Benchmarking de Modelos Estado da Arte: Um Estudo sobre a Robustez de Modelos de Transcrição frente aos Dialetos do Português

Tech for Humans, Anna Júlia de Souza Ferreira

***Resumo*— Este artigo apresenta análises e demonstrações de benchmarks de modelos de ASR aplicados à transcrição do português brasileiro. Além disso, discute uma pipeline de pré-processamento de áudio baseada em técnicas de deep learning, avaliando como diferentes tipos de áudio respondem a esse processamento. O estudo também investiga o comportamento de distintos modelos de speech-to-text diante do uso de gírias, regionalismos e variações dialetais da língua portuguesa.**

***Abstract*— This article presents analyses and benchmark demonstrations of ASR models applied to the transcription of Brazilian Portuguese. In addition, it discusses an audio preprocessing pipeline based on deep learning techniques, evaluating how different types of audio respond to this processing. The study also investigates the behavior of different speech-to-text models when dealing with slang, regionalisms, and dialectal variations of the Portuguese language.**

***Palavras-Chave*— ASR, LLMs, Word Error Rate (WER), Português Brasileiro, Open-Source AI, Benchmarking, Robustez Dialetal, Deep Learning.**

***Keywords*— ASR, LLMs, Word Error Rate (WER), Brazilian Portuguese, Open-Source AI, Benchmarking, Dialectal Robustness, Deep Learning**

I. Introdução

Embora a geração atual de agentes de IA seja predominantemente fundamentada em interações textuais (text-to-text), o mercado avança rapidamente rumo à consolidação de interfaces multimodais. No ecossistema Tech for Humans, observa-se que a integração de voz desponta como o próximo grande catalisador da experiência do cliente, especialmente em fluxos de alta criticidade e complexidade operacional, como a regulação de sinistros.

Entretanto, a implementação dessa tecnologia no contexto brasileiro impõe desafios sociolinguísticos e técnicos relevantes. Torna-se imprescindível o uso de sistemas de Automatic Speech Recognition (ASR) que transcendam a simples transcrição fonética e apresentem robustez diante da diversidade de dialetos regionais, da informalidade inerente à fala espontânea e das disfluências comuns à linguagem oral. Ademais, o português brasileiro caracteriza-se por uma elevada incidência de homófonos — palavras com fonética idêntica, mas grafias e significados distintos — o que exige que os modelos de ASR não apenas processem o sinal acústico, mas também incorporem compreensão contextual e semântica, garantindo precisão na identificação de entidades numéricas e terminologias contratuais críticas.

Neste contexto, o presente artigo apresenta uma análise comparativa de modelos de ASR do estado da arte, avaliando pilares fundamentais para a viabilidade operacional e o cumprimento de Service Level Agreements (SLAs) de negócio: acurácia, mensurada pela taxa de erro de palavras (Word Error Rate – WER); eficiência computacional, avaliada por meio do Real-Time Factor (RTF); latência de ponta a ponta; custos de infraestrutura; e rastreabilidade modular do sistema. Complementarmente, descreve-se a aplicação de um pipeline de pré-processamento baseado em Deep Learning para remoção de ruídos (denoising) e normalização acústica (resampling), assegurando a integridade do sinal de entrada antes da etapa de inferência e possibilitando uma auditoria granular, transparente e reproduzível de todo o fluxo de transcrição.

II. Trabalhos Relacionados e Estado da Arte

O "Estado da Arte"(SOTA) em modelos de ASR ou Speech-to-Text (STT) é definido, atualmente, por um *trade-off* crítico entre precisão e eficiência operacional. Para medir a qualidade da transcrição, utiliza-se primordialmente a métrica **WER** (*Word Error Rate*). Atualmente, o cenário é dominado por duas abordagens principais: arquiteturas baseadas em **Conformer** e, mais recentemente, sistemas integrados a **Large Language Models** (LLM).

A. Arquiteturas Conformer e Eficiência Computacional

O Encoder do tipo Conformer [1] é uma arquitetura de *deep learning* híbrida projetada especificamente para o processamento de fala. Sua inovação reside na combinação do mecanismo de *Self-Attention* dos Transformers, capazes de capturar dependências globais e o contexto de longo alcance das frases, com camadas convolucionais (CNNs), que são excepcionais na extração de padrões acústicos locais e características fonéticas em escala de milissegundos.

Para mitigar a alta demanda computacional do Conformer original, surgiram variantes como o **Efficient Conformer**[1]. Esta evolução introduz técnicas como o **Downsampling Progressivo** (ou *subsampling*), que compacta a sequência de áudio codificada ao longo das camadas. Em vez de processar cada quadro individualmente, o modelo agrupa informações em **vetores de características** densos, permitindo uma inferência significativamente mais veloz sem perda de integridade semântica.

No que tange à velocidade, o **Real Time Factor** (RTF) é a métrica decisiva: valores menores que 1 indicam que o modelo processa o áudio mais rápido do que o tempo de fala real. Modelos SOTA frequentemente utilizam decodificadores

baseados em **CTC** (*Connectionist Temporal Classification*) [3] ou a inovação da NVIDIA, o **TDT** (*Token-and-Duration Transducer*)[15], uma evolução do **RNN-T** (*Recurrent Neural Network Transducer*). Enquanto o **CTC** é amplamente adotado por sua simplicidade e velocidade em modelos como Wav2Vec2 e HuBERT, o TDT destaca-se por prever simultaneamente o token e sua duração em *frames*. Isso permite que o decodificador "pule"silêncios e quadros redundantes, otimizando drasticamente o RTF sem sacrificar a precisão.

B. Modelos de Fundação e LALMs

A convergência entre o Automatic Speech Recognition (ASR) e os Large Language Models (LLMs) consolidou a categoria dos LALMs (Large Audio-Language Models). Diferente das abordagens tradicionais em cascata — que dependem de um modelo de ASR para gerar texto e um LLM subsequente para interpretá-lo —, esses sistemas multimodais operam de forma nativa e end-to-end. Essa arquitetura permite que o modelo processe o sinal acústico diretamente em seu espaço latente, preservando nuances prosódicas e paralinguísticas que frequentemente se perdem na transcrição textual simples.

Tal integração viabiliza não apenas a transcrição de alta fidelidade, mas o raciocínio sobre o áudio (audio reasoning). Isso permite a execução de tarefas complexas em um único passo de inferência, como a análise de sentimento baseada na entonação, tradução simultânea com preservação de voz e a identificação dinâmica de falantes (diarização). Expoentes recentes dessa categoria que definem o estado da arte incluem o GPT-4o-audio, o Qwen2-Audio e o Microsoft Phi-4 Multimodal Instruct, que demonstram uma capacidade superior de compreensão contextual e robustez frente a ruídos de fundo e sobreposição de falas.

III. Metodologia

A metodologia adotada neste trabalho foi projetada para avaliar, de forma sistemática, o desempenho, a estabilidade e a viabilidade operacional de diferentes arquiteturas de ASR aplicadas ao português brasileiro. O desenho experimental priorizou reprodutibilidade, controle de variáveis e observabilidade, aspectos fundamentais em cenários de produção.

Os experimentos foram organizados a partir de uma infraestrutura híbrida (APIs gerenciadas, endpoints dedicados e execução local), combinada com um pipeline padronizado de pré-processamento de áudio, coleta de métricas e análise de resultados, permitindo comparar modelos sob condições equivalentes de carga e complexidade acústica.

A. Ambiente Experimental e Infraestrutura

A condução dos testes comparativos de ASR foi centralizada no Colab Enterprise, integrado à plataforma Vertex AI da Google Cloud. Esta infraestrutura foi selecionada por sua capacidade de orquestrar processamento em nuvem de alta performance e por oferecer ferramentas de MLOps que aceleram o ciclo de desenvolvimento e a rastreabilidade dos experimentos. A estratégia de execução foi segmentada em três pilares principais, visando garantir condições ideais de inferência para modelos proprietários e soluções Open Source:

1) Modelos Gerenciados e APIs Proprietárias: Este pilar foca na avaliação de modelos operados via endpoints escaláveis, onde a gestão da infraestrutura é abstraída pelo provedor:

- **Google Gemini**: Os modelos Gemini-2.0-Flash-Lite e Gemini-2.5-Flash-Lite foram executados nativamente via API da Google, aproveitando o processamento multimodal do modelo direto da plataforma.
- **OpenAI GPT-4o-mini-transcribe**: A integração deste modelo foi realizada através do framework LiteLLM, permitindo a padronização das chamadas de API e a coleta unificada de métricas de custo e performance.

2) Endpoints de Alta Performance:

- Whisper (Groq): As variantes Whisper Large v3 e Whisper Large v3 Turbo foram testadas através da plataforma Groq. A escolha desta infraestrutura justifica-se pela utilização de LPU (Language Processing Units), que minimizam drasticamente o fator de tempo real (RTF) e a latência de ponta a ponta.

3) Infraestrutura Provisionada e Modelos Open Source: Diferente do consumo via API, a utilização de modelos de código aberto exigiu o provisionamento de recursos dedicados para garantir o monitoramento granular de hardware e evitar falhas críticas de memória:

- **Vertex AI Model Garden**: Os modelos Gemma 3n-e4b-it e Qwen2-Audio-7b-instruct foram implantados via Model Garden, exigindo o provisionamento de instâncias no Google Compute Engine (GCE). Devido à alta demanda de VRAM da arquitetura multimodal do Qwen2-Audio (7B), a infraestrutura foi configurada com aceleradores NVIDIA H100 de 80GB para evitar o encerramento abrupto de contêineres e garantir que a inferência permanecesse dentro dos limites do Liveness Probe. A configuração técnica desse ambiente é detalhada na configuração de endpoint a seguir.

```python

import vertexai
from vertexai import model_garden

vertexai.init(project="value_project",
    location="europe-west4")

model = model_garden.OpenModel("hf-model-id")
endpoint = model.deploy(
  accept_eula=True,
  machine_type="a3-highgpu-2g",
  accelerator_type="NVIDIA_H100_80GB",
  accelerator_count=2,
  serving_container_image_uri="vllm-inference"
    ,
  endpoint_display_name="asr-benchmark-ep",
  model_display_name="asr-benchmark-model",
  use_dedicated_endpoint=True,
)
```

Listing 1
Configuração do Endpoint VertexAI

A Listagem 1 ilustra o processo de implantação de um modelo open source via Model Garden. O modelo é carregado a partir do repositório Hugging Face por meio da abstração `OpenModel`, que viabiliza a integração de arquiteturas abertas ao ecossistema gerenciado do Vertex AI, preservando compatibilidade e rastreabilidade. A versão específica do modelo é explicitamente fixada, garantindo reprodutibilidade experimental.

A etapa de *deploy* configura um endpoint dedicado para inferência, incluindo o aceite dos termos de uso do modelo. A infraestrutura computacional é definida com instâncias do tipo `a3-highgpu-2g`, equipadas com dois aceleradores **NVIDIA H100 de 80 GB de VRAM** cada, compondo um ambiente de alta capacidade voltado a workloads multimodais intensivos em memória.

O contêiner de inferência especificado (`vllm-inference`) utiliza a biblioteca *vLLM*, otimizada para modelos de grande porte, permitindo gerenciamento eficiente do *Key-Value Cache* e maior throughput durante a inferência. Essa escolha é particularmente relevante para modelos com muitos parâmetros, cuja arquitetura multimodal impõe elevada demanda de VRAM durante o processamento de áudios longos.

Por fim, a utilização de um *dedicated endpoint* (`use_dedicated_endpoint=True`) assegura isolamento de recursos e maior previsibilidade de latência, fatores essenciais para a condução de benchmarks consistentes e para a avaliação da estabilidade operacional dos modelos sob carga contínua.

- **Hospedagem em Runtime Local (Persistent Disk)**: Os modelos NVIDIA Parakeet-TDT-0.6b-v3 e Voxtral-Mini-3B-2507 foram instalados diretamente no armazenamento persistente do ambiente de execução. Esta configuração simula um ambiente de produção local, permitindo avaliar o desempenho das arquiteturas sem o overhead de orquestradores de endpoints gerenciados.

B. Dataset e Diversidade Dialetal

A composição do dataset utilizado para o benchmarking dos modelos de ASR foi projetada para avaliar a robustez sistêmica em três dimensões críticas: integridade acústica em cenários ruidosos, consistência em dados sintéticos e sensibilidade a variações dialetais. A diversificação das fontes de áudio permite validar não apenas a precisão textual (WER), mas a eficácia da pipeline de pré-processamento e normalização implementada.

1) Amostras Empíricas e Validação de Denoising: Para simular cenários reais de alta variância ambiental, como tráfego intenso e ruídos impulsivos, foram realizadas gravações controladas pela autora em ambientes não estacionários. Estas amostras são fundamentais para testar a capacidade de isolamento seletivo da pipeline de denoising baseada no DeepFilterNet [5], garantindo que as propriedades fonéticas essenciais sejam preservadas após a remoção de ruídos estocásticos. Além disso, datasets de áudio liberados para uso em pesquisas disponíveis no GitHub foram aproveitados.

2) Dados Sintéticos e Validação de Fluxo: Com o intuito de validar a integridade da pipeline completa, abrangendo a restauração acústica, reamostragem (resampling) para 16 kHz e a decodificação final, utilizou-se a plataforma ElevenLabs para a geração de áudios sintéticos de alta fidelidade. O dataset foi balanceado entre vozes femininas e masculinas com diferentes tons e cadências, permitindo aferir se o processo de interpolação sinc de banda limitada introduz artefatos que possam prejudicar a fidelidade da transcrição.

3) Amostragem Dialetal e Robustez Regional: Considerando que o português brasileiro apresenta desafios sociolinguísticos significativos e uma vasta densidade dialetal, utilizou-se a plataforma Localingual como fonte de fala espontânea. Através do mapeamento interativo da ferramenta, foram extraídas contribuições de usuários de diversas regiões do Brasil, provendo um espectro abrangente de sotaques e regionalismos. Esta diversidade é crucial para avaliar a capacidade dos modelos de compreender o contexto semântico regional, mitigando erros em homófonos e expressões locais.

4) Repositório de Resultados e Gestão de Dados: Visando garantir a reprodutibilidade dos experimentos e consolidar uma base de conhecimento para expansões futuras do corpus, todos os áudios processados e suas respectivas transcrições foram documentados em um repositório centralizado na plataforma Hugging Face. Esta estratégia de curadoria permite não apenas a centralização dos dados, mas a rastreabilidade completa de cada inferência realizada durante o benchmarking.

Para assegurar o máximo rigor estatístico e a confiabilidade dos benchmarks, cada modelo foi submetido a 50 iterações de transcrição por amostra de áudio. Este procedimento permitiu mitigar variações sazonais de latência de rede (para APIs) e oscilações de processamento de hardware (para modelos locais), garantindo que as métricas coletadas reflitam a performance real e estável de cada arquitetura.

