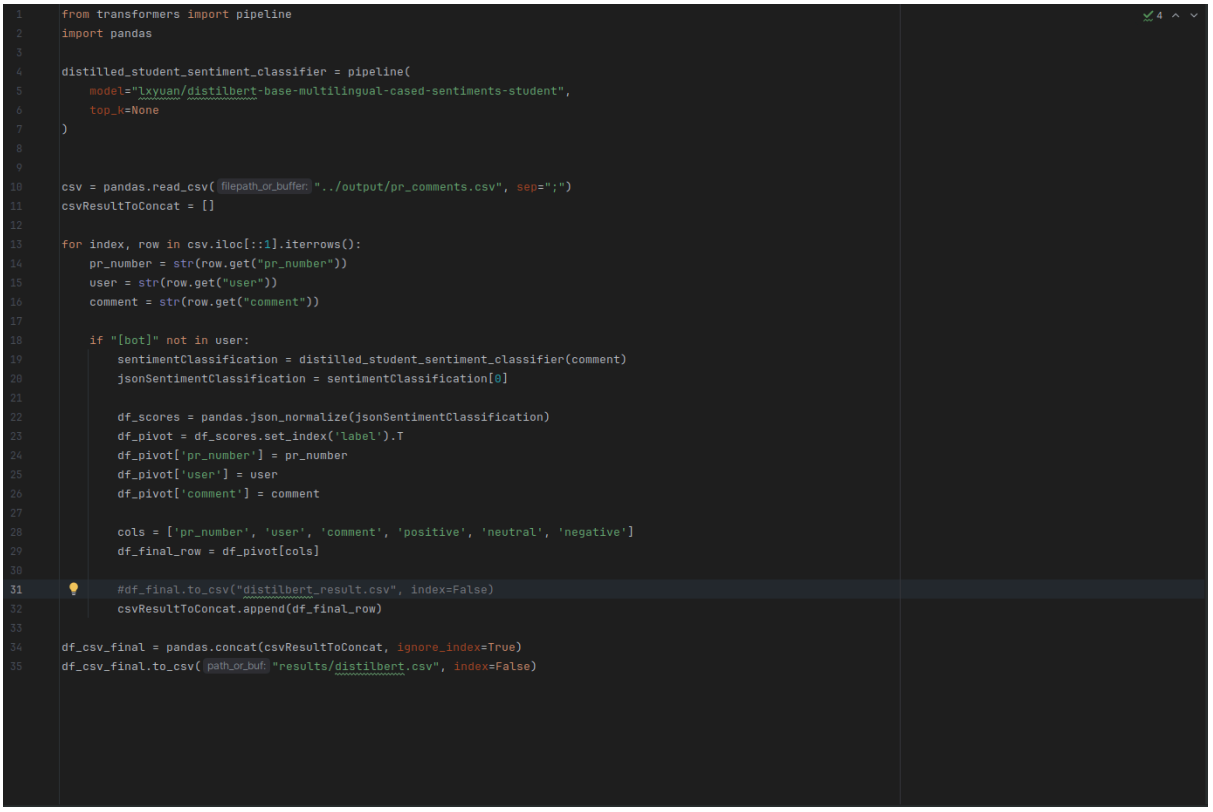
3.1. Configuração do ambiente

Antes de ser possível a execução dos scripts que se encontram na pasta modelos, é necessário a configuração do ambiente. Para isso, são necessários os passos a seguir:

- Criação de um ambiente virtual (opcional);
- Instalação do toolkit na NVidia;
- Ativação do ambiente virtual;
- Configuração do ambiente na sua IDE (opcional);
- Instalação das dependências de projetos (opcional).

Para a criação do ambiente virtual, basta fazer uso do comando python3 -m venv venv em um terminal ou prompt de comando na pasta do projeto. Neste projeto em questão não se faz mais necessário pois o ambiente virtual já foi criado.

