UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Engenharia de Computação

Trabalho de Conclusão de Curso

VideoLearnAI: LLM Powered Web Application para aprendizagem ativa com vídeos do Youtube

Kevin Castro Weitgenant

Pelotas, 2025

Kevin Castro Weitgenant

VideoLearnAI: LLM Powered Web Application para aprendizagem ativa com vídeos do Youtube

Trabalho de Conclusão de Curso apresentado ao Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Tiago Primo
Coorientador: Prof. Dr. Marilton Sanchotene de Aguiar

Pelotas, 2025

Insira AQUI a ficha catalográfica
Quando finalizado o trabalho, deve ser solicitada através do Sistema Cobalto
Biblioteca – Cadastro – Ficha catalográfica.

Kevin Castro Weitgenant

VideoLearnAI: LLM Powered Web Application para aprendizagem ativa com vídeos do Youtube

Trabalho de Conclusão de Curso aprovado, como requisito parcial, para obtenção do grau de Bacharel em Engenharia de Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 16 de março de 2025

Banca Examinadora:

Prof. Dr. Marilton Sanchotene de Aguiar (orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Paulo Roberto Ferreira Jr.
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Ricardo Matsumura Araujo
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Luciano da Silva Pinto
Doutor em Biotecnologia pela Universidade Federal de Pelotas.

Dedico. . .

AGRADECIMENTOS

Agradeço. . .

Só sei que nada sei.
— SÓCRATES

RESUMO

WEITGENANT, Kevin Castro. **VideoLearnAI: LLM Powered Web Application para aprendizagem ativa com vídeos do Youtube**. Orientador: Tiago Primo. 2025. 40 f. Trabalho de Conclusão de Curso (Engenharia de Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

Este trabalho apresenta o desenvolvimento de uma plataforma educacional como serviço (SaaS) que utiliza Inteligência Artificial para aprimorar a experiência de aprendizagem com conteúdo em vídeo. O sistema implementa cinco funcionalidades principais: melhoria automática da legibilidade de legendas, geração de capítulos, transcrição sincronizada, geração de quizzes interativos e um sistema de bate-papo contextual com o conteúdo do vídeo. A solução emprega Large Language Models (LLMs) e arquitetura Transformer para processar e transformar o conteúdo audiovisual em material educacional interativo. A implementação foi realizada com foco em escalabilidade e performance, utilizando processamento em GPU e técnicas modernas de desenvolvimento de software. Os resultados demonstram o potencial da plataforma para transformar vídeos em experiências de aprendizagem mais engajadoras e efetivas.

Palavras-chave: palavrachave-um; palavrachave-dois; palavrachave-tres; palavrachave-quatro.

RESUMO

WEITGENANT, Kevin Castro. **AI-Powered Educational Platform: Transforming Video Content into Interactive Learning Experiences**. Orientador: Tiago Primo. 2025. 40 f. Trabalho de Conclusão de Curso (Engenharia de Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

This work presents the development of an educational Software as a Service (SaaS) platform that leverages Artificial Intelligence to enhance video-based learning experiences. The system implements five main functionalities: automatic subtitle readability improvement, chapter generation, synchronized transcription, interactive quiz generation, and a contextual chat system for video content. The solution employs Large Language Models (LLMs) and Transformer architecture to process and transform audiovisual content into interactive educational material. The implementation focused on scalability and performance, utilizing GPU processing and modern software development techniques. The results demonstrate the platform's potential for transforming videos into more engaging and effective learning experiences.

Palavras-chave: keyword-one; keyword-two; keyword-three; keyword-four.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
NUMA	Non-Uniform Memory Access
SIMD	Single Instruction Multiple Data
SMP	Symmetric Multi-Processor
SPMD	Single Program Multiple Data

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o consumo de conteúdo educacional em vídeo tem crescido exponencialmente, impulsionado por plataformas como YouTube, Coursera e Udemy. Hoje, é possível encontrar aulas completas de universidades de altíssimo nível, como MIT, Harvard e Stanford, gratuitamente disponíveis online. No entanto, apesar da abundância de material de qualidade, muitos usuários enfrentam dificuldades em absorver e reter conhecimento de forma eficiente. A maioria das pessoas consome esses conteúdos de maneira passiva, apenas assistindo aos vídeos sem um envolvimento ativo com o material. Isso limita a retenção e a compreensão das informações.