O dataset foi estruturado para fornecer uma visão multidimensional do desempenho dos modelos, integrando:

- **Métricas Quantitativas:** Registro sistemático do *Word Error Rate* (WER), do *Real Time Factor* (RTF) e do cálculo do Desvio Padrão dos tempos de processamento. A análise dessas métricas permite avaliar não apenas a velocidade, mas a consistência operacional de cada modelo frente a diferentes cargas.
- **Análise Qualitativa de Erros**: Documentação detalhada das discrepâncias entre o áudio original e a saída dos modelos, com foco na identificação de alucinações fonéticas, erros em homófonos e falhas de pontuação.
- **Mapeamento de Sucesso**: Identificação de cenários de "transcrição perfeita", servindo como base para entender em quais condições acústicas ou dialetais cada arquitetura atinge sua performance máxima.

Esta infraestrutura de dados assegura que a auditoria do sistema possa ser realizada de forma transparente e granular, provendo insumos valiosos para o ajuste fino de modelos e para a melhoria contínua da experiência do cliente em fluxos de alta criticidade.

C. Plataforma de Experimentação: Weights & Biases

Para garantir a transparência e a acessibilidade dos resultados, utilizou-se o Weights & Biases como camada de registro e análise. A ferramenta permite que leitores interajam diretamente com os dados brutos e métricas consolidadas, transformando tabelas estáticas em visualizações dinâmicas. Além de facilitar a colaboração, o W&B atua como um "caderno de laboratório digital", preservando o histórico de experimentos e possibilitando a análise comparativa detalhada entre o desempenho de modelos atuais e benchmarks pretéritos. Isso não apenas enriquece a documentação do projeto, mas também estabelece um padrão de versionamento que facilita a auditoria de resultados e a comparação evolutiva entre as arquiteturas testadas.

IV. Pré-processamento e Aprimoramento de Áudio

Embora arquiteturas modernas de ASR possuam mecanismos nativos de atenção acústica para lidar com ruídos moderados, sua eficácia é severamente comprometida em cenários de alta variância ambiental, como tráfego intenso, música de fundo ou ruídos impulsivos. Para mitigar esse impacto, implementou-se uma pipeline de *denoising* baseada em Deep Learning, utilizando modelos de Speech Enhancement de última geração.

O diferencial desta abordagem reside no uso de isolamento seletivo, uma técnica que permite distinguir entre padrões complexos de fala humana e ruídos estocásticos não estacionários. Ao contrário de filtros estatísticos convencionais, esta pipeline foi calibrada para preservar as propriedades fonéticas essenciais, como dialetos regionais, sotaques e variações de entonação. Essa manutenção é crucial, pois garante que as nuances das frases permaneçam intactas para análises semânticas subsequentes ou para a identificação de emoções pelo LLM, caso seja necessário.

A. Denoising e Resampling Estratégico

Para a etapa de restauração acústica, optou-se pela integração do framework de código aberto DeepFilterNet [5], uma arquitetura de aprimoramento de fala (speech enhancement) de baixa complexidade, otimizada para a remoção de ruído em banda total (full-band) em tempo real. A escolha desta solução externa justifica-se pela sua eficiência comprovada em ambientes ruidosos e sua arquitetura otimizada para produção. O DeepFilterNet opera nativamente em uma taxa de amostragem de 48 kHz; por conseguinte, a pipeline de pré-processamento realiza um upsampling inicial para esta faixa, garantindo que o modelo capture e remova resíduos de ruído em todo o espectro audível antes da decodificação final.

A premissa fundamental desta ferramenta é a decomposição da fala humana em dois pilares: um componente periódico (harmônico) e um componente ruidoso (estocástico). Esta separação permite uma economia drástica de recursos computacionais, uma vez que o framework distribui o processamento de forma assimétrica entre essas tarefas.

O fluxo de dados da pipeline inicia-se com a Transformada de Fourier de Curto Termo (STFT), que projeta o sinal do domínio do tempo para o domínio da frequência. Na fase

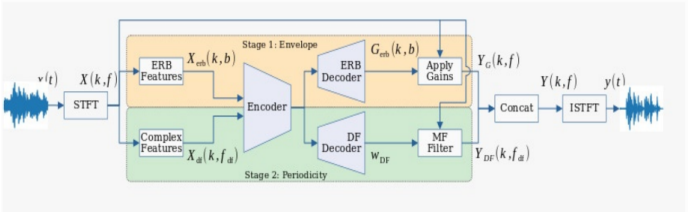

Fig. 1
Arquitetura do DeepFilterNet - DeepFilterNet.

de extração de características (features), o espectrograma é processado em duas frentes:

- **Escala ERB (Equivalent Rectangular Bandwidth)**: Utiliza uma resolução que mimetiza a audição humana para reduzir a dimensionalidade dos dados sem perder a percepção psicoacústica.
- **Características Complexas**: Preserva informações de fase e magnitude para uma filtragem de alta precisão.

Ambos os vetores alimentam um codificador (encoder) baseado em redes neurais que estima a Relação Sinal-Ruído (SNR) para direcionar dois estágios de processamento. O estágio superior prevê ganhos reais na escala ERB, que são aplicados ao sinal ruidoso para atenuar o ruído de fundo e restaurar o envelope espectral da fala (Apply Gains). Simultaneamente, o estágio inferior emprega o conceito de Deep Filtering (através do DF Decoder). Para otimizar o consumo de VRAM e CPU, esta técnica de filtragem complexa atua predominantemente em frequências baixas, onde a energia da voz humana é mais densa e a percepção de qualidade é mais crítica. O resultado é um processamento Multi-Frame, que utiliza coeficientes previstos para limpar o sinal com precisão cirúrgica. Ao final do processo, os componentes são recombinados e processados por uma Transformada Inversa de Fourier (iSTFT), gerando um áudio limpo, com latência ultra-baixa e pronto para o resampling final de 16 kHz, para garantir equilíbrio entre qualidade de áudio e eficiência dos dados.

Ademais, para assegurar a viabilidade em tempo real e a segurança de memória exigida em sistemas de produção, o framework utiliza originalmente um loop de processamento implementado na linguagem Rust, enquanto o treinamento e a pesquisa foram conduzidos em Python utilizando o ecossistema Pytorch. A Figura 2 apresenta a análise temporal comparativa de um áudio ruidoso antes e após a aplicação da ferramenta:

A eficácia da restauração é evidenciada pela análise das formas de onda. No sinal original (em azul), nota-se um elevado nível de ruído de fundo que mascara as pausas e degrada a SNR, fator crítico para a "alucinação"de fonemas em modelos de ASR. Após o processamento (em verde), observa-se uma atenuação drástica do ruído, resultando em um silêncio digital quase absoluto nos intervalos entre as falas, preservando, contudo, os envelopes de amplitude originais.

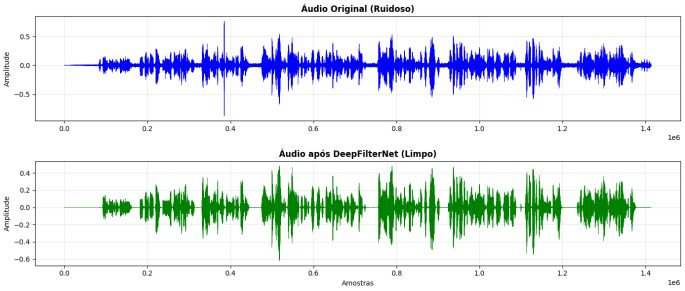

Fig. 2
DIFERENÇA DE AMPLITUDE APÓS A APLICAÇÃO DO DEEPFILTERNET

B. Padronização e Reamostragem (Resampling)

Após a etapa de restauração, a pipeline executa a normalização de frequência (resampling) [6] para 16 kHz. Este passo é vital para assegurar a compatibilidade com o encoder do modelo de ASR e reduzir o Real Time Factor (RTF). Em modelos de fala, a taxa de 16 kHz é considerada o ponto de equilíbrio ideal: frequências abaixo de 8 kHz (como em sistemas de telefonia legado) tornam fonemas fricativos e sibilantes indistinguíveis, enquanto taxas acima de 16 kHz carregam dados ultrassônicos redundantes que aumentam o custo computacional desnecessariamente.

Para esta implementação, utilizou-se a biblioteca **torchaudio** (*PyTorch*), aplicando o método de interpolação sinc de banda limitada. Este processo matemático, fundamental para evitar o fenômeno de *aliasing*, ou serrilhamento, um dos maiores fatores que evitam a fidelidade na transcrição, especialmente ao lidar com a diversidade fonética do português brasileiro. São utilizadas operações de convolução, componente central de redes neurais convolucionais (CNNs). O sistema aplica um kernel de pesos sobre a entrada de áudio para calcular os novos valores do sinal em passos de tempo arbitrários, extraindo as características espectrais essenciais de forma eficiente.

A configuração do reamostrador foi parametrizada para maximizar a fidelidade do sinal, conforme demonstrado no código abaixo:

```
1
2 resample_rate = 16000
3 resampler = T.Resample(sample_rate,
      resample_rate, dtype=waveform.dtype,
      lowpass_filter_width=6, rolloff=0.85,
      resampling_method="sinc_interp_hann")
4 resampled_waveform = resampler(waveform)
5
6 plot_sweep(resampled_waveform, resample_rate,
      title="Resampled Waveform")
7 Audio(resampled_waveform.numpy()[0], rate=
      resample_rate)
```

Listing 2
CONFIGURAÇÃO DO RESAMPLER

A escolha dos hiperparâmetros reflete os objetivos de precisão e desempenho da pesquisa:

- **Largura do Filtro Passa-Baixa**: Controla quantos pontos vizinhos o algoritmo consulta para calcular o valor de cada nova amostra de áudio.
- **Rolloff**: ele define onde o filtro começa a cortar as frequências para respeitar o Limite de Nyquist, que é sempre a metade da taxa de amostragem.
- **Hann**: é o padrão `sinc_interp_hann`. Uma função de cosseno ponderada e rápida de calcular, o que garante a rapidez do processo.

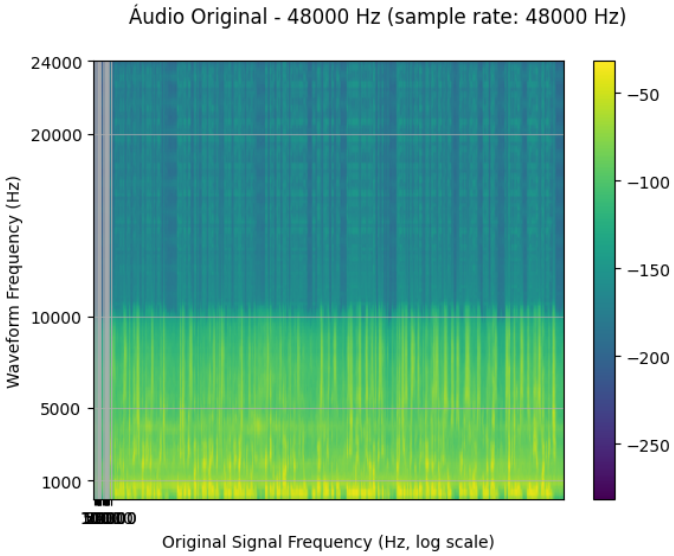

Fig. 3
GRÁFICO DO ÁUDIO ORIGINAL

A validação deste processo é observada através da análise espectrográfica. A Figura 3 ilustra o sinal capturado originalmente a 48 kHz. Embora a largura de banda teórica se estenda até 24 kHz (*Nyquist*), a maior densidade espectral da voz humana concentra-se abaixo dos 10.000 Hz. A presença reduzida de energia nas frequências superiores valida a decisão técnica de aplicar o *downsampling*.

Após a conversão, a Figura 4 demonstra o novo Limite de Nyquist estabelecido em 8 kHz. É possível notar o corte abrupto de frequências acima desta marca, protegendo o sinal contra o *aliasing*. A manutenção da estrutura morfológica do espectrograma abaixo de 8 kHz confirma que as *features* críticas para o reconhecimento de fala foram preservadas integralmente. O efeito do *rolloff* é visível na atenuação controlada na borda da banda passante, assegurando um sinal limpo para o modelo de ASR.

Por fim, todos os áudios processados são padronizados no formato WAV (PCM). Ao contrário de formatos comprimidos (*lossy*) como o MP3, que introduzem artefatos de quantização e perda de dados, o formato WAV garante uma reprodução *lossless* exata. Esta padronização elimina discrepâncias entre diferentes dispositivos de captura, garantindo que o modelo de transcrição receba insumos com integridade total, minimizando o *Word Error Rate* (WER).

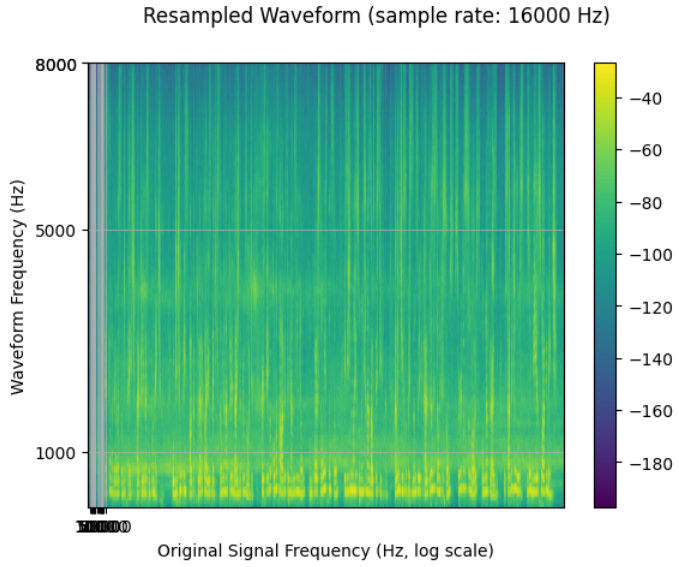

Fig. 4
GRÁFICO DO ÁUDIO APÓS A CONVERSÃO DE FREQUÊNCIA

V. ARQUITETURA DOS MODELOS DE ASR TESTADOS

A análise das arquiteturas subjacentes aos modelos de Reconhecimento Automático de Fala (ASR) é o ponto de partida para qualquer implementação estratégica. Essa análise revela as disparidades de desempenho e as lógicas operacionais que impactam diretamente o produto final. No contexto contemporâneo, especialmente frente à complexidade dos diálogos em português brasileiro — marcados por variações dialetais, gírias e sobreposições — não existe uma solução única que atinja a excelência em todos os cenários. O desenho arquitetural de um modelo atua como o determinante primário dos trade-offs entre precisão (WER), latência (RTF), custo computacional (VRAM e GFLOPS) e funcionalidades avançadas.

Atualmente, a transição de sistemas puramente acústicos para modelos de aprendizado ponta a ponta (End-to-End) e, mais recentemente, para os Large Audio-Language Models (LALMs), permite avaliar não apenas a robustez ao ruído, mas as capacidades cognitivas e multimodais dos sistemas. Esta seção investiga se a arquitetura em questão se comporta como um generalista multilíngue, que prioriza a versatilidade e a compreensão contextual global, ou como um especialista linguístico, otimizado para a extração de características fonéticas finas da língua portuguesa, definindo assim sua aplicabilidade em ambientes de produção.

