UNIVERSIDADE DE SÃO PAULO - USP

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO - ICMC

DAVI FAGUNDES FERREIRA DA SILVA - 12544013
ANTONIO MOREIRA - 9779242
DANIEL PENNA CHAVES BERTAZZO - 10349561
FABRÍCIO NASCIMENTO DE LIMA - 15824942

SCC0633 - PROCESSAMENTO DE LINGUAGEM NATURAL
GRAMATIQUIZ

SÃO CARLOS
2025

SUMÁRIO

INTRODUÇÃO

Com o objetivo de facilitar e tornar mais divertido o estudo de gramática, o grupo apresenta o *GramatiQuiz* uma solução capaz de ouvir uma sentença dita pelo aluno e propor um *quiz* para adivinhar qual a função morfológica das palavras contidas na sentença. A Figura 1 demonstra o funcionamento básico do projeto.

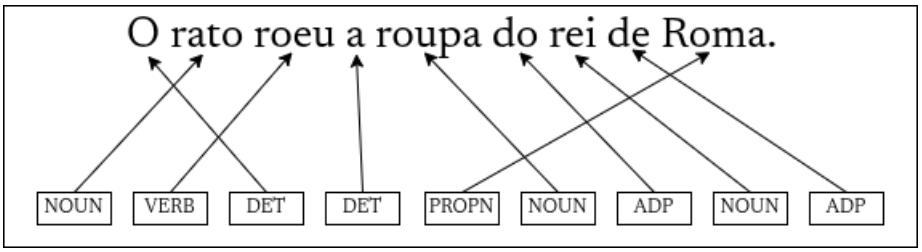

Figura 1 – Quiz esquematizando a proposta do grupo. Gerado com draw.io.

O software desenvolvido utiliza tecnologias de processamento de linguagem natural (PLN), como reconhecimento automático de fala (*ASR*) e análise morfológica (*POS tagging*) para transcrever a fala e analisar o conteúdo da mesma. Dessa forma, é possível fornecer um retorno educacional imediato, permitindo que o aluno compreenda melhor a estrutura gramatical da língua portuguesa por meio da própria fala.

1 PROPOSTA SIMBÓLICA

1.1 A Aplicação

Visando a feira de ciências na USP, a aplicação tem como público alvo os alunos do ensino médio. Deste modo, os itens a seguir descrevem as iterações dos usuários com a plataforma:

- Após habilitar o microfone do navegador o aluno aperta o botão de gravar e diz uma sentença à sua escolha. Quando a sentença é finalizada ele deve parar a gravação;
- A sentença dita é exibida para a confirmação do aluno, caso deseje alterar algo. Em caso de aceite deverá clicar em *confirmar*;
- Por fim, uma tela será exibida para o aluno realizar o *quiz* morfológico. Ao concluir será exibida uma mensagem informando em quantas tentativas ele conseguiu.

Uma representação visual do fluxo da aplicação pode ser vista na Figura 2.

1.2 Arquitetura do projeto

A arquitetura proposta é composta por três módulos principais que serão explicadas adiante:

- *Frontend* (*Vue.js* + *Vite* + *Bulma*)
- ASR-API (Reconhecimento Automático de Fala com *Whisper* via *FastAPI*)
- POS-API (Análise Morfológica com *Stanza* via *FastAPI*)

Cada módulo é responsável por uma etapa fundamental no processo de entrada e análise da fala.

1.3 Explicação da arquitetura

1.3.1 Frontend

Desenvolvido com *Vue.js*, utilizando o *Vite* como ambiente de desenvolvimento e o *Bulma* como *framework CSS*, o *frontend* permite a captação de áudio diretamente do navegador e exibe os resultados da transcrição e análise morfológica. É também no *frontend* onde é feito o *quiz* para aluno. A interface é responsiva, amigável e de fácil interação para estudantes.