A aprendizagem ativa, por outro lado, é um modelo comprovadamente mais eficaz, pois envolve o estudante em processos como resumo, questionamento, reorganização do conteúdo e interação com o material. Pesquisas mostram que métodos ativos de estudo, como fazer perguntas sobre o conteúdo, testar-se frequentemente e organizar a informação de forma estruturada, levam a um aprendizado mais profundo e duradouro.

Diante desse cenário, este trabalho apresenta o desenvolvimento de um Software as a Service (SaaS) voltado para transformar o consumo passivo de vídeos educacionais em um processo de aprendizagem ativa. A solução utiliza modelos de linguagem natural (LLMs) para reestruturar legendas em textos mais legíveis, gerar capítulos automáticos, fornecer resumos e permitir interações como perguntas e respostas sobre o conteúdo. Além disso, o sistema oferece quizzes dinâmicos para reforçar o aprendizado e um mecanismo para salvar o progresso dos usuários, incentivando um envolvimento mais estruturado com os vídeos.

O desenvolvimento do SaaS seguiu uma abordagem iterativa. A arquitetura da aplicação integra tecnologias como FastAPI, Next.js e Transformers, além de estratégias de otimização para garantir eficiência e escalabilidade. A validação da ferramenta inclui métricas de desempenho e feedback dos usuários, avaliando sua eficácia na melhoria da compreensão e retenção do conhecimento.

Com esta pesquisa, buscamos não apenas oferecer uma ferramenta inovadora para aprendizado com vídeos, mas também contribuir para a democratização da edu-

cação de qualidade, permitindo que qualquer pessoa tenha acesso a um método mais eficaz para extrair o máximo de conhecimento dos conteúdos disponíveis online.

1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um Software as a Service (SaaS) que transforme o consumo passivo de vídeos educacionais em um processo de aprendizagem ativa. Para isso, a plataforma utilizará inteligência artificial para melhorar a legibilidade das legendas, gerar resumos, estruturar conteúdos em capítulos, criar quizzes interativos e permitir interações diretas com o conteúdo por meio de perguntas e respostas. O foco é tornar o aprendizado com vídeos mais eficiente, estruturado e acessível, permitindo que qualquer pessoa aproveite melhor o vasto acervo educacional disponível online.

1.2 Objetivos específicos

Para atingir o objetivo geral, este trabalho busca:

- Desenvolver um sistema que reestruture legendas de vídeos em textos mais legíveis e organizados, facilitando a compreensão.
- Implementar um mecanismo para geração automática de capítulos e resumos, permitindo uma navegação mais eficiente pelo conteúdo.
- Criar um módulo de perguntas e respostas, possibilitando interações com o vídeo de forma contextualizada.
- Desenvolver um sistema de quizzes automáticos baseados no conteúdo dos vídeos, reforçando o aprendizado ativo.
- Implementar um sistema de salvamento de progresso para permitir que usuários retomem facilmente seus estudos.

1.3 Estrutura do trabalho

Este trabalho está organizado da seguinte forma:

- **Capítulo 2 – Revisão da Literatura**: apresenta os conceitos fundamentais de aprendizagem ativa e modelos de linguagem natural (LLMs), que embasam o desenvolvimento da aplicação.
- **Capítulo 3 – Metodologia**: descreve o processo de desenvolvimento do SaaS, incluindo a definição de requisitos, prototipação, escolha de tecnologias e critérios de avaliação.