A. *OpenAI Whisper(large-v3, large-v3-turbo)*

Lançado em setembro de 2022 pela OpenAI, o Whisper [7] consolidou-se como um modelo de fundação para o reconhecimento automático de fala (ASR). Seu diferencial reside no treinamento massivo com 680.000 horas de dados multilingues e multitarefas coletados da web, o que lhe confere uma robustez excepcional na transcrição de diversos idiomas e sotaques, mesmo em condições acústicas adversas.

A arquitetura do Whisper adota uma abordagem End-to-End (ponta a ponta), estruturada como um Transformer do tipo Encoder-Decoder. O processo de processamento de sinal inicia-se com a conversão do áudio bruto para o domínio da frequência:

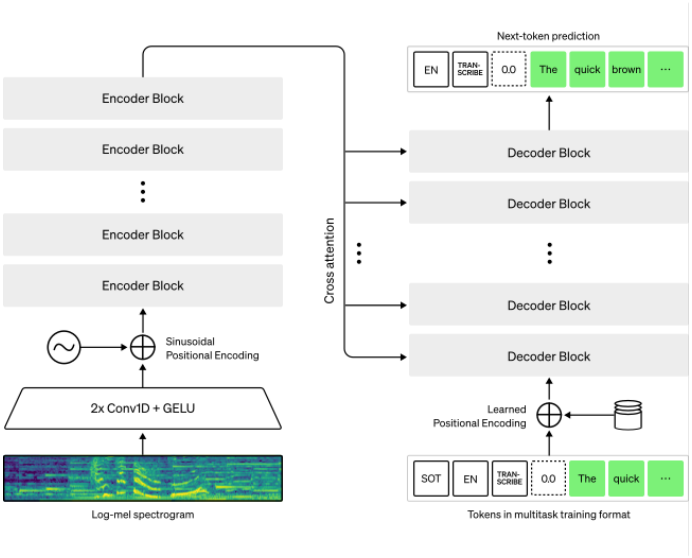

Fig. 5
ESQUEMA DE ARQUITETURA DO MODELO WHISPER - OPENAI

Neste pipeline, o sinal de áudio é amostrado a 16 kHz, normalizado e convertido em um Espectrograma de Mel de 128 canais. Esta representação mapeia as frequências de forma não linear, mimetizando a percepção auditiva humana e facilitando a extração de características fonéticas essenciais.

O funcionamento interno divide-se em duas frentes distintas:

- **Encoder (Codificador)**: Executado uma única vez para cada segmento de 30 segundos, ele transforma o espectrograma em uma representação latente de alta dimensionalidade. Por operar de forma paralela sobre o bloco de áudio, sua carga computacional é relativamente otimizada.
- **Decoder (Decodificador)**: Atua de forma autorregressiva, prevendo sequencialmente a probabilidade de cada token. O decodificador pode realizar até 224 inferências por segmento.

Essa característica do decodificador representa o principal gargalo computacional do Whisper. Em transcrições de longa duração, a natureza iterativa do decodificador, que precisa ser executado repetidamente para cada bloco de 30 segundos, consome significativamente mais recursos de processamento do que o codificador, impactando diretamente o fator de tempo real (RTF) e os custos de infraestrutura em larga escala.

Para este estudo, foram selecionadas as variantes large-v3 e large-v3-turbo. Enquanto o primeiro representa a capacidade máxima de precisão do modelo, a versão turbo utiliza um decodificador otimizado (com menos camadas), visando reduzir drasticamente a latência sem comprometer severamente o WER (Word Error Rate).

B. OpenAI GPT-4o-mini-transcribe

A arquitetura do modelo GPT-4o-mini-transcribe não foi divulgada publicamente pela OpenAI. No entanto, sabe-se que ele é uma derivação otimizada do GPT-4o, obtida por meio de um processo de *knowledge distillation* (destilação de conhecimento). Essa técnica consiste em transferir o conhecimento e as capacidades de alto desempenho de um modelo maior, denominado *professor* (GPT-4o), para um modelo menor, o *aluno* (mini), mantendo níveis elevados de acurácia e compreensão multimodal, ao mesmo tempo em que reduz significativamente a latência e o custo computacional.

Uma das principais inovações do GPT-4o-mini-transcribe está relacionada ao seu modelo de tokenização otimizado. Ao diminuir a quantidade de tokens necessários para representar um determinado texto, o modelo reduz a carga computacional associada a cada palavra processada. Para idiomas que não utilizam o alfabeto latino, essa eficiência chega a ser duplicada. No caso do português, essa otimização resulta em um processamento multilíngue mais ágil e economicamente eficiente.

Diferentemente do Whisper, que se baseia em uma arquitetura *encoder-decoder*, o GPT-4o-mini-transcribe é um modelo autorregressivo fundamentado na arquitetura *Transformer Decoder-Only* [8]. Essa abordagem permite que áudio e texto sejam processados de forma nativa dentro de um único fluxo contínuo, possibilitando o tratamento eficaz de dependências de longo alcance e a captura de nuances contextuais presentes no sinal de áudio.

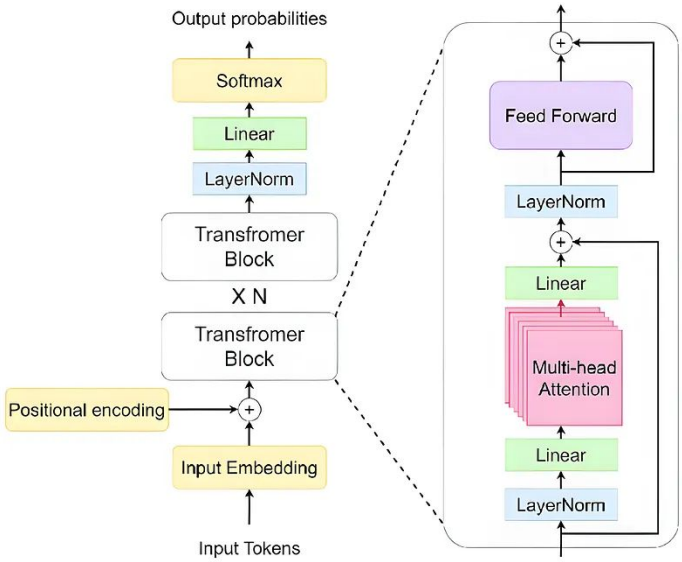

Fig. 6
ESQUEMA DA ARQUITETURA TRANSFORMER DECODER-ONLY

Nessa arquitetura, os principais componentes organizam-se da seguinte forma:

- **Mecanismo de Atenção**: Trata-se do elemento central do modelo. Diferentemente de abordagens mais antigas, que processavam palavras de maneira isolada, o mecanismo de autoatenção permite que o modelo considere toda a sequência de tokens simultaneamente, atribuindo pesos distintos a cada elemento conforme sua relevância para o contexto global.
- **Modelagem de Contexto**: O modelo constrói representações vetoriais (embeddings) baseadas em médias ponderadas, levando em conta tanto a proximidade quanto as relações semânticas entre os tokens da sequência. Esse processo é fundamental para a compreensão contextual do discurso.
- **Tratamento da Polissemia**: A atenção contextual possibilita ao modelo diferenciar múltiplos significados de uma mesma palavra, resolvendo ambiguidades semânticas de acordo com o contexto em que o termo é utilizado.

Como redes neurais não possuem, por natureza, uma noção explícita de ordem sequencial, a arquitetura Transformer incorpora informações posicionais às representações dos tokens. Esses vetores de posição informam ao modelo a localização exata de cada token na sequência, bem como suas relações temporais e estruturais, preservando a sintaxe e a coerência temporal do áudio transcrito.

O estágio de decodificação opera de forma autorregressiva, prevendo cada novo token com base em todos os tokens previamente gerados. Esse mecanismo se assemelha ao processamento humano da linguagem, no qual a continuidade da fala é construída a partir do histórico contextual. Embora estudos e fontes técnicas [8] indiquem a utilização dessa arquitetura, os detalhes internos específicos do GPT-4o-mini-transcribe permanecem proprietários e não são divulgados pela OpenAI.

C. Google Gemini-2.0-Flash-Lite

O Gemini 2.0 Flash Lite utiliza uma arquitetura Transformer baseada em Mistura de Especialistas Esparsa (Sparse Mixture of Experts - MoE) [10]. Embora herde fundamentos da família Gemini 1.5, esta versão incorpora refinamentos de design focados em eficiência extrema. O modelo é nativamente multimodal, o que significa que foi treinado para processar e raciocinar sobre diferentes modalidades em um único fluxo, eliminando a necessidade de componentes separados para traduzir cada tipo de mídia.

O modelo demonstra versatilidade ao aceitar diversas formas de entrada de maneira integrada:

- Strings de texto;
- Imagens;
- Áudio;
- Vídeo.

O grande diferencial da arquitetura MoE reside na sua capacidade de escalabilidade. Embora o modelo possua bilhões de parâmetros em sua estrutura total, ele ativa de forma seletiva apenas um subconjunto de "especialistas"necessários para resolver um token ou tarefa específica. Essa ativação esparsa reduz consideravelmente a demanda computacional durante o pré-treinamento e, crucialmente, resulta em uma baixa latência de inferência. Essa característica permite que o Gemini 2.0 Flash Lite ofereça um desempenho de ponta com um custo operacional reduzido, tornando-o ideal para aplicações que exigem alta escalabilidade em tempo real.

D. Google Gemini-2.5-Flash-Lite

O Gemini 2.5 Flash Lite, consolidado no mercado em meados de 2025, representa o estado da arte em eficiência dentro do ecossistema de modelos de baixa latência do Google. Sua arquitetura preserva a estrutura de Mistura de Especialistas Esparsa (Sparse MoE), mas introduz um diferencial qualitativo: o Modo de Pensamento (Thinking Mode) [11]. Diferente das arquiteturas puramente reativas, este recurso permite a configuração de um "Thinking Budget"(orçamento de pensamento), habilitando o modelo a executar passos de raciocínio interno e autorreflexão antes de concluir a geração da resposta.

No entanto, como a premissa da linha "Lite"é a minimização extrema de custos e latência, o recurso de raciocínio avançado apresenta as seguintes particularidades operacionais:

- **Configuração por Padrão:** O modo *thinking* permanece desativado por padrão, priorizando o tempo de resposta imediato e o menor consumo de tokens possível.
- **Trade-off de Performance:** Quando habilitado, o modelo melhora significativamente a precisão em tarefas de lógica complexa e transcrição de áudios com terminologias técnicas densas, porém à custa de um aumento proporcional no tempo para o primeiro token (TTFT).
- **Multimodalidade Nativa:** Mesmo em sua versão Lite, o modelo mantém o processamento nativo de áudio, o que, somado à arquitetura MoE, permite uma identificação de intenção mais refinada em diálogos do português brasileiro sem sobrecarregar a infraestrutura.

Esta flexibilidade arquitetural permite que o Gemini 2.5 Flash Lite atue tanto como um motor de transcrição ultrarrápido quanto como um agente de análise criteriosa, dependendo da necessidade de negócio e do orçamento computacional disponível.

E. Google Gemma-3n-e4b-it

O Gemma 3n introduz uma mudança de paradigma ao adotar a arquitetura MatFormer (Matryoshka Transformer) [12], projetada com uma filosofia mobile-first. Como ilustrado na Figura 7, essa estrutura permite a inferência elástica através de um design aninhado onde submodelos menores coexistem dentro do modelo principal. No treinamento do modelo de 4 bilhões de parâmetros efetivos (E4B), composto por 35 camadas (7 blocos) e uma rede Feed-Forward (FFN) de expansão 8x, um submodelo de 2 bilhões (E2B) é otimizado simultaneamente.

O mecanismo de elasticidade baseia-se em dois pilares fundamentais observados no diagrama:

- **Profundidade Flexível (Flexible Depth):** O modelo E2B opera com 30 camadas (6 blocos), utilizando a técnica de skip block (como o salto do bloco 5) para reduzir a latência.
- **FFN Flexível:** Enquanto o E4B, modelo utilizado, faz uso de uma FFN de 8x, o submodelo pode ser reduzido para uma configuração de 4x, permitindo que desenvolvedores ajustem o modelo via técnica *Mix-n-Match* para equilibrar precisão e velocidade.

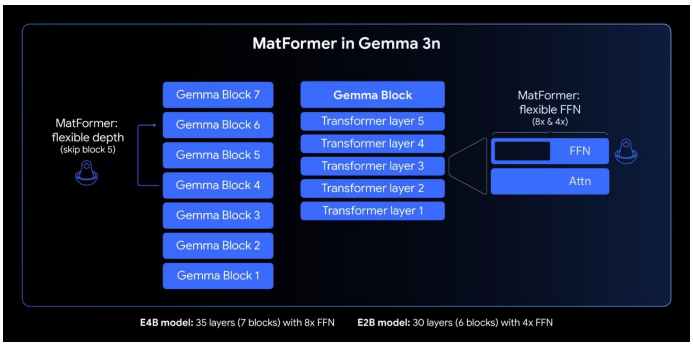

Fig. 7
ESQUEMA DE ARQUITETURA MATFORMER NO GEMMA 3N- GOOGLE FOR DEVELOPERS

A eficiência de memória é viabilizada pelos Per-Layer Embeddings (PLE), que desacoplam as representações de cada camada para processamento na CPU, mantendo apenas os pesos críticos na VRAM. Para o processamento multimodal nativo, o Gemma 3n integra o codificador USM (Universal Speech Model) para áudio, que processa sinais em fragmentos de 160ms, permitindo ASR de alta fidelidade em tempo real. Adicionalmente, a latência de sequências longas é otimizada via KV Cache Sharing, acelerando a fase de *prefill* ao compartilhar chaves e valores entre as camadas superiores da rede.

F. Qwen2-Audio-7b-instruct

O Qwen2-Audio é um LALM, com um total de 8.2 bilhões de parâmetros. A arquitetura deste modelo foi projetada para simplificar o processamento multimodal [13] (áudio e texto) e melhorar a capacidade de seguir instruções complexas.