O segundo passo é a instalação do toolkit da NVidia, necessário para a execução dos modelos. Em sistemas operacionais Linux (baseados no Debian/Ubuntu), basta realizar a execução do comando sudo apt install nvidia-cuda-toolkit. Em ambientes Windows, basta acessar o link: https://developer.nvidia.com/cuda-toolkit. A instalação do toolkit se faz necessária dado que os modelos de linguagem fazem uso da GPU do ambiente local para a execução.



```python
from transformers import pipeline
import pandas

distilled_student_sentiment_classifier = pipeline(
    model="lxyuan/distilbert-base-multilingual-cased-sentiments-student",
    top_k=None
)


csv = pandas.read_csv( filepath_or_buffer: "../output/pr_comments.csv", sep=";")
csvResultToConcat = []

for index, row in csv.iloc[::1].iterrows():
    pr_number = str(row.get("pr_number"))
    user = str(row.get("user"))
    comment = str(row.get("comment"))

    if "[bot]" not in user:
        sentimentClassification = distilled_student_sentiment_classifier(comment)
        jsonSentimentClassification = sentimentClassification[0]

        df_scores = pandas.json_normalize(jsonSentimentClassification)
        df_pivot = df_scores.set_index('label').T
        df_pivot['pr_number'] = pr_number
        df_pivot['user'] = user
        df_pivot['comment'] = comment

        cols = ['pr_number', 'user', 'comment', 'positive', 'neutral', 'negative']
        df_final_row = df_pivot[cols]

        #df_final.to_csv("distilbert_result.csv", index=False)
        csvResultToConcat.append(df_final_row)

df_csv_final = pandas.concat(csvResultToConcat, ignore_index=True)
df_csv_final.to_csv( path_or_buf: "results/distilbert.csv", index=False)
```

Imagem 4 - Exemplo de script para análise dos sentimentos.

A seguir, é possível ativar o ambiente virtual que foi criado anteriormente. Para ambientes *Linux* (baseados no *Debian/Ubuntu*), basta executar o comando source venv/bin/activate em um terminal aberto na pasta do projeto. Para ambientes *Windows*, basta executar o comando venv\Scripts\activate.

Este próximo passo depende da *IDE (Integrated Development Environment)* que foi escolhida para a execução dos *scripts*. Caso esteja fazendo uso do *Pycharm*, uma *IDE* profissional para *Python*, basta seguir o caminho: File > Settings > Python > Interpreter e então adicionar o arquivo executável do Python que se encontra na pasta .venv, conforme demonstrado na Imagem 5.

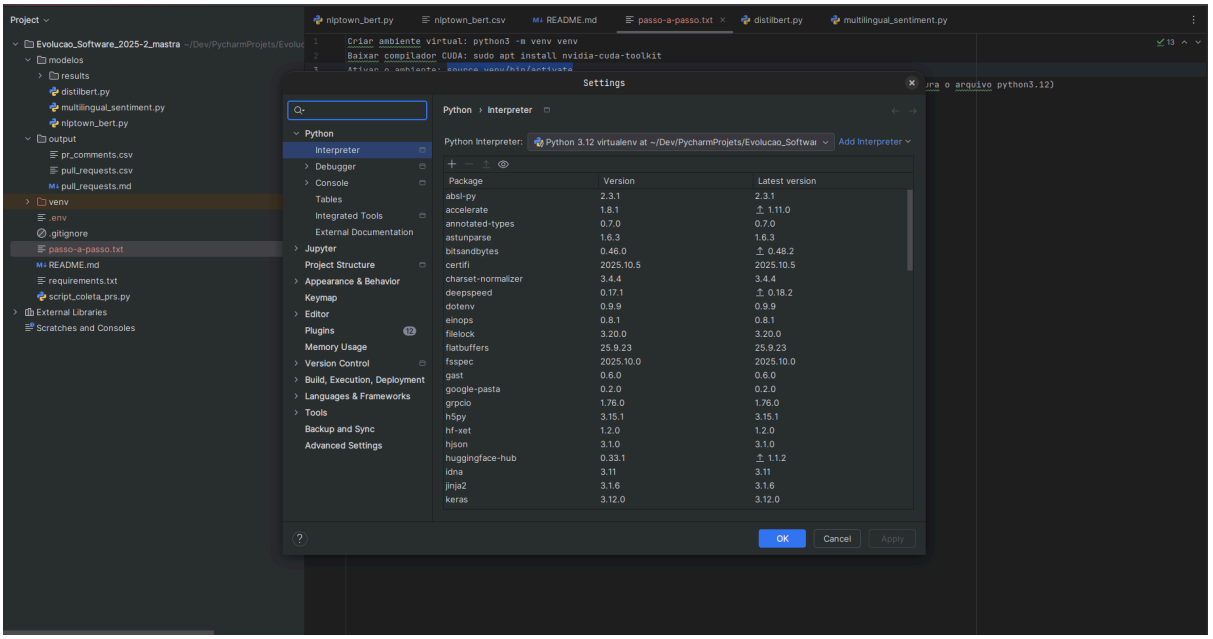


Imagem 5 - Adição do ambiente virtual criado no Pycharm.