- **Capítulo 4 – Desenvolvimento da Aplicação**: detalha as funcionalidades do sistema, explicando a implementação de cada módulo e os desafios enfrentados.
- **Capítulo 5 – Resultados**: analisa o desempenho da aplicação e apresenta o feedback dos usuários, avaliando o impacto da ferramenta na experiência de aprendizado.
- **Capítulo 6 – Conclusão**: discute os objetivos alcançados, as principais contribuições do trabalho e sugestões para aprimoramentos futuros.

2 SOLUÇÕES RELACIONADAS

2.1 youlearn

2.2 studyfetch

3 TECNOLOGIAS UTILIZADAS

Este capítulo apresenta as principais tecnologias utilizadas no desenvolvimento do projeto, que adota uma arquitetura distribuída com Next.js como principal framework full-stack, complementado por um backend adicional em Python para funcionalidades específicas.

3.1 Linguagens de Programação

3.1.1 Python

Python foi utilizado como backend complementar, focado em duas funcionalidades específicas: extração de dados do YouTube e operações com o modelo Segment Anything (SaT). Esta escolha foi motivada pela disponibilidade de bibliotecas especializadas no ecossistema Python que facilitam estas operações específicas, não prontamente disponíveis no ambiente Node.js/TypeScript.

3.1.2 TypeScript

TypeScript foi a linguagem principal do projeto, utilizada tanto no frontend quanto no backend principal desenvolvido com Next.js. Sua tipagem estática trouxe maior segurança e manutenibilidade ao código, especialmente importante em uma aplicação full-stack onde a consistência entre frontend e backend é crucial.

3.2 Frameworks e Bibliotecas

3.2.1 FastAPI

FastAPI foi utilizado para criar um serviço backend complementar em Python, expondo endpoints para funcionalidades específicas de extração de dados do YouTube e operações com o modelo SaT. Um diferencial importante foi a geração automática de clients TypeScript através do OpenAPI, permitindo uma integração tipo-segura e seamless com o frontend Next.js. Esta capacidade de TypeScript code generation eliminou a necessidade de definir tipos manualmente e garantiu consistência na co-

municação entre os serviços.

3.2.2 Next.js

Next.js atuou como o framework full-stack principal, gerenciando tanto o frontend quanto a maior parte do backend da aplicação. Através de seus recursos de API Routes e Server Components, foi possível construir uma aplicação completa com renderização híbrida, manipulação de estado servidor/cliente e APIs RESTful. Sua arquitetura permitiu manter a maioria das operações de backend centralizadas, recorrendo ao serviço Python apenas para funcionalidades específicas.

3.2.3 Vercel AI SDK

O **Vercel AI SDK** desempenhou um papel fundamental na implementação de funcionalidades de inteligência artificial diretamente no **Next.js**, proporcionando uma integração eficiente com modelos de linguagem natural (LLMs).

Uma das principais vantagens dessa biblioteca é a facilidade na construção de interfaces de usuário interativas e dinâmicas para aplicações baseadas em IA. O SDK permite o **streaming de respostas**, garantindo uma experiência mais fluida para o usuário, sem a necessidade de aguardar a conclusão total do processamento. Essa abordagem melhora significativamente a experiência do usuário (**UX**), tornando as interações mais ágeis e responsivas.

Além do texto, outros elementos da interface, como **quizzes gerados dinamicamente**, também são entregues por meio de streaming. Isso significa que as perguntas e respostas dos quizzes não precisam ser completamente processadas antes de serem exibidas, pois são disponibilizadas conforme são geradas pelo modelo de IA. Todo esse processo é abstraído pelo **Vercel AI SDK**, simplificando a implementação e reduzindo a complexidade no desenvolvimento da interface.

Dessa forma, o uso do **Vercel AI SDK** permitiu a criação de uma **UI altamente responsiva e eficiente**, eliminando a necessidade de comunicação constante com o backend Python para determinadas operações de IA, otimizando o desempenho e a escalabilidade da aplicação.

3.2.4 Drizzle ORM

O **Drizzle ORM** foi utilizado como ORM principal no **Next.js**, oferecendo uma interface *type-safe* para interações com o banco de dados **PostgreSQL**. Sua integração direta com **TypeScript** permitiu manter a consistência de tipos entre o banco de dados e a aplicação.