Como ilustrado na Figura 8, o modelo é estruturado em dois módulos principais integrados:

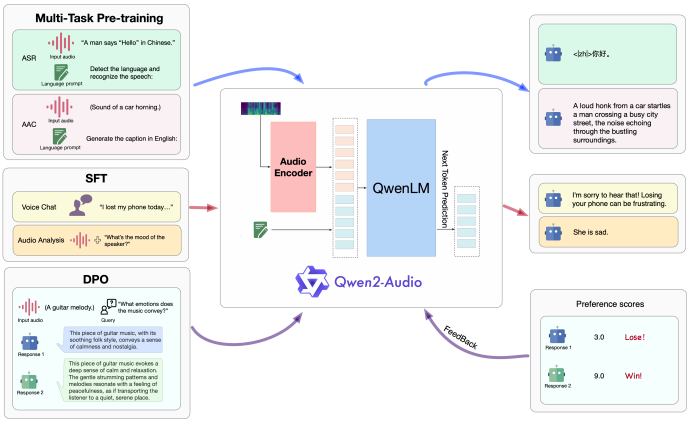

Fig. 8
ESQUEMA DE ARQUITETURA DO QWEN2 AUDIO - QWEN

- **Codificador de Áudio (Audio Encoder):** A percepção acústica é baseada no **Whisper-large-v3**. Nesta versão, o áudio é reamostrado para 16 kHz e convertido em um espectrograma de Mel de 128 canais. Para otimizar a

eficiência, utiliza-se uma camada de *pooling* com *stride* de 2, reduzindo a extensão da representação para que cada quadro de saída corresponda a aproximadamente 40 ms de áudio real.

- **Large Language Model (LLM):** O núcleo de processamento linguístico é o **Qwen2-7B**. Este módulo atua como o "cérebro"do sistema, recebendo as representações latentes do áudio e realizando a previsão do próximo token (Next Token Prediction) com base no contexto multimodal.

O diferencial competitivo deste modelo reside no seu pipeline de treinamento em três estágios, destacados no diagrama:

1) **Multi-Task Pre-training:** O modelo é exposto a tarefas de Reconhecimento Automático de Fala (**ASR**) e Legendagem Automática de Áudio (**AAC**) para aprender a mapear sons em texto.
2) **Supervised Fine-Tuning (SFT):** Refina a capacidade de conversação (Voice Chat) e análise profunda de áudio, permitindo entender o "sentimento"ou o "clima"de uma gravação.
3) **Direct Preference Optimization (DPO):** O estágio final utiliza pontuações de preferência para treinar o modelo a escolher a resposta mais útil e segura, garantindo que o *output* final seja o mais alinhado possível com a expectativa humana.

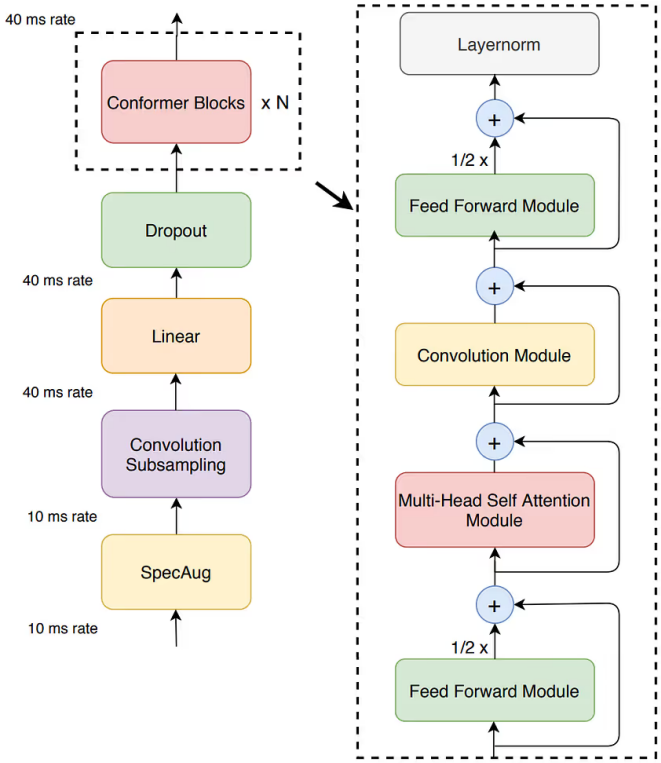

Fig. 9
ESQUEMA DE ARQUITETURA DO PARAKEET - ASSEMBLYAI

G. *Nvidia/Parakeet-tdt-0.6b-v3*

A família de modelos Parakeet, desenvolvida pela NVIDIA, representa uma solução híbrida de ponta, projetada para maximizar a velocidade de inferência (throughput) sem sacrificar a precisão da transcrição. Sua arquitetura integra dois componentes inovadores: o FastConformer como codificador e o Token-and-Duration Transducer (TDT) [4] como decodificador.

1) O Codificador: FastConformer: O FastConformer é uma evolução otimizada do modelo Conformer padrão, alcançando uma velocidade de processamento entre 2,4 vezes e 2,8 vezes superior. Como observado no diagrama, o processamento inicia-se com o SpecAug a uma taxa de 10 ms, seguido por modificações estruturais críticas:

- **Subamostragem Agressiva**: O modelo aplica uma subamostragem convolucional logo no estágio inicial, reduzindo a taxa de amostragem de 10 ms para 40 ms. Essa compressão de 8x (ou 4x dependendo da configuração) diminui drasticamente o comprimento da sequência, aliviando a carga computacional das camadas subsequentes.
- **Convoluções Separáveis e Kernel Reduzido**: O uso de convoluções separáveis em profundidade (depthwise separable) reduz a contagem de parâmetros. Além disso, a redução do kernel convolucional de 31 para 9 acelera os cálculos mantendo a fidelidade acústica.
- **Estrutura de Bloco Macaron**: O detalhamento interno do bloco Conformer revela uma estrutura "Macaron", composta por dois módulos Feed Forward que envolvem os componentes de Multi-Head Self-Attention e o módulo de convolução, garantindo estabilidade no treinamento.

2) O Decodificador: TDT (Token-and-Duration Transducer): Considerado o diferencial competitivo da arquitetura, o TDT rompe com a lógica tradicional de processamento quadro a quadro dos modelos RNN-T.

- **Previsão Dupla e Salto de Quadros**: O decodificador prevê simultaneamente o token e sua duração temporal (número de quadros). Ao determinar que um som se estende por múltiplos quadros, o modelo executa um "salto", ignorando quadros redundantes ou silêncios.
- **Ganho de Eficiência**: Essa abordagem elimina o desperdício computacional típico de prever símbolos "em branco", tornando o Parakeet-TDT até 64% mais rápido que seus predecessores, sendo ideal para processar áudios de longa duração (até 24 minutos) com baixa latência.

H. *MistralAI/Voxtral-Mini-3B-2507*

O Voxtral-Mini é um modelo multimodal de aproximadamente 4,7 bilhões de parâmetros, projetado especificamente para processar áudios de longa duração com baixo consumo de memória. Sua arquitetura é composta por três blocos sequenciais: um codificador de áudio, uma camada adaptadora e um decodificador de linguagem baseado no Ministral 3B [14].

Como observado na Figura 10, o processamento de áudio no Voxtral-Mini-3B-2507 inicia-se com a conversão do sinal bruto em um espectrograma log-Mel, que é posteriormente refinado por camadas de autoatenção bidirecionais. Uma das principais inovações desta arquitetura reside na técnica de

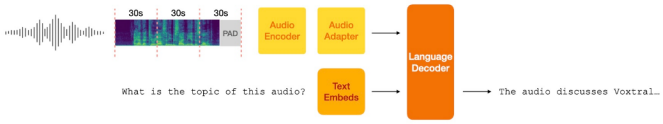

Fig. 10
ESQUEMA DE ARQUITETURA DO MISTRAL - MISTRAL VOXTRAL PAPER [14]

janelamento dinâmico: para contornar o campo receptivo fixo de 30 segundos, uma limitação comum em modelos derivados do Whisper, o Voxtral segmenta áudios de longa duração em blocos (chunks) independentes. Ao concatenar os embeddings resultantes na saída, o sistema viabiliza o processamento de arquivos que excedem drasticamente as janelas temporais convencionais, garantindo a continuidade da análise acústica em gravações extensas.

A eficiência operacional em sequências de longa duração é garantida pela Camada Adaptadora (Audio Adapter), que funciona como uma ponte tecnológica entre os sinais de áudio e o decodificador de linguagem. Utilizando um Multi-Layer Perceptron (MLP) [14], este módulo realiza uma redução de amostragem (downsampling) de 4x, comprimindo a taxa original de 50 Hz para 12,5 Hz. Esse processo de compressão temporal é vital para evitar o transbordamento da memória de trabalho, permitindo que o modelo gerencie até 40 minutos de conteúdo sonoro dentro de sua janela de contexto limitada a 32.000 tokens.

No núcleo cognitivo do sistema encontra-se o decodificador Ministral 3B, um modelo de linguagem multimodal otimizado especificamente para ambientes de borda (edge computing) e hardware com recursos limitados. Este decodificador integra de forma fluida os embeddings de áudio e texto, permitindo que o modelo execute um raciocínio multimodal profundo. Essa integração possibilita que o sistema não apenas transcreva, mas responda a perguntas complexas e contextuais sobre o conteúdo sonoro, elevando o Voxtral ao patamar de um assistente de análise de áudio inteligente, robusto e altamente eficiente para aplicações locais.

VI. VIABILIDADE ECONÔMICA E OPERACIONAL

A presente pesquisa não se limita a avaliar o desempenho dos modelos na transcrição de áudios em língua portuguesa, desafio que impõe o tratamento de variações dialetais e nuances prosódicas intrínsecas à vasta dimensão territorial brasileira. Um dos pilares centrais deste estudo reside na análise da viabilidade econômica e operacional. Para modelos proprietários, foi investigada a métrica de custo por token/minuto em provedores de nuvem (Cloud Providers). Paralelamente, para modelos Open Source, a análise foca no custo de infraestrutura local ou instâncias dedicadas, correlacionando a contagem de parâmetros e requisitos de hardware (GPU/CPU e VRAM) com a escalabilidade da hospedagem.

A. Modelos Open Source

A viabilidade financeira de modelos Open Source está atrelada ao tempo de alocação de hardware dedicado. Nesse contexto, o custo operacional é definido pelo uso de aceleradores gráficos (GPU) e memória de vídeo (VRAM). Para este benchmark, utilizou-se a infraestrutura do Vertex AI para testar modelos de Reconhecimento Automático de Fala (ASR) em duas modalidades: *endpoints* de predição e *runtimes* locais.

B. Model Garden: Gemma-3n-e4b-it e Qwen2-Audio-7b-instruct

Os modelos Qwen2 Audio e Gemma 3n foram implementados através do ecossistema Model Garden no Vertex AI, utilizando infraestrutura provisionada de última geração. Devido a restrições de disponibilidade de recursos na região geográfica selecionada, optou-se por configurações de instâncias distintas para cada modelo, garantindo a continuidade dos testes sem comprometer a performance dos aceleradores.

- **Qwen2-Audio-7b-instruct:** Foi servido em uma instância `a3-highgpu-2g`, equipada com dois aceleradores NVIDIA H100 de 80GB (representando 1/4 dos recursos de um nó A3 completo).
- **Gemma-3n-e4b-it:** Foi servido em uma instância `a3-highgpu-1g`, utilizando um acelerador NVIDIA H100 de 80GB (equivalente a 1/8 dos recursos do nó).

O custo operacional desta infraestrutura é calculado de forma granular, integrando o valor do acelerador, a taxa de gerenciamento da plataforma e o armazenamento persistente [17]. A Tabela I detalha os valores unitários, enquanto a Tabela II consolida o investimento horário e mensal para as instâncias específicas utilizadas neste benchmark.

TABELA I
COMPONENTES DE CUSTO UNITÁRIO PARA INFRAESTRUTURA NVIDIA H100 NO VERTEX AI.

Componente de Infraestrutura	Custo Unitário (USD/Hora)
Acelerador NVIDIA H100 80GB	9,7966
Taxa de Gerenciamento Vertex AI	1,4695
Instância Base (vCPU/RAM/Rede)*	1,3598
Disco PD-SSD (por GiB)	0,0003

*Valor residual para composição do custo total da instância.

A tarifação segue uma lógica linear baseada no número de GPUs alocadas. Como demonstrado na Tabela II, a instância a3-highgpu-1g estabelece o custo base de US$ 12,6259/h, enquanto a a3-highgpu-2g totaliza US$ 25,2518/h.

TABELA II
DETALHAMENTO PROPORCIONAL DE CUSTOS PARA INSTÂNCIAS A3 (H100).

Configuração	Custo Horário	Custo Mensal
a3-highgpu-8g (Base)	US$ 101,0073	US$ 73.735,3670
a3-highgpu-2g	US$ 25,2518	US$ 18.433,8417
a3-highgpu-1g	US$ 12,6259	US$ 9.216,9209

Ressalta-se que a escolha de uma GPU de alto desempenho para os benchmarks visou extrair o potencial máximo de cada arquitetura. Contudo, todos os modelos de código aberto selecionados são versáteis e compatíveis com hardwares que possuam menor poder de processamento.

C. Runtime Local: Nvidia/Parakeet-tdt-0.6b-v3 e MistralAI/Voxtral-Mini-3B-2507

Para os modelos Parakeet-TDT-0.6b e Voxtral-Mini-3B, executados em ambiente de *runtime* local com armazenamento SSD persistente, a estrutura de custos segue o padrão de instâncias gerenciadas do Vertex AI. Nesta configuração, embora não haja a implantação de um *endpoint* de predição, o faturamento incorpora a taxa de gerenciamento de instância da plataforma, mantendo o custo operacional idêntico ao processamento em larga escala.

O custo efetivo para transcrever o sinal de áudio nesta modalidade é determinado pela Equação 1, que pondera o preço da instância somado ao armazenamento persistente pelo fator de eficiência do modelo (RTF):

$$C_{\text{audio\_hora}} = (P_{\text{hora\_instância}} + C_{\text{disco}}) \times RTF \quad (1)$$

Onde:

- $P_{\text{hora\_instância}}$: Representa o custo total da VM `a3-highgpu-1g`, integrando o acelerador NVIDIA H100, a taxa de gerenciamento do Vertex AI e os recursos de computação base (US$ 12,6259/h).
- C_{disco}: Refere-se à tarifação do armazenamento persistente PD-SSD (US$ 0,0002678/GiB-h), com faturamento contínuo enquanto o recurso estiver provisionado.
- RTF: Fator de Tempo Real (*Real-Time Factor*), que define a eficiência da inferência local (ex: 0,00813 para Parakeet-TDT).

A Tabela III consolida os valores horários baseados na alocação linear de aceleradores para os modelos testados.

TABELA III
CUSTO ESCALONADO DE INSTÂNCIAS POR UNIDADE DE ACELERADOR (NVIDIA H100).

Modelo	**Instância (Config.)**	**VRAM Total**	**USD / Hora**
Gemma-3n	`a3-highgpu-1g`	80 GB	12,6259
Qwen2-Audio	`a3-highgpu-2g`	160 GB	25,2518

Nota: O custo escala linearmente com base no número de GPUs alocadas.

A tarifação da infraestrutura de alto desempenho seguiu os valores nominais da família de instâncias A3 do Google Cloud. O custo unitário por acelerador NVIDIA H100 foi estabelecido em US$ 12,6259190/h. Consequentemente, para o modelo Gemma-3n (alocado em uma instância `a3-highgpu-1g`), o custo horário da VM reflete o valor unitário, enquanto para o Qwen2-Audio (alocado em uma instância `a3-highgpu-2g`), o custo foi calculado de forma linear, totalizando US$ 25,2518380/h.

