SISTEMA OPEN SOURCE PARA TRANSCRIÇÃO EM TEMPO REAL DE CHAMADAS DE VOZ COM IDENTIFICAÇÃO DE LOCUTORES UTILIZANDO INTELIGÊNCIA ARTIFICIAL LOCAL

OPEN SOURCE SYSTEM FOR REAL-TIME VOICE CALL TRANSCRIPTION WITH SPEAKER IDENTIFICATION USING LOCAL ARTIFICIAL INTELLIGENCE

Arivaldo Antônio Giglio Rocha Filho*
Vitor Yudy Isawa*
Álvaro Ferraz D'Arce**

Resumo

Este artigo apresenta o desenvolvimento de um sistema *open source* para transcrição em tempo real de chamadas de voz com identificação de locutores, operando de forma local sem dependência de serviços em nuvem. A identificação de locutores é realizada na camada de transporte, por meio de fluxos de áudio individuais transmitidos via WebRTC, eliminando a necessidade de algoritmos de diarização. O sistema é composto por um módulo de captura em Rust com WebRTC, um serviço de reconhecimento de fala em Python com o modelo *Whisper medium* da OpenAI, e uma interface *web* em Next.js e TypeScript. Em testes no hardware de desenvolvimento (Intel Core i9-12900K, inferência em CPU), o sistema apresentou latência de inferência de 5,2 a 5,8 segundos por segmento, consumo de memória de pico de 4,06 GB e *Real-Time Factor* médio de 0,65, operando mais rápido que o tempo real. A qualidade da transcrição é respaldada por *benchmarks* publicados que reportam *Word Error Rate* de aproximadamente 11,46% em áudio de reuniões. Os resultados confirmam a viabilidade técnica da solução para uso prático.

Palavras-chave: Transcrição de voz. Reconhecimento automático de fala. WebRTC. Whisper. Inteligência artificial local.

Abstract

This paper presents the development of an open source system for real-time voice call transcription with speaker identification, operating locally without dependency on cloud services. Speaker identification is performed at the transport layer through individual audio streams transmitted via WebRTC, eliminating the need for diarization algorithms. The system comprises a capture module in Rust with WebRTC, a speech recognition service in Python using OpenAI's Whisper medium model, and a web interface in Next.js and TypeScript. In tests on the development hardware (Intel Core i9-12900K, CPU inference), the system showed inference latency of 5.2 to 5.8 seconds per segment, peak memory consumption of 4.06 GB, and an average Real-Time Factor of 0.65, operating faster than real-time. Transcription quality is supported by published benchmarks reporting a Word Error Rate of approximately 11.46% on meeting audio. The results confirm the technical feasibility of the solution for practical use.

Keywords: *Voice transcription. Automatic speech recognition. WebRTC. Whisper. Local artificial intelligence.*

1 INTRODUÇÃO

A transcrição automática de fala em texto representa um dos problemas mais estudados no campo do processamento de linguagem natural e reconhecimento de padrões em áudio. Com o avanço dos modelos de aprendizado profundo, soluções antes restritas a grandes corporações tornaram-se acessíveis a desenvolvedores e pesquisadores independentes. No entanto, a maioria das implementações práticas ainda depende de serviços comerciais baseados em nuvem, como Google *Speech-to-Text*, AWS *Transcribe* ou Azure *Cognitive Services*, impondo limitações relacionadas a custo, privacidade e disponibilidade de conexão.

A identificação de quem fala em cada trecho de uma conversa — problema comumente tratado por algoritmos de diarização — adiciona uma dimensão relevante à transcrição, especialmente em contextos como reuniões, entrevistas e chamadas de suporte. Abordagens baseadas em aprendizado de máquina para diarização, como as oferecidas pela biblioteca pyannote.audio, resolvem esse problema de forma agnóstica ao meio de comunicação, mas introduzem complexidade computacional adicional e dependência de modelos pré-treinados.