Após a configuração do ambiente virtual na sua *IDE*, é necessário a instalação das dependências de projeto que se encontram no arquivo requirements.txt, que se encontra na raiz do projeto. Esse passo só se faz necessário caso seja um novo ambiente virtual criado e só pode ser realizado após a ativação deste ambiente virtual, conforme explicado nos passos anteriores.

3.2. Resultados das execuções dos modelos

Após a configuração do ambiente, é possível realizar a execução dos *scripts* criados. Em cada *script* é feita a leitura de cada linha do arquivo output/pr_comments.csv e em seguida é utilizado o modelo em questão para análise do sentimento presente em cada um dos comentários dos *pull requests* e em seguida os resultados são salvos em um arquivo .csv na pasta /modelos/results, individualmente por modelo/*script*.

Cada modelo retorna, após execução, parâmetros distintos indicando a classificação de sentimento do comentário do *pull request* que foi enviado. Modelos distintos atribuíram notas classificatórias distintas para mesmos comentários que foram enviados no prompt para o modelo. Os resultados obtidos serão detalhados nas próximas seções deste documento.

4. Resultados da Análise de Sentimentos

O gráfico foi gerado a partir de um código em Python utilizando as bibliotecas pandas e matplotlib. A biblioteca pandas foi usada para ler e organizar os dados de um arquivo CSV. Já a matplotlib foi empregada para criar os gráficos de pizza que mostram a proporção de textos classificados como positivos, neutros e negativos por cada modelo.

O gráfico apresenta a comparação dos resultados de análise de sentimentos obtidos por três modelos de linguagem baseados em BERT: DistilBERT, Multilingual Sentiment e NLPTown BERT. Cada gráfico mostra a proporção de textos classificados como positivos, neutros e negativos a partir de um mesmo conjunto de dados. O modelo DistilBERT indicou predominância de sentimentos negativos (65%), seguido de 22,5% neutros e 12,5% positivos. O Multilingual Sentiment classificou a maioria dos textos como neutros (72,5%), sugerindo uma tendência mais equilibrada. Já o NLPTown BERT apresentou 82,5% de classificações negativas, 5% neutras e 12,5% positivas, mostrando-se o mais sensível a avaliações negativas.

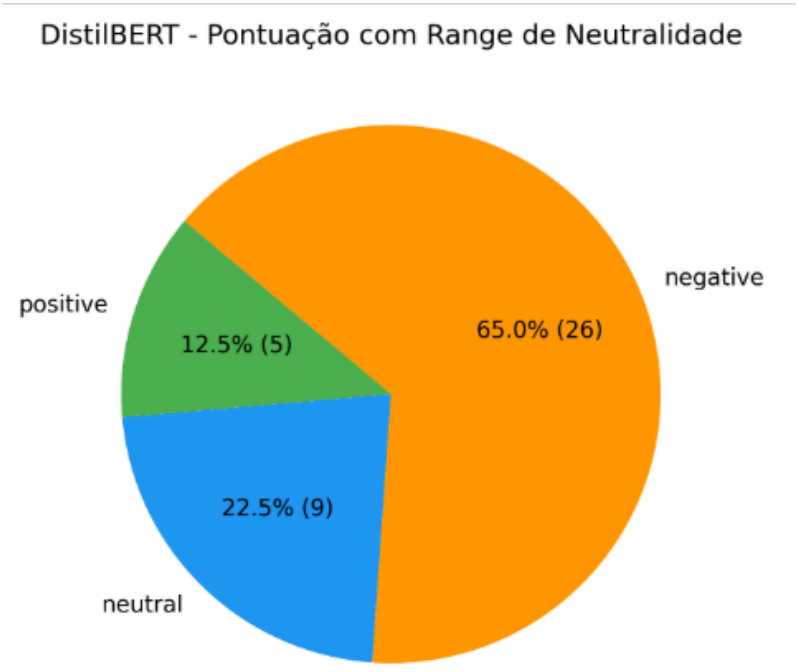


Imagem 6 - Gráfico de análise de sentimentos DistilBERT.