1) Custos de Oferecer os Modelos no Gateway LiteLLM: Para assegurar a viabilidade comercial e técnica do projeto em larga escala, é imperativo transitar da infraestrutura de alto desempenho (H100) para aceleradores otimizados para inferência de baixo custo. Esta otimização permite que o gateway LiteLLM opere com alta densidade de requisições, minimizando o custo por hora de áudio transcrito sem comprometer a estabilidade operacional.

Modelos de arquitetura compacta, como o Parakeet-TDT e o Voxtral-Mini, apresentam baixa pegada de memória, sendo candidatos ideais para a GPU NVIDIA Tesla T4 (16GB GDDR6). Conforme observado nos testes de perfilamento, o Parakeet consome entre 2 GB e 4 GB de VRAM, enquanto o Voxtral-Mini demanda de 7 GB a 9 GB. Nestas condições, a T4 opera com margem de segurança adequada para o processamento de sinais acústicos complexos.

Por outro lado, os modelos Gemma 3n e Qwen2-Audio (7B) apresentam requisitos de VRAM situados no limiar da capacidade da arquitetura Turing (T4), exigindo entre 15 GB e 16 GB apenas para a carga dos pesos em precisão nativa. A operação desses modelos em uma T4 resultaria em saturação da memória física e interrupções frequentes por erros de Out of Memory (OOM).

Para mitigar este gargalo, propõe-se a utilização da NVIDIA Tesla L4 (24GB GDDR6). A arquitetura Ada Lovelace da L4 não apenas provê o buffer de memória necessário para inferências estáveis de modelos de 7B parâmetros, mas também oferece suporte superior para precisão BF16, otimizando o throughput global do gateway.

2) Viabilidade em Escala: Cenário de Produção com NVIDIA T4: Para modelos compactos, propõe-se o uso de instâncias da família N1 com aceleradores T4. A Tabela IV detalha a composição de preços [16], demonstrando uma redução drástica no custo fixo horário em relação ao ambiente de teste.

TABELA IV
DETALHAMENTO DE CUSTOS NOMINAIS PARA INSTÂNCIAS N1 E ACELERADORES T4.

Componente	**USD / Hora**	**USD / Mês (730h)**
Instância `n1-standard-8`	0,4369977	319,008321
GPU NVIDIA Tesla T4	0,4025000	293,825000*
Total da Infraestrutura	**0,8394977**	**612,833321**

*Custos da GPU calculados a partir da taxa horária nominal $(0,4025 \times 730)$.

3) Viabilidade em Escala: Cenário de Produção com NVIDIA L4: A arquitetura L4 mostra-se necessária para a estabilidade dos modelos de 7B parâmetros. A Tabela V evidencia que o incremento financeiro para migrar da T4 para a L4 é marginal diante do ganho em memória e eficiência.

D. Modelos Proprietários (SaaS/API)

As soluções SaaS operam sob um modelo de precificação pay-as-you-go, caracterizado por custos estritamente variáveis atrelados ao volume de dados (tokens ou tempo de áudio) processados. Em contraste com a arquitetura self-hosted — que exige provisionamento de infraestrutura fixa independente

TABELA V

TAXAS DE CUSTO DE INFRAESTRUTURA PARA INSTÂNCIAS G2 E ACELERADORES L4.

Componente	USD / Hora	USD / Mês (730h)
Instância `g2-standard-8`	0,644046276	470,153781
GPU NVIDIA Tesla L4	0,210953724	153,996219**
Total da Infraestrutura	**0,855000000**	**624,150000**

**Cálculo proporcional baseado no valor residual para o total mensal informado.

da demanda — o SaaS alinha a despesa operacional (OpEx) diretamente à curva de utilização, eliminando custos com ociosidade.

E. Gpt-4o-Mini-Transcribe

A adoção do GPT-4o-mini-transcribe no benchmark representa uma mudança estratégica na gestão de recursos, migrando de um modelo de Despesas de Capital (CapEx), caracterizado pelo alto custo de provisionamento e manutenção de hardware físico ou instâncias de GPU dedicadas, para um modelo de Despesas Operacionais (OpEx). Esta abordagem elimina os custos fixos associados à infraestrutura local, como a manutenção de servidores e o consumo energético de GPUs em estado de ociosidade, permitindo que o sistema escale de forma puramente reativa ao volume de demanda acústica.

Para assegurar a precisão na coleta de métricas financeiras, utilizou-se a camada de abstração do framework LiteLLM. Através do acesso programático ao dicionário de metadados `model_cost`, foi possível extrair a tarifação exata aplicada pelo provedor (Azure/OpenAI) [18] durante as fases de inferência. O trecho de código a seguir exemplifica a recuperação desses dados, garantindo que o faturamento reportado reflita as condições reais de mercado.

```
1
2 from litellm import model_cost
3
4 print(model_cost["gpt-4o-mini-transcribe"])
```

Listing 3

PRINT PARA OBTER O CUSTO DOS MODELOS SERVIDOS NO LITELLM

A análise dos dados consolidados na Tabela VI evidencia uma arquitetura de custos otimizada para operações em larga escala. Conforme as especificações técnicas da Azure AI Foundry, o modelo `gpt-4o-mini-transcribe` opera com uma janela de contexto de 16.000 tokens. Esta delimitação técnica viabiliza a transcrição contínua de áudios com duração média de até 7 minutos por requisição, estabelecendo um equilíbrio ideal entre densidade de dados, baixa latência e previsibilidade de custos em fluxos de trabalho OpEx.

Dessa maneira, foram disponibilizados os seguintes dados:

Esta eficiência na tokenização permite o processamento de sequências longas com um custo de entrada fixo de apenas $3,00 por 1 milhão de tokens de áudio. Tal característica garante alta viabilidade para a transcrição de diálogos extensos

TABELA VI

CUSTOS E LIMITES: GPT-4O-MINI-TRANSCRIBE.

Componente	$/Token	$/1M	Limite
Áudio (In)	3×10^{-6}	$3,00	7 min*
Texto (In)	$1,25 \times 10^{-6}$	$1,25	-
Texto (Out)	5×10^{-6}	$5,00	2k tokens

*Capacidade aferida empiricamente, superior aos 16k nominais.

sem a necessidade de fragmentação agressiva do sinal (chunking), o que preserva a coerência contextual necessária para lidar com homófonos e regionalismos brasileiros. Adicionalmente, a limitação de saída fixada em 2.000 tokens induz a geração de transcrições concisas e objetivas, mantendo o custo final da geração de texto em $5,00 por milhão de tokens.

F. Whisper

O modelo Whisper é Open Source, porém, como foi utilizado via Groq, os custos computados serão retirados da API Groq. Os modelos operam sob um paradigma de faturamento por cobrança por hora de áudio, ao invés de tokens, simplificando a previsibilidade de custos em comparação aos modelos baseados em janelas de tokens. Esta plataforma utiliza LPUs (Language Processing Units) para oferecer um alto throughput, permitindo que o processamento ocorra em frações do tempo real (RTF reduzido).

Os indicadores de tarifação e os limites operacionais extraídos da plataforma estão consolidados na Tabela VII. Verifica-se que a variante Whisper Large V3 Turbo apresenta uma otimização econômica de aproximadamente 64

TABELA VII

CUSTOS E LIMITES: WHISPER (GROQ).

Modelo	Preço/Hora	ASH*	Max File
Whisper V3	$0,111	200k	100 MB
Whisper Turbo	$0,040	400k	-

*ASH: Segundos de áudio processados por hora (Audio Seconds per Hour).

G. Gemini 2.0-Flash-Lite e Gemini 2.5-Flash-Lite

No espectro das soluções proprietárias, as arquiteturas **Gemini 2.0 Flash Lite** e **Gemini 2.5 Flash Lite** representam a fronteira de eficiência e baixo custo da Google [19]. Diferente dos modelos hospedados em infraestrutura dedicada, estas versões operam sob um regime de tarifação por volume de tokens, eliminando custos fixos de manutenção de instâncias.

Ambos os modelos são projetados com capacidades multimodais nativas, permitindo a ingestão direta de sinais de áudio, sem a necessidade de pipelines explícitos de ASR seguidos de pós-processamento textual. A estrutura de custos é segmentada entre tokens de entrada (*input*) e tokens de saída (*output*), conforme detalhado na Tabela VIII.

Para 730 horas mensais de uso dos modelos, contando que, para arquivos de áudio é possível processar até 32 tokens por segundo [21], os cálculos resultam em:

- **Eficiência de Escala**: O Gemini 2.0 Flash-Lite é aproximadamente 3,8 vezes mais barato que a versão 2.5 para

TABELA VIII
ESTRUTURA DE TARIFAÇÃO PARA MODELOS GEMINI VIA GOOGLE API (POR 1M DE TOKENS).

Modelo	Entrada (USD)	Saída (USD)
Gemini 2.0 Flash Lite	0,075	0,30
Gemini 2.5 Flash Lite	0,300	0,40

A tarifação de entrada abrange o processamento direto de áudio e texto.

TABELA IX
PROJEÇÃO DE CUSTOS MENSAIS (730H DE ÁUDIO) PARA MODELOS GEMINI.

Modelo	Entrada	Saída	Total Mensal
Gemini 2.0 Flash-Lite	6,31	0,44	**6,75**
Gemini 2.5 Flash-Lite	25,23	0,58	**25,81**

o processamento massivo de áudio, mantendo uma janela de contexto robusta de 1 milhão de tokens.

- **Tokens Nativos**: Como ambos processam áudio sem necessidade de transcrição intermediária para "entender"o sinal, o custo de entrada (áudio) é o componente predominante no faturamento, representando mais de 93% do custo total mensal.

A integração desses modelos no ecossistema de testes é realizada via gateway LiteLLM, que atua como uma camada de abstração unificada. O uso do LiteLLM permite a padronização das chamadas de API e o roteamento inteligente entre modelos proprietários e instâncias locais, facilitando o monitoramento centralizado de custos e latência.

H. Dados Finais de Custos Consolidados

Com todos os valores calculados, consolidam-se custos operacionais mensais para o processamento de 730 horas de áudio, revelando disparidades econômicas significativas entre as diferentes arquiteturas e modelos de provisionamento analisados.

TABELA X
CUSTOS MENSAIS CONSOLIDADOS PARA 730 HORAS DE ÁUDIO TRANSCRITO

Modelo	Provedor / Infra	Tipo	USD/Mês
Gemini 2.0 Flash Lite	Google Cloud	API	**6,75**
Gemini 2.5 Flash Lite	Google Cloud	API	**25,81**
Whisper Turbo	Groq (LPU)	API	**29,20**
Whisper V3	Groq (LPU)	API	**81,03**
GPT-4o-mini	OpenAI	API	**51,34**
Parakeet / Voxtral	`n1-std-8` + T4	Local	**612,83**
Gemma-3n / Qwen2	`g2-std-8` + L4	Local	**624,15**

A análise dos dados consolidados revela **três regimes econômicos distintos**, cada um com características operacionais e financeiras específicas:

1) Regime 1: APIs de Baixíssimo Custo : Os modelos Gemini 2.0 e 2.5 Flash Lite demonstram uma vantagem competitiva substancial, operando com custos mensais de $6,75 e $25,81, respectivamente. Esta eficiência deriva de três fatores arquiteturais:

- **Processamento Nativo de Áudio:** Os modelos multimodais avaliados realizam ingestão direta de áudio em seu fluxo de inferência, eliminando a necessidade de pipelines explícitos baseados em ASR tradicional seguidos de etapas adicionais de interpretação textual. Internamente, o sinal acústico é convertido em representações latentes multimodais, permitindo que o modelo opere de forma end-to-end sobre o conteúdo sonoro e textual de maneira integrada.
- **Janela de Contexto Expansiva:** A disponibilidade de janelas de contexto de até 1 milhão de tokens permite o processamento de longas sequências multimodais dentro de uma única requisição, reduzindo a necessidade de fragmentação excessiva das entradas. No caso de áudio, a tokenização ocorre a uma taxa fixa documentada de 32 tokens por segundo, o que viabiliza a estimativa do consumo de contexto para sinais de longa duração.

O Gemini 2.0 Flash Lite posiciona-se como a solução mais econômica do benchmark, oferecendo uma redução de custos de:

- **4,3 vezes frente ao Whisper Turbo** ($29,20)
- **7,6 vezes frente ao GPT-4o-mini** ($51,34)
- **91 vezes frente ao Parakeet-TDT self-hosted** ($612,83)

2) Regime 2: APIs de Custo Moderado: O Whisper Turbo ($29,20) e o GPT-4o-mini ($51,34) operam em patamares de custo entre 4,3 e 7,6 vezes superiores ao Gemini 2.0, porém ainda **12 a 21 vezes inferiores** às soluções self-hosted. Essas alternativas se justificam em cenários específicos:

- **GPT-4o-mini:** Indicado quando a transcrição requer análise semântica profunda ou raciocínio contextual avançado, aproveitando as capacidades do modelo de linguagem subjacente.
- **Whisper Turbo:** Preferível quando a previsibilidade absoluta de custos é crítica, com tarifação direta por hora ($0,040/h) e infraestrutura LPU garantindo RTF ultra-baixo para latência mínima.

3) Regime 3: Self-Hosting (O Paradoxo da Escala): Observa-se que manter infraestrutura própria (GPU T4/L4) tornou-se economicamente ineficiente para volumes baixos ou médios. Como o custo do servidor é fixo ($612/mês) independentemente do uso, o modelo self-hosted sofre com a ociosidade.

O ponto de equilíbrio financeiro (break-even), onde o custo do servidor dedicado empata com o custo variável da API do Gemini 2.5, ocorre apenas em volumes superiores a **17.300 horas de áudio/mês**.

Isso significa que, a menos que a aplicação processe o equivalente a **24 fluxos de áudio ininterruptos (24x7)**, é financeiramente mais vantajoso utilizar as APIs. O Self-Hosting, portanto, restringe-se a casos de uso onde há imperativo regulatório estrito (LGPD, HIPAA) que proíba o envio de dados para nuvens públicas, independentemente do custo.

VII. ANÁLISE DE RESULTADOS E QUALIDADE DE TRANSCRIÇÃO

Para a análise e o benchmark dos modelos de transcrição, avaliou-se o desempenho de cada modelo frente a

áudios em português brasileiro. Foram selecionados áudios representativos de diferentes sotaques regionais, presença de homófonos — palavras com pronúncia idêntica, mas grafias e significados distintos — e entidades numéricas. Essas amostras foram escolhidas estrategicamente para permitir uma avaliação abrangente dos modelos.