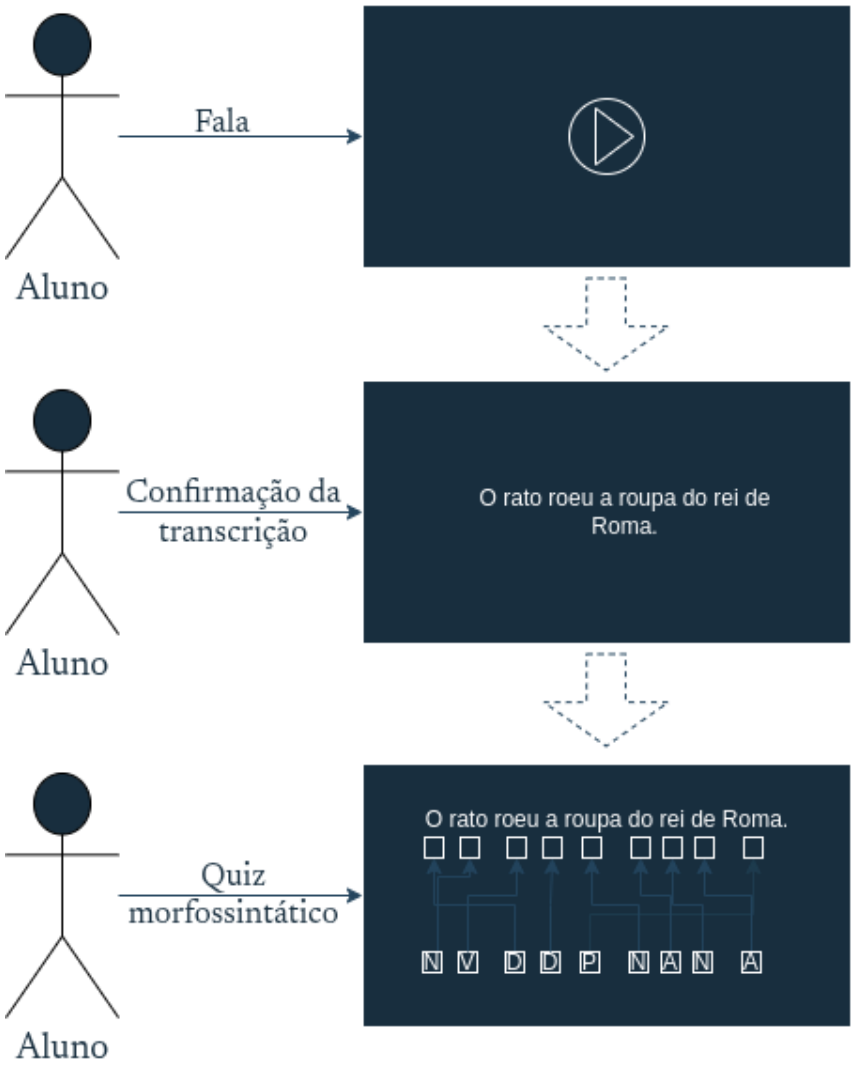

Figura 2 – Iterações do aluno com a aplicação.

1.3.2 ASR-API

Este módulo funciona como um intermediário entre o *frontend* e o modelo *Whisper* da *OpenAI*. Ele recebe os dados de áudio, em *base64*, enviados pelo usuário, envia ao *Whisper* para transcrição (áudio → texto) e retorna o texto transcrito para ser processado. Devido à grande complexidade computacional para processar com velocidade e assertividade os áudios recebidos, a API de transcrição fica hospedada no Google Cloud Platform.

1.3.3 POS-API

Após a transcrição, o texto é enviado para o módulo de análise morfológica, que utiliza o *Stanza* com modelo treinado para o idioma português. A POS-API realiza a etiquetagem das palavras (*part-of-speech tagging*), identificando a classe gramatical de cada termo da frase. O resultado é então devolvido ao *frontend* para exibição ao usuário.

