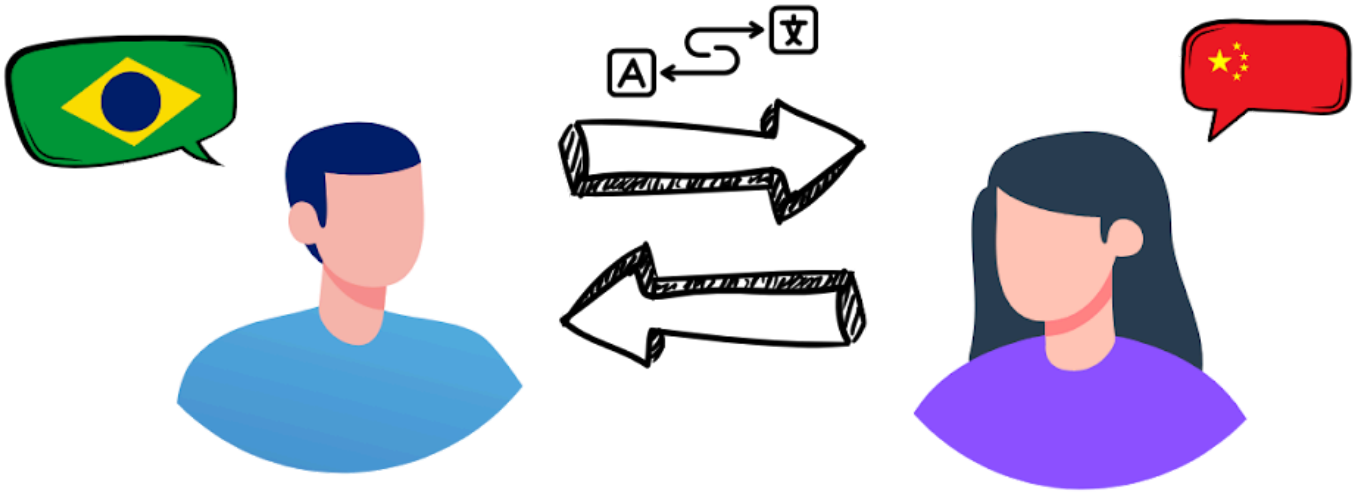
UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural
na Comunicação Multilíngue em Tempo Real

Rafaela Mota Silva



UFG
UNIVERSIDADE
FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

RAFAELA MOTA SILVA

Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): RAFAELA MOTA SILVA

Título do trabalho: Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Rafaela Mota Silva**, **Discente**, em 14/01/2025, às 21:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 15/01/2025, às 16:27, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5089805** e o código CRC **CF8F6D27**.

Referência: Processo nº 23070.001599/2025-37 SEI nº 5089805

RAFAELA MOTA SILVA

Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.
Orientador: Prof. Dr. Fernando Marques Federson

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

SILVA, RAFAELA MOTA
Tradução Simultânea de Fala [manuscrito] : Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real / RAFAELA MOTA SILVA. - 2025.
101 f.

Orientador: Prof. Dr. Fernando Marques Federson.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. inteligência artificial. 2. processamento de linguagem natural. 3. tradução em tempo real. I. Federson, Fernando Marques , orient. II. Título.

CDU 004

RAFAELA MOTA SILVA

Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 17 de dezembro de 2024.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Profa. Dra. Telma Woerle de Lima Soares
(INF-UFG)

RAFAELA MOTA SILVA

Tradução Simultânea de Fala

Exploração do Processamento de Áudio e da Linguagem Natural na Comunicação Multilíngue em Tempo Real

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Tradução Simultânea de Fala**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: inteligência artificial, modelos grandes de linguagem, geração automática de datasets.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Simultaneous Speech Translation**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: artificial intelligence, large language models, automatic dataset generation.

Goiânia

2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Minha Jornada

Rafaela Mota Silva

Especialista em: Tradução Simultânea de Fala

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

MINHA JORNADA

Nome: **Rafaela Mota Silva**

Especialidade: **Tradução Simultânea de Fala**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha Jornada teve início antes mesmo da **Semana 1**, quando foi recomendada a análise de uma vasta lista de tópicos que, de alguma maneira ou outra, estão relacionados à Inteligência Artificial para que pudéssemos definir uma área específica para dar continuidade à residência. Após destacar os tópicos e ponderar sobre a escolha de com qual seguir, foi decidido que o campo de aplicação seria na parte de Jogos. Diante disso, na **Semana 1** foram analisados dois livros (AI for Games e Artificial Intelligence and Games) e um congresso (SBGames) que estão disponíveis no **Apêndice 1**, todos eles detalhando as mais variadas aplicações de diferentes áreas de IA no campo de jogos. Em decorrência disso, a área escolhida foi **Tradução Simultânea de Fala** (NLP e PAV) aplicada no contexto de jogos multiplayer.

Nas **Semanas 2 e 3**, foi feita uma busca por artigos nas áreas de processamento de linguagem natural, processamento de áudio e voz e sobre como criar um sistema de tradução simultânea, com um destaque especial ao artigo **Attention Is All You Need**. Nesses materiais, foi possível definir algumas ferramentas clássicas e outras essenciais

[1] Dez semanas, entre setembro de 2024 e dezembro de 2024.

para a continuidade do trabalho (incluir as principais). Além disso, a busca por implementações atuais de tradução simultânea de fala foi iniciada.

Entendendo a importância de transformers e suas adaptações para diversos domínios de IA, foi possível começar a selecionar alguns modelos que pudessem performar bem na tarefa definida. Foram analisadas quatro implementações diferentes durante a **Semana 4**, todas elas utilizavam um modelo de transcrição baseado em transformers, o **Whisper**, e sua variação mais rápida, o **Faster Whisper**. Após uma análise de vantagens e desvantagens de ambos os modelos, presente no **Apêndice 3**, foi proposto o teste prático de todas as implementações selecionadas, com um destaque especial para o **RealtimeSTT**.

Após testes, foi estabelecido, na **Semana 5,** o uso do **RealtmeSTT** devido à sua versatilidade e capacidade de adaptação para o problema em questão. A partir disso, alguns testes envolvendo diferentes tipos de tamanhos de modelos e diferentes abordagens foram feitas, levando em consideração a qualidade das traduções e o tempo de inferência, todas essas informações estão presentes no **Apêndice 4**. As duas vertentes de tradução usadas foram o uso da função ‘translate’ da biblioteca **RealtimeSTT** para realizar a tradução, e outra utilizando um modelo de tradução (**M2M-100**). Contudo, ainda havia espaço para melhora e testes com outras ferramentas e, por isso, algumas APIs foram usadas para a tarefa de tradução durante a **Semana 6**. Uma lista de APIs que podem ser usadas para tradução foi feita e está disponível no **Apêndice 4**, mas somente três foram selecionadas: **Google Cloud Translate, DeepL e OpenAI API (ChatGPT)**. Além disso, a abordagem com o modelo de tradução **M2M-100** foi otimizada, reduzindo o tempo de inferência de 6 a 18 segundos para 0.24 a 1.93 segundos. A principal diferença visualizada é a capacidade de adaptação por meio de APIs como a da **OpenAI**, devido à possibilidade de moldar os prompts para de adequarem ao contexto na qual a pessoa está inserida. Para parâmetros de comparação, foi decidido continuar o processo utilizando as duas abordagens: com APIs e com o modelo de tradução. O próximo passo foi tornar a tradução mais fluida pois, até o presente momento, o intervalo de captura de um segmento de áudio para o outro era relativamente longo, resultando em perda de informação e um retorno tardio da tradução da fala para a pessoa, o que foi devidamente feito na **Semana 7**. As mudanças feitas não atingiram as expectativas iniciais, se mostrando um pouco mais complexas do que o imaginado inicialmente. Uma

alteração que realmente impactou na fluidez foi a redução do tempo de pausa detectada entre uma frase e outra para segmentar o áudio capturado em tempo real. Também foi proposta uma alternativa para adaptar o modelo de tradução para contextos específicos utilizando o **M2M-100**, mas que não foi testado, pois na **Semana 8** o modelo de tradução foi alterado para o **NLLB-200**. Além de ser uma alternativa mais viável para tradução para idiomas de baixo e médios recursos, ainda abrange 100 idiomas a mais que o **M2M-100**. O tempo de inferência também não sofreu alterações, sendo mantido praticamente o mesmo. Durante essa mesma Semana, foram feitos alguns testes com a abordagem do modelo **NLLB-200** para testar seu desempenho com algumas frases de diferentes categorias de jogos e os resultados podem ser encontrados no **Apêndice 4**.

Por fim, com os modelos de transcrição, tradução e APIs definidos, foi dado início à fase de implementação de um sistema de tradução no contexto de jogos multiplayer nas **Semanas 9 e 10**. Três propostas diferentes foram apresentadas na **Semana 8**, cada uma possuindo seus respectivos protótipos, que estão disponíveis no **Apêndice 5**, e uma análise da viabilidade de cada um. É preciso ressaltar que a abordagem do Teste Pop-Up foi importante, pois utiliza uma plataforma de comunicação amplamente utilizada na área de jogos, o **Discord**. Dessa maneira, é possível implementar um sistema que roda localmente e é transmitido pelo usuário para as pessoas com as quais ele está se comunicando. Como resultado final, foi produzida uma interface interativa, na qual é possível selecionar a API ou modelo de tradução a ser usado (**DeepL, OpenAI API, M2M-100** ou **NLLB-200**) e os idiomas de entrada e de saída (para a demonstração foram selecionados seis idiomas, incluindo dois com um alfabeto completamente diferente do ocidental para testar a qualidade da tradução). Um detalhe importante é que, apesar de não implementado, é de suma importância selecionar corretamente o dispositivo de entrada e, mesmo que não tenha sido adicionado à interface, há um código disponibilizado no **Apêndice 5** que permite identificar o dispositivo, assim como seu respectivo índice. Como resultado final, temos o código do protótipo do sistema de tradução de fala simultânea, assim como uma lista com os requisitos para executá-lo localmente.

Em função de tudo que vivi nesta Jornada, gostaria de deixar registrado que todo o trabalho até o presente momento permitiu a implementação de um sistema de tradução que

pode ser adaptado para qualquer contexto que fosse necessário. O resultado final demonstra, em parte, o que pode ser feito utilizando as técnicas e ferramentas certas para um determinado objetivo. Apesar da Jornada ter sido encerrada na Residência, o que foi aprendido durante este processo foi de suma importância para o desenvolvimento da minha trajetória profissional e pessoal. Cada desafio enfrentado, cada solução encontrada e cada conhecimento adquirido contribuíram para expandir minha compreensão sobre o potencial das tecnologias de tradução em tempo real e sobre a importância da inovação contínua.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 19 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva e Lucas Brandão Rodrigues

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante esta etapa consistiram em:

- **Busca e análise de áreas/campos de aplicações para a residência:**
  - Tópicos para a Residência ;
- **Definição do campo de aplicação para a residência:**
  - IA e Jogos;
- **Busca por referências bibliográficas no campo de aplicação escolhido:**
  - Book 1 - AI for Games (Ian Millington);
  - Book 2 - Artificial Intelligence and Games (Georgios N. Yannakakis e Julian Togelius);s
  - Congress 1 - SBGames .
- **Análise do material encontrado para definir o tema de especialização:**
  - Stage 1 - 190924 ;
- **Definição do tema:**
  - Tradução Simultânea de Fala em Jogos Multiplayer.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Buscar trabalhos referentes às áreas de Processamento de Linguagem Natural, Processamento de Áudio e Voz (especificamente sobre Reconhecimento Automático de Fala) e Redes Neurais Profundas aplicadas no contexto da tarefa designada (tradução simultânea de fala);

- Analisar e resumir os trabalhos encontrados.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Algumas partes do trabalho foram realizadas em conjunto com o aluno Lucas Brandão. Para haver distinção entre as alterações feitas por ambos, decidimos atribuir uma cor para cada um, facilitando a visualização. Dessa forma, as cores para destacar decididas individualmente foram:

Lucas
Rafaela

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Início

Durante a Semana 0, foi proposta a análise de uma ampla lista de tópicos relacionados à Inteligência Artificial, com o objetivo de identificar uma área específica para guiar a residência. Após revisar e refletir sobre as opções, decidi que o campo de aplicação seria a área de Jogos. Lucas Brandão, outro estudante da residência, também escolheu este campo, e, por isso, decidimos colaborar em um trabalho conjunto até que nossos objetivos divergissem.

Contudo, o objetivo era definir uma área de conhecimento para a residência, não apenas um campo de aplicação. Por isso, durante a Semana 1, selecionamos livros como *AI for Games* e *Artificial Intelligence and Games* para explorar os diferentes usos de IA no campo de jogos. Além disso, também analisamos as trilhas do SBGames e destacamos os tópicos que mais chamaram nossa atenção.

Tópicos de Interesse

Os tópicos listados provenientes das conferências internacionais nortearam a escolha do campo de aplicação (Jogos), enquanto as trilhas do Simpósio Brasilero de Jogos e Entretenimento Digital (SBGames) e dos livros ajudaram na decisão da área de conhecimento e atuação da residência.

