**REDE NACIONAL DE ENSINO E PESQUISA - RNP**

**GERÊNCIA DE ATENDIMENTO INTEGRADO - GAI**

**Pedro Henrique Pena Neves**

**AudioSentimentAnalyzer**

**BRASÍLIA**

**2024**

**DESCRIÇÃO DO PROJETO**

Este projeto analisa sentimentos em áudios de entrada, determinando se são positivos, neutros ou negativos. Utiliza um script que processa o áudio, extrai características relevantes e aplica técnicas de processamento de linguagem natural (NLP) para a classificação de sentimentos. O objetivo é fornecer insights emocionais sobre o conteúdo do áudio. A análise é feita de maneira eficiente e precisa, facilitando a compreensão do tom emocional dos áudios. Ideal para aplicações em análise de feedback, atendimento ao cliente e estudos de comportamento.

**INSTALAÇÃO**

Setup do projeto:

* Ferramentas
  + VScode
* Tecnologias
  + Python 3.11.2
* Bibliotecas Usadas
  + Scikit Learn version 1.4.2
  + NumPy 1.26.4
  + SpeechRecognition 3.10.3
  + pydub 0.25.1

* Clone o projeto **git clone** [**https://github.com/pedropenaa/IA-do-Alem.git**](https://github.com/pedropenaa/IA-do-Alem.git)
* Navegue até o diretório do projeto
  + cd IA-do-Alem
* Crie um ambiente virtual **(opcional, mas recomendado**)
  + python -m venv venv
  + source venv/bin/activate, no Windows use `venv\Scripts\activate`
* Instale as dependências
  + pip install -r requirements.txt

**ESTRUTURA DO PROJETO**

* transcricao\_audio.py
  + Contém a parte do código que receberá a entrada principal do projeto, o áudio para podermos analisar.
* analise\_sentimento.py
  + Neste arquivo iremos realizar todo o treinamento do áudio transcrito para que o modelo de **machine learning** possa classificar a nossa entrada.
* main.py
  + Onde iremos importar todas as classes criadas nos outros arquivos, e rodar como um arquivo principal do projeto.
* CR7.mp3
  + O arquivo de áudio servirá como entrada do projeto.

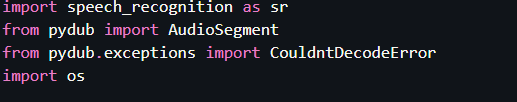
**EXPLICANDO O CÓDIGO**

* **Explicação do Código: Transcrição de Áudio**

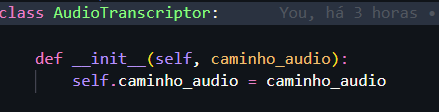
Este código é responsável por transcrever o conteúdo de um arquivo de áudio em texto. Ele utiliza as bibliotecas speech\_recognition para reconhecimento de fala e pydub para manipulação de áudio. O processo inclui verificação do arquivo de áudio, conversão para o formato necessário e transcrição usando o serviço Google Speech Recognition.

* Estrutura do código

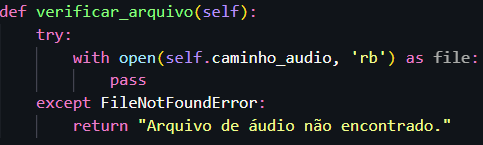
1. Importação das bibliotecas
   1. speech\_recognition: usada para reconhecimento de fala.
   2. pydub: usada para manipulação de áudio.
   3. os: usada para operações do sistema de arquivos.

****

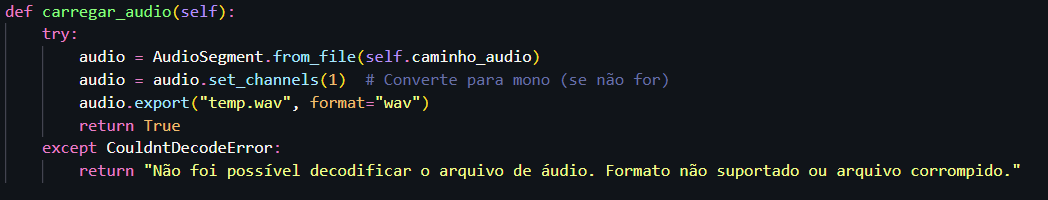
1. Criação da Classe e inicialização
   1. Esta classe encapsula a funcionalidade de transcrição de áudio.
   2. Inicializa a classe com o caminho do arquivo do áudio.