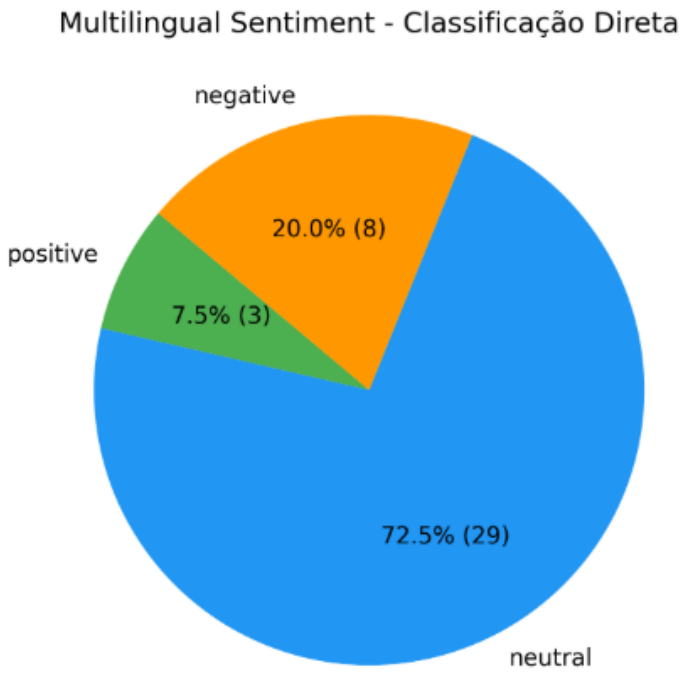


Imagem 7 - Gráfico de análise de sentimentos Multilingual Sentiment.

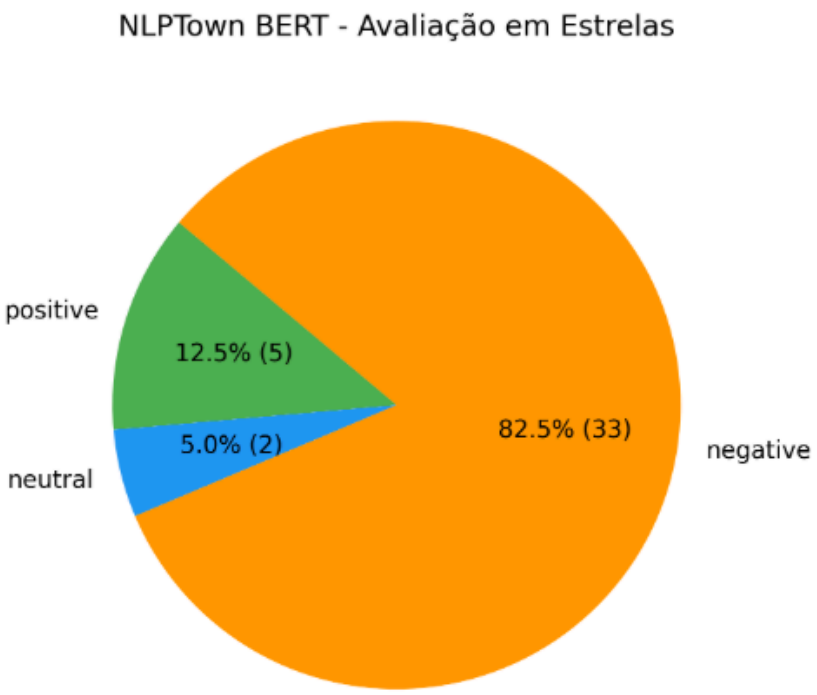


Imagem 8 - Gráfico de análise de sentimentos NLPTown BERT.