The 8th International Conference on Applied Cognitive Computing (ACC'24)

- Application of chaos Engineering in machine intelligence
- Dynamical learning systems
- Self-Adaptive and Self Organizing Systems

The 25th International Conference on Bioinformatics & Computational Biology (BIOCOMP'24)

- Molecular dynamics and simulation; Molecular interactions
- Molecular sequence classification, alignment and assembly
- Molecular sequence and structure databases

The 10th International Conference on Biomedical Engineering & Sciences (BIOENG'24)

- Virtual Reality

The 22nd International Conference on Scientific Computing (CSC'24)

- Chaos modeling

The 23rd International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government (EEE'24)

- Methods and tools for e-Government
- Designing web services for e-Government
- Inter-administration and G2G issues

The 20th International Conference on Foundations of Computer Science (FCS'24)

- Game Theory and Methods

The 3rd International Conference on Emergent Quantum Technologies (ICEQT'24)

- AI for Quantum Computing

The 25th International Conference on Internet Computing & IoT (ICOMP'24)

- Augmented reality games
- Computer games and education
- Artificial intelligence and computer games
- Audio-video communication tools for network 3D games

Livro 1: AI For Games

- Speed and Memory
- Movement
    - Movement in the Third Dimension
- Action Execution
- Decision Learning
- Neural Networks

- World Interfacing
    - Getting Knowledge Efficiently
- Real-Time Strategy
    - Decision Making
- Representing Actions
- Representing the World

Livro 2: Artificial Intelligence and Games

- Player Experience and Behavioral Data Analytics
- Evolutionary Algorithms
- Artificial Neural Networks
- Neuroevolution
- Reinforcement Learning

SBGames

- Design de jogo
- Tecnologias em jogos
- Computer Games
- AI for Games
- Game Technology
- Gamification
- Comunicação, marketing e mediação cultural com jogos
- Culturas de live streaming, machinima, realidade aumentada e virtual
- Aprendizagem baseada em jogos
- Métricas na Indústria de Jogos Digitais

Conclusão

Após essa busca inicial, algumas áreas se destacaram como interesses em potencial. Decidimos como foco da nossa pesquisa:

- Lucas: **“Aprendizado por Reforço Aplicado a Games”**
- Rafaela: **“Tradução Simultânea Usando Modelos de Linguagem Natural e Processamento de Áudio e Voz”**

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 25 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante esta etapa consistiram em:

- **Busca por materiais de apoio referentes às áreas de Processamento de Linguagem Natural e Processamento de Áudio e Voz:**
  - Natural language processing state of the art, current trends and challenges.pdf ;
  - Automatic Speech Recognition: Systematic Literature Review.pdf ;
  - A Review on Automatic Speech Recognition Architecture and Approaches.pdf ;
  - An Overview of End-to-End Automatic Speech Recognition.pdf ;
  - Fundamentals of Artificial Intelligence (2020).pdf ;
  - Deep Learning for NLP and Speech Recognition (2019).pdf .

- **Busca por materiais de apoio referentes à Tradução Simultânea de Fala:**
  - Real Time Direct Speech-to-Speech Translation.pdf ;
  - Classification techniques for automatic speech recognition (ASR) algorithms used wit…
    .

- **Análise do material selecionado:**
  - Stage 2 - 250924 .
    - Articles ;
    - Fundamentals of Artificial Intelligence (2020) ;
    - Deep Learning for NLP and Speech Recognition (2019) .

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Revisar os artigos selecionados durante esse estágio;
- Analisar e resumir sobre as ferramentas e métodos presentes nos artigos selecionados;
- Começar a pesquisar por implementações de tradução simultânea de fala, entender e documentar o processo particular de cada uma.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

LEONARDO ALVES: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 2 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 3 - 021024 consistiram em:

- **Revisão dos artigos selecionados:**
  - Além da revisão dos artigos anteriores, houve uma adição significativa aos materiais base, principalmente devido ao seu impacto, não somente na área de NLP, como também em Visão Computacional e Processamento de Áudio e Voz;
  - Attention Is All You Need (2017) (Transformer).

- **Análise e resumo de ferramentas, métodos e/ou conceitos presentes nos artigos selecionados:**
  - MFCC, HMM, LSTM, DTW, RNNs e Redes Neurais Profundas foram aparições recorrentes;
  - Transformers começaram a ser usados para outras áreas além de NLP por volta de 2020 e foi citado somente em um dos artigos selecionados;
  - Alguns frameworks foram apresentados, mas não eram o foco para essa semana.

- **Início da busca por aplicações de tradução simultânea de fala:**
  - Realtime Speech to Speech Translation (Kenson Hui).

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Continuar a busca por implementações de sistemas de tradução simultânea de fala, entender e documentar o processo particular de cada uma;
- Analisar os frameworks recorrentes nas implementações e entender as vantagens e desvantagens de cada um;

- Estudar a arquitetura Transformer e quais são os modelos estabelecidos para tarefas específicas, especialmente tradução.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Pesquisa Bibliográfica

Após a definição do tema (Tradução Simultânea de Fala em Jogos Multiplayer), foi feita a busca por trabalhos nas áreas de Processamento de Linguagem Natural (NLP) e Reconhecimento Automático de Fala (ASR).

Um dos critérios de exclusão consistiu na data de publicação (qualquer um publicado antes de 2016 não foi incluído), justamente pelo fato de serem áreas com avanços significativos nos últimos anos/meses, com seus respectivos estados da arte sendo alterados frequentemente.

Um outro fator de escolha foi a capacidade de operar em tempo real. Apesar de não excluir aqueles que não possuíam essa característica, foram incluídos trabalhos com um foco neste problema específico para fornecer uma visão geral sobre as ferramentas e métodos utilizados. No total, foram selecionados sete artigos e dois livros, um de 2019 e outro de 2020, que, ao contrário dos livros mostrados anteriormente, possuem capítulos inteiros dedicados somente ao Processamento de Linguagem Natural (NLP) e Reconhecimento de Fala. O primeiro livro tem como foco os fundamentos da inteligência artificial. O segundo tem um foco maior especificamente na parte de NLP e Reconhecimento de Fala.

Artigos Selecionados

Classification techniques for automatic speech recognition (ASR) algorithms used with real time speech translation

Classification techniques for automatic speech recognition (ASR) algorithms used with …

Autores: Dr Hebah H. O. Nasereddin, Ayoub Abdel Rahman Omari

Tamanho: 8 páginas

Ano: 2017

Citações: 30

Resumo: O tema principal desta pesquisa foi medir a precisão da técnica de classificação do sistema ASR com o sistema base MT para tradução de fala em tempo real do inglês para árabe. Experimentos foram efetuados para cobrir vários ambientes e técnicas para calcular a precisão das sentenças reconhecidas e traduzidas em cada combinação. Nove combinações foram testadas nos experimentos conduzidos de extração de características usando MFCC com abordagens de classificação (ou seja, HMM, DTW e DBN) e abordagens de tradução MT (abordagem baseada em conhecimento, abordagem baseada em regras e abordagem híbrida).

Para cada combinação, foram usados cinquenta arquivos de áudio de fala que cobriram quatro categorias de tipo de fala (ou seja, 14 sentenças na categoria de notícias, 15 sentenças na categoria de conversação, 10 sentenças na categoria de frases científicas e 11 sentenças na categoria de controle).

Apesar de focar em alguns aspectos que não se aplicam ao tema da residência, as explicações matemáticas das técnicas de ASR fornecem uma visualização boa sobre o problema.

Natural language processing: state of the art, current trends and challenges

Natural language processing state of the art, current trends and challenges.pdf

Autores: Diksha Khurana, Aditya Koli, Kiran Khatter, Sukhdev Singh

Tamanho: 32 páginas

Ano: 2023

Citações: 1354

Resumo: O artigo aborda a evolução da área de NLP, descrevendo seus componentes principais, como a compreensão da linguagem natural (Natural Language Understanding - NLU) e a geração de linguagem natural (Natural Language Generation - NLG). No passado, ferramentas como o modelo baseado em gramática e a tradução automática (Machine Translation - MT) formavam a base dos sistemas NLP. Nos últimos anos, o uso de redes neurais, particularmente as redes neurais convolucionais (CNNs) e as redes neurais recorrentes (RNNs), transformou a forma como os modelos são treinados e aplicados, aumentando sua eficiência.

Os principais modelos atuais no campo do NLP incluem o BERT (Bidirectional Encoder Representations from Transformers), que revolucionou o campo com a capacidade de analisar texto de forma bidirecional, fornecendo um melhor entendimento contextual. Além disso, modelos como LSTM's (Long Short-Term Memory) e redes neurais transformadoras (transformers) desempenham um papel crucial na evolução do NLP. Redes neurais são amplamente utilizadas para tarefas como classificação de texto, sumarização, tradução de idiomas e análise de sentimentos.

Nesse contexto de NLP, temos o BERT como um dos modelos de ponta que oferece capacidades de processamento bidirecional para várias tarefas de NLP. Além disso, também existem RNN's e LSTM's, que são utilizados para dados sequenciais, como texto e fala, e retêm informações críticas por períodos mais longos. Finalmente, há CNN's, frequentemente usadas em tarefas de classificação de frases e análise de sentimentos.

O campo do NLP está atualmente sendo impulsionado por avanços em aprendizado profundo e big data, com Transformers se tornando o modelo dominante para várias tarefas devido ao seu desempenho exemplar, além de existirem ferramentas de Geração de Linguagem Natural, como o ChatGPT, para gerar textos que se assemelham à linguagem humana.

Entre os desafios mais comuns de NLP estão a ambiguidade da linguagem natural, a necessidade de grandes volumes de dados de treinamento e a dificuldade em lidar com contextos e variações linguísticas complexas. Além disso, a presença de ambiguidades

semânticas dificulta o desenvolvimento de sistemas de NLP que possam entender todas as nuances da linguagem humana.

No contexto do tema escolhido para a residência, não seria viável utilizar modelos com tempo de inferência alto, justamente por se tratar de uma aplicação em tempo real.

Real Time Direct Speech-to-Speech Translation

Real Time Direct Speech-to-Speech Translation.pdf

Autores: Sanchit Chaudhari, Aniket Shukla, Tanvi Gaware
Tamanho: 3 páginas
Ano: 2022
Citações: 0

Resumo: Logo no início, há uma ressalva sobre a importância da tradução de fala para a comunicação, principalmente tendo em mente que o mundo globalizado facilita o contato com pessoas de diferentes nacionalidades e culturas e a barreira da linguagem é um fator limitante para isso.

O foco principal está nas ferramentas usadas para permitir uma aplicação que realize essa tarefa de tradução de fala em tempo real. Bibliotecas Python e Spyder são utilizadas, juntamente com a API de tradução do Google.

Abordagens tradicionais desta tarefa demandam tempo e esforço para produzir os elementos necessários, que podem ser treinados e calibrados separadamente. Contudo, a abordagem de fala para fala tem a vantagem de usar um método com somente um passo, o que resulta em menos poder computacional demandado e um tempo de inferência melhor. O modelo do tradutor proposto utiliza 4 módulos, como representado na tabela abaixo.

Módulo	Descrição

Login	Recebe as credenciais do usuário.
Entrada (Input)	Recebe a entrada de áudio e envia para o sistema.
Tradução (Translation)	Traduz a entrada de voz para a saída desejada com o auxílio da API do Google.
Saída (Output)	Exibe o resultado final da tradução, seja em texto ou em áudio.

Há também uma visão geral do sistema, como descrito abaixo.

	Descrição
Sistema Operacional	Windows 10+
Python	Linguagem de programação de alto nível usada.
Spyder	Ambiente de desenvolvimento escrito em Python e para Python.
Anaconda Navigator	Pacote e gerenciador de ambiente usado para Python, permite que vários ambientes com suas respectivas especificações sejam utilizados separadamente e sem interferências.
OpenCV	Permite o uso de bibliotecas e funções de Visão Computacional em tempo real.
Pillow	Adiciona a funcionalidade de processamento de imagem ao interpretador Python.

Além disso, foram apresentadas diferentes técnicas utilizadas no processo de tradução, que podem ser testadas posteriormente, sendo elas o uso de Neural Machine

Translation, Tacotron, Speech Translation Without Transcription, Speech-to-Speech Translation for Non-transcribed Languages e Multilingual Corpus for Speech Translation.

Automatic speech recognition: Systematic Literature Review

Automatic Speech Recognition: Systematic Literature Review.pdf

Autores: Sadeen Alharbi, Muna Alrazgan, Alanoud Alrashed, Turkiayh Alnomasi, Raghad Almojel, Rimah Alharbi, Saja Alharbi, Sahar Alturki, Fatimah Alshehri, Maha Almojil
Tamanho: 19 páginas
Ano: 2021
Citações: 93

Resumo: Primeiramente, o trabalho realiza uma pesquisa geral com os termos de interesse em cinco bases de dados (IEEE, ACM, Scopus, Web of Science, Science Direct) e remove os que estavam duplicados. Posteriormente, alguns artigos foram removidos baseado em critérios de exclusão e, finalmente, foram removidos os artigos com uma pontuação menor que 3 na avaliação de qualidade. Com isso, o trabalho final foi embasado em 82 artigos cuidadosamente selecionados.

O artigo apresenta uma revisão sistemática sobre Reconhecimento Automático de Fala (Automatic Speech Recognition - ASR), destacando os avanços da área nos últimos anos, suas aplicações, desafios e lacunas de pesquisa. O ASR evoluiu de sistemas simples que reconheciam um número limitado de sons para sistemas sofisticados que compreendem fluentemente a linguagem natural. O estudo visa oferecer uma visão abrangente dos principais tópicos abordados na pesquisa de ASR entre 2015 e 2020.

O Reconhecimento Automático de Fala começou como sistemas limitados, mas com o avanço da inteligência artificial, especialmente de redes neurais, houve um avanço significativo. As principais áreas abordadas incluem o reconhecimento de ambientes

ruidosos, variações de fala e técnicas de adaptação. O uso de Redes Neurais Profundas e de modelos melhorou significativamente a precisão de sistemas de ASR.

As técnicas de ASR incluem o uso de modelos acústicos baseados em redes neurais, como CNN's e DNN's, para melhorar o desempenho do reconhecimento em ambientes ruidosos e com variações de fala, e de redes E2E (End-to-End), que incluem modelos baseados em CTC (Connectionist Temporal Classification) e transformers, que são capazes de superar limitações dos modelos tradicionais ao lidar com a sequência temporal de fala diretamente, sem precisar de segmentação explícita.

Algumas das ferramentas e métodos comumente mencionados na revisão são técnicas de redução de ruído, para aumentar a robustez do reconhecimento em ambientes ruidosos (DNN's e beamfoaming)

O artigo destaca o crescente interesse em ASR multimodal, onde a fusão de dados de áudio e vídeo pode melhorar a robustez do sistema. Além disso, técnicas para lidar com ambientes ruidosos e fala espontânea são focos de pesquisa recentes, como a utilização de modelos seq2seq e modelos de atenção. Outras áreas emergentes incluem o reconhecimento de fala em tempo real (com a adoção de transformers otimizados, como o Transformer-XL, para reduzir a latência em aplicações de tradução simultânea e assistentes virtuais).

Os principais desafios enfrentados pelos sistemas de ASR incluem ambientes ruidosos (o desempenho diminui em ambientes com muito ruído, o que afeta a qualidade do reconhecimento de fala), sobreposição de fala (o problema do "cocktail party", onde múltiplos falantes estão presentes simultaneamente) e a diversidade de sotaques e variações linguísticas (muitos sistemas são treinados com dados limitados de sotaques não nativos).