****

1. Verificação do arquivo
   1. Verifica se o arquivo de áudio existe no caminho especificado.

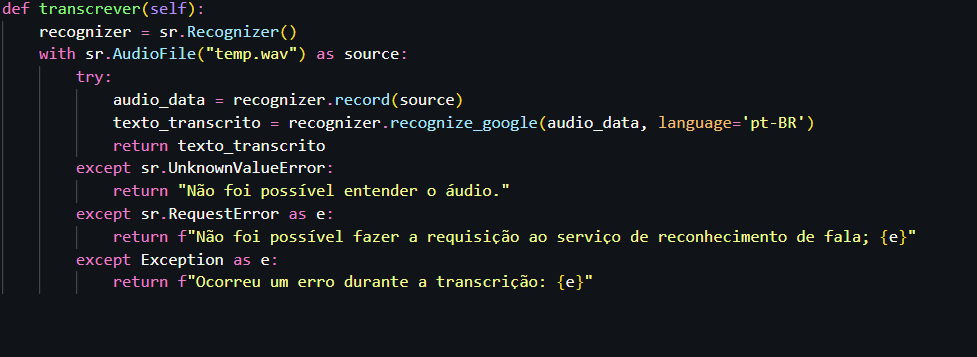


1. Carregamento e Conversão do Áudio
   1. Carrega o arquivo de áudio, converte para mono se necessário e exporta para o formato WAV.



]

1. Transcrição do Áudio
   1. Usa o Google Speech Recognition para transcrever o áudio e retorna o texto transcrito.

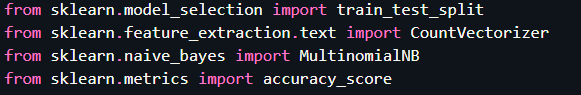


* **Fluxo de Trabalho**
  + Inicialização: Cria-se um objeto AudioTranscriptor com o caminho do arquivo de áudio.
  + Verificação: Verifica se o arquivo de áudio existe.
  + Carregamento: Carrega o arquivo de áudio, converte para mono e salva como temp.wav.
  + Transcrição: Transcreve o áudio do arquivo temp.wav usando o Google Speech Recognition.
  + Limpeza: Remove o arquivo temporário temp.wav.
* **Explicação do Código: Análise de Sentimentos**

Este código implementa uma classe SentimentoAnalyzer que utiliza técnicas de aprendizado de máquina para classificar textos como positivos ou negativos. Ele utiliza bibliotecas do scikit-learn para dividir dados, extrair características textuais, treinar um modelo de Naive Bayes e avaliar sua precisão.

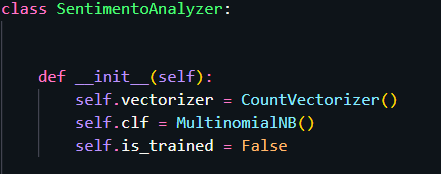
* Estrutura do código

1. Importação das bibliotecas
2. train\_test\_split: utilizado para dividir os dados em conjuntos de treino e teste.
3. CountVectorizer: converte texto em uma matriz de contagem de tokens.
4. MultinomialNB: implementação do algoritmo Naive Bayes para classificação multiclasse.
5. accuracy\_score: métrica para avaliar a precisão do modelo.



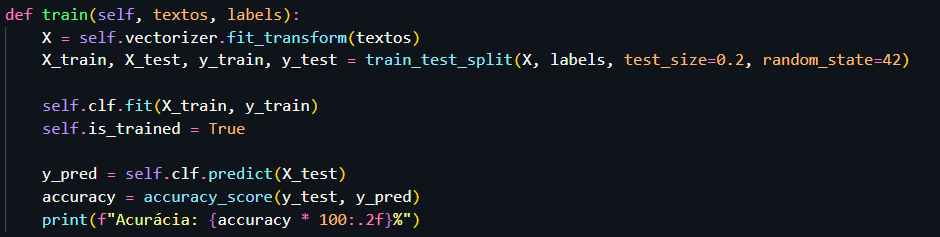
2. Criação da Classe e inicialização

1. Esta classe encapsula a funcionalidade de treinamento e predição de sentimentos.
2. Inicializa a classe com um CountVectorizer para transformar textos em vetores de contagem e um classificador MultinomialNB. A variável is\_trained é usada para verificar se o modelo foi treinado.



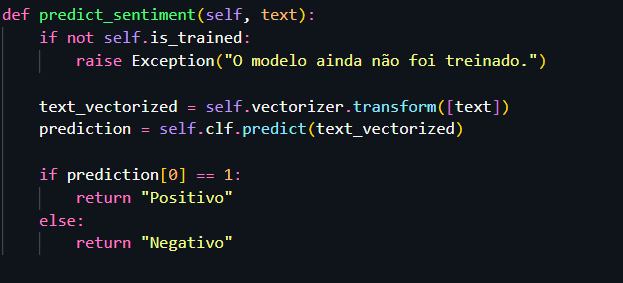
3. Treinamento do Modelo

1. Transformação dos Textos: Converte os textos em vetores de contagem usando CountVectorizer.
2. Divisão dos Dados: Divide os dados em conjuntos de treino e teste.
3. Treinamento: Treina o classificador Naive Bayes com os dados de treino.
4. Avaliação: Avalia a precisão do modelo no conjunto de teste e imprime a acurácia.