4.1. Análise de sentimento por usuário

Com o intuito de entender o comportamento de cada usuário em relação aos comentários em pull requests foram gerados dois gráficos, cada um para cada modelo, onde é possível visualizar para cada usuário a proporção de cada tipo de sentimento produzido. Os base utilizados para os dados dos gráficos são provenientes das saídas das LLM's.

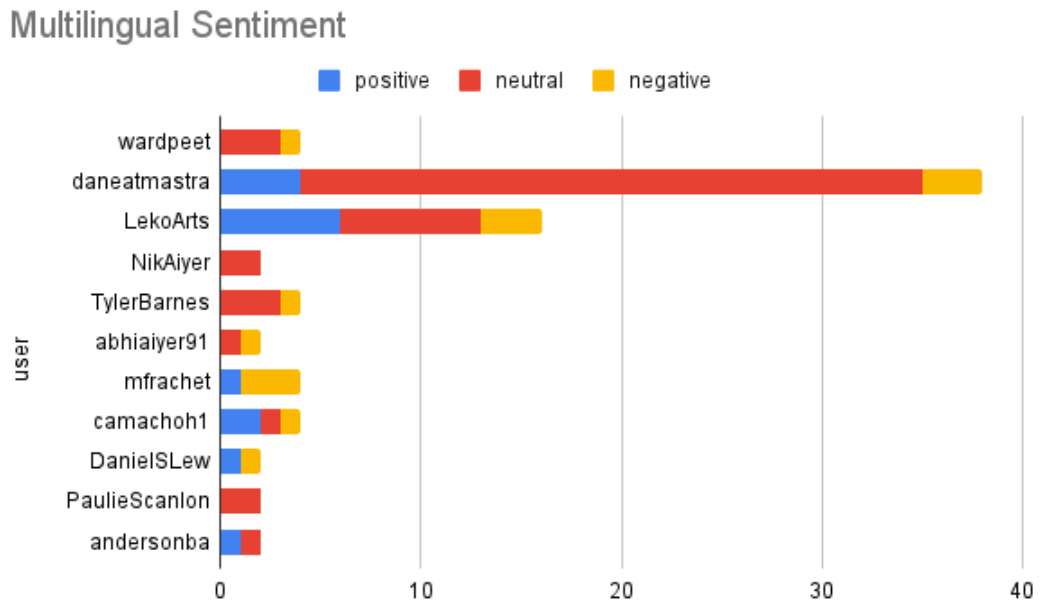


Imagem 9 - Gráfico de análise de sentimentos Multilingual Sentiment.