A Review on Automatic Speech Recognition Architecture and Approaches

A Review on Automatic Speech Recognition Architecture and Approaches.pdf

---

Autores: Karpagavalli S., Chandra E.

Tamanho: 12 páginas

Ano: 2016

Citações: 154

Resumo: O artigo revisa as principais arquiteturas e abordagens do Reconhecimento Automático de Fala (Automatic Speech Recognition - ASR), explorando sua evolução, componentes fundamentais e desafios. A revisão cobre métodos de extração de características, modelagem acústica, modelos de linguagem, algoritmos de decodificação e diferentes abordagens para reconhecimento de padrões, desde técnicas clássicas baseadas em fonética até o uso de aprendizado profundo. O estudo destaca a importância da robustez a ruídos e variações na fala, além de apontar ferramentas e bancos de dados amplamente utilizados para pesquisa em ASR. Contudo, é importante ressaltar que ele foi publicado em 2016 e, nestes oito anos, muitos avanços ocorreram e que mudaram o estado da arte em ASR.

O ASR evoluiu significativamente com o uso de Modelos Ocultos de Markov (HMMs) combinados com Redes Neurais Profundas (DNNs) e técnicas de aprendizado profundo. Os sistemas modernos dependem da extração de características como MFCCs (Mel-Frequency Cepstral Coefficients) e modelos de linguagem baseados em n-grams, além de arquiteturas híbridas, como HMM-DNN.

As metodologias predominantes incluem abordagem acústico-fonética, reconhecimento de padrões e aprendizado profundo.

O artigo detalha os principais métodos para extração de características, modelagem acústica e decodificação, que consistem em extração de características (utiliza técnicas como MFCC, análise de espectro e transformadas de Fourier para converter sinais de áudio em vetores de características), modelos acústicos (tradicionalmente baseados em HMMs, os modelos acústicos são treinados com grandes bases de dados de fala, utilizando fonemas como unidades básicas), modelos de linguagem (usados modelos de n-grams para prever sequências de palavras com base no contexto de fala. Ferramentas como o CMU Statistical

Language Modeling Toolkit e o Stanford Research Institute Language Modeling Toolkit são comumente usadas) e decodificação (algoritmos como Viterbi e beam search são empregados para encontrar as sequências de palavras mais prováveis com base nos modelos acústico e de linguagem).

Na época da publicação, o uso de redes neurais profundas (DNNs) e redes neurais convolucionais (CNNs) estava substituindo abordagens tradicionais, como GMM-HMM. A combinação de aprendizado supervisionado com redes profundas permitiu melhorias significativas no reconhecimento de fala em ambientes ruidosos.

An Overview of End-to-End Automatic Speech Recognition

An Overview of End-to-End Automatic Speech Recognition.pdf

Autores: Dong Wang, Xiaodong Wang, Shaohe Lv
Tamanho: 26 páginas
Ano: 2019
Citações: 276

Resumo: O artigo revisa os avanços e o estado da arte no Reconhecimento Automático de Fala (ASR) end-to-end, destacando a transição das abordagens tradicionais baseadas em Modelos Ocultos de Markov (HMM) e GMM para as arquiteturas mais modernas, baseadas em aprendizado profundo. O foco principal está nos modelos end-to-end, como aqueles baseados em Connectionist Temporal Classification (CTC), Recurrent Neural Network (RNN) transducer e nos mecanismos de atenção. Esses novos modelos apresentam uma série de vantagens sobre os antigos sistemas HMM-GMM, incluindo a simplificação da arquitetura, a eliminação de módulos intermediários e o mapeamento direto de sinais de áudio para a transcrição, o que resulta em treinamento conjunto e maior flexibilidade. O artigo explora detalhadamente o funcionamento dessas abordagens end-to-end, analisando suas vantagens e desvantagens, e também traça um panorama dos possíveis rumos que o ASR pode seguir no futuro.

Historicamente, o ASR evoluiu de uma arquitetura baseada em HMM-GMM para modelos híbridos que integram redes neurais profundas, como o HMM-DNN, e, mais recentemente, para os modelos end-to-end. Esses últimos trazem benefícios substanciais, como a eliminação da necessidade de alinhamento explícito dos dados e dos módulos intermediários que outrora eram essenciais, como os modelos de pronúncia e de linguagem. Com os modelos end-to-end, o processo de mapeamento de sinais de áudio para transcrições tornou-se mais direto, o que simplifica a arquitetura do sistema e melhora a performance geral.

Os três principais tipos de modelos end-to-end discutidos no artigo são os baseados em CTC, RNN-transducer e Attention. Os modelos CTC utilizam a classificação temporal connectionista para lidar com a segmentação temporal, gerando transcrições sem a necessidade de alinhamento explícito. Já o RNN-transducer integra uma rede de predição e uma de codificação, o que oferece maior flexibilidade no mapeamento entre entrada e saída, embora seja mais complexo e difícil de treinar. Por sua vez, os modelos baseados em atenção utilizam mecanismos que focam nas partes mais relevantes da entrada de áudio, proporcionando maior precisão e uma capacidade mais robusta de modelagem linguística, porém à custa de maior complexidade computacional e latência.

Apesar das melhorias proporcionadas pelos modelos end-to-end, alguns desafios permanecem. Modelos como o RNN-transducer e os baseados em atenção exigem maior cuidado no treinamento devido à necessidade de ajustar o alinhamento dos dados e garantir que a correspondência entre entrada e saída seja consistente. Enquanto isso, o modelo CTC, embora mais simples e eficiente em termos de complexidade computacional, enfrenta limitações devido à sua suposição de independência entre os labels, o que pode comprometer a precisão da transcrição final.

O artigo também compara o desempenho entre essas diferentes abordagens. O CTC, por exemplo, se destaca pela simplicidade no treinamento e pela eficiência computacional, mas sua incapacidade de modelar dependências linguísticas reduz sua precisão em algumas tarefas. O RNN-transducer, ao contrário, consegue modelar interdependências entre os elementos da saída, mas sua complexidade torna o treinamento mais desafiador. Os modelos baseados em atenção, embora sejam os mais precisos e

adequados para capturar nuances linguísticas, sofrem com problemas de latência e maior custo computacional. O estudo conclui que, embora o CTC continue sendo amplamente utilizado devido à sua simplicidade, os modelos baseados em atenção são mais promissores para tarefas de ASR mais complexas. A combinação de diferentes modelos, como a hibridização entre CTC e atenção, aparece como uma tendência emergente para superar as limitações individuais de cada abordagem e alcançar melhores resultados em sistemas de reconhecimento de fala.

Attention Is All You Need

Attention Is All You Need

Autores: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin
Tamanho: 15 páginas
Ano: 2017
Citações: 90.000+

Resumo: Este artigo apresenta o Transformer, uma arquitetura revolucionária para processamento de sequência que substitui mecanismos recorrentes e convolucionais pelo uso exclusivo de atenção. O modelo introduz o mecanismo de atenção "self-attention" e a atenção escalonada por múltiplas cabeças (multi-head attention), permitindo maior eficiência em paralelismo computacional e melhor captura de dependências de longo alcance. O estudo detalha como o Transformer supera métodos tradicionais em tarefas de tradução automática, utilizando pares de idiomas como base de treinamento.

Os experimentos incluem uma comparação com arquiteturas como LSTM e CNN, mostrando que o Transformer alcança melhores resultados em benchmarks como o WMT 2014 English-to-German e English-to-French, reduzindo significativamente o tempo de treinamento. O artigo enfatiza as contribuições da normalização de camadas, feedforward posicional, embeddings de posição e escalonamento do mecanismo de atenção. Apesar do

foco em tradução, as técnicas apresentadas têm aplicações amplas em NLP e outras áreas de aprendizado profundo.

Livros Selecionados

Fundamentals of Artificial Intelligence (2020)

O capítulo 19, "Natural Language Processing," apresenta uma visão abrangente sobre o processamento de linguagem natural (NLP), iniciando com uma introdução ao campo e suas aplicações práticas. Ele explora os componentes fundamentais do NLP, como análise sintática, semântica e discursiva, e discute a importância das gramáticas, incluindo estruturas frasais e a hierarquia de Chomsky. Técnicas de parsing, como parsing probabilístico e análise de construções em nível de sentença, são abordadas em seguida. O capítulo também examina a extração de informações, destacando processos de pré-processamento e interpretação semântica, além de sistemas de perguntas e respostas baseados em redundância de dados e gramáticas estruturadas. Por fim, discute interfaces baseadas em conhecimento comum e apresenta ferramentas como o NLTK, ilustrando suas funcionalidades e exemplos práticos.

O capítulo 20, "Automatic Speech Recognition," explora o reconhecimento automático de fala, começando com uma introdução ao tema e os recursos disponíveis. Ele aborda a aplicação de algoritmos de reconhecimento de fala e a estrutura dos modelos que sustentam o processo, como o léxico, modelos acústicos e modelos de linguagem. Também discute as ferramentas utilizadas no desenvolvimento de sistemas de ASR, incluindo motores de reconhecimento de fala e softwares específicos. O capítulo se encerra com uma síntese dos principais conceitos apresentados, oferecendo exercícios práticos e referências para aprofundamento.

Deep Learning for NLP and Speech Recognition (2019)

O terceiro capítulo, "Text and Speech Basics," aborda os fundamentos do processamento de texto e fala, começando com uma introdução à linguística computacional e os modelos de linguagem que sustentam essas tecnologias. Em seguida, explora a análise morfológica, destacando métodos como stemming e lematização para tratar variações linguísticas. Representações lexicais, como tokens, stop words e n-grams, são apresentadas como ferramentas centrais na estruturação de dados textuais. O capítulo avança para representações sintáticas, como análise de classes gramaticais e parsing de dependência, e semânticas, com foco em reconhecimento de entidades nomeadas, extração de relações e papéis semânticos. Representações discursivas também são discutidas, abordando coesão, coerência e referência, seguidas de um estudo sobre modelos de linguagem e suas métricas. Por fim, há seções dedicadas a tarefas práticas, como classificação de texto, análise de sentimentos, clustering, tradução automática, sistemas de perguntas e respostas e sumarização.

O oitavo capítulo, "Automatic Speech Recognition," foca no reconhecimento automático de fala, começando com uma visão geral da produção de fala e das características acústicas, como MFCCs e formas de onda. Discute modelos estatísticos de fala, destacando os componentes acústicos, modelos de linguagem e o uso de HMMs para decodificação. O capítulo também apresenta métricas de erro e uma análise do modelo híbrido DNN/HMM, que combina técnicas modernas de redes neurais e estatísticas tradicionais. Um estudo de caso detalhado ilustra a aplicação prática usando ferramentas como Sphinx e Kaldi, com ênfase nos resultados obtidos e exercícios para praticantes.

O capítulo doze, "End-to-End Speech Recognition," explora abordagens modernas de reconhecimento de fala com modelos de ponta a ponta, como a classificação temporal conexionista (CTC). Modelos como Deep Speech e Deep Speech 2 são analisados em detalhes, assim como o Wav2Letter e outros avanços baseados em aprendizado profundo. O capítulo oferece uma visão abrangente sobre como essas abordagens transformaram o

reconhecimento de fala ao eliminar a necessidade de modelos intermediários complexos, proporcionando maior simplicidade e precisão nos resultados.

Ferramentas e Conceitos

Após analisar os artigos, focando especialmente nos conceitos e ferramentas citados, foram encontradas 27 ferramentas/conceitos, com algumas delas se repetindo nos artigos (MFCC, HMM, PLP, LSTM, DTW, RNNs e Redes Neurais Profundas). Abaixo, há uma breve descrição de cada uma delas.

Técnicas/Conceitos

- **MFCC (Mel Frequency Cepstral Coefficients)**: Técnica para captura de características acústicas.
- **LPCC (Linear Predictive Cepstral Coefficients)**: Coeficientes preditivos usados na análise de espectro de áudio.
- **PLP (Perceptual Linear Prediction Coefficient)**: Método de transformação de sinais de áudio para uma representação perceptual.
- **AF (Acoustic Feature)**: Conversão de sinais de áudio para vetores representativos.
- **AM (Acoustic Model)**: Modelagem das características acústicas dos fonemas.
- **LM (Language Model)**: Modelagem da sequência de palavras.
- **HMM (Hidden Markov Model)**: Modelo probabilístico para reconhecimento de fala.
- **DTW (Dynamic Time Wrapping)**: Técnica para medir similaridade entre sequências de tempo.
- **DBN (Dynamic Bayesian Networks)**: Modelagem flexível de processos estocásticos temporais.
- **SVM (Support Vector Machine)**: Algoritmo para tarefas de classificação, como modelagem acústica.
- **ANN (Artificial Neural Network)**: Usado para tarefas de classificação de fala.

- **DNN (Deep Neural Network)**: Rede neural profunda usada para melhorar a precisão do reconhecimento.
- **RNNs (Redes Neurais Recorrentes)**: Usadas para modelagem de sequências de fala.
- **LSTM (Long Short-Term Memory)**: Variante das RNNs para lidar com dependências de longo prazo.
- **Transformers**: Usados para reconhecimento de fala com base em autoatenção.
- **Modelos Baseados em Markov (HMMs)**: Para reconhecimento de padrões de fala.
- **Modelos Geradores e Discriminativos**: Métodos como Naive Bayes (gerador) e regressão logística (discriminativo).
- **CNNs (Convolutional Neural Networks)**: Usadas para modelagem de sinais de fala.
- **Time-Delay Neural Networks (TDNN)**: Aplicadas para ASR em ambientes ruidosos.
- **CTC (Connectionist Temporal Classification)**: Técnica para treinamento sem segmentação explícita.
- **Beamforming**: Técnica para melhorar reconhecimento de fala multicanal.
- **Noise Robust ASR**: Métodos como filtros cepstrais para ambientes ruidosos.
- **PCA, LDA, ICA**: Métodos para redução de dimensionalidade.
- **Wavelet-based Features**: Técnica para representar sinais de fala.
- **Zero Crossing Peak Amplitude (ZCPA), GFCC (Gammatone Frequency Cepstral Coefficients)**: Métodos robustos ao ruído.
- **Viterbi Algorithm**: Usado para determinar a sequência de palavras mais provável.
- **Busca com Foco no Feixe (Beam Search)**: Técnica para aceleração da decodificação.