4. Predição de Sentimentos

1. Verificação de Treinamento: Verifica se o modelo já foi treinado.
2. Transformação do Texto: Converte o texto de entrada em um vetor de contagem.
3. Predição: Prediz o sentimento do texto e retorna "Positivo" ou "Negativo".



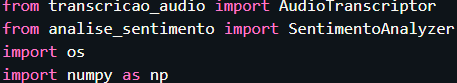
* **Fluxo de Trabalho**
  + Inicialização: Cria-se um objeto SentimentoAnalyzer, que inicializa o CountVectorizer e o classificador MultinomialNB.
  + Treinamento:
    - Os textos são transformados em vetores de contagem.
    - Os dados são divididos em conjuntos de treino e teste.
    - O modelo é treinado com os dados de treino.
    - A precisão do modelo é avaliada e exibida.
  + Predição:
    - O texto de entrada é transformado em um vetor de contagem.
    - O sentimento é predito usando o classificador treinado.
    - O resultado da predição é retornado como "Positivo" ou "Negativo".

* **Explicação do Código: Arquivo Principal**

Este código integra duas funcionalidades principais: transcrição de áudio e análise de sentimentos. Primeiro, ele converte um arquivo de áudio em texto usando a classe AudioTranscriptor. Em seguida, ele utiliza a classe SentimentoAnalyzer para analisar o sentimento do texto transcrito, classificando-o como positivo ou negativo.

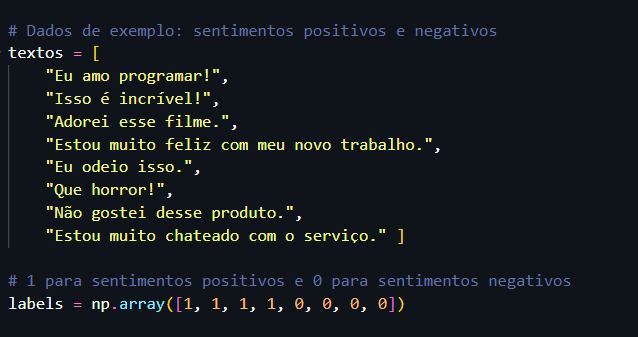
* Estrutura do código

1. Importação de Bibliotecas e Módulos
2. AudioTranscriptor: classe para transcrição de áudio.
3. SentimentoAnalyzer: classe para análise de sentimentos.
4. os: utilizado para operações do sistema de arquivos.
5. numpy: utilizado para manipulação de arrays.

****

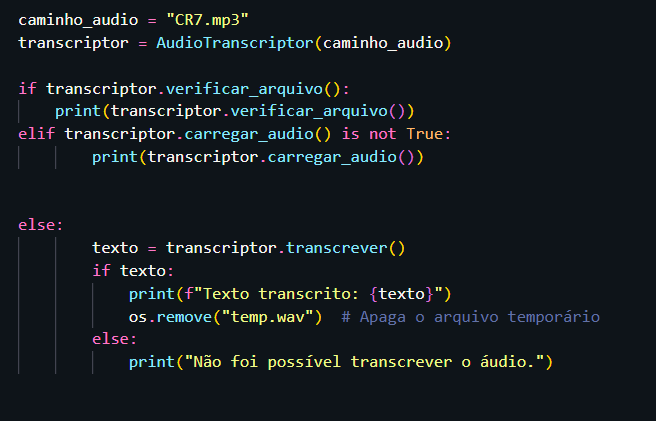
2. Definição de Dados de Exemplo

1. textos: lista de textos de exemplo com sentimentos positivos e negativos.
2. labels: array de rótulos correspondentes, onde 1 indica sentimento positivo e 0 indica sentimento negativo.



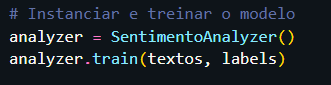
3. Transcrição de Áudio

1. caminho\_audio: caminho para o arquivo de áudio.
2. Cria uma instância de AudioTranscriptor com o caminho do arquivo de áudio.
3. Verifica se o arquivo de áudio existe.
4. Carrega e converte o arquivo de áudio, se necessário.
5. Transcreve o áudio para texto e imprime o resultado. Remove o arquivo temporário temp.wav após a transcrição.