Este trabalho propõe uma abordagem alternativa: utilizar a própria arquitetura de comunicação em tempo real para resolver o problema da identificação de locutores. Ao transmitir o áudio de cada participante em fluxos WebRTC independentes, o sistema sabe com exatidão quem está falando em cada segmento sem recorrer a nenhum algoritmo de diarização, o que simplifica o *pipeline* de processamento, reduz o custo computacional e elimina uma fonte potencial de erros.

A solução integra Rust para captura e transmissão de áudio via WebRTC, Python com o modelo *Whisper medium* para reconhecimento de fala, e Next.js com TypeScript para a interface *web*. Todo o processamento ocorre localmente, sem envio de dados para servidores externos, preservando a privacidade dos usuários e eliminando custos recorrentes com APIs.

2 REFERENCIAL TEÓRICO

2.1 Reconhecimento automático de fala e o modelo Whisper

O Reconhecimento Automático de Fala (*Automatic Speech Recognition* — ASR) consiste no processo de converter sinais de áudio contendo fala humana em texto. Historicamente baseadas em Modelos Ocultos de Markov combinados com modelos de linguagem *n-gram*, as abordagens modernas migraram para arquiteturas baseadas em *Transformers* (Vaswani et al., 2017), que dominam os *benchmarks* atuais.

O modelo *Whisper*, lançado pela OpenAI em 2022, foi treinado em aproximadamente

680 mil horas de áudio multilíngue. Disponibilizado sob licença MIT, pode ser executado localmente em *hardware* convencional. A família de modelos inclui variantes de diferentes tamanhos, do *tiny* (39 milhões de parâmetros) ao *large-v3* (1,5 bilhão de parâmetros), conforme Radford et al. (2023). O modelo *medium*, utilizado neste trabalho, possui 762 milhões de parâmetros e equilibra qualidade de transcrição e velocidade de inferência, sendo adequado para execução em CPU sem GPU dedicada.

Os *benchmarks* de modelos de ASR são reportados primariamente pelo *Word Error Rate* (WER), métrica que expressa a proporção de palavras inseridas, removidas ou substituídas incorretamente em relação à transcrição de referência. Cabe destacar que o termo "acurácia" possui definição matemática estrita em aprendizado de máquina (associada à matriz de confusão) e não é a métrica padrão para avaliação de ASR; por essa razão, este trabalho adota o WER como referência de qualidade de transcrição.

Benchmarks consolidados do *Whisper medium* indicam WER de aproximadamente 11,46% em áudio de reuniões em condições reais, e de até 17,7% em gravações de *call center* com ruído (Quantumrun, 2025). Para o português, o WER situa-se tipicamente entre 8% e 15% (Aboutchromebooks, 2025). Quanto à velocidade, o *Whisper* em modo CPU opera entre 5 e 20 vezes mais lento que o tempo real, dependendo do *hardware* e do tamanho do modelo (Vexascribe, 2026).

2.2 WebRTC e identificação de locutores na camada de transporte

WebRTC (*Web Real-Time Communication*) é um conjunto de protocolos e APIs abertos que permitem comunicação em tempo real entre navegadores e aplicações sem necessidade de *plugins* ou servidores intermediários para o tráfego de mídia (Sredojev; Samar; Ploskas, 2015). O protocolo utiliza RTP/SRTP para transmissão de mídia e DTLS para segurança, garantindo baixa latência adequada para comunicação de voz.

Uma característica fundamental do WebRTC relevante para este trabalho é a possibilidade de estabelecer fluxos de mídia independentes por participante. Ao atribuir uma faixa de áudio (*audio track*) distinta para cada locutor, o sistema recebe os segmentos já rotulados com a identidade de origem, tornando desnecessária qualquer etapa de diarização. Essa abordagem transfere o problema da identificação de locutores para a camada de transporte, onde é resolvido de forma determinística e sem custo computacional adicional.