Frameworks

- **HTK, Julius, Sphinx-4, Kaldi**: Conjuntos de ferramentas de código aberto usadas para construção de sistemas de reconhecimento de fala.
- **CMU Statistical Language Modeling (SLM) Toolkit e Stanford Research Institute Language Modeling Toolkit**: Ferramentas para modelagem de linguagem.

- **Kaldi**: Usado especificamente para reconhecimento de fala baseado em transdutores de estados finitos.
- **OpenFst**: Usado em conjunto com o Kaldi para transdutores de estados finitos.
- **Pytorch:** Framework de aprendizado profundo de código aberto que facilita a criação e treinamento de redes neurais, oferecendo flexibilidade com grafos computacionais dinâmicos.
- **TensorFlow:** Framework de aprendizado profundo e machine learning de código aberto, desenvolvido pelo Google, que facilita a criação, treinamento e implementação de modelos de IA em várias plataformas.

Implementações (Início)

Uma das atividades propostas para a Semana 3 consistia em começar a busca por implementações atuais de sistemas de tradução simultânea de fala. No GitHub, o usuário **kensonhui** apresenta um projeto que captura a entrada de áudio local, traduz utilizando Whisper v3 e transforma o texto em áudio com a Microsoft SpeechT5 TTS.

O nome do projeto é Realtime Speech to Speech Translation e detalha todo o processo, permitindo que haja uma reprodução do trabalho feito. Apesar de ter sido publicado inicialmente em 2023, ainda recebe atualizações, com a última delas feita há dois meses.

Conclusão

Após uma análise das ferramentas, técnicas e conceitos, foi decidido aprofundar um pouco mais em Transformers (tanto a arquitetura quanto os modelos) para dar continuidade à residência devido às suas capacidades de processamento paralelo, atenção de longo alcance e existência de modelos que já demonstram resultados excepcionais na tarefa de

tradução simultânea, como o Whisper, que é um modelo de reconhecimento de fala baseado em uma arquitetura Transformer.

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 10 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 4 - 101024 consistiram em:

- **Análise sobre a evolução de Transformers:**
  - Transformers ;
  - O foco principal foi definir modelos que podem ser usados na tarefa de Tradução Simultânea de Fala.

- **Busca por implementações que realizem transcrição/tradução em tempo real:**
  - Realtime Speech to Speech Translation (Kenson Hui);
  - Whisper Realtime Translation (mldljyh);
  - Whisper Streaming (ÚFAL);
  - RealtimeSTT (KoljaB).

- **Vantagens e desvantagens dos "Frameworks" (modelos) das implementações:**
  - Whisper;
  - Faster-Whisper.

Em decorrência das informações pesquisadas, foi estabelecido inicialmente o uso do Whisper/Faster-Whisper para seguir com o tema da residência.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Testar na prática todas as implementações selecionadas;
- Caso seja possível, alterar algumas partes do código do RealtimeSTT para devolver a tradução, não somente a transcrição.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

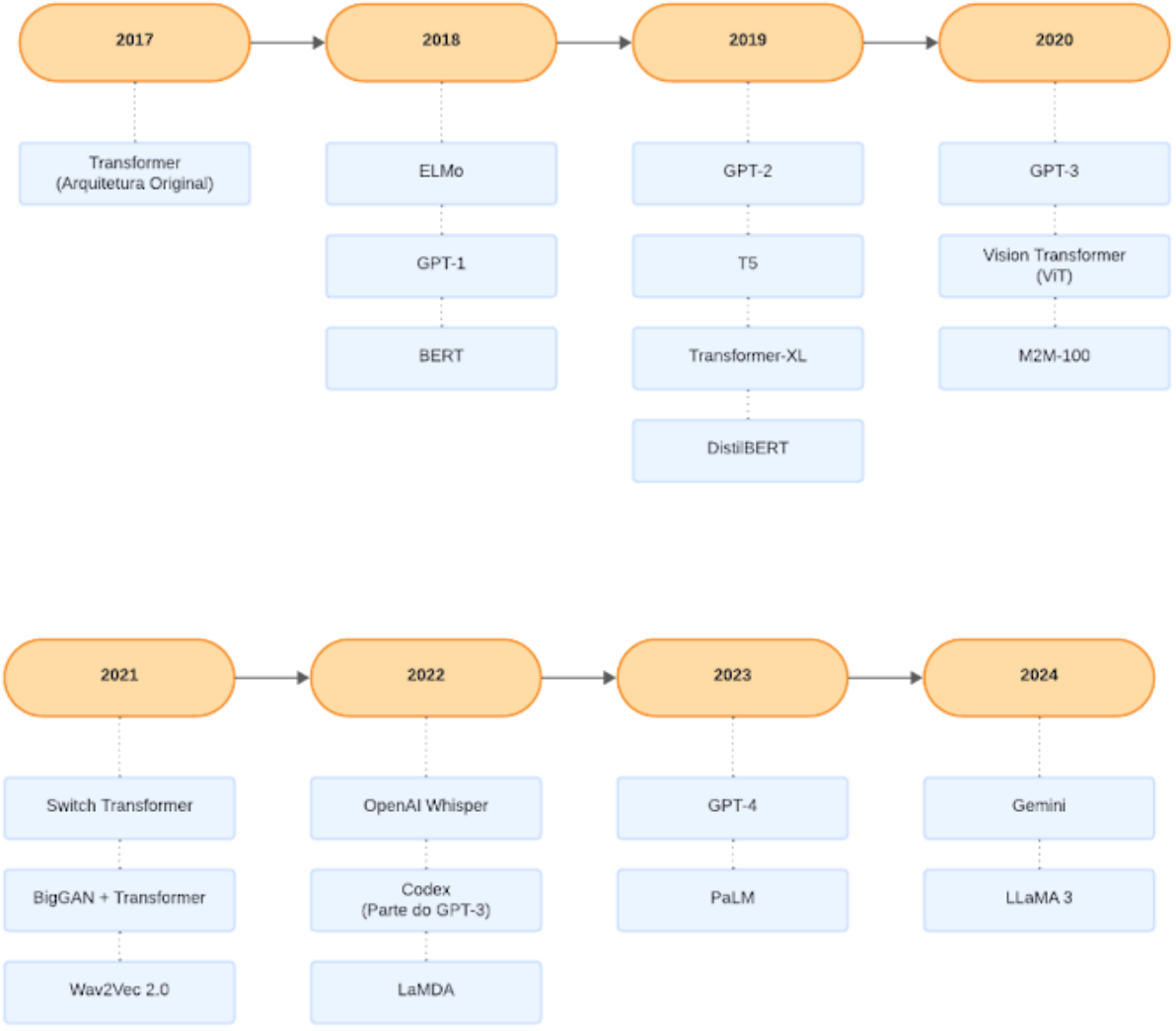
Transformers

Foi efetuado um simples resumo sobre a evolução dos principais marcos nas áreas de Transformers e, com isso, algumas ressalvas devem ser feitas. Existem diferentes possibilidades de criar um sistema de tradução de fala em tempo real, seja capturando o áudio diretamente e imediatamente realizando a tradução, ou utilizando um ASR para gerar a transcrição e, posteriormente, um modelo de tradução cuja entrada foi o texto transcrito.

Alguns destaques aqui foram o Wav2Vec 2.0, OpenAI Whisper e M2M-100. Apesar de o M2M-100 não necessitar do passo extra de ter a língua inglesa como uma ponte intermediária, um sistema que o incluísse precisaria de um ASR para gerar a transcrição. É uma opção viável de teste, porém pode ter um tempo de latência alto, o que prejudicaria o objetivo de ser usado em tempo real.

Já o Whisper se destaca por lidar bem com ruído de fundo, sotaques diversos, e múltiplas línguas. Além de transcrever a fala, ele oferece suporte para traduzir diretamente o texto gerado em outra língua, o que se alinha perfeitamente com os objetivos da tarefa escolhida (Tradução Simultânea de Fala).

Evolução de Modelos Baseados em Transformers



2017

Transformer (Arquitetura Original): Desenvolvido pelos pesquisadores do Google Brain (Vaswani et al.) no artigo *Attention is All You Need*. Tem como diferencial o mecanismo de self-attention para processamento paralelo de sequências.

2018

ELMo: Criado pelo grupo AllenNLP. Tem como diferencial embeddings dinâmicos e contextuais de palavras, ajustando-se ao contexto de cada uso.

GPT-1: Desenvolvido pela OpenAI. Tem como diferencial o uso do transformer unidirecional para geração de texto.

BERT: Desenvolvido pelo Google AI. Tem como diferencial o pré-treinamento bidirecional, permitindo maior compreensão de contexto ao analisar as palavras à esquerda e à direita simultaneamente.

2019

GPT-2: Lançado pela OpenAI. A principal diferença é o aumento significativo de parâmetros em relação ao GPT-1, melhorando a fluidez e a coesão na geração de texto.

T5: Desenvolvido pelo Google. Tem como diferencial o fato de tratar todas as tarefas de NLP como problemas de transformação de texto para texto, usando tanto encoder quanto decoder.

Transformer-XL: Criado pela Google DeepMind. Tem como diferencial o aprimoramento do transformer para lidar com dependências de longo prazo em sequências extensas.

DistilBERT: Desenvolvido pela Hugging Face. Tem como diferencial ser uma versão reduzida e mais rápida do BERT, mantendo 97% da precisão com apenas 60% dos parâmetros.

2020

GPT-3: Desenvolvido pela OpenAI. Tem como diferencial sua escala massiva, com 175 bilhões de parâmetros, e capacidades avançadas de geração de texto.

Vision Transformer (ViT): Criado pelo Google. Tem como diferencial a aplicação do transformer para visão computacional, tratando imagens como sequências de patches.

M2M-100: Desenvolvido pela Facebook/Meta AI. Tem como diferencial a tradução direta entre múltiplas línguas, sem usar o inglês como intermediário.

2021

Switch Transformer: Desenvolvido pelo Google. Tem como diferencial a arquitetura de mistura de especialistas (mixture of experts), ativando apenas partes do modelo, tornando-o mais eficiente.

BigGAN + Transformer: Combina transformers com GANs para geração de imagens de alta qualidade.

Wav2Vec 2.0: Desenvolvido pela Facebook/Meta AI. Tem como diferencial o reconhecimento de fala baseado em aprendizado não supervisionado, facilitando a transcrição de áudio.

2022

OpenAI Whisper: Desenvolvido pela OpenAI. Tem como diferencial sua robustez no reconhecimento de fala, mesmo em ambientes ruidosos e com diferentes sotaques.

Codex (parte do GPT-3): Desenvolvido pela OpenAI. Tem como diferencial sua especialização na geração de código de programação, utilizado no GitHub Copilot.

LaMDA: Desenvolvido pelo Google. Tem como diferencial ser um modelo de diálogo avançado para conversas abertas e mais naturais.

2023

GPT-4: Desenvolvido pela OpenAI. Tem como diferencial sua maior capacidade multimodal, lidando com texto e imagem de forma mais integrada e com maior contexto.

PaLM: Criado pelo Google. Tem como diferencial ser um modelo de linguagem de grande escala, com suporte para tarefas multimodais (texto e imagem).

2024

Gemini: Desenvolvido pelo Google DeepMind. Tem como diferencial ser uma IA multimodal avançada, combinando capacidades de linguagem e visão em um único modelo.

LLaMA 3: Desenvolvido pela Meta AI. Tem como diferencial suas melhorias em eficiência e redução de tamanho, focado em manter modelos de grande escala mais acessíveis para pesquisas abertas.

Considerações

Dentre os modelos mencionados, somente alguns se destacam na tarefa de tradução simultânea de fala, sendo eles o Wav2Vec 2.0, OpenAI Whisper e M2M-100, com abertura de uso do Codex e LaMDA, que não foram projetados para tradução de fala, mas podem ser usados em adaptações para a realização da tarefa.

O Wav2Vec é excepcional em reconhecimento de fala, usando aprendizado não supervisionado em grandes volumes de dados de áudio, o que o torna muito eficaz para converter fala em texto, uma etapa crítica para a tradução simultânea.

Já o Whisper, é um dos modelos mais robustos para reconhecimento e tradução de fala em texto. Ele se destaca por lidar bem com ruído de fundo, sotaques diversos, e múltiplas línguas. Além de transcrever a fala, ele oferece suporte para traduzir diretamente o texto gerado em outra língua, tornando-o uma das opções mais avançadas para tradução simultânea.

Finalmente, temos o M2M-100, que é especializado em tradução multilinguística, podendo realizar traduções diretas entre várias línguas sem passar pelo inglês como intermediário. Embora seu foco principal seja a tradução de texto, ele também pode ser utilizado para tradução simultânea de fala, se combinado com sistemas de reconhecimento de fala como o Wav2Vec 2.0.

Implementações

Realtime Speech to Speech Translation (Kenson Hui)

Modelo: Whisper

Detalhes: Permite tradução em tempo real de áudio para áudio usando sockets. Ele utiliza o modelo Whisper da OpenAI para transcrição e o Microsoft SpeechT5 para a síntese de fala. O fluxo de dados envolve um cliente que envia áudio para um servidor, que processa a tradução e devolve o áudio traduzido.

Whisper Realtime Translation (mldljyh)

Modelo: Faster-Whisper

Detalhes: Utiliza o modelo Faster-Whisper para transcrição de fala em tempo real e a biblioteca TranslatePy para tradução. Os resultados são exibidos em uma janela pop-up semi-transparente, facilitando a visualização. O sistema é leve, adequado para dispositivos

com menos de 6GB de VRAM. Os usuários podem personalizar a experiência, incluindo opções para o modelo e idioma.

Whisper Streaming (ÚFAL)

Modelo: Whisper

Detalhes: Implementa a transcrição e tradução em tempo real usando o modelo Whisper da OpenAI. Ele permite a transcrição de longos áudios de forma contínua e eficiente, com suporte para múltiplos idiomas. O projeto utiliza um mecanismo de streaming para processar o áudio em tempo real e é compatível com várias bibliotecas e frameworks, facilitando a integração.

RealtimeSTT (KoljaB)

Modelo: Faster-Whisper

Detalhes: Usa o modelo Whisper da OpenAI para a transcrição em tempo real. O projeto é projetado para ser eficiente e de baixa latência, com suporte para detecção de atividade de voz e ativação por palavra-chave. É uma solução robusta para aplicações de transcrição instantânea.