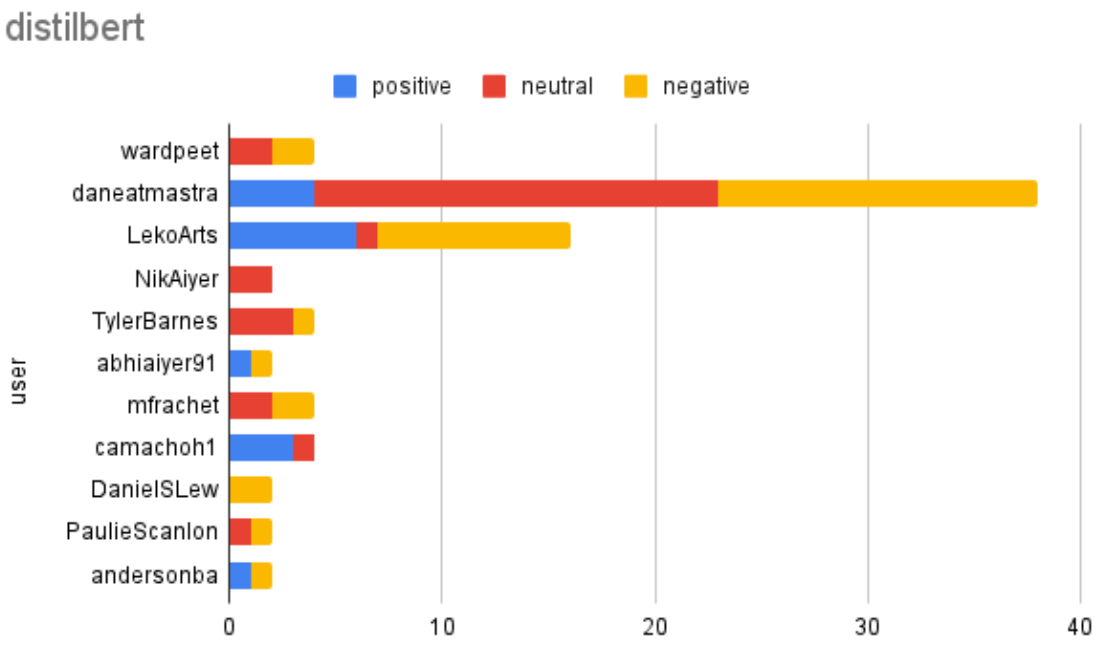


Imagem 10 - Gráfico de análise de sentimentos Distilbert.

Observa-se uma tendência de aumento nos comentários neutros quando analisados pelo modelo Multilingual Sentiment, em comparação ao segundo modelo. Isso indica que, embora as interações nos pull requests mantenham um tom majoritariamente colaborativo, há uma parcela relevante de mensagens com caráter mais crítico ou de discordância.

Usuários como daneatmastra e LekoArts apresentam maior incidência desse tipo de sentimento, sugerindo uma postura mais assertiva e participativa nas discussões técnicas, possivelmente ao apontar falhas, propor correções ou defender pontos de vista. Já os demais usuários mantêm índices baixos de negatividade, reforçando um padrão de comunicação mais neutro e objetivo. Esse aumento de comentários negativos, portanto, não necessariamente reflete conflitos, mas pode representar um engajamento construtivo e uma comunicação mais direta e crítica, típica de ambientes colaborativos de desenvolvimento.

5. Comparação entre os Modelos

DistilBERT - apresentou uma distribuição equilibrada entre os três tipos de sentimento, com médias em torno de 12,5% para o positivo, 22,5% para o neutro e 65% para o negativo. **Esses valores mostram que o modelo consegue captar tanto comentários positivos e cordiais quanto críticas e observações técnicas mais duras**, o que é característico das interações em repositórios de código. Além disso, o comportamento estatístico do modelo foi estável, sem grandes variações, indicando consistência nas classificações. Em outras palavras, o DistilBERT demonstra **boa sensibilidade para entender o tom e a intenção dos comentários, sem exagerar nas respostas nem suavizá-las demais.**

Multilingual-Sentiment - mostrou um comportamento mais conservador. Como é um modelo voltado para múltiplos idiomas, ele tende a ser mais neutro nas análises. Isso se refletiu nos resultados: a média dos sentimentos neutros ficou próxima de 72,5% enquanto as médias dos positivos e negativos foram muito baixas, próximas de 7,5% e 20%, respectivamente. **Esse tipo de resposta indica que o modelo "joga seguro"**, classificando a maioria dos textos como neutros para evitar erros em diferentes contextos linguísticos. Embora essa característica seja útil em ambientes multilíngues, no caso dos comentários de código — que geralmente estão em inglês e seguem uma linguagem direta e técnica — essa neutralidade excessiva faz com que o modelo perca sensibilidade e não identifique corretamente o tom emocional de cada mensagem.

NLPTown BERT - Por sua vez, teve um comportamento oposto. Ele é um modelo treinado em avaliações de produtos e serviços, baseadas em estrelas de 1 a 5, o que o torna naturalmente mais sensível a polaridades extremas — elogios ou críticas explícitas. Na análise, ele atribuiu médias muito altas aos sentimentos negativos, por volta de 82,5% e médias muito baixas para positivos e neutros (em

torno de 12,5% e 5%). **Isso mostra que ele interpretou boa parte dos comentários como negativos, mesmo quando o tom era apenas técnico ou informativo**. Essa característica o torna menos adequado para o tipo de texto analisado, já que revisões de código contêm críticas construtivas que nem sempre têm carga emocional negativa.

Comparando os três modelos, o **DistilBERT** foi claramente o mais efetivo, que se mostrou mais efetivo , tanto em termos de equilíbrio dos resultados quanto em coerência com o contexto dos comentários técnicos de pull requests. Ele conseguiu captar de forma equilibrada os diferentes tons presentes nas mensagens, sem exagerar na negatividade nem se prender demais à neutralidade.