A análise considerou tanto a precisão quantitativa, medida pela taxa de erro de palavras (Word Error Rate – WER), quanto o desempenho de latência, calculado por meio de uma métrica que relaciona o tempo de processamento do modelo à duração do sinal acústico. Para garantir robustez estatística e maior confiabilidade dos resultados, cada transcrição foi executada em 50 iterações, possibilitando a obtenção de dados consistentes sobre latência e qualidade de transcrição. Ademais, foi utilizado o mesmo prompt de transcrição para todos os modelos.

A. *Métricas de Precisão Quantitativa*

Nas métricas de precisão quantitativa, a avaliação foi conduzida por meio da Word Error Rate (WER), definida como a razão entre o número total de edições necessárias para transformar a transcrição gerada pelo modelo na transcrição de referência e o número total de palavras da referência original. Essas edições incluem substituições, deleções (omissões) e inserções, conforme formalizado na Equação 2.

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{2}$$

Onde:

- S é o número de substituições;
- D é o número de deletações (omissões);
- I é o número de inserções;
- C é o número de palavras corretas;
- N é o número total de palavras na referência ($N = S + D + C$).

Para cada áudio transcrito por cada modelo avaliado, o cálculo do Word Error Rate (WER) foi realizado de forma individual. Essa abordagem possibilitou não apenas a obtenção de métricas agregadas de desempenho, mas também uma análise detalhada dos padrões de erro, permitindo a comparação direta entre os tipos de falhas cometidas por cada modelo. O valor médio do WER, obtido ao longo das iterações, é apresentado na Figura 11, evidenciando diferenças significativas de desempenho entre os modelos analisados.

Com o objetivo de analisar o desempenho individual de cada modelo de forma mais granular, foi construído um heatmap, apresentado na Figura 12. Essa visualização permite observar, para cada combinação de áudio e modelo, a variação do WER, evidenciando padrões relacionados à maior presença de sotaques regionais, à complexidade fonética dos trechos e aos cenários em que os modelos apresentaram maior dificuldade ou facilidade na compreensão e transcrição das palavras.

A partir da análise do heatmap e do dataset do projeto, disponibilizado publicamente na plataforma Hugging Face, que contém as análises detalhadas de erro para cada áudio transcrito, foi possível identificar padrões qualitativos relevantes no comportamento de cada modelo avaliado, descritos a seguir.

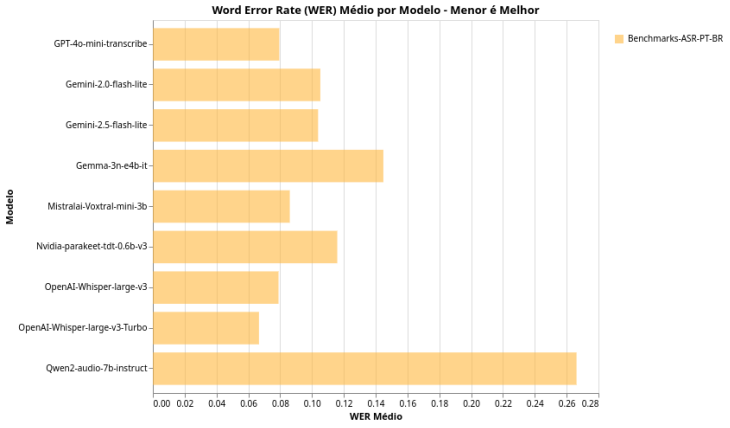

Fig. 11
TAXA DE ERRO (WER) MÉDIO DE CADA MODELO.

- **GPT-4o-mini-transcribe**: O modelo apresentou desempenho limitado na desambiguação de homófonos, utilizando de forma recorrente o termo “sessão” em contextos nos quais a grafia correta seria semanticamente distinta. No que se refere à transcrição de entidades numéricas, observou-se apenas um erro pontual, no qual o termo “cento” foi incorretamente transcrito como “cent”, caracterizando uma falha isolada de normalização lexical.
- **Qwen2-audio-7b-instruct**: Este foi o modelo com pior desempenho geral em termos de WER. Observou-se um alto grau de alucinação em praticamente todas as transcrições, acompanhado por numerosas substituições fonéticas inadequadas. No caso dos homófonos, o modelo substituiu o termo “cessão” por “substituição”, além de inserir palavras inexistentes no áudio original. Destaca-se ainda a inserção recorrente de termos em inglês, mesmo quando ausentes no conteúdo de entrada.
- **Nvidia-parakeet-tdt-0.6b-v3**: O modelo apresentou erros predominantemente leves, como substituições funcionais (“no” por “do”, “de” por “do”), omissões de artigos e trocas pronominais (“este” por “esse”). Contudo, em uma das transcrições, observou-se a inserção indevida de termos em inglês. No tratamento de homófonos, utilizou corretamente o termo “cessão” em sua primeira ocorrência.
- **Mistralai-Voxtral-mini-3b**: O modelo apresentou inserções indevidas de artigos antes de algumas palavras e desempenho inconsistente na desambiguação de homófonos, errando a maioria das ocorrências do termo “sessão”. Destaca-se ainda uma substituição foneticamente distante, na qual “cruzamento” foi transcrito como “empreza-mento”.
- **Gemini-2.5-flash-lite**: O modelo realizou diversas substituições semânticas relevantes, como “cruzamento” por “endividamento”, além de omissões de pronomes e trocas pronominais (“esse” por “este”). Esses erros indicam dificuldades na preservação do significado original da sentença.
- **Gemma-3n-e4b-it**: Em relação aos homófonos, o modelo

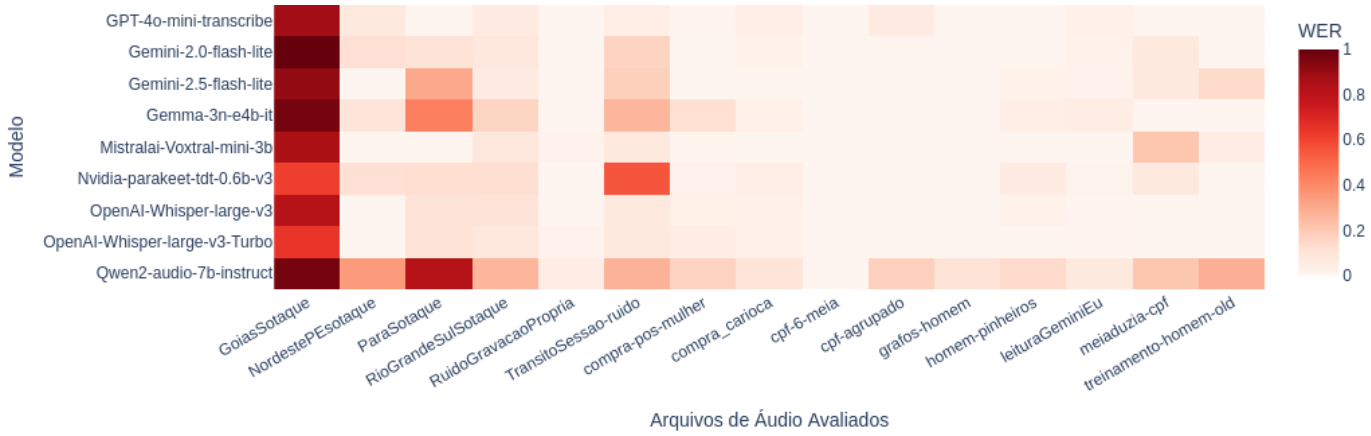

Fig. 12

HEATMAP DE ANÁLISE DO DESEMPENHO DE MODELOS POR ÁUDIO.

utilizou corretamente o termo "sessão", porém substituiu "cessão" por "cessamento", alterando significativamente o significado. Observou-se ainda a substituição de tempos verbais ("era" por "é"), erros de preposição e omissões pontuais de palavras.

- **Gemini-2.0-flash-lite**: O modelo apresentou substituições morfológicas, omissões de conjunções e artigos, além da substituição inadequada de preposições, como "desde" por "das".
- **OpenAI-Whisper-large-v3**: Observou-se omissão de preposições, troca de pronomes possessivos ("meu" por "o") e substituições pronominais recorrentes ("este" por "esse"), erros que, embora frequentes, tendem a ter menor impacto semântico.
- **OpenAI-Whisper-large-v3-Turbo**: O modelo apresentou erros de repetição de caracteres, caracterizando alucinação em nível de token, além de omissões de preposições e substituições pronominais semelhantes às observadas na versão não turbo.

B. Estratégias de Segmentação

Os modelos Gemma 3n e Qwen2 Audio não foram capazes de transcrever diretamente áudios com duração superior a um minuto. Inicialmente, foi adotada uma estratégia de chunking temporal, na qual os áudios eram segmentados em janelas de 30 segundos com sobreposição (overlap), visando preservar informações relevantes na transição entre segmentos. No entanto, por se tratar de um particionamento baseado exclusivamente em tempo, essa abordagem resultou na duplicação de trechos transcritos, com repetição de frases e palavras em regiões de sobreposição.

Com o objetivo de garantir maior consistência metodológica e manter um elevado padrão de qualidade nos benchmarks, optou-se pela aplicação de um pipeline de Voice Activity Detection (VAD) [20] baseado em redes neurais, utilizando o Silero VAD. Essa abordagem permite identificar automaticamente os intervalos do sinal acústico nos quais há presença ou ausência de fala, possibilitando a segmentação do áudio em pontos de silêncio natural. Dessa forma, o chunking passou a ser realizado de maneira semanticamente coerente, reduzindo significativamente o risco de duplicação de conteúdo e assegurando maior fidelidade na transcrição final.

Entretanto, ao aplicar o *chunking* com VAD no endpoint do modelo **Qwen2 Audio** hospedado no Vertex AI, o serviço passou a retornar o erro **HTTP 503 — No healthy backend servers**. Diante dessa instabilidade, foi necessário manter, especificamente para esse modelo, a estratégia de *chunking* temporal padrão, sem o uso de VAD.

Contudo, a fim de preservar a comparabilidade dos resultados e avaliar a latência intrínseca dos modelos em cenários padrão, essas técnicas de segmentação não foram aplicadas aos áudios de menor duração, nos quais os modelos conseguiram realizar a transcrição de forma direta. Essa decisão metodológica permitiu isolar o impacto do chunking e do VAD sobre a latência, garantindo uma avaliação mais fiel do desempenho temporal natural de cada modelo.

VIII. ANÁLISE DE ROBUSTEZ LINGUÍSTICA E REGIONALISMOS

Com o objetivo de avaliar a robustez dos modelos de reconhecimento automático de fala (ASR) frente à diversidade linguística e dialetal da língua portuguesa, foram utilizados áudios provenientes da plataforma *LocaLingual*. Essa base colaborativa reúne gravações de falantes de diferentes regiões do Brasil e de outros países, permitindo a análise sistemática de variações regionais, prosódicas e fonéticas do português. Nesse contexto, a documentação do comportamento de modelos de ASR em português brasileiro torna-se particularmente relevante, dada a elevada heterogeneidade linguística presente no território nacional.

Para a avaliação qualitativa da capacidade de transcrição em português brasileiro, os modelos foram analisados a partir de dois eixos principais: *(i) Regionalismos e variações fonéticas* e *(ii) Transcrição de entidades numéricas*. Esses critérios permitiram uma análise mais granular dos tipos de erro e das limitações linguísticas apresentadas por cada arquitetura.

A. *Regionalismos e variações fonéticas*

- **Áudio - Sotaque do Rio Grande do Sul**: Neste áudio, a maioria dos modelos apresentou dificuldade na transcrição de expressões idiomáticas e regionalismos característicos do sul do Brasil. A expressão "*Mas bah, gurizada*" foi incorretamente transcrita como "Margarida", "Margotizada", "Marjorie" e "barulhosa", evidenciando falhas na desambiguação fonética e no reconhecimento de construções regionais. Entre os modelos avaliados, apenas o **Gemini 2.5** conseguiu transcrever corretamente a expressão completa, bem como o uso do termo regional "*tri*" no contexto adequado.
- **Áudio - Sotaque do Estado de Goiás**: Este foi o cenário mais crítico de desempenho. O áudio apresenta um sotaque fortemente regional, com variações fonéticas acentuadas, o que resultou em altos níveis de alucinação em praticamente todos os modelos avaliados. O **Gemini 2.0 Flash Lite** não gerou transcrição; o **Gemini 2.5 Flash Lite** produziu um texto desconectado do áudio, com estrutura poética; o **GPT-4o-mini-transcribe** identificou apenas dois termos isolados.
  Os modelos **Whisper Large v3** e **Whisper Large v3 Turbo** não conseguiram capturar corretamente o conteúdo semântico, sendo que o Turbo apresentou inserções aleatórias. O **Gemma 3n** indicou ausência de áudio, o **Qwen2 Audio** respondeu como um assistente conversacional em vez de transcrever, o **Nvidia Parakeet** teve dificuldades com nomes próprios e regionalismos, e o **Mistral Voxtral** gerou uma frase inexistente e semanticamente incorreta.
- **Áudio - Sotaque do Nordeste (PE)**: Neste cenário, os modelos apresentaram desempenho significativamente melhor em comparação aos demais sotaques avaliados. Os modelos **Gemini 2.5**, **Whisper Large v3**, **Whisper Large v3 Turbo** e **Mistral Voxtral** realizaram transcrições corretas, sem perdas de conteúdo ou distorções relevantes.
  Os erros observados foram majoritariamente estilísticos, como a substituição de formas informais ("pra") por formas mais formais ("para"), sem impacto semântico. O modelo **Qwen2 Audio**, por sua vez, apresentou maior tendência à normalização linguística, substituindo expressões regionais por termos do português padrão.
- **Áudio — Sotaque do Pará (ParaSotaque)**: Neste áudio, houve grande variação de desempenho, especialmente no reconhecimento de nomes próprios e datas. O **Gemini 2.0** inseriu uma frase introdutória inexistente. O **Gemini 2.5** apresentou erros em entidades geográficas, substituindo "Tucuruí" por "Itaipu" e "Turim", além de errar a data mencionada, transcrevendo 1974 em vez de 1984.
  Os modelos **Whisper Large v3** e **Whisper Large v3 Turbo** erraram apenas o nome próprio "Tucuruí". O **Gemma 3n** apresentou alucinações graves, com erros simultâneos de data e localidade. O **Qwen2 Audio** substituiu "Tucuruí/Pará" por "Curitiba/Paraná" e gerou termos inexistentes. O **Nvidia Parakeet** teve erros pontuais em nomes próprios, enquanto os modelos **Mistral Voxtral** e **GPT-4o-mini-transcribe** realizaram a transcrição corretamente.