Observação: O código referente a essa biblioteca foi testado. Existem opções de idiomas de entrada, entretanto, é possível que uma transcrição seja gerada em outra língua quando a mesma for detectada, mesmo que o idioma selecionado seja outro. Um problema identificado foi a dificuldade de gerar uma boa transcrição com diferentes sotaques. A palavra ‘verdade’ não é identificada corretamente quando o sotaque goiano é falado, porém, foi identificada com excelência quando o sotaque usado foi o paulista. Em inglês e espanhol há o mesmo problema.

“Frameworks”

Nas implementações selecionadas, é possível identificar a recorrência do Whisper e sua variante, o Faster-Whisper. Por isso, as vantagens e desvantagens referentes a eles foram analisadas em prol de determinar qual seria a melhor escolha para ser usado em um sistema de tradução simultânea de fala.

Whisper

Vantagens:

- **Precisão**: Alta taxa de acerto em várias línguas e sotaques.
- **Versatilidade**: Funciona para transcrição e tradução.
- **Modelo aberto**: Disponível para uso público, permitindo personalização.
- **Robustez**: Capaz de lidar com áudio ruidoso e variações na qualidade do som.

Desvantagens:

- **Requisitos de hardware**: Pode demandar GPUs potentes para desempenho ideal.
- **Latência**: Pode haver um atraso em transcrições em tempo real.
- **Complexidade**: Configuração inicial pode ser desafiadora para iniciantes.

Faster-Whisper

Vantagens:

- **Velocidade**: Projetado para processamento mais rápido em comparação com o Whisper original.
- **Eficiência**: Requer menos recursos de hardware, tornando-o mais acessível.
- **Precisão**: Mantém uma alta taxa de acerto similar ao modelo original.
- **Flexibilidade**: Compatível com diversas plataformas e aplicações.

Desvantagens:

- **Menos otimizado para tarefas específicas**: Pode não ser tão preciso quanto o Whisper em alguns cenários.
- **Complexidade de instalação**: A configuração inicial ainda pode ser desafiadora.
- **Limitações em hardware mais antigo**: Apesar de mais eficiente, ainda pode exigir GPUs decentes.

Conclusão

A partir das observações até o presente momento, foi decidido utilizar o modelo Whisper e suas variações para dar continuidade à residência. Uma possibilidade futura seria usar o M2M-100 juntamente com um ASR para gerar resultados comparativos.

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 17 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 5 - 171024 consistiram em:

- **Implementações de Tradução Simultânea de Fala:**
  - Uso do PyCharm e de microfone externo;
  - Durante as implementações dos códigos, o RealtimeSTT se mostrou como o mais adaptável e, por essa razão, os testes foram feitos utilizando-o como base;
  - As alterações feitas para efetuar os testes foram:
    - Whisper (task='translation');
      - O modelo Medium apresenta um bom equilíbrio entre desempenho e tempo de inferência;
    - Whisper + M2M100.
  - Testes das Implementações.

- **Considerações sobre as implementações:**
  - A implementação que utiliza a ferramenta disponibilizada pelo Whisper para traduzir automaticamente a fala transcrita é mais rápida do que utilizando um modelo de tradução externo, contudo, essa tradução só pode ser feita para o inglês;
  - A implementação que utiliza o M2M100 tem muitas limitações, pois, apesar de a transcrição ser excelente em alguns modelos, o modelo de tradução (que é fixo), não foi capaz de traduzir corretamente.

Em decorrência dos resultados obtidos, foi decidido continuar o uso do RealtimeSTT, contudo, fazendo alterações para reduzir o tempo de inferência e, caso seja necessário, otimizar o suficiente para que seja viável em sua tarefa designada. Além disso, API's também serão utilizadas para parâmetros de comparação da qualidade da tradução.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Testar o RealtimeSTT com algumas API's de tradução;
- Otimizar o processamento e tradução de áudio para diminuir o tempo de inferência;
- Caso seja necessário para os testes, implementar uma interface simples para ajudar com a coleta dos resultados.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 30 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 6 - 301024 consistiram em:

- **Testar o RealtimeSTT com APIs de Tradução:**
  - Google Cloud Translate;
  - DeepL;
  - OpenAI API (ChatGPT):
    - Dependendo do prompt, pode alucinar e retornar respostas que não são tão satisfatórias para o usuário (ex.: retorno de uma frase que não foi originalmente falada, ausência de xingamentos com o prompt padrão, etc).

- **Otimizar a abordagem Whisper + Modelo de Tradução (M2M-100):**
  - Alterações no código para torná-lo mais rápido;
  - Redução do tempo de inferência de 6-18 para 0.24-1.93 segundos;
  - O tamanho do modelo Whisper não altera o tempo de inferência nesta abordagem;
  - A qualidade das traduções depende do tamanho da frase.

Ao comparar os resultados obtidos com as APIs e a abordagem do Whisper + M2M-100, é possível perceber que uma das principais diferenças é a capacidade das APIs de se adaptarem ao linguajar do usuário e traduzir corretamente ditos populares, gírias e afins. Todas as abordagens testadas são capazes de detectar o idioma falado, independente da configuração inicial. Em decorrência dos resultados, foi decidido continuar utilizando tanto as APIs quanto a abordagem Whisper + M2M-100, mas agora com o objetivo de tornar a tradução mais fluida.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Testar maneiras de tornar a tradução mais fluida, sem a necessidade de finalizar a frase para, finalmente, traduzir o texto.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

LEONARDO ALVES: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 7 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 7 - 071124 consistiram em:

- **Modificações no Código**
  - Não foram apresentadas mudanças significativas no código;
  - A biblioteca é complexa e adaptá-la para a tradução simultânea de fala de qualquer idioma para qualquer outro idioma se mostrou uma tarefa mais difícil do que imaginada anteriormente;
  - Redução do tempo de pausa entre frases para realizar a transcrição.

- **Adaptação ao Linguajar Específico**
  - Uma alternativa a ser testada e implementada é realizar o fine-tuning no modelo de tradução M2M-100, tendo como base o projeto do GitHub de TartuNLP.

Não foi feito o avanço esperado para a Semana, com dificuldades referentes à adaptação do código da biblioteca para a aplicação desejada.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Testar mais maneiras de deixar o processo mais fluido;
- Testar o fine-tuning do modelo de tradução.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 14 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 08 - 141124 consistiram em:

- **Alterações do Modelo**
  - Mudança do modelo de tradução do M2M-100 para o NLLB-200.

- **Adaptação ao Linguajar Específico**
  - Para ambientes com um vocabulário mais "nichado", é recomendado o uso de APIs, como o da OpenAI;
  - Quanto mais específico é o prompt em relação às particularidades do vocabulário, melhor será a tradução resultante, mesmo que uma palavra ou outra não sejam transcritas corretamente;
  - Algumas frases específicas de um nicho de jogos foram traduzidas perfeitamente com o NLLB-200 (Traduções em Contextos Específcos - Jogos).

Resumidamente, o tempo de inferência está satisfatório, o modelo de transcrição é capaz de identificar o idioma e transcrever palavras de diferentes línguas em uma mesma frase e o modelo de tradução também consegue traduzir relativamente bem frases com contextos específicos e em um tempo viável.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Produzir uma interface para seleção de idiomas de entrada e saída, assim como seleção de dispositivo de entrada e saída de resultados (tradução);
- Caso seja possível, tentar implementar em algum jogo.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Ferramentas Utilizadas

Os códigos foram adaptados para o uso no PyCharm para possibilitar o início dos testes. A ferramenta de clonar um GitHub específico foi usada para obter os códigos, sem modificar os trabalhos originais e permitir alterações para testes futuros. Um microfone externo foi utilizado para a captura de áudio, juntamente com as configurações da NVIDIA Broadcast para assegurar a supressão de ruídos e melhorar a qualidade do áudio.

Testes

Como anteriormente foi proposto o teste de integração do RealtimeSTT com outra ferramenta que fosse capaz de utilizar a transcrição resultante para realizar a tradução da fala captada, foram propostas duas abordagens: o uso da função de tradução presente no próprio Whisper e o uso de um modelo de tradução.

O modelo de tradução escolhido foi M2M-100 por ser projetado para traduzir diretamente entre 100 idiomas diferentes sem a necessidade de um idioma intermediário, como o inglês. Isso significa que o modelo pode realizar traduções de forma mais eficiente e precisa, sem depender de pares de treinamento específicos. Uma das grandes qualidades do M2M-100 é sua capacidade de lidar com idiomas menos comuns, oferecendo uma tradução de maior qualidade para línguas com menos recursos. Ao traduzir diretamente entre os pares de idiomas, o modelo reduz erros e preserva melhor o significado e as nuances do texto original.

RealtimeSTT (task=’translation’)

As mesmas frases foram testadas com os diferentes tamanhos de modelos para gerar comparativos, juntamente com seus respectivos tempos de inferência e traduções resultantes. Um detalhe a ser observado é que as traduções foram feitas somente para o

inglês, pois este é o único idioma que permite realizar a tradução automática. Para qualquer outra língua, deve-se usar um modelo de tradução ou API.

As frases de teste escolhidas foram:

- **(Frase 1)** Eu sou uma estudante de Inteligência Artificial
- **(Frase 2)** Esse é um teste para mensurar a qualidade da tradução e da transcrição
- **(Frase 3)** O rápido zangão voou sobre o vale florido, zumbindo de forma errática, enquanto as folhas balançavam suavemente com a brisa.

Esta última frase foi escolhida por conter uma combinação de sons complexos, com diferentes ritmos e palavras com similaridade fonética, o que pode ajudar a avaliar a qualidade de uma transcrição de fala.

Essa implementação alterou uma parte da classe presente na biblioteca original para permitir a tarefa de tradução, o que permitiu tempos mais baixos do que os apresentados posteriormente.

Modelo	Tempo de Inferência (Frase 1)	Tempo de Inferência (Frase 2)	Tempo de Inferência (Frase 3)
Tiny	0.70 segundos	0.56 segundos	2.65 segundos
Small	0.89 segundos	0.69 segundos	1.34 segundos
Base	0.94 segundos	0.45 segundos	2.30 segundos
Medium	1.17 segundos	0.90 segundos	1.18 segundos
Large-v2	1.83 segundos	1.11 segundos	2.29 segundos

Modelo	Tradução Obtida (Frase 1)	Tradução Obtida (Frase 2)	Tradução Obtida (Frase 3)
Tiny	I am a student of artificial intelligence.	This is a test to ensure the quality of the translation and the transcription.	The fast-earned, soft-robed, is a beautiful form of magic, while the flowers were balanced with its brightness.
Small	I am an Artificial Intelligence student.	This is a test to measure the quality of the translation and translation.	The fast zangam flew over the flowery valley, zooming in on the way it was erratic, while the leaves were slowly balancing with the breeze.
Base	I am a student of artificial intelligence.	This is a test to measure the quality of the translation and transcription.	The rapid zangão flew over the flower valley, zumbindo de forma erática, while the leaves balanced their avi-mente with the breeze.
Medium	I am an Artificial Intelligence student.	This is a test to measure the quality of the translation and transcription.	The fast zangão flew over the florid valley, zooming erratically, while the

			leaves swayed gently with the breeze.
Large-v2	I am an Artificial Intelligence student.	This is a test to measure the quality of translation and transcription.	The fast Zangão flew over the flowery valley, zooming erratically, while the leaves swayed gently with the breeze.

RealtimeSTT + M2M-100

Devido às limitações quanto ao idioma para o qual se deseja traduzir, foi feita uma alteração no código para implementar o modelo de tradução M2M-100, do Facebook/Meta. Essa alteração resultou em um aumento significativo no tempo de inferência, porém, possui mais possibilidades no uso da tradução e, provavelmente, pode ser otimizado para diminuir a latência entre a entrada de áudio e a tradução. Contudo, seu desempenho piorou a qualidade da tradução, por mais que as transcrições obtidas usando modelos mais robustos do Whisper fossem excelentes.

As frases utilizadas para o teste aqui foram as mesmas.

Modelo	**Tempo de Inferência (Frase 1)**	**Tempo de Inferência (Frase 2)**	**Tempo de Inferência (Frase 3)**
Tiny	6.502 segundos	15.229 segundos	11.081 segundos

Small	12.040 segundos	18.300 segundos	12.620 segundos
Base	12.480 segundos	18.620 segundos	11.023 segundos
Medium	13.180 segundos	13.249 segundos	11.203 segundos
Large-v2	16.119 segundos	16.640 segundos	11.818 segundos

Modelo	Tradução Obtida (Frase 1)	Tradução Obtida (Frase 2)	Tradução Obtida (Frase 3)
Tiny	I am a student of artificial intelligence.	This is a test to measure the quality of translation and transcription.	The rapidus Angão wants to survive florido zumbindo de farmerratica, while as folias balançavam su befimente com a brisa.
Small	I am a student of artificial intelligence.	This is a test to measure the quality of the translation and transcription.	Rapido zangão voou sobre o vale florido, zumbindo de forma errática, in how many folhas balançavam suavemente can abriza.
Base	I am a student of artificial intelligence.	This is a test to measure the quality of translation and transcription.	A fast zangão, vol sobrevali florido, zumbindo de forma erratica, while as folhas balançavam

			suavimente com abrisa.
Medium	I am a student of artificial intelligence.	This is a test to measure the quality of the translation and transcription.	The fast zangão flew over the flowery valley, zombing erratically while the leaves swayed gently with the breeze.
Large-v2	I am a student of artificial intelligence.	This is a test to measure the quality of translation and transcription.	A quick zangão voou sobre o vale florido, zumbindo de forma errática, while as folhas balançavam suavemente com a brisa.

Implementações

Durante os Testes das Implementações dos códigos, o RealtimeSTT se mostrou como o mais adaptável e, por essa razão, os testes foram feitos utilizando-o como base. Foram escolhidas três frases bases, com uma delas contendo uma estrutura específica para mensurar a qualidade da transcrição, uma vez que possui uma combinação de sons complexos, com diferentes ritmos e palavras com similaridade fonética.

Um aspecto importante observado é que o modelo Medium do Whisper é capaz de obter uma transcrição satisfatória e em menos tempo que o Large-v2, o que seria mais ideal para a tarefa de Tradução Simultânea de Fala.