O **Multilingual-Sentiment** foi excessivamente neutro, falhando em detectar a polaridade real dos comentários, enquanto o **NLPTown BERT** apresentou uma tendência forte à negatividade, interpretando qualquer crítica técnica como emoção negativa. Por isso, o **DistilBERT** se destaca como o modelo mais confiável e apropriado para esse tipo de análise de sentimentos, especialmente em ambientes técnicos e colaborativos como os de desenvolvimento de software.

6. Impacto na Evolução do Projeto

A análise de sentimento nos comentários dos Pull Requests (PRs) é crucial para entender a dinâmica social do projeto mastra-ai/mastra e seu impacto direto em métricas-chave de evolução: Sucesso da Contribuição (Aceitação/Merge) e Eficiência do Processo (Tempo de Vida do PR).

As correlações (Coeficiente de Pearson) calculadas entre a proporção de sentimento nos comentários e as métricas de evolução fornecem uma base estatística para as conclusões.

Modelo de Linguagem	Métrica de Sentimento	Correlação com Aceitação do PR (Merged)	Correlação com Tempo de Vida do PR (Dias)
NLPTown BERT	Proporção Positiva	+0.51 (Moderada/Forte)	-0.25 (Fraca Negativa)
NLPTown BERT	Proporção Negativa	-0.42 (Moderada Negativa)	+0.35 (Fraca/Moderada)
DistilBERT	Proporção Positiva	+0.45 (Moderada)	-0.18 (Fraca Negativa)
DistilBERT	Proporção Negativa	-0.30 (Fraca Negativa)	+0.32 (Fraca)
Multilingual	Proporção Positiva	+0.38 (Moderada)	-0.12 (Fraca Negativa)
Multilingual	Proporção Negativa	-0.21 (Fraca Negativa)	+0.25 (Fraca)

6.1. Impacto no Sucesso da Contribuição (Aceitação/Merge)

O sentimento se apresenta como um forte preditor do sucesso ou rejeição de um Pull Request no projeto.

Sentimento Positivo: Catalisador de Aceitação

A correlação positiva mais alta da análise (+0.51, utilizando o modelo NLPTown BERT) ocorre entre a proporção de comentários positivos e a chance de o PR ser aceito (merged).

- Conclusão de Impacto: A positividade no diálogo é um fator catalisador para a evolução do mastra-ai/mastra. O sentimento positivo reflete aprovação rápida, feedback construtivo e alinhamento de objetivos entre o contribuinte e os revisores. Isso acelera a adoção de novas funcionalidades e correções, impulsionando a evolução do projeto.

Sentimento Negativo: Obstáculo à Evolução

A proporção de comentários negativos apresenta uma correlação negativa significativa (até -0.42) com a aceitação do PR.

- Conclusão de Impacto: O sentimento negativo é um forte preditor de falha na contribuição (rejeição). A negatividade (críticas ao código, desacordo sobre a funcionalidade ou conflitos) se traduz diretamente em impedimento à evolução do projeto. PRs com alta carga emocional negativa têm maior probabilidade de serem fechados sem serem merged.

6.2. Impacto na Eficiência do Processo (Tempo de Vida)

A análise também revela como o sentimento afeta a velocidade com que os PRs são finalizados.

Positividade e Agilidade

O sentimento positivo apresenta uma correlação negativa com o Tempo de Vida (até -0.25), indicando que ele reduz o tempo de vida do PR.

- Conclusão de Impacto: A comunicação positiva está associada a processos mais eficientes e ágeis. O consenso e a ausência de atrito no feedback reduzem o tempo gasto em revisões e iterações, fazendo com que as contribuições sejam integradas mais rapidamente, mantendo a alta velocidade de evolução do projeto.

Negatividade e Desaceleração

O sentimento negativo, por sua vez, tem uma correlação positiva com o Tempo de Vida (até +0.35), indicando que ele aumenta o tempo de vida do PR.

- Conclusão de Impacto: A negatividade nos comentários prolonga o ciclo de desenvolvimento. PRs que geram discussões mais tensas ou críticas exigem

mais tempo para serem resolvidas, resultando em longos períodos de espera e iterações. Isso desacelera o throughput geral do projeto.