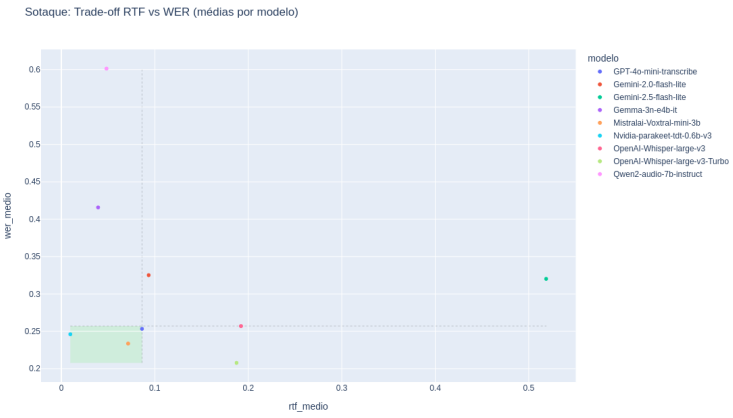

Fig. 13
TRADE-OFF ENTRE O FATOR DE TEMPO REAL (RTF) E A TAXA DE ERRO DE PALAVRAS (WER) DOS MODELOS AVALIADOS NA CATEGORIA DE SOTAQUES REGIONAIS.

De acordo com o gráfico, os modelos que apresentaram melhor desempenho na categoria de sotaques foram o **Nvidia Parakeet**, o **MistralAI Voxtral Mini** e o **GPT-4o-mini-transcribe**.

B. *Transcrição de Entidades Numéricas*

- No que se refere à transcrição de entidades numéricas, observou-se variação significativa entre os modelos quanto às estratégias de normalização lexical e formatação numérica. Os modelos **GPT-4o-mini-transcribe**, **Gemini 2.5 Flash Lite**, **Nvidia Parakeet** e **Qwen2 Audio** optaram predominantemente pela transcrição dos numerais por extenso. O modelo **Mistral Voxtral** apresentou um comportamento híbrido, alternando entre representações numéricas e lexicalizadas ao longo das transcrições. Todos os modelos demonstraram compreensão adequada da expressão coloquial "meia" como o dígito *6*, indicando robustez no mapeamento semântico de numerais informais. Entretanto, o modelo **Qwen2 Audio** apresentou inconsistências na normalização, convertendo corretamente "meia" para "seis", mas transcrevendo o número "98" como "nove e oitenta", caracterizando um erro semântico de decomposição numérica.
  O modelo **GPT-4o-mini-transcribe** apresentou um erro pontual de truncamento lexical, transcrevendo o termo "cento" como "cent", evidenciando uma falha isolada de geração textual, sem impacto significativo na compreensão global do conteúdo. Já o modelo **Gemini 2.0 Flash Lite** apresentou omissão de dígitos e inserção indevida, substituindo a expressão "meia dúzia" por um dígito "2", sem preservar a equivalência semântica.
  O modelo **Mistral Voxtral**, em um dos áudios avaliados, tentou inferir e impor uma formatação de documento do

tipo CPF, porém errou os dois dígitos finais, indicando uma extrapolação contextual indevida. Por fim, o modelo **Nvidia Parakeet** interpretou corretamente "meia" como *6*, mas manteve o termo "dúzia" residual na transcrição, resultando em uma inconsistência parcial de normalização.

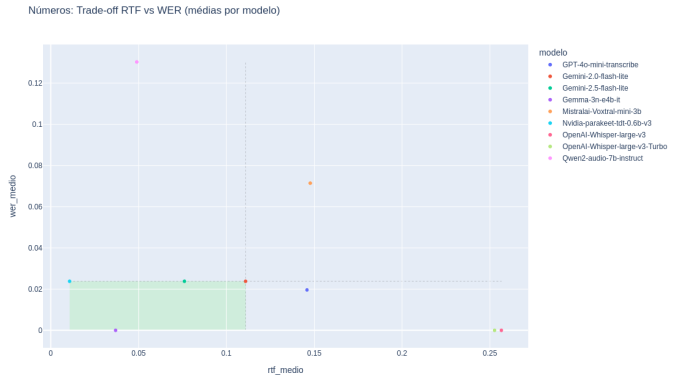

Fig. 14

TRADE-OFF ENTRE O FATOR DE TEMPO REAL (RTF) E A TAXA DE ERRO DE PALAVRAS (WER) DOS MODELOS DE RECONHECIMENTO AUTOMÁTICO DE FALA (ASR) AVALIADOS NA TRANSCRIÇÃO DE ENTIDADES NUMÉRICAS.

Considerando a categoria de entidades numéricas, os melhores desempenhos foram observados nos modelos **Gemma 3n**, **Nvidia Parakeet**, **Gemini 2.5 Flash Lite** e **Gemini 2.0 Flash Lite**. De forma geral, os resultados reforçam que um WER médio mais baixo não garante, necessariamente, melhor interpretação semântica, especialmente em cenários com variações linguísticas, como homófonos e sotaques regionais.

IX. DESEMPENHO TEMPORAL E LATÊNCIA

Para a avaliação do desempenho temporal e da latência dos modelos, utilizou-se como métrica principal o Real-Time Factor (RTF), amplamente empregado na literatura de Automatic Speech Recognition (ASR). O RTF é definido como a razão entre o tempo de processamento do áudio e a duração do próprio sinal acústico, refletindo a capacidade do modelo de operar em tempo real ou em regimes mais rápidos que o tempo real.

1) Interpretação operacional do RTF: Do ponto de vista operacional, o Real-Time Factor (RTF) é um indicador fundamental para determinar a viabilidade de um modelo de ASR em cenários de transcrição ao vivo, uma vez que expressa a relação entre o tempo de processamento e a duração do sinal de entrada. A interpretação do RTF pode ser sintetizada da seguinte forma:

- **RTF < 1:** Indica que o sistema é capaz de processar o áudio mais rapidamente do que ele é produzido. Por exemplo, um modelo que processa 60 segundos de áudio em 30 segundos apresenta um RTF de 0,5, caracterizando-se como adequado para aplicações em tempo real e escalável para ambientes com múltiplas requisições simultâneas.
- **RTF > 1:** Indica que o tempo de processamento excede a duração do áudio, tornando o sistema mais lento do que a fala humana. Nesses casos, o modelo torna-se inadequado para aplicações de transcrição em tempo real, pois o atraso cresce progressivamente ao longo da interação.
- **RTF = 1:** Representa um regime de processamento em tempo real estrito, no qual o sistema transcreve o áudio exatamente na mesma velocidade em que ele é falado, sem margem para atrasos acumulativos.

Considerando que cada áudio foi processado em 50 iterações por modelo, o valor de latência utilizado no cálculo do RTF correspondeu à média do tempo de processamento dessas iterações, normalizada pela duração do áudio. Dessa forma, o RTF pode ser formalmente representado pela Equação 3:

$$RTF = \frac{T_{medioproc}}{D_{audio}} \tag{3}$$

Onde:

- $T_{medioproc}$ representa o tempo médio das 50 iterações, individualmente, necessário para o processamento da inferência;
- D_{audio} é a duração total do sinal acústico (áudio) original.

A partir do cálculo do RTF para cada processamento, foi possível obter o valor médio de RTF por modelo, apresentado na Figura 15.

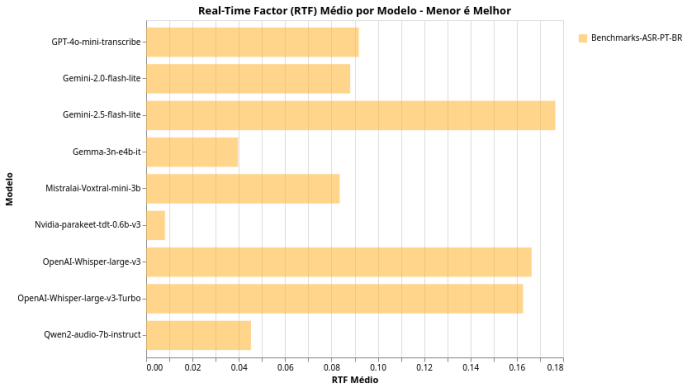

Fig. 15

FATOR DE TEMPO REAL (RTF) MÉDIO DE CADA MODELO.

A análise dos resultados mostra diferenças claras de desempenho entre os modelos avaliados. O modelo Nvidia Parakeet, baseado na arquitetura Token-and-Duration Transducer (TDT), destacou-se pela alta velocidade de processamento, alcançando um RTF médio de 0,00813, o que indica processamento muito acima do tempo real.

Em contraste, o Gemini 2.5 Flash Lite apresentou o maior RTF médio, de 0,17664. Embora ainda opere abaixo do tempo real, esse valor é cerca de 21,7 vezes superior ao do Nvidia Parakeet, evidenciando o impacto da arquitetura na latência de inferência.

De forma geral, todos os modelos apresentaram RTF inferior a 1, porém com diferenças relevantes de margem operacional,

que afetam diretamente a escalabilidade, a latência percebida pelo usuário e os custos de infraestrutura.

Mesmo nos casos em que não houve transcrição válida, o RTF foi mantido na análise, pois ele mede o custo computacional do processamento. Essa abordagem permite quantificar o impacto da falha: um modelo que falha rapidamente é menos oneroso do que aquele que consome mais tempo antes de concluir que não consegue transcrever, especialmente em aplicações sensíveis à latência.

A. *Desvio Padrão*

Para uma análise mais abrangente do desempenho temporal dos modelos, torna-se fundamental avaliar não apenas o *Real-Time Factor* (RTF), mas também a variabilidade do tempo de processamento ao longo das iterações, representada pelo desvio padrão. Essa métrica permite caracterizar a estabilidade temporal dos modelos de transcrição e identificar comportamentos anômalos que não são capturados por métricas baseadas exclusivamente em médias.

De modo geral, os modelos avaliados apresentaram desvios padrão reduzidos, indicando elevada consistência no tempo de inferência entre as diferentes execuções. Contudo, exceções relevantes foram observadas em cenários caracterizados por maior complexidade acústica e linguística. No áudio contendo um sotaque regional fortemente marcado do estado de Goiás, o modelo **Gemini 2.5** registrou um tempo máximo de processamento de **165,082 segundos** em uma das iterações. De forma semelhante, no áudio denominado *"Trânsito Sessão Ruído"*, o mesmo modelo atingiu um tempo máximo de **191,2332 segundos** para a conclusão da transcrição. Adicionalmente, o modelo **Whisper large v3 Turbo** apresentou um tempo máximo de processamento de **61,31 segundos** nesse mesmo contexto experimental.

Esses resultados indicam que, em situações nas quais os modelos enfrentam maior dificuldade na interpretação, desambiguação ou alinhamento do conteúdo acústico, ocorre um aumento expressivo e pontual da latência de inferência. Tal comportamento sugere uma correlação direta entre a complexidade do sinal de entrada e a instabilidade temporal do modelo, evidenciando limitações importantes para aplicações sensíveis a atraso.

Do ponto de vista operacional, esses picos de latência podem comprometer o cumprimento de *Service Level Agreements* (SLAs), que estabelecem formalmente os níveis mínimos de desempenho, qualidade e disponibilidade esperados de um serviço. Em sistemas de atendimento síncrono ou de transcrição em tempo real, essa variabilidade temporal impacta negativamente a previsibilidade operacional e a experiência do usuário final, reforçando a necessidade de avaliar não apenas métricas médias, mas também a dispersão e os valores extremos do tempo de processamento.

A Figura 16 apresenta o desvio padrão médio do tempo de inferência para cada um dos modelos de *Speech-to-Text* (STT) avaliados neste estudo.

X. Trade-off entre RTF, WER e Custo

Para avaliar o equilíbrio entre a qualidade da transcrição e a velocidade de resposta dos modelos, é fundamental analisar

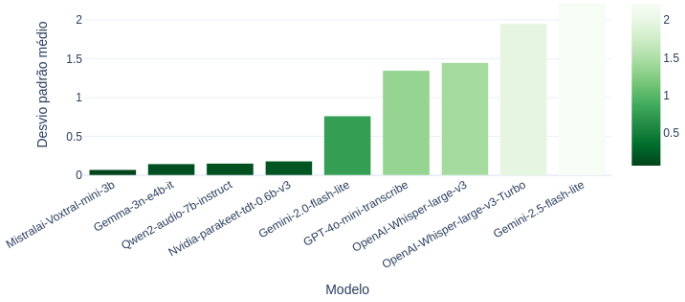

Fig. 16
Desvio padrão médio de cada modelo.

conjuntamente as métricas de WER e RTF. Idealmente, um modelo deve apresentar valores baixos para ambas as métricas. No entanto, na prática, observa-se um compromisso entre precisão e eficiência: modelos com WER muito baixo podem ser lentos para uso em cenários reais, enquanto modelos muito rápidos podem introduzir erros que comprometem a compreensão, geram retrabalho ou causam falhas operacionais.

Diante disso, foi gerado um gráfico de trade-off entre RTF e WER, permitindo a visualização comparativa do posicionamento de cada modelo em relação a essas duas métricas.

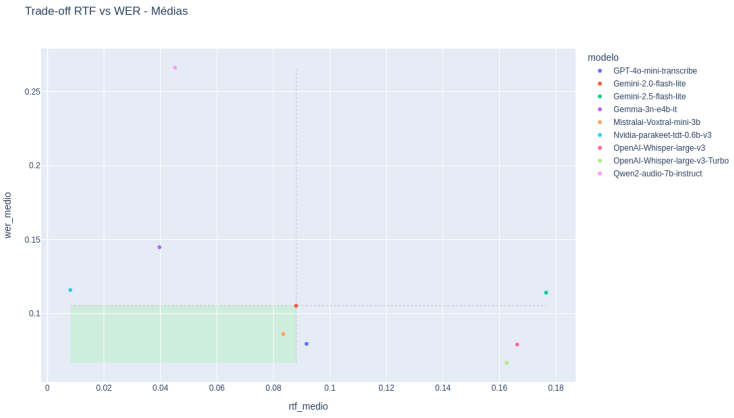

Fig. 17
Trade-off entre o fator de tempo real (RTF) e a taxa de erro de palavras (WER) dos modelos avaliados.

A partir da análise do gráfico, os modelos que apresentaram o melhor equilíbrio entre precisão e desempenho foram o **Mistral Voxtral Mini** e o **Gemini 2.0 Flash Lite**.

Além das métricas de qualidade de transcrição e latência discutidas anteriormente, a análise de custo operacional é um fator determinante para a adoção prática de modelos de ASR, especialmente em cenários de produção e larga escala. Avaliações baseadas exclusivamente em desempenho técnico podem conduzir a decisões subótimas quando aspectos como escalabilidade, previsibilidade de gastos e restrições orçamentárias são considerados.

Com base nos valores de WER previamente calculados e nos

custos associados a cada modelo, foi construída uma análise comparativa entre *WER médio* e *custo de inferência*. Para essa avaliação, adotou-se o conceito de *Fronteira de Pareto*, que permite identificar soluções que apresentam o melhor equilíbrio entre múltiplos objetivos concorrentes, neste caso, minimizar simultaneamente o erro de transcrição e o custo operacional.