Além disso, o uso do **Drizzle ORM** faz sentido dentro da arquitetura **serverless** do projeto. Diferente de ORMs mais tradicionais, que podem apresentar desafios com conexões persistentes em ambientes serverless, o **Drizzle ORM** foi projetado para

ser leve e eficiente, garantindo melhor compatibilidade com essa abordagem. Dessa forma, a aplicação pode escalar dinamicamente sem sobrecarga desnecessária na comunicação com o banco de dados.

3.2.5 ShadCN

ShadCN forneceu componentes de UI reutilizáveis e personalizáveis, sendo utilizado exclusivamente na camada de frontend do Next.js para construir interfaces consistentes e acessíveis.

3.2.6 Zustand

Zustand gerenciou o estado global do frontend, complementando o gerenciamento de estado servidor/cliente do Next.js com uma solução leve e eficiente para estados efêmeros do cliente.

3.3 Modelos de Machine Learning

3.3.1 API GPT

A API GPT foi integrada primariamente através do Next.js, utilizando o Vercel AI SDK para streaming de respostas e gerenciamento eficiente de prompts.

3.3.2 Segment Anything (SaT)

O modelo **Segment Anything (SaT)** foi utilizado para segmentar as legendas extraídas de vídeos do **YouTube** em parágrafos, com o objetivo de melhorar a legibilidade e organização do texto transcrito. Essa segmentação estruturada facilitou a compreensão e a análise do conteúdo, tornando a experiência do usuário mais intuitiva e agradável.

Durante a fase de desenvolvimento, o modelo foi executado no backend utilizando **FastAPI**. No entanto, devido ao alto custo associado à manutenção de servidores com **GPU**, optou-se por migrar a execução para a infraestrutura **Beam Serverless GPU** em produção. Esse serviço permitiu a execução do modelo de forma escalável e sob demanda, reduzindo os custos operacionais sem comprometer significativamente o desempenho. A principal desvantagem foi uma pequena latência ocasionada pelos *cold starts*, mas isso foi minimizado devido a várias técnicas disponibilizadas pelo serviço.

A **Beam Serverless GPU** será explorada com mais detalhes no próximo capítulo.

3.4 Infraestrutura e Serviços em Nuvem

3.4.1 Open Next.js

Open Next.js otimizou o deploy da aplicação Next.js full-stack, garantindo performance adequada tanto para o frontend quanto para as API Routes do backend.

3.4.2 Google Cloud

Google Cloud Platform hospedou o backend Python complementar, enquanto também fornecia outros serviços de infraestrutura necessários para a aplicação.

3.4.3 Beam Serverless GPU

Beam Serverless GPU foi utilizado especificamente para o backend Python, fornecendo recursos de GPU para processamento de modelos de machine learning mais pesados.

3.4.4 Supabase

Supabase forneceu o banco de dados PostgreSQL e serviços de autenticação, sendo acessado principalmente através do backend Next.js via Drizzle ORM.

3.4.5 Sentry

Sentry monitorou tanto o ambiente Next.js quanto o serviço Python, fornecendo visibilidade completa sobre erros e performance em toda a aplicação.

3.5 Banco de Dados

3.5.1 PostgreSQL (via Supabase)

PostgreSQL foi utilizado como banco de dados principal, sendo acessado principalmente através do Next.js com Drizzle ORM, e ocasionalmente pelo backend Python quando necessário.

3.6 Ferramentas de Desenvolvimento

3.6.1 Visual Studio Code (VS Code)

VS Code foi o IDE principal, oferecendo excelente suporte tanto para o desenvolvimento em Next.js/TypeScript quanto para Python.

3.6.2 Cursor

Cursor complementou o ambiente de desenvolvimento com recursos de IA para autocomplete e refatoração, sendo especialmente útil no desenvolvimento full-stack.

3.6.3 v0.dev

v0.dev auxiliou na prototipagem rápida de componentes frontend para o Next.js, acelerando o desenvolvimento da interface do usuário.