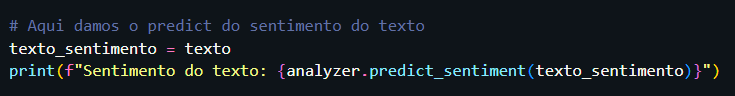
4. Treinamento do Modelo de Análise de Sentimento

1. Cria uma instância de SentimentoAnalyzer.
2. Treina o modelo com os textos de exemplo e seus rótulos correspondentes.



5. Predição do Sentimento do texto Transcrito.

1. Usa o texto transcrito como entrada para a análise de sentimentos.
2. Prediz e imprime o sentimento do texto (positivo ou negativo).

* **Fluxo de Trabalho**
  + Transcrição de Áudio:
    - Inicialização: Define o caminho do arquivo de áudio e cria um objeto AudioTranscriptor.
  + Verificação: Verifica se o arquivo de áudio existe.
  + Carregamento e Conversão: Carrega o arquivo de áudio e o converte para o formato necessário.
  + Transcrição: Transcreve o áudio para texto e remove o arquivo temporário.
  + Análise de Sentimentos:
    - Definição de Dados: Define exemplos de textos e seus rótulos de sentimentos.
    - Treinamento do Modelo: Treina o modelo SentimentoAnalyzer com os dados de exemplo.
    - Predição de Sentimento: Usa o texto transcrito para prever o sentimento, impresso como resultado final.

**CONSIDERAÇÕES FINAIS**

O projeto de transcrição de áudio e análise de sentimentos integra duas tecnologias poderosas para extrair e interpretar informações a partir de dados de áudio. Utilizando a classe AudioTranscriptor para converter arquivos de áudio em texto e a classe SentimentoAnalyzer para classificar o sentimento do texto transcrito, este projeto oferece uma solução robusta para análise de sentimentos baseada em entrada de áudio.

**Pontos Fortes:**

* **Automação e Eficiência:**
  + A automação da transcrição de áudio e da análise de sentimentos permite um processamento rápido e eficiente de grandes volumes de dados de áudio.
* **Uso de Tecnologias Populares:**
  + A utilização do speech\_recognition e pydub para transcrição de áudio, e do scikit-learn para análise de sentimentos, garante a aplicação de métodos comprovados e amplamente aceitos na comunidade de ciência de dados.
* **Facilidade de Implementação:**
  + A estrutura modular e a simplicidade do código facilitam a integração e a adaptação para diferentes contextos e tipos de dados de áudio.

**Desafios e Limitações:**

* **Qualidade do Áudio:**
  + A precisão da transcrição depende significativamente da qualidade do áudio de entrada. Ruídos, sotaques fortes e baixa qualidade de gravação podem afetar o desempenho da transcrição.
* **Generalização do Modelo de Sentimentos:**
  + O modelo de análise de sentimentos foi treinado com um conjunto pequeno e específico de dados de exemplo. Para aplicações em cenários reais, é essencial treinar o modelo com um dataset maior e mais diverso para melhorar sua capacidade de generalização.
* **Suporte a Múltiplos Idiomas:**
  + Atualmente, a transcrição está configurada para o português (pt-BR). Adaptações adicionais são necessárias para suportar múltiplos idiomas e dialetos.

**Próximos Passos:**

* **Aprimoramento da Base de Dados:**
  + Expandir a base de dados de treinamento para incluir uma maior variedade de textos e sentimentos, melhorando assim a precisão do modelo de análise de sentimentos.
* **Melhorias na Transcrição:**
  + Implementar técnicas de pré-processamento de áudio para reduzir ruídos e melhorar a qualidade do áudio antes da transcrição.
* **Integração com Outras APIs:**
  + Considerar a integração com outras APIs de reconhecimento de fala, como IBM Watson ou Microsoft Azure Speech, para comparar e melhorar possivelmente a precisão da transcrição.
* **Interface de Usuário:**
  + Desenvolver uma interface de usuário amigável, seja web ou móvel, para facilitar o uso do sistema por pessoas sem conhecimento técnico.
* **Implementação em Tempo Real:**
  + Trabalhar na implementação de um sistema de análise de sentimentos em tempo real a partir de transmissões ao vivo ou gravações contínuas, expandindo o alcance e aplicabilidade do projeto.

Este projeto demonstra como combinar transcrição de áudio e análise de sentimentos para fornecer insights valiosos a partir de dados não estruturados. Com melhorias contínuas e expansões, ele tem o potencial de se tornar uma ferramenta essencial para diversas aplicações, incluindo atendimento ao cliente, monitoramento de redes sociais, análise de feedbacks e muito mais. O uso de tecnologias modernas e a abordagem modular garantem que o sistema possa ser facilmente adaptado e escalado para atender às necessidades específicas de diferentes usuários e indústrias.