A implementação que utiliza a ferramenta disponibilizada pelo Whisper para traduzir automaticamente a fala transcrita é mais rápida do que utilizando um modelo de tradução externo, contudo, essa tradução só pode ser feita para o inglês.

Entretanto, é importante ressaltar que a implementação que utiliza o M2M-100 tem muitas limitações, pois, apesar de a transcrição ser excelente em alguns modelos, o modelo de tradução (que é fixo), não foi capaz de traduzir corretamente, o que resultou em uma péssima qualidade de tradução.

Uma alternativa para contornar este problema pode ser utilizar API's para realizar o processo de tradução, sendo necessário somente a entrada da fala.

Abaixo, estão os códigos principais usados para realizar os testes

Whisper (task='translation')

Whisper + M2M100

Otimização RealtimeSTT + M2M-100

A partir das observações até o presente momento, foi decidido continuar o uso do RealtimeSTT, contudo, fazendo alterações para reduzir o tempo de inferência e otimizar o suficiente para que seja viável em sua tarefa designada.

As principais diferenças de otimização entre o código anterior e o atual estão relacionadas ao uso da GPU (se disponível), à economia de memória e à modularização do código para facilitar o processamento e evitar redundância.

Na primeira versão, somente a CPU era utilizada para o processamento do modelo, o que é menos eficiente em termos de velocidade, especialmente em tarefas de tradução de linguagem natural que envolvem modelos grandes. Além disso, o modelo era carregado com

precisão padrão (float32), consumindo mais memória e podendo ser mais lento, especialmente em sistemas com menos capacidade de memória.

Atualmente, o código otimiza o uso de hardware, verificando se há uma GPU disponível e, caso haja, transferindo o modelo para a GPU, além disso, a precisão do modelo também foi reduzida para float16, o que diminui a quantidade de memória usada e pode aumentar a velocidade de inferência.

Outra modificação foi referente à verificação/transferência de tensores explicitamente para a GPU. Na primeira versão, todo o processamento é feito na CPU, o que tornava o código mais lento. No intuito de corrigir isso, agora, o código verifica a disponibilidade de GPU e, caso disponível, transfere os tensores para a GPU, evitando operações de cópia desnecessárias e aumentando a eficiência. A transferência condicional dos tensores (encoded_sentences) para a GPU é feita diretamente na função de tradução, evitando a repetição de código e otimizando o uso da GPU.

Finalmente, a última alteração foi em relação aos parâmetros padding e truncation. Na versão inicial, não se usa os parâmetros padding e truncation, o que pode levar a resultados inconsistentes e menos otimizados em termos de processamento. Ao definir padding=True e truncation=True ao tokenizar, o desempenho melhora, especialmente em tarefas de tradução em lote. Com o padding, as entradas são padronizadas, o que evita que o modelo tenha que lidar com entradas de diferentes tamanhos no mesmo batch.

Uma observação em relação à qualidade das traduções é que, quanto maior a frase falada, maiores as chances de a tradução apresentar uma boa qualidade, uma vez que ao falar somente uma palavra, ela pode ser erroneamente interpretada como parte de um outro idioma. Ao apresentar mais elementos na frase, também são fornecidas mais informações sobre o idioma falado, o que resulta em uma transcrição mais certeira e, consequentemente, uma tradução melhor.

Outro fator é que, aparentemente, a ordem na qual a frase é dita interfere na qualidade da tradução. Por exemplo, ao dizer ‘tradução e transcrição’, a tradução é perfeita.

Entretanto, quando a ordem deles é invertida, ambos são traduzidos somente como ‘Translation’.

Otimização	Código Antigo	Código Atual
Uso de GPU e Redução de Precisão	CPU e float32	GPU (se disponível) e float16
Modularização de Função	Fluxo completo no loop principal	Função translate_text modularizada
Transferência de Tensores	CPU apenas	Verifica e transfere para GPU
Configuração de Padding/Truncação	Não aplicados	padding=True, truncation=True

Com essas alterações, o tempo de inferência caiu de 6-18 para 0.19-1.93 segundos, apresentando uma melhora extremamente significativa em relação aos resultados apresentados na anteriormente.

RealtimeSTT + APIs de Tradução

Além dos testes com os modelos de tradução, uma abordagem que utilizou APIs para a tradução também foi testada.

Foram pesquisadas algumas APIs de tradução para substituir o M2M-100 e comparar os resultados obtidos entre eles. É preciso ressaltar que poucas dessas APIs são

gratuitas e, quando são, não possuem a mesma capacidade e qualidade que as versões pagas.

API	Qualidade da Tradução	Idiomas Suportados	Recursos Adicionais	Custo	Pontos Fortes
Google Cloud Translate	Excelente, especialmente em uma variedade de contextos	100+	Detecção automática de idioma, glossário customizado, suporte para websites	Gratuito (limitado) e pago por caracteres	Alta precisão, ampla documentação e integração com Google Cloud
DeepL	Excelente, qualidade próxima à humana, ideal para idiomas europeus	30+	Controle de formalidade	Gratuito (limitado) e pago por caracteres	Tradução de alta qualidade, especialmente para idiomas europeus
Microsoft Azure Translator	Muito boa, competitiva com Google	70+	Detecção automática de idioma, glossário customizado, integração com Azure	Pago por caracteres	Integração com Azure e facilidade de personalização
Amazon Translate	Boa, com alta velocidade e escalabilidade	70+	Integração com AWS, detecção de idioma, customização de terminologia	Pago por caracteres	Excelente desempenho em grande escala e integração com AWS

IBM Watson Language Translator	Boa, com foco em contextos técnicos	50+	Modelos customizados para setores específicos	Pago por caracteres (plano gratuito limitado)	Integrado ao Watson e IBM Cloud, foco em modelos específicos
Yandex Translate	Boa, mas menos precisa em contextos complexos	90+	Detecção de idioma, tradução de websites	Pago por caracteres	Custo-benefício e facilidade de uso
OpenAI API (ChatGPT)	Alta, especialmente para traduções contextuais	50 - 100	Tradução contextual, suporte a nuances e gírias	Pago por token	Flexibilidade e contexto avançado de linguagem
LibreTranslate	Funcional, menos precisa em contextos complexos	10+	Código aberto, pode ser personalizado	Gratuito (para uso próprio) ou pago na nuvem	Open-source, ideal para soluções privadas e sem custo

API vs Modelos de Tradução

Ao comparar os resultados obtidos com as APIs e a abordagem do Whisper + M2M-100, é possível perceber que uma das principais diferenças é a capacidade das APIs de se adaptarem ao linguajar do usuário e traduzir corretamente ditos populares, gírias e expressões idiomáticas. Todas as abordagens testadas são capazes de detectar o idioma falado, independente da configuração inicial.

Em decorrência dos resultados, foi decidido continuar utilizando tanto as APIs quanto a abordagem Whisper + M2M-100, mas agora com o objetivo de tornar a tradução mais fluida.

Detecção de Pausas

Para tornar a aplicação mais fluida, o intervalo de pausa da fala para realizar a transcrição foi reduzido e, com isso, pausas pequenas em uma mesma frase segmentam a transcrição e tornam o processo um pouco mais fluido. Uma característica presente na biblioteca RealtimeSTT é a transcrição em tempo real, que é apresentada até mesmo no GitHub do projeto. Apesar de ativar essa função específica, a transcrição não ocorre com a fluidez desejada, pois está reservada a exibir este comportamento somente quando o idioma de entrada é inglês.

Adaptação ao Linguajar Específico

Uma desvantagem de se usar o M2M-100 é a inflexibilidade do mesmo em relação a aspectos mais específicos e intrínsecos de um idioma, como por exemplo, gírias e ditados populares. Em decorrência disso, uma alternativa que poderia ser testada e implementada é realizar o fine-tuning no modelo de tradução M2M-100, tendo como base o projeto do GitHub de TartuNLP.

Inicialmente, o objetivo seria testar de português para inglês e, posteriormente, caso seja bem sucedido, adaptar para alguns outros idiomas amplamente utilizados.

Alterações do Modelo de Tradução

Durante o processo de pesquisa sobre como melhorar a qualidade da tradução do modelo M2M-100 (2020), foi encontrado um modelo mais recente, também da Meta/Facebook, e que possui mais suporte para linguagens como recursos baixos ou médios. Este modelo de tradução é o No Language Left Behind (NLLB-200).

O NLLB-200 e o M2M-100 são dois modelos avançados de tradução automática, ambos desenvolvidos para lidar com múltiplos idiomas e aprimorar a tradução entre línguas com diferentes características. Embora ambos busquem superar as limitações dos modelos de tradução tradicionais, eles possuem diferenças marcantes em termos de escopo, foco e arquitetura.

Uma das diferenças principais entre os dois modelos está no número de idiomas que cada um pode suportar. O M2M-100, desenvolvido pelo Facebook AI, foi projetado para trabalhar com 100 idiomas, permitindo traduções diretas entre qualquer par de idiomas, sem a necessidade de passar pelo inglês como intermediário. Isso é particularmente vantajoso para sistemas que precisam lidar com múltiplas línguas de forma eficiente e precisa, sem perder a fluidez e o contexto nas traduções. Já o NLLB-200, também criado pelo Facebook AI, foi desenvolvido para lidar com uma gama ainda mais ampla, oferecendo suporte para 200 idiomas. Esse modelo foi projetado para lidar de forma mais eficaz com idiomas de baixa e média recursos, que muitas vezes têm menos dados paralelos disponíveis para treinamento de modelos de tradução.

A abordagem do NLLB-200 é focada em idiomas que historicamente têm sido negligenciados em modelos de tradução, incluindo muitos idiomas africanos, asiáticos e de outras regiões com menos corpora de dados. Essa ampliação para 200 idiomas reflete a tentativa de reduzir as desigualdades na qualidade da tradução automática entre idiomas de alto e baixo recurso. Enquanto o M2M-100 também oferece suporte para uma boa variedade de idiomas menos comuns, o NLLB-200 leva isso a um nível superior, com uma atenção específica para melhorar as traduções em línguas com escassez de dados de treinamento.

Outra diferença importante é a abordagem de treinamento. O NLLB-200 foi projetado com técnicas que visam reduzir a falta de recursos de treinamento, permitindo que o modelo gere traduções de qualidade, mesmo para idiomas para os quais há poucos dados. Isso é feito por meio de ajustes na arquitetura e no processo de treinamento, o que o torna mais robusto para situações em que a quantidade de dados disponíveis é limitada. Por outro lado, o M2M-100, embora seja igualmente poderoso, não foi tão explicitamente otimizado para lidar com idiomas de baixo recurso, o que pode resultar em uma qualidade de tradução ligeiramente inferior quando comparado ao NLLB-200 em contextos específicos.

Em termos de performance, enquanto o M2M-100 oferece boas traduções em idiomas populares e de médio porte, o NLLB-200 tem um desempenho superior quando o foco está em idiomas menos representados. Isso é particularmente evidente em cenários multilíngues, onde a diversidade de idiomas e a quantidade de dados paralelos podem variar amplamente. No geral, o NLLB-200 se destaca pela sua capacidade de lidar com a diversidade linguística de uma forma mais inclusiva e adaptada às necessidades globais, enquanto o M2M-100 oferece uma solução mais sólida para tradução entre idiomas populares com uma estrutura mais focada.

O NLLB-200 se destaca em contextos multilíngues altamente diversos, onde as combinações de idiomas podem ser muito diferentes. Em cenários de jogos, por exemplo, onde a comunicação entre jogadores pode envolver uma combinação de termos técnicos, gírias locais e expressões culturais, o NLLB-200 consegue fornecer traduções mais contextualizadas e com menor risco de distorção de significado, mesmo quando essas expressões são de idiomas com poucos dados.

Apesar de mais recente e mais robusto, o tempo de inferência com as alterações de otimização permaneceu o mesmo, não interferindo no objetivo de permanecer com um tempo de inferência baixo.

Contextos Específicos (Jogos)

Em muitas áreas, como no setor de jogos eletrônicos, existe um vocabulário altamente especializado que é utilizado por jogadores, desenvolvedores e fãs, o que torna a tradução e interpretação de mensagens mais complexa. Esse linguajar técnico pode incluir termos como “aggro”, “gank”, “buff”, “cooldown”, entre outros, que possuem significados muito específicos dentro do contexto do jogo. A dificuldade de tradução surge quando esses termos são aplicados em contextos de jogos diferentes ou quando não há equivalentes diretos em outras línguas, tornando necessária uma adaptação cultural e linguística.

Para adaptar-se a esse vocabulário técnico, é importante usar ferramentas capazes de aprender e se ajustar a esse tipo de contexto. APIs como as da OpenAI oferecem uma solução eficaz, uma vez que podem ser configuradas para entender e gerar respostas baseadas em um corpus específico, permitindo uma tradução mais precisa de termos nichados. O uso dessas tecnologias, que permitem que o modelo "aprenda" sobre os diferentes contextos de linguagem, pode melhorar significativamente a tradução em ambientes com vocabulário técnico.

Quando se trabalha com sistemas de tradução automatizada, a especificidade do prompt é crucial para obter uma tradução precisa. Quanto mais detalhado e direcionado for o prompt, mais eficaz será a tradução. Por exemplo, ao traduzir uma frase de um jogo de RPG, se o prompt incluir termos como "farming XP" ou "boss fight", o modelo pode gerar traduções mais adequadas ao contexto, já que entenderá que esses termos têm um significado técnico específico em comparação com sua tradução literal.

Embora isso melhore a qualidade geral da tradução, pode haver casos em que palavras específicas não sejam traduzidas corretamente, especialmente se não existirem equivalentes diretos na língua de destino. Mesmo assim, ao manter o foco no contexto geral, a tradução resultante será mais precisa, pois o modelo levará em conta o significado das expressões no contexto do jogo, ao invés de traduzir palavra por palavra.

A tradução de vocabulários técnicos em jogos envolve não apenas a tradução de palavras, mas também o entendimento de contextos específicos, como mecânicas de jogo,

interação entre jogadores, e elementos que compõem a experiência de um jogo. Isso significa que a tradução precisa ser capaz de capturar a funcionalidade e o tom do jogo, além de garantir que a comunicação entre jogadores não seja distorcida por falhas de tradução.