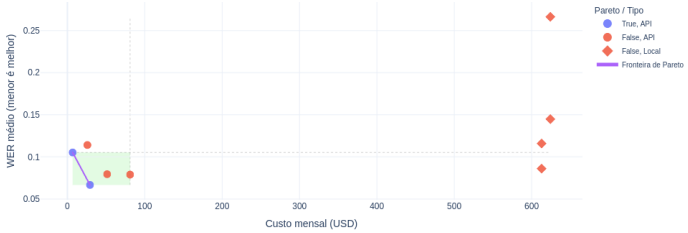

Fig. 18

RELAÇÃO ENTRE O CUSTO MENSAL E A TAXA DE ERRO DE PALAVRAS (WER) MÉDIO DOS MODELOS AVALIADOS.

A partir da Figura 18, observa-se que os modelos **Gemini 2.0 Flash Lite** e **Whisper Large v3** posicionam-se na Fronteira de Pareto, indicando o melhor compromisso entre custo e qualidade de transcrição dentre os modelos avaliados. Esses modelos alcançam baixos valores de WER sem incorrer em custos elevados, tornando-se opções particularmente atrativas para aplicações em larga escala.

O modelo **GPT-4o-mini-transcribe**, embora não pertença estritamente à Fronteira de Pareto, encontra-se próximo a esse conjunto ótimo, apresentando um bom equilíbrio entre custo e qualidade. Isso sugere que, dependendo dos requisitos específicos da aplicação, como integração com outros serviços ou suporte multimodal, ele ainda pode ser considerado uma alternativa viável do ponto de vista econômico.

Complementarmente, foi construída uma segunda análise de Pareto com o objetivo de avaliar o trade-off entre custo operacional e fator de tempo real (RTF), métrica diretamente associada à latência de inferência e à capacidade de atendimento em tempo real.

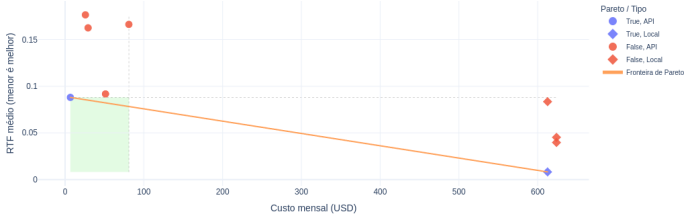

Fig. 19

RELAÇÃO ENTRE O CUSTO MENSAL E O FATOR DE TEMPO REAL (RTF) MÉDIO DOS MODELOS AVALIADOS.

Conforme ilustrado na Figura 19, os modelos Gemini 2.0 Flash Lite e NVIDIA Parakeet compõem a Fronteira de Pareto no espaço custo–RTF. Isso indica que, dentro do conjunto analisado, não existem alternativas que ofereçam simultaneamente menor custo e menor latência de processamento.

Embora o modelo da NVIDIA apresente custos significativamente mais elevados, sua presença na fronteira de Pareto se justifica pelo fato de atingir valores de RTF substancialmente inferiores aos demais modelos. Esse resultado evidencia que a Fronteira de Pareto não representa necessariamente as opções mais econômicas, mas sim os melhores compromissos possíveis entre objetivos conflitantes, neste caso custo e desempenho temporal.

Por fim, o modelo GPT-4o-mini-transcribe, apesar de não integrar formalmente a fronteira de Pareto nesse cenário, posiciona-se próximo ao quadrante mais vantajoso do gráfico, indicando novamente um bom equilíbrio entre custo e latência. Tal característica o torna uma alternativa prática para cenários de produção nos quais restrições orçamentárias coexistem com requisitos moderados de baixa latência.

XI. CONCLUSÃO

Para concluir a análise dos modelos avaliados, é essencial considerar conjuntamente três fatores centrais discutidos ao longo deste trabalho: o Real-Time Factor (RTF), o Word Error Rate (WER) e os custos de implementação. A escolha do modelo mais adequado não deve se basear exclusivamente na maior precisão ou na menor latência de forma isolada, mas sim no equilíbrio entre qualidade de transcrição, tempo de resposta e viabilidade econômica.

Em aplicações práticas de transcrição de áudio em português, especialmente em ambientes de produção, esse balanceamento é determinante para garantir bom desempenho operacional, uma experiência de uso satisfatória e sustentabilidade de custos ao longo do tempo.

A. Custos

A inclusão dos modelos Gemini Flash Lite na análise comparativa altera significativamente o cenário de decisão para sistemas de transcrição em língua portuguesa. Considerando um regime de 730 horas mensais, representativo de operações de pequeno e médio porte, as APIs multimodais nativas demonstraram ampla vantagem econômica. Nesse contexto, o Gemini 2.0 Flash Lite apresentou um custo até 91 vezes inferior quando comparado às soluções self-hosted mais eficientes.

O self-hosting permanece justificável exclusivamente quando há:

1) **Processamento de dados sensíveis ou regulados**, nos quais não é permitido o envio de áudio para serviços externos via API;
2) **Ambientes com exigência de controle total dos dados**, como sistemas governamentais, jurídicos ou de saúde;
3) **Volumes extremamente elevados**, superiores a 17.300 horas mensais, onde a amortização do custo de infraestrutura torna o self-hosting economicamente vantajoso;
4) **Requisitos rigorosos de latência**, em aplicações que demandam resposta sub-50 ms e não toleram latência de rede.

Fora desses cenários, a adoção de APIs proprietárias se mostra a alternativa mais eficiente do ponto de vista econômico e operacional, permitindo escalabilidade elástica sem a necessidade de investimento contínuo em infraestrutura de GPU.

B. Latência e Word Error Rate

Para definir o modelo mais adequado para uso em produto, é necessário equilibrar latência e qualidade de transcrição, representada pelo WER. Esse balanceamento foi analisado por meio das métricas apresentadas nas Figuras 18 e 19, que destacam tanto a fronteira de Pareto quanto os quadrantes de maior relevância operacional.

A seleção final considerou os modelos que se destacaram simultaneamente nos dois gráficos, refletindo um bom compromisso entre custo, RTF e WER. Nesse contexto, o modelo que apresentou melhor equilíbrio global foi o Gemini 2.0 Flash Lite, figurando de forma consistente nas fronteiras de Pareto analisadas.

Em segundo lugar, destaca-se o GPT-4o-mini-transcribe, que, embora não tenha integrado formalmente a fronteira de Pareto, apresentou desempenho próximo aos quadrantes ideais, configurando-se como uma alternativa robusta dependendo dos requisitos específicos da aplicação.

XII. Desafios de Implementação e Resiliência de Infraestrutura

Os desafios identificados durante a fase de implementação são fundamentais para a avaliação da viabilidade prática dos modelos analisados em ambientes reais de produção. Nesse contexto, esta seção discute os principais erros e limitações observados ao longo da execução dos experimentos de benchmarking.

Durante o processo de avaliação, ao lidar com falhas relacionadas à segmentação de áudio por meio de Voice Activity Detection (VAD) no modelo **Qwen2-Audio** implantado em um Endpoint do **Vertex AI**, foi identificada uma instabilidade crítica caracterizada por erros **HTTP 503 (No healthy backend servers)** e mensagens de **Server out of service**. A análise detalhada dos logs e das métricas de telemetria indicou que tais falhas não estavam associadas a erros lógicos na aplicação, mas sim à saturação de recursos computacionais no nó de serviço do Vertex AI. A investigação permitiu isolar dois fatores técnicos principais, descritos a seguir.

1) Saturação de VRAM e Erros de Out-of-Memory (OOM): A arquitetura multimodal do **Qwen2-Audio (7B)** apresenta elevada demanda por memória de vídeo. O carregamento simultâneo dos pesos do modelo e dos componentes responsáveis pelo processamento de áudio reduz significativamente a quantidade de VRAM disponível para a alocação do *Key-Value (KV) Cache*. Ao processar requisições contendo áudios de maior duração, essa limitação resulta em erros de *Out-of-Memory (OOM)*, levando ao encerramento abrupto do contêiner.

Esse cenário é especialmente crítico em ambientes de produção que não utilizam técnicas de quantização, como *AWQ* ou *GPTQ*, que poderiam reduzir a pegada de memória do modelo e aumentar a estabilidade do serviço.

2) Timeout do Liveness Probe e Ciclos de Reinicialização: Outro fator relevante foi o tempo de inferência excedendo os limites configurados para o **Liveness Probe** do Vertex AI. Quando o processamento de um áudio ultrapassa o intervalo definido para os *health checks*, o orquestrador do Google Cloud interpreta o contêiner como não responsivo e inicia um ciclo de reinicialização automática (*restart loop*). Esse comportamento impede que o serviço atinja um estado operacional estável, resultando em indisponibilidade contínua do *backend*.

Além disso, ao utilizar Endpoints no Vertex AI, observou-se que o tempo de provisionamento das máquinas virtuais apresenta alta variabilidade. Durante a execução dos benchmarks, foram registrados tempos de espera que variaram de aproximadamente **5 minutos** a **35 minutos** para a disponibilização do hardware solicitado. Essa imprevisibilidade impacta diretamente a execução de experimentos e a operação contínua do serviço.

Por fim, também foi constatado que nem sempre o tipo de máquina desejado está disponível na região geográfica selecionada. Nesses casos, torna-se necessário migrar a implantação para outras localidades, assumindo o risco de indisponibilidade do mesmo hardware, o que adiciona complexidade operacional e reduz a previsibilidade da infraestrutura.

A. Pipeline de remoção de ruídos

Como discutido anteriormente, a pipeline de remoção de ruído foi aplicada a todos os áudios utilizados nos experimentos. No entanto, observou-se um comportamento atípico em dois casos específicos: os áudios grafos-homem e homem-pinheiros, ambos gerados sinteticamente pela plataforma ElevenLabs.

Nesses exemplos, apesar de os áudios apresentarem baixo nível de ruído na versão original, o processo de denoising introduziu degradações perceptíveis no sinal, resultando em perdas locais de informação acústica. Como consequência, os modelos de ASR demonstraram maior dificuldade de compreensão, refletida em transcrições menos precisas.

Em contraste, para os demais áudios avaliados, incluindo gravações reais com ruído de fundo, a pipeline de remoção de ruído apresentou desempenho consistente, contribuindo positivamente para a qualidade da transcrição.

Esses resultados indicam que a eficácia do denoising pode variar significativamente de acordo com a natureza do áudio de entrada, especialmente em sinais sintéticos com características acústicas já bem controladas. Dessa forma, trabalhos futuros devem incluir a avaliação de um conjunto mais amplo de áudios sintéticos, bem como experimentos com ruídos artificiais e reais injetados de forma controlada, a fim de compreender melhor os limites e possíveis efeitos colaterais da pipeline de pré-processamento.

Referências

[1] ASSEMBLYAI. Conformer-1: State-of-the-art Speech Recognition Model. AssemblyAI Blog, 2023. Disponível em: https://www.assemblyai.com/blog/conformer-1/.

[2] ZHENG, L.; WANG, X.; ZHAO, Q.; LI, T. Two-Stage Domain Adaptation for LLM-Based ASR by Decoupling Linguistic and Acoustic Factors. **Appl. Sci.**, v. 16, n. 60, 2026. Disponível em: https://doi.org/10.3390/app16010060.

[3] SRIVASTAV, Vaibhav; ZHENG, Steven; BEZZAM, Eric; LE BIHAN, Eustache; MOUMEN, Adel; GANDHI, Sanchit. Open ASR Leaderboard: Towards Reproducible and Transparent Multilingual Speech Recognition Evaluation. arXiv:2510.06961, 2025. Disponível em: https://arxiv.org/abs/2510.06961.

[4] QED42. NVIDIA Parakeet-TDT-0.6B-V2: A Deep Dive into State-of-the-Art Speech Recognition Architecture. QED42 Insights, 2025. Disponível em: https://www.qed42.com/insights/nvidia-parakeet-tdt-0-6b-v2-a-deep-dive-into-state-of-the-art-speech-recognition-architecture.
[5] SCHRÖTER, Hendrik. DeepFilterNet: A Low Complexity Speech Enhancement Framework for Full-Band Audio. GitHub, 2025. Disponível em: https://github.com/Rikorose/DeepFilterNet.
[6] PYTORCH. Audio Resampling Tutorial. PyTorch Documentation, 2025. Disponível em: https://docs.pytorch.org/audio/stable/tutorials/audio_resampling_tutorial.html.
[7] OPENAI. Introducing Whisper. OpenAI Blog, 2022. Disponível em: https://openai.com/pt-BR/index/whisper/.
[8] YASH. Decoder-Only Transformers Explained: The Engine Behind LLMs. Medium, 2024. Disponível em: https://medium.com/@yash9439/decoder-only-transformers-explained-the-engine-behind-llms-3a3224086afe.
[9] IBM. O que são modelos codificadores-decodificadores? IBM Think, 2025. Disponível em: https://www.ibm.com/br-pt/think/topics/encoder-decoder-model.
[10] GOOGLE. Gemini 2.0 Flash Model Card. Google DeepMind, 2025. Disponível em: https://modelcards.withgoogle.com/assets/documents/gemini-2-flash.pdf.
[11] GOOGLE. Gemini 2.5: Thinking Model Updates. Google for Developers Blog, 2026. Disponível em: https://developers.googleblog.com/en/gemini-2-5-thinking-model-updates/.
[12] GOOGLE. Introducing Gemma 3n: Developer Guide. Google for Developers Blog, 2026. Disponível em: https://developers.googleblog.com/en/introducing-gemma-3n-developer-guide/.
[13] CHU, Yunfei et al. Qwen2-Audio Technical Report. Qwen Blog, 2024. Disponível em: https://qwenlm.github.io/blog/qwen2-audio/.
[14] LIU, Alexander H. et al. Voxtral. arXiv:2507.13264, 2025. Disponível em: https://arxiv.org/abs/2507.13264.
[15] XU, Hainan; KOLUGURI, Nithin Rao; MAJUMDAR, Somshubra. Turbocharge ASR Accuracy and Speed with NVIDIA NeMo Parakeet-TDT. NVIDIA Technical Blog, 2024. Disponível em: https://developer.nvidia.com/blog/turbocharge-asr-accuracy-and-speed-with-nvidia-nemo-parakeet-tdt/.
[16] GOOGLE. Google Cloud Pricing Calculator. Google Cloud, 2026. Disponível em: https://cloud.google.com/products/calculator.
[17] GOOGLE. Preços da Vertex AI. Google Cloud, 2026. Disponível em: https://cloud.google.com/vertex-ai/pricing.
[18] MICROSOFT. Azure OpenAI Service pricing. Microsoft Azure, 2026. Disponível em: https://azure.microsoft.com/en-us/pricing/details/azure-openai/.
[19] GOOGLE. Preços da API Gemini. Google AI for Developers, 2026. Disponível em: https://ai.google.dev/gemini-api/docs/pricing?hl=pt-br.
[20] SNAKERS4. Silero VAD: Voice Activity Detection. PyTorch Hub, 2026. Disponível em: https://pytorch.org/hub/snakers4_silero-vad_vad/.
[21] GOOGLE. Tokens. Google AI for Developers, 2026. Disponível em: https://ai.google.dev/gemini-api/docs/tokens?hl=pt-br&lang=python.