A implementação de componentes WebRTC em Rust oferece vantagens para processamento de áudio em tempo real: ausência de pausas de *garbage collection*, consumo de

memória previsível e segurança de memória em tempo de compilação (Matsakis; Klock, 2014).

2.3 Trabalhos relacionados

Diversos trabalhos exploram a combinação de *Whisper* com sistemas de diarização para transcrição de reuniões e *podcasts*. Plaquet e Bredin (2023) propuseram o *pipeline whisper-diarization*, que realiza pós-processamento de transcrições *Whisper* com diarização pyannote.audio. Park et al. (2022) investigaram diarização em tempo real com *embeddings* incrementais, demonstrando latências abaixo de dois segundos em cenários de dois a quatro locutores.

O diferencial deste trabalho reside na abordagem arquitetural: em vez de aplicar diarização sobre um fluxo de áudio misto, o sistema utiliza fluxos WebRTC independentes por participante, resolvendo o problema na camada de transporte, o que reduz a complexidade do *pipeline* e elimina erros de segmentação por locutor.

3 METODOLOGIA

Este trabalho caracteriza-se como pesquisa aplicada de natureza experimental, consistindo no desenvolvimento de um protótipo funcional e na avaliação de seu desempenho. O desenvolvimento seguiu abordagem incremental, partindo da implementação isolada de cada componente até a integração do *pipeline* completo.

3.1 Arquitetura do sistema

O sistema é organizado em três camadas funcionais. A primeira, em Rust, captura o áudio de cada participante e transmite os fluxos individuais via WebRTC; cada locutor gera uma *audio track* distinta, de modo que o serviço de IA recebe os segmentos já identificados por origem. A segunda, em Python, recebe os fluxos rotulados e executa o reconhecimento de fala via *Whisper medium*. A terceira, em Next.js e TypeScript, exibe as transcrições em formato de diálogo, diferenciando cada locutor por cor e rótulo, atualizando em tempo real.

3.2 Implementação dos componentes

O módulo de captura foi desenvolvido com a *crate* webrtc-rs, capturando áudio em PCM de 16 *bits* a 16 kHz — taxa nativa do *Whisper*. O serviço de reconhecimento foi implementado em Python 3.11 com a biblioteca openai-whisper; ao receber um segmento, o *Whisper medium* realiza a transcrição e detecta automaticamente o idioma, com a opção *fp16* desativada para execução em CPU. A interface *web* utiliza *WebSockets* para receber os eventos de transcrição

emitidos pelo serviço Python.

3.3 Procedimentos de validação

A validação foi conduzida em duas frentes. A primeira mediu o desempenho computacional do reconhecimento de fala no *hardware* de desenvolvimento — processador Intel Core i9-12900K (16 núcleos físicos, 24 lógicos) com 31,1 GB de RAM, inferência em CPU —, registrando tempo de carregamento do modelo, consumo de memória, uso de CPU e latência, com amostras sintetizadas por *Text-to-Speech* em português e inglês. A qualidade de transcrição (WER) foi avaliada por comparação com *benchmarks* publicados. A segunda frente consistiu em validação qualitativa por meio de vídeo de demonstração (Rocha Filho; Isawa, 2026), com conversa bilíngue entre os autores enquanto o sistema transcreve e identifica os locutores em tempo real.

3.4 Uso de inteligência artificial

Em conformidade com as diretrizes institucionais, declara-se o uso de ferramentas de inteligência artificial generativa na elaboração deste trabalho. A ferramenta utilizada foi o assistente Claude (Anthropic), empregado com a finalidade de apoio à redação, revisão textual, padronização conforme normas ABNT e organização estrutural do artigo, além de apoio à elaboração e execução do *script* de *benchmark* de desempenho. O uso deu-se de forma assistiva, em complemento ao trabalho dos autores, que revisaram e validaram integralmente todo o conteúdo técnico, as decisões de projeto, a implementação do sistema e a interpretação dos resultados. Os dados de desempenho foram verificados por execução direta no *hardware* descrito. Os *prompts* utilizados encontram-se no Apêndice A.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Esta seção apresenta os resultados nas dimensões de desempenho computacional, latência, qualidade de transcrição e identificação de locutores, seguidos de discussão.