Ao traduzir frases dentro de um jogo, por exemplo, os termos como "Ult", "gank", e "jungler" devem ser traduzidos de maneira que não percam seu significado no jogo, e que, idealmente, mantenham a fluidez da conversa. Uma tradução mal feita pode resultar em uma experiência frustrante para o jogador, especialmente em jogos de ritmo rápido, onde o timing e a compreensão precisa das instruções são essenciais.

Um dos objetivos citados anteriormente era realizar o fine-tuning no modelo de tradução M2M-100 para entender melhor expressões idiomáticas. Mas como houve a troca do modelo de tradução em questão pelo NLLB-200, essa tarefa se tornou inviável, uma vez que a ferramenta que seria utilizada para realizá-la não tem suporte para o NLLB-200. Em compensação, esse novo modelo apresentou uma capacidade alta de tradução em contextos mais específicos, como em jogos. As traduções não são perfeitas, mas apresentam resultados muito bons sem fine-tuning algum, como pode ser analisado nos testes de Traduções em Contextos Específcos - Jogos.

Testes de Traduções em Contextos Específicos

1. Jogo de Estratégia (como "Age of Empires" ou "Starcraft"):

1. **"O 'macro' tá bom, mas eu preciso melhorar meu 'micro' para garantir que as tropas não morram tão facilmente nas batalhas."**

   The macro is good, but I need to upgrade my microphone to make sure the troops don't die so easily in the battles.

2. **"Não posso expandir agora, o 'timing' do 'rush' inimigo está muito forte. Melhor me concentrar em defender."**

   I can't expand now, the timing of the enemy Rush is very strong. I better focus on defending.

3. **"Acabei de 'upar' minha 'economia', agora posso treinar unidades mais poderosas para o 'push'."**

   I just cleaned up my economy, now I can train more powerful units for push.

4. **"Cuidado com a 'scout', eles vão ver o meu 'build' e tentar uma 'counter'!"**

   Watch out for the Scout, they'll see my build and try a counter.

5. **"Eu só preciso de mais 'gas' para avançar para as tecnologias de nível 3, se não, vou ficar para trás na pesquisa."**

   I just need more gas to advance level 3 technologies, or I'm gonna be left behind in research.

2. Jogo de Tiro em Primeira Pessoa (FPS, como "Call of Duty" ou "Counter-Strike"):

1. **"Estão 'campando' no bomb, não vou passar por ali sem uma 'bang'."**

They're camping at the bomb. I'm not gonna go through there without a bang.

2. **"Boa! Você deu 'cover' perfeito e limpei o 'site' B."**

   You gave me the perfect cover and I cleaned up site B.

3. **"O 'clutch' foi lindo, eu consegui fazer o '1v3' com a ajuda da minha 'smoke' para bloquear a visão."**

   The clutch was beautiful, I was able to do the 1v3 with the help of my smoke to block my vision.

4. **"Preciso de mais munição, vou 'recarregar' antes de entrar na próxima sala!"**

   I'll recharge before I go into the next room.

5. **"Eles estão 'stackando' no 'B', precisamos 'rotar' para o 'A' antes que o tempo acabe."**

   They're stalling in B. We need to rotate to A before time runs out.

3. Jogo de RPG (como "The Witcher" ou "Elder Scrolls"):

1. **"Estou tentando completar essa 'side quest', mas o 'loot' não está vindo como eu esperava. Vou tentar outra área."**

   I'm trying to complete this sidequest, but the loot isn't coming as I expected. I'm trying to complete this sidequest, but the loot isn't coming as I expected.

2. **"Preciso 'farmar' mais pontos de habilidade para desbloquear a 'ultimate' e melhorar o 'dps'."**

   I need to get more skill points to unlock my ultimate and improve DPS.

3. **"Acho que não vou conseguir derrotar esse 'boss', ele está muito 'tank' e o meu dano está baixo."**

   I don't think I'm gonna be able to beat that boss, he's too big and my damage is low.

4. **"Tirei 'aggro' do tanque e usei meu W para manter os inimigos ocupados enquanto a equipe cuida do resto."**

   I took the farm from the tank and used my W to keep the enemy occupied.

5. **"A história é incrível, mas os 'NPCs' são poucos. Eu só quero pegar mais 'quests'!"**

   The story is amazing, but the NPCs are few, I want to get more quests.

4. Jogo de Luta (como "Tekken" ou "Street Fighter"):

1. **"Usei o 'counter' no momento certo e agora estou fazendo o 'combo' perfeito!"**

   I used the counter at the right time and now I'm making the perfect combo.

2. **"Eu precisei usar o 'block' perfeito"**

   I had to use the perfect block.

3. **"Não me deixa 'punir' o teu 'whiff'! Eu posso encaixar um 'jab' rápido para punir o erro."**

   Don't let me punish your if. I need to fit a fast jib to punish the mistake.

4. **"Foi um 'perfect'! Não tomei nenhum 'hit' durante a luta, fiquei no 'frame data' o tempo todo."**

It was a perfect one, I didn't take any hits during the fight, I was in the date frame the whole time.

5. Jogo de MOBA (como "League of Legends" ou "Dota 2"):

1. **"Eu estou fazendo 'roaming' pela 'jungle', tentando encontrar o jungler' e dar o 'gank' no top."**

   I'm roaming the jungle, trying to find the jungler and give a gank at the top.

2. **"Acho que o 'ADC' está dando 'over' na 'bot lane', preciso evitar um 'gank' inimigo."**

   I think the ADC's overtaking the bot lane. Avoid an enemy gank.

3. **"Vamos fazer o 'drag' agora, a prioridade é garantir o buff antes da próxima 'team fight'."**

   We're gonna drag it now, and the priority is to secure the buff before the next team fight.

4. **"Estamos com vantagem, mas o 'split push' do 'top laner' deles pode ser um problema se não controlarmos o mapa."**

   We have the advantage, but… Their top laner split push could be a problem if we don't control the map.

5. **"Ele saiu do 'stun', mas o 'flash' do 'sup' foi perfeito para garantir a eliminação."**

   He's out of the stun, but the flash from the sup was perfect to ensure elimination.

Considerações Sobre a Implementação

O tempo de inferência está satisfatório, o modelo de transcrição é capaz de identificar o idioma e transcrever palavras de diferentes línguas em uma mesma frase e o modelo de tradução é capaz de traduzir relativamente bem frases com contextos específicos e em um tempo viável.

Em decorrência disso, a próxima etapa consiste na criação de uma interface interativa, que permita o uso do sistema de uma forma mais simples e intuitiva.

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 27 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 09 - 271124 consistiram em:

- **Comunicação Server / Client**
  - Solução momentânea para captura, transcrição e tradução de áudio em tempo real;
  - Composto por um servidor responsável pelo processamento centralizado e um cliente que envia comandos e exibe os resultados;
  - Uso da API DeepL para testes (devido ao menor tempo de espera para iniciar o serviço da API em relação aos modelos de tradução);

- **Protótipo da Interface**
  - Interface que permite a seleção do modelo usado, os idiomas de entrada e de saída e o dispositivo de entrada de áudio utilizado;

- **Teste Pop-Up**
  - Alternativa para ser transmitida, permitindo que quem assiste receba a tradução em tempo real do que a pessoa que está transmitindo está falando.

Em suma, alguns protótipos foram testados para analisar qual é o mais viável para dar continuidade ao atual momento. Cada um apresenta desafios particulares de diferentes dificuldades. Ainda não houve uma decisão definitiva sobre qual abordagem seguir.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Definir uma abordagem para seguir;
- Implementar a abordagem.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

LEONARDO ANTÔNIO ALVES: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 4 de dez. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Rafaela Mota Silva

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

As atividades realizadas durante o Stage 10 - 041224 consistiram em:

- **Resultado Final:**
  - Código;
  - Interface;
  - Seleção de Modelos;
  - Seleção de Idiomas de Entrada e Saída;
  - Avisos na Interface;
  - Prompt que permite lidar bem com ditados populares, expressões idiomáticas, gírias e afins.

- **Demonstração:**
  - Demo.mp4

Todo o trabalho até o presente momento permitiu que a implementação de um sistema de tradução que pode ser adaptado para qualquer contexto fosse possível. O resultado final está longe de ser perfeito, mas demonstra uma parcela do que pode ser feito utilizando as técnicas e ferramentas certas para um determinado objetivo.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Aplicar a tradução em um jogo que eu goste e me motive, não precisando de intermediadores como o Discord e sem um limite de tempo estabelecido para entrega.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Um agradecimento especial a Lucas Brandão, por tirar um pouco de seu tempo para testar comigo a eficiência e qualidade do resultado da Residência.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Comunicação Server / Client

O sistema implementa uma solução momentânea para captura, transcrição e tradução de áudio em tempo real. Ele é composto por dois componentes principais que trabalham de forma integrada: um servidor responsável pelo processamento centralizado e um cliente que envia comandos e exibe os resultados.

No servidor, implementado no arquivo server.py, o fluxo inicia com a configuração de um serviço que escuta conexões no endereço 0.0.0.0 e na porta 5025. A partir daí, ele gerencia os clientes conectados, registrados em um conjunto chamado connected_clients. Esses clientes podem enviar comandos, como o de iniciar gravação, que é processado pela função handler. Esse comando ativa a captura de áudio, realizada pelo componente AudioToTextRecorder. O gravador é configurado para detectar segmentos de áudio e realizar transcrições em tempo real. Sempre que uma transcrição estabilizada é detectada, o servidor a processa e adiciona a uma fila de mensagens (message_queue).

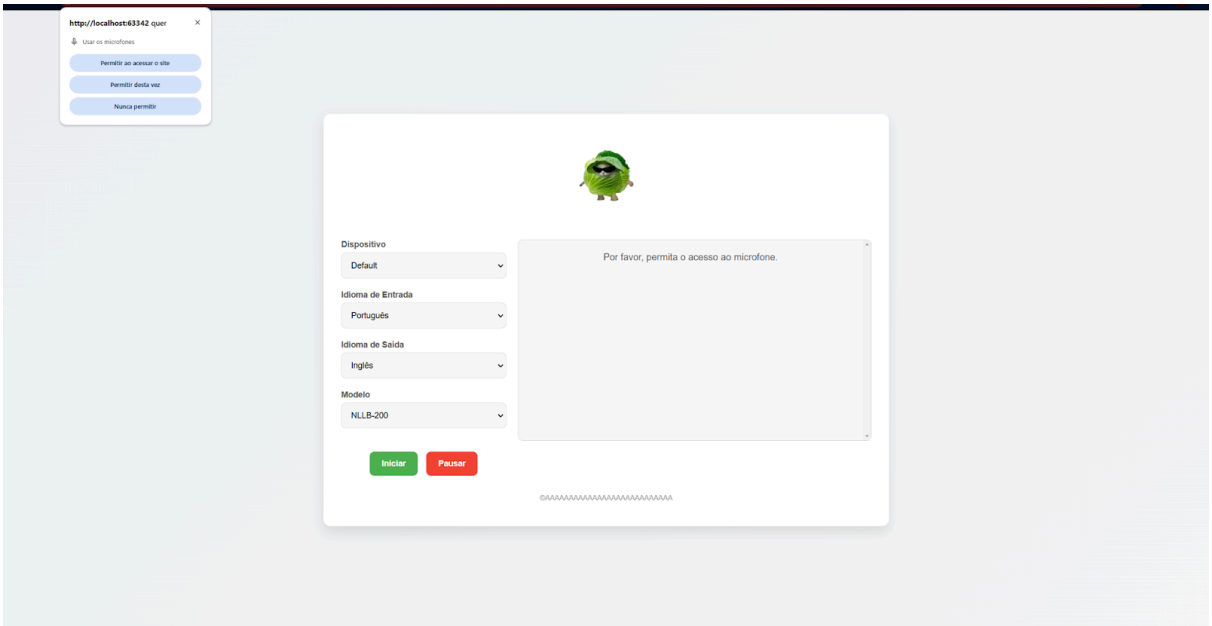
Após a captura e transcrição, o texto é traduzido para o idioma de destino usando a API DeepL, mas o código pode ser facilmente adaptado para usar os modelos de tradução apresentados anteriormente. A decisão de usar a API DeepL foi consequência do menor tempo de espera para iniciar o serviço da API em relação aos modelos de tradução devido ao tamanho dos mesmos. O processamento da tradução é realizado por uma thread dedicada, que aguarda a detecção de um segmento de áudio, executa a transcrição e envia o texto traduzido de volta aos clientes conectados. Esses resultados são enviados de forma assíncrona, garantindo uma comunicação contínua e eficiente.

O cliente, por sua vez, conecta-se ao servidor e é responsável por enviar comandos e receber mensagens. Implementado no arquivo client.py, assim que o servidor processa o áudio e realiza a transcrição e tradução, as mensagens resultantes são recebidas pelo cliente e exibidas no terminal.

Utilizando a biblioteca colorama, as transcrições em tempo real são exibidas de forma contínua, enquanto as traduções completas são destacadas em amarelo para facilitar

sua identificação. Um fator limitante em relação à fluidez do texto reside no fato de a transcrição para o inglês ter uma acurácia muito melhor do que qualquer outro idioma testado, ou seja, quando se fala em inglês, a transcrição ocorre palavra por palavra, não frase por frase. Apesar de tudo, a simplicidade do terminal é funcional, mas limita as possibilidades de interação, sugerindo espaço para melhorias com uma interface gráfica.

Protótipo da Interface



Em decorrência dos pontos levantados anteriormente, uma interface foi pensada para ser capaz de lidar com estes problemas. O primeiro deles é em relação a qual é o dispositivo de entrada do usuário, que precisa ser manualmente definido no código até então.