3.6.4 OpenAPI TypeScript Code Generator

A geração automática de código TypeScript a partir das especificações OpenAPI do FastAPI foi uma ferramenta crucial no desenvolvimento, criando automaticamente tipos e clients para todas as APIs Python. Isto garantiu uma integração tipo-segura entre o frontend Next.js e o backend Python, reduzindo erros de integração e melhorando a experiência de desenvolvimento.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 Processo de Desenvolvimento

4.1.1 Definição de Requisitos

4.1.2 Prototipação

4.1.3 Desenvolvimento Iterativo

4.2 Arquitetura

4.2.1 Visão Geral da Arquitetura

5 DESENVOLVIMENTO DA APLICAÇÃO

5.1 Melhoria da Legibilidade das Legendas

5.1.1 O Problema da Legibilidade

As legendas automáticas de vídeos frequentemente apresentam problemas de formatação e segmentação, dificultando a compreensão do conteúdo. Esse problema ocorre porque as transcrições brutas costumam ser geradas como um fluxo contínuo de palavras, sem uma estrutura clara de frases e parágrafos. Além disso, quebras de linha mal posicionadas afetam a fluidez da leitura.

Além de melhorar a legibilidade das legendas, tornar o conteúdo escrito mais organizado pode ser útil para quem prefere ler um vídeo em vez de assisti-lo. A leitura permite revisar rapidamente uma parte específica sem precisar voltar no vídeo, além de ajudar a decidir se vale a pena assistir aquele trecho ou se a informação já foi absorvida apenas lendo. Para algumas pessoas, ler pode ser simplesmente uma forma mais eficiente e rápida de consumir o conteúdo.

Para resolver essa questão, foram testadas abordagens baseadas em modelos de linguagem natural, buscando aprimorar a estrutura das legendas sem alterar seu conteúdo original.

5.1.2 Primeira Abordagem com LLMs

Inicialmente, foi utilizado um modelo de linguagem de grande porte (LLM) para segmentar e melhorar a legibilidade do texto das legendas. A implementação foi feita utilizando a biblioteca `instructor`, que permite a validação do texto gerado através do `pydantic`, garantindo conformidade com um formato estruturado.

Entretanto, essa abordagem apresentou desafios significativos:

- **Latência elevada**: O tempo de resposta do modelo era relativamente alto, tornando a solução pouco eficiente para processar grandes quantidades de legendas.
- **Alterações não desejadas no texto**: Apesar das instruções explícitas no

prompt para evitar adições ou remoções de palavras, o modelo ocasionalmente incluía frases como "Aqui está o seu texto otimizado"ou alterava partes do conteúdo original.

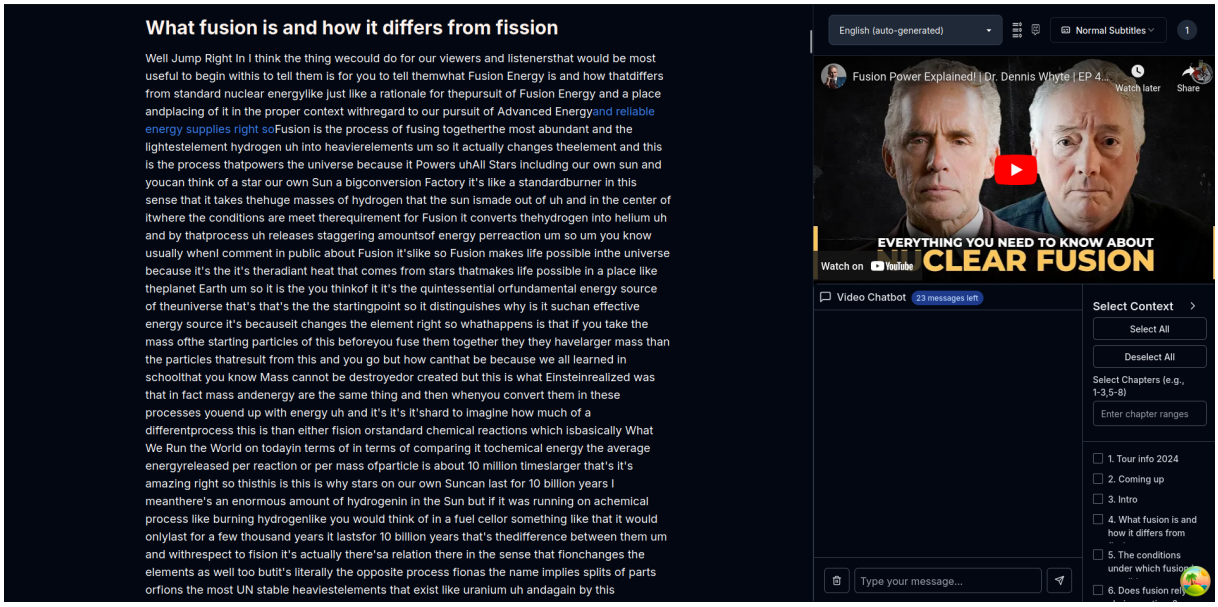
Esses fatores tornaram a abordagem com LLMs menos viável para o problema proposto.