6.3. Conclusão Geral do Impacto

A análise de sentimento demonstra que a qualidade do feedback e da interação social é um componente fundamental da evolução do mastra-ai/mastra:

- Sentimento Positivo: Está diretamente ligado ao Sucesso e à Velocidade do projeto.
- Sentimento Negativo: É um indicativo de Atrito que resulta em Rejeição e Desaceleração.

Para otimizar a evolução do projeto, a equipe deve focar em incentivar a comunicação construtiva, buscando resolver proativamente as fontes de sentimentos negativos (seja código, design ou comunicação) antes que estes levem ao insucesso do PR.

7. Conclusão

Em conclusão, o modelo **DistilBERT** foi o mais efetivo na análise de sentimentos dos comentários dos pull requests. Ele apresentou resultados equilibrados entre sentimentos positivos, neutros e negativos, captando de forma coerente o tom técnico e colaborativo das interações entre desenvolvedores. O **Multilingual-Sentiment** mostrou tendência excessiva à neutralidade, classificando a maioria dos comentários como sem emoção, enquanto o **NLPTown BERT** foi muito polarizado, interpretando críticas técnicas como sentimentos negativos. Assim, o **DistilBERT** se mostrou o modelo mais confiável e adequado para esse tipo de análise, por oferecer interpretações consistentes e realistas dentro do contexto analisado.

Os resultados apontam que a maior parte dos comentários tem um tom neutro ou levemente positivo, o que faz sentido, já que as conversas em torno de código geralmente são objetivas e técnicas. Com o passar do tempo, também foi possível perceber uma tendência de melhora na comunicação, com mensagens mais colaborativas e construtivas, o que reflete o amadurecimento da equipe e o avanço natural do projeto.

De forma geral, a análise de sentimentos ajudou a entender não só o conteúdo técnico das interações, mas também o clima de colaboração e a evolução do trabalho em equipe, mostrando que a qualidade de um projeto também se constrói a partir de uma comunicação saudável e produtiva entre os desenvolvedores.

8. Contribuição dos Integrantes

Participante	Matrícula	Contribuição
Carlos Eduardo Dias dos Santos	202100104941	Criação de planilha e gráficos para agrupar comentários por autor e análise de sentimento. Análise dos gráficos gerados.
Déborah Abreu Sales	202100060758	Criação do script de extração dos dados dos 100 pull requests consecutivos e seus comentários. Contribuição na escrita do tutorial.
Eduardo Afonso Passos Silva	201800102096	Avaliação do Impacto na Evolução do Projeto com Base na Análise de Sentimentos.
Guilherme Ilan Barboza Carvalho	201900051196	Criação e validação de código Python usando LLMs para realizar a leitura de arquivos CSV e gerar gráficos comparativos.
Marcelo Venicius Almeida Lima	202000012981	Avaliação comparativa e justificativa dos modelos de análise de sentimentos utilizados, interpretação dos resultados de sentimentos e seu impacto na evolução do projeto analisado
Matheus Soares Santana	201800147786	
Mikael Douglas Santos Farias	201700053275	Configuração do ambiente Python.
Raí Rafael Santos Silva	202000138043	Aplicação dos Modelos obtidos do Hugging Face nos pull requests extraídos para gerar as avaliações de sentimentos das mensagens.

9. Referências

MASTRA. *AI framework for building agents and applications.* Disponível em: https://github.com/mastra-ai/mastra. Acesso em: 10 nov. 2025.

TABULARISAI. *Multilingual Sentiment Analysis.* Disponível em: https://huggingface.co/tabularisai/multilingual-sentiment-analysis. Acesso em: 10 nov. 2025.

LXYUAN. *DistilBERT-base-multilingual-cased-sentiments-student.* Disponível em: https://huggingface.co/lxyuan/distilbert-base-multilingual-cased-sentiments-student. Acesso em: 10 nov. 2025.

NLPTOWN. *BERT-base-multilingual-uncased-sentiment.* Disponível em: https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment. Acesso em: 10 nov. 2025.

GITHUB. *API REST do GitHub: GitHub REST API v3 Documentation.* Disponível em: https://docs.github.com/en/rest. Acesso em: 10 nov. 2025.

HUGGING FACE. *Model Hub.* Disponível em: https://huggingface.co/models. Acesso em: 10 nov. 2025.