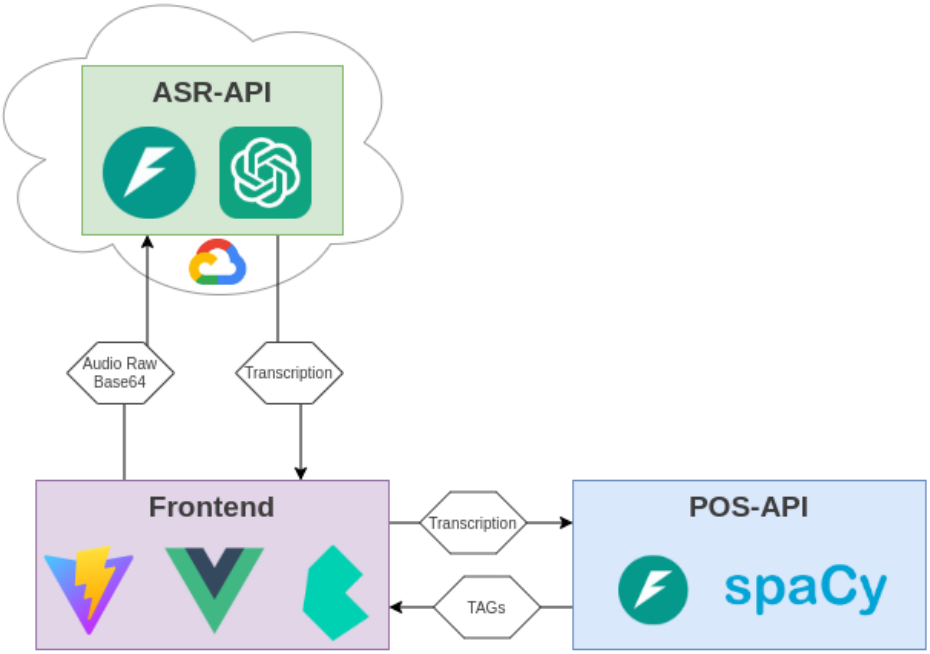

Figura 3 – Arquitetura do projeto

1.4 Recursos e ferramentas utilizados

- Whisper (OpenAI): Modelo de reconhecimento automático de fala (ASR). Converte áudio em texto com alta precisão, suporta múltiplos idiomas e é ideal para tarefas de transcrição, legendagem e acessibilidade.
- Stanza: Uma biblioteca de Python desenvolvida pela Universidade de Stanford para processamento de linguagem natural (NLP). Ela oferece ferramentas como tokenização, lematização, análise morfológica, análise sintática e reconhecimento de entidades nomeadas, funcionando em vários idiomas.
- FastAPI: Framework moderno para construção de APIs web em Python, com foco em alta performance e suporte nativo a validação de dados, documentação automática e integração com ferramentas de machine learning.
- Vue.js: Framework JavaScript progressivo e reativo para desenvolvimento de interfaces ricas e dinâmicas. Fácil de integrar e escalar, possui ótimo suporte à componentização.
- Vite: Ferramenta de *build* e servidor de desenvolvimento extremamente rápida. Utiliza ES Modules para melhorar o tempo de atualização durante o desenvolvimento e Rollup para builds finais otimizadas.
- Bulma: Framework CSS leve e moderno, baseado em Flexbox. Permite construir interfaces responsivas de forma simples e elegante, com uma sintaxe intuitiva e limpa.

1.5 Córpus Escolhidos

Para demonstrar a qualidade do *Whisper* e do analisador morfológico do *Stanza* o grupo selecionou dois corpus para analizar.

- Common Voice: conjunto de dados fornecido pela Mozilla composto por gravações de voz e suas respectivas transcições, coletadas de voluntários em várias línguas, no nosso caso usaremos apenas a língua portuguesa.
- Porttinari: O corpus Porttinari é um conjunto de dados brasileiro público composto por textos jornalísticos em português, que estão etiquetados com categorias morfossintáticas para facilitar a análise linguística.

1.6 Testes e Resultados

Fez-se testes para averiguar o funcionamento das ferramentas escolhidas para a parte de processamento de linguagem natural do GramatiQuiz.

1.6.1 Whisper

O modelo de transcrição escolhido para ser utilizado no projeto foi o *Whisper* da *OpenAI*. A variante testada em um ambiente virtual do Google Colaboratory foi o *whisper-tiny* devido a mudanças no projeto: inicialmente, o objetivo era o desenvolvimento de um software embarcado, por isso o emprego de um modelo menor. A métrica escolhida foi o *BLEU* score, no qual o modelo obteve uma pontuação de 0,3. Já o modelo maior (*whisper large-v3*, o qual será utilizado no projeto final), testado em uma amostra menor do dataset (500 instâncias) devido a limitações de *GPU*, obteve um *BLEU* score de 0,5, bem superior, podendo ser mais alto ainda caso avaliado em todo o corpus (aproximadamente 21 mil instâncias). Este modelo, quando testado anteriormente em seu lançamento, obteve taxa de 5.9% de erro.(JONGWOOK,)

1.6.2 Stanza

O Stanza, pacote feito pela universidade de Stanford para PLN, foi testado utilizando o Porttinari-base, um treebank multigênero feito pelo NILC ICMC USP (foram utilizados os primeiros 500 registros por limitações do ambiente de teste.). Em ambos, teve uma performance bem aceitável, sendo que no último teve 92% de acurácia.

1.7 Considerações Finais

O sistema desenvolvido oferece uma proposta inovadora e acessível para o aprendizado da língua portuguesa, unindo tecnologias de ponta em reconhecimento de fala e processamento de linguagem. A arquitetura modular garante escalabilidade e flexibilidade para futuras melhorias,

como incorporação de correções automáticas, feedback personalizado e expansão para outras funcionalidades educacionais. Para a segunda parte do projeto já está planejada uma melhora no *frontend* visando uma melhora na experiência do usuário (*UX*).

REFERÊNCIAS

JONGWOOK. <https://github.com/openai/whisper/discussions/1762>. Acesso em: 27 de abril de 2025. Citado na página 6.