O segundo reside no fato apontado de que os idiomas de entrada e saída influenciam na qualidade da transcrição e, consequentemente, na tradução das falas. Logo, o ideal seria definir quais são os idiomas de entrada e saída para cada uma das alternativas

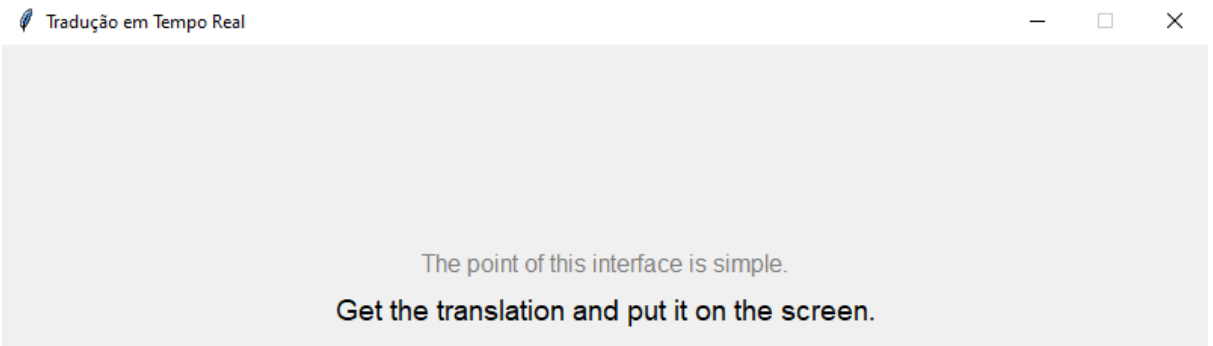
de tradução, flexibilizando o uso dos modelos e da API, de modo que um ou outro seja usado quando não há a opção esperada disponível no atual.

O terceiro se encontra no fato de que usar somente os modelos de tradução poderiam tornar o processo mais lento para pessoas que precisam imediatamente dele, e somente o uso da API poderia ser muito limitante por não ter apoio a idiomas de médio e baixo recursos. Por isso, o ideal também seria que os usuários pudessem escolher qual o meio de tradução utilizado com base em suas necessidades.

Teste Pop-Up

Um fato ressaltado após a apresentação do Stage 08 foi o fato de que um lugar onde muitas pessoas vão para conversar ou jogar é o Discord. Nele, você é capaz de compartilhar sua tela com outros usuários, mostrando o conteúdo presente ali. Uma alternativa sugerida foi o uso de bots do Discord para auxiliarem nesta tarefa de tradução de fala em tempo real, porém, ainda não houve uma tentativa de fato em fazer isso.

Por isso, uma alternativa seria transmitir a tela que exibe as traduções em uma interface mais bonita, ainda com as funcionalidades de escolher o idioma de entrada e de saída, o modelo utilizado e afins. Um protótipo foi feito, pois as implementações para a escolha dos modelos e display de idiomas de entrada e saída baseadas nos modelos ainda não estão na versão ideal.



Na interface, o usuário recebe a tradução do que falou em tempo real para o idioma escolhido, com a possibilidade de transmissão para que outra pessoa veja e compreenda o que foi dito.

No terminal, é possível ver a frase traduzida e o tempo de inferência para a tradução.

```
Texto traduzido: The point of this interface is simple.
Tempo decorrido: 0.28685498237609863

Texto traduzido: Get the translation and put it on the screen.
Tempo decorrido: 0.32681751251220703
```

Resultado Final

Com base nos protótipos produzidos anteriormente, foi feita uma interface simples, usando elementos diferentes de cada uma das alternativas apresentadas para validar o processo de construção de um sistema de tradução simultânea de fala.

O resultado final está disponível no arquivo translator.py, os requisitos estão presentes em requirements.txt. Abaixo, há uma descrição geral do sistema e suas funcionalidades.

Código

Este código cria uma aplicação gráfica para tradução em tempo real, permitindo ao usuário interagir de forma fluida com a interface e realizar traduções instantâneas a partir de comandos de voz. A interface gráfica é construída utilizando o Tkinter e o ttkbootstrap, que proporcionam um layout moderno e agradável. O usuário é capaz de selecionar entre quatro modelos de tradução diferentes: DeepL, OpenAI, M2M-100 e NLLB-200. A escolha do modelo influencia não só o processo de tradução, mas também as opções de idiomas disponíveis, que são ajustadas dinamicamente com base na seleção do modelo. Ao

selecionar os idiomas de entrada e saída, o usuário pode garantir que a tradução seja realizada conforme suas necessidades específicas.

A conversão de áudio em texto é feita em tempo real utilizando a biblioteca RealtimeSTT. Assim que o usuário começa a falar, a aplicação grava e converte o áudio em texto, enviando-o para o modelo de tradução escolhido. A tradução é realizada de forma contínua, permitindo que o texto seja traduzido enquanto o usuário continua falando. Dependendo do modelo escolhido, diferentes APIs e bibliotecas são utilizadas para o processamento da tradução. O modelo OpenAI, por exemplo, usa a API gpt-3.5-turbo para realizar a tradução, garantindo um alto nível de fluidez e adaptação cultural na conversão entre os idiomas. Já o DeepL e os modelos M2M-100 e NLLB-200 utilizam suas respectivas APIs, que são altamente otimizadas para realizar traduções para uma gama maior de idiomas.

A organização da interface gráfica é bem definida, com a tela dividida em duas áreas principais. Uma área de configurações permite que o usuário escolha o modelo de tradução e os idiomas desejados, enquanto a outra exibe o resultado da tradução em tempo real. O design da interface é intuitivo, com botões claros que permitem ao usuário iniciar e parar a gravação de áudio de forma simples. A aplicação também mantém o usuário informado sobre o status da tradução, com mensagens de feedback visíveis na tela, alertando-o sobre o progresso, erros ou falhas de operação.

O código implementa a tradução em tempo real, utilizando threads para garantir que o processamento de áudio e tradução não bloqueie a interface gráfica. Esse sistema de threads permite que a interface permaneça interativa, sem interrupções, enquanto o áudio é processado e traduzido. Assim, o usuário pode continuar interagindo com a aplicação enquanto o sistema está em funcionamento, sem se preocupar com lentidão ou travamentos.

Adicionalmente, as chaves de API necessárias para acessar os serviços de tradução, como o OpenAI e o DeepL, são armazenadas de forma segura em variáveis de ambiente, protegendo dados sensíveis. A configuração do ambiente é feita por meio do uso do arquivo .env, onde as chaves de API são carregadas automaticamente. Isso garante que as

credenciais não sejam expostas diretamente no código e possam ser facilmente alteradas ou configuradas em diferentes ambientes de execução.

A aplicação também garante que o processo de gravação de áudio seja encerrado corretamente quando o usuário fechar a janela da aplicação. Isso é feito por meio de uma função que interrompe o gravador de áudio e realiza o encerramento seguro da aplicação. Essa abordagem assegura que o processo de gravação não seja interrompido de maneira abrupta, prevenindo potenciais perdas de dados ou falhas.

Modelos Disponíveis

Os modelos disponíveis no código estão listados abaixo e são selecionáveis pelo usuário para realizar as traduções no sistema.

- **OpenAI**: Utiliza a API do OpenAI para tradução com o modelo "gpt-3.5-turbo". Ele pode traduzir entre os idiomas listados para o modelo OpenAI.
- **DeepL**: Utiliza a API do DeepL para tradução, que oferece suporte a vários idiomas. No código, é configurado para tradução entre os idiomas listados para o modelo DeepL.
- **M2M-100**: Um modelo de tradução neural treinado pela Meta, que suporta múltiplos idiomas diretamente (não requer tradução intermediária para um idioma comum). É uma opção local que utiliza a biblioteca transformers para realizar a tradução.
- **NLLB-200**: Modelo de tradução neural da Meta, que suporta 200 idiomas e é utilizado com a biblioteca transformers. Este modelo também é uma opção local, sem a necessidade de uma API externa.

Idiomas Disponíveis

Os idiomas disponíveis no código são definidos pelos modelos e mapeados na variável LANGUAGES. Eles são utilizados de acordo com o modelo selecionado para a tradução. O usuário pode escolher entre essas opções de idiomas de entrada e saída ao configurar a interface. Idiomas como japonês e coreano foram escolhidos para serem testados devido ao fato de seus alfabetos e sistemas de escrita não serem baseados no ocidental.

- **OpenAI:**
    - Portuguese (PT)
    - English (EN)
    - German (DE)
    - Spanish (ES)
    - French (FR)
    - Korean (KO)
- **DeepL:**
    - Portuguese (PT)
    - English (EN-US)
    - German (DE)
    - Spanish (ES)
    - French (FR)
    - Korean (KO)

- **M2M-100:**
    - Portuguese (pt)
    - English (en)
    - Spanish (es)
    - French (fr)
    - German (de)
    - Korean (ko)
- **NLLB-200:**
    - Portuguese (por_Latn)
    - English (eng_Latn)
    - Spanish (es_Latn)
    - French (fra_Latn)
    - German (deu_Latn)
    - Korean (kor_Hang)

Avisos de Interface

Os avisos são exibidos em uma interface gráfica com feedback ao usuário, geralmente utilizando o widget status_frame.

1. **Erro: Nenhum modelo selecionado**: Este aviso é exibido quando o usuário tenta selecionar os idiomas, mas nenhum modelo foi escolhido.

2. **Erro: Selecione ambos os idiomas**: Aparece quando o usuário não escolhe o idioma de entrada ou de saída ao selecionar os idiomas.
3. **Erro: Nenhum modelo ou idioma selecionado**: Este aviso é mostrado quando o usuário tenta iniciar o processo de tradução sem ter escolhido um modelo ou idiomas.
4. **Já há uma gravação em andamento**: Exibido quando o usuário tenta iniciar a gravação enquanto já há uma gravação em andamento.
5. **Erro ao traduzir: {e}**: Caso ocorra algum erro durante o processo de tradução, este aviso mostra a mensagem de erro gerada pela exceção.
6. **Gravação encerrada**: Aviso que informa que a gravação foi parada corretamente.
7. **Processando... Fale agora**: Mensagem informando que o sistema está pronto para processar a fala do usuário.
8. **Carregando sistema de tradução...**: Aviso que indica que o sistema está sendo carregado e preparado para a tradução.

Observações

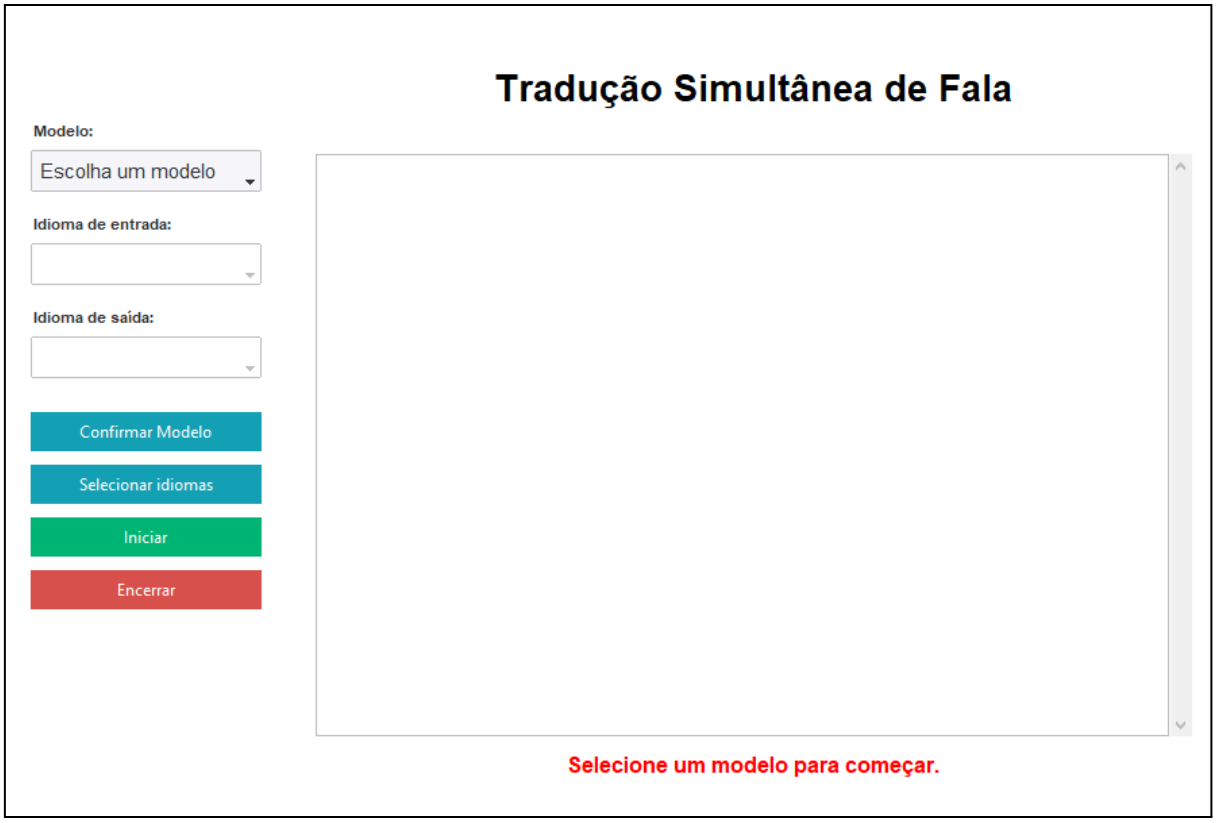
Algumas alterações não foram implementadas, mas são de uma importância primordial para a flexibilidade do uso do sistema em diferentes plataformas e dispositivos. Uma delas é a seleção do dispositivo de entrada de áudio, que não pode ser feita com o código atual, somente alterando manualmente o index referente ao microfone que o usuário deseja utilizar. Para analisar qual o index do dispositivo de entrada de áudio desejado, é possível usar o código abaixo para obter todas as informações necessárias.

```
import pyaudio
```

```
def listar_dispositivos_audio():
    audio = pyaudio.PyAudio()
    print("Dispositivos de áudio disponíveis:\n")

    for i in range(audio.get_device_count()):
        dispositivo = audio.get_device_info_by_index(i)
        print(f"Índice: {i}")
        print(f"Nome: {dispositivo['name']}")
        print(f"Entradas (input channels):
{dispositivo['maxInputChannels']}")
        print(f"Saídas (output channels):
{dispositivo['maxOutputChannels']}")
        print(f"Taxa padrão: {dispositivo['defaultSampleRate']} Hz")
        print("-" * 40)

    audio.terminate()

listar_dispositivos_audio()
```

Uma outra observação é sobre a capacidade de idiomas suportados por cada modelo. Como a quantidade de idiomas de entrada e saída variam de acordo com o modelo usado, foi decidido que para esta etapa seriam utilizados somente alguns idiomas para teste, incluindo aqueles que apresentam um alfabeto completamente diferente do ocidental.

Demonstração



Demo.mp4

Foi feita uma demonstração, apresentando os tempos de inferência em tempo real, a qualidade das traduções e as funcionalidades do sistema de tradução.