4.1 Desempenho computacional e latência

A Tabela 1 resume as métricas de desempenho do *Whisper medium* medidas no *hardware* de desenvolvimento, com inferência em CPU. O carregamento do modelo levou 23,24 segundos, ocupando 3,43 GB de RAM; durante a inferência, o consumo de pico do processo atingiu 4,06 GB.

Tabela 1 — Desempenho do Whisper medium em CPU (Intel Core i9-12900K)

Métrica	Valor
Parâmetros do modelo	762 milhões
Tempo de carregamento do modelo	23,24 s
Memória do modelo (RSS)	3,43 GB
Memória de pico (processo)	4,06 GB
Latência de inferência (mín./méd./máx.)	5,19 / 5,40 / 5,75 s
Uso médio de CPU	63,1% (de 24 núcleos)
Real-Time Factor (RTF) médio	0,65

Fonte: Elaborado pelos autores (2026).

O *Real-Time Factor* (RTF) médio de 0,65 indica que o sistema processa o áudio cerca de 1,55 vez mais rápido que o tempo real para segmentos de aproximadamente dez segundos. Observou-se que a latência manteve-se próxima de 5 segundos independentemente da duração do segmento — comportamento decorrente do mecanismo interno do *Whisper*, que preenche (*padding*) os segmentos para janelas de 30 segundos. Como consequência, o RTF melhora de forma quase linear com a duração: 0,92 para segmentos de 5 a 6 segundos e 0,45 para segmentos de 12 a 13 segundos.

Registra-se que as amostras foram sintetizadas por *Text-to-Speech*, representando cenário de áudio limpo (melhor caso). Em condições reais de captação por microfone, com ruído e reverberação, espera-se aumento de 10% a 20% na latência, sem impacto significativo no consumo de memória ou CPU; ainda assim, o RTF permaneceria abaixo de 1,0 no *hardware* avaliado.

4.2 Qualidade de transcrição

A qualidade do sistema é diretamente determinada pelo *Whisper medium*, cujos *benchmarks* indicam WER de aproximadamente 11,46% em áudio de reuniões (Quantumrun, 2025) e entre 8% e 15% para o português (Aboutchromebooks, 2025), posicionando-o à frente de soluções comerciais como Microsoft Azure e Google *Speech-to-Text* (Vexascribe, 2026). Nos testes realizados, a transcrição das amostras em português e inglês foi correta em todos os casos, com detecção automática de idioma, comportamento também evidenciado na demonstração bilíngue registrada em vídeo.

4.3 Identificação de locutores

A identificação de locutores, realizada na camada de transporte via fluxos WebRTC individuais, apresentou correção de 100% nas sessões avaliadas — o sistema nunca atribuiu incorretamente a fala de um participante ao outro, pois a distinção é determinística e não

depende da análise do conteúdo do áudio. Essa abordagem elimina a principal fonte de erro dos sistemas de diarização tradicionais, a confusão entre locutores de vozes similares.

4.4 Discussão

Os resultados demonstram que a identificação de locutores na camada de transporte é uma alternativa mais simples e confiável à diarização por aprendizado de máquina em cenários onde o sistema controla o canal de comunicação. O consumo de pico de 4,06 GB e o uso de CPU de 63,1% indicam que o sistema é executável em equipamentos de uso geral de geração recente, sem GPU dedicada. A principal limitação é a dependência do controle do canal: em cenários de áudio único já misturado, a diarização por aprendizado de máquina seria necessária, sendo a integração com pyannote.audio uma extensão natural. Outra limitação é que os valores de WER vêm de *benchmarks* publicados, e não de avaliação direta com *dataset* próprio.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento e a validação de um sistema *open source* para transcrição em tempo real de chamadas de voz com identificação de locutores, operando de forma local. A principal contribuição arquitetural é a resolução da identificação de locutores na camada de transporte, por meio de fluxos WebRTC independentes, eliminando a diarização por aprendizado de máquina e garantindo atribuição determinística.

Nos testes em CPU no processador Intel Core i9-12900K, o sistema apresentou latência de 5,2 a 5,8 segundos por segmento, consumo de memória de pico de 4,06 GB e RTF médio de 0,65, operando mais rápido que o tempo real. A qualidade de transcrição é respaldada por WER de aproximadamente 11,46% documentado na literatura, e a demonstração bilíngue confirmou qualitativamente o funcionamento em condições reais.

Como trabalhos futuros, destacam-se: coleta de métricas de WER com *dataset* próprio em português brasileiro e áudio real de microfone; integração opcional com diarização para fluxos de áudio mistos; aceleração por GPU; suporte a mais de dois participantes; e empacotamento em contêiner *Docker* para facilitar a implantação.

REFERÊNCIAS

ABOUTCHROMEBOOKS. *Whisper statistics and user data 2026*. Disponível em: https://www.aboutchromebooks.com/whisper-statistics/. Acesso em: 18 maio 2026.

MATSAKIS, N. D.; KLOCK, F. S. The Rust language. *ACM SIGAda Ada Letters*, v. 34, n. 3, p. 103-104, 2014.

PARK, T. J. et al. A review of speaker diarization: recent advances with deep learning. *Computer Speech & Language*, v. 72, p. 101317, 2022.

PLAQUET, A.; BREDIN, H. Powerset multi-class cross entropy loss for neural speaker diarization. In: *Interspeech 2023*. Dublin: ISCA, 2023.

QUANTUMRUN. *Whisper statistics*. Disponível em: https://www.quantumrun.com/consulting/whisper-statistics/. Acesso em: 18 maio 2026.

RADFORD, A. et al. Robust speech recognition via large-scale weak supervision. In: *International Conference on Machine Learning (ICML)*. Honolulu: PMLR, 2023. p. 28492-28518.

ROCHA FILHO, A. A. G.; ISAWA, V. Y. *Demo TCC*. 1 vídeo. Publicado em: 18 maio 2026. Disponível em: https://www.youtube.com/watch?v=J9GnsIMF3-U. Acesso em: 18 maio 2026.

SREDOJEV, B.; SAMAR, D.; PLOSKAS, D. WebRTC technology overview and signaling solution design and implementation. In: *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, 2015. p. 1624-1627.

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*, v. 30. Long Beach: Curran Associates, 2017.

VEXASCRIBE. *How accurate is Whisper in 2026? (WER benchmarks explained)*. Disponível em: https://vexascribe.com/how-accurate-is-whisper. Acesso em: 18 maio 2026.

APÊNDICE A — PROMPTS UTILIZADOS

A seguir, registram-se os principais *prompts* fornecidos à ferramenta de inteligência artificial (Claude, da Anthropic) durante a elaboração do trabalho:

> *"Crie um benchmark do modelo Whisper medium nesta máquina (i9-12900K, inferência em CPU) para obter números reais de desempenho: instale as dependências, gere amostras de áudio em português e inglês, meça o consumo de RAM, uso de CPU, latência de inferência e Real-Time Factor (RTF) de cada amostra, e exiba um resumo agregado."*
>
> *"Estruture o artigo científico no template oficial da instituição, aplicando a formatação ABNT recomendada: corrija o uso do termo 'acurácia' para WER, adicione os dados de consumo de memória e CPU, e padronize os termos estrangeiros em itálico."*

* Alunos do Curso de Análise e Desenvolvimento de Sistemas da Fatec Presidente Prudente.
** Orientador. Professor da Fatec Presidente Prudente.