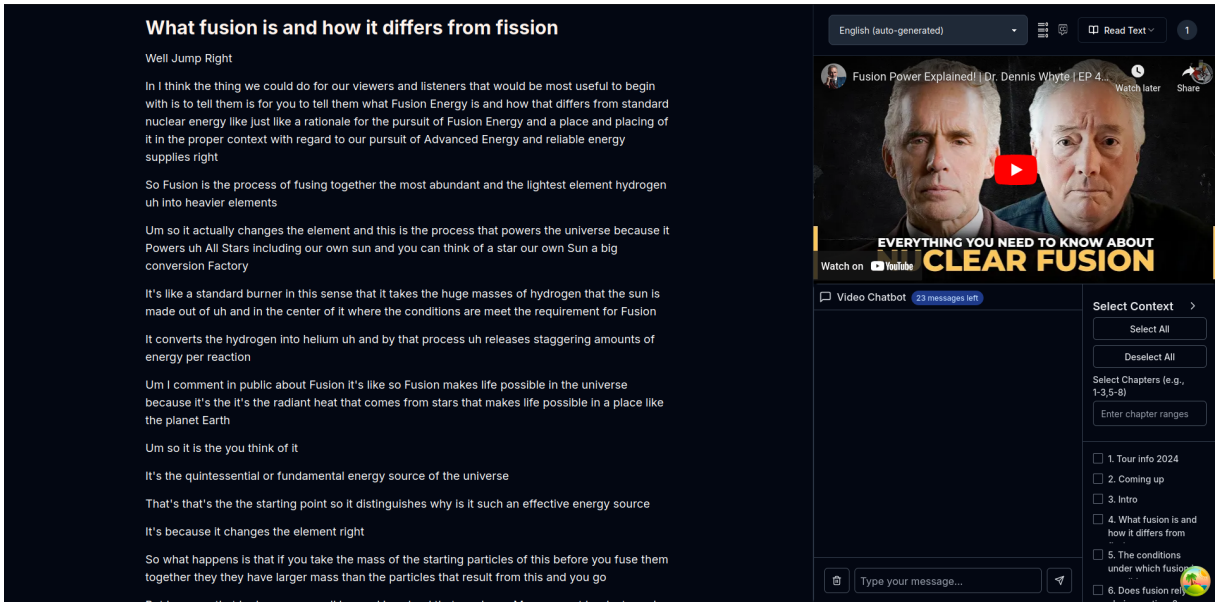
5.1.3 Implementação com Transformers

Diante das limitações dos LLMs, foi testada uma alternativa baseada em transformers especializados para segmentação de texto. A ferramenta wtpsplit mostrou-se uma solução eficiente para a melhoria da legibilidade das legendas.

Diferente dos LLMs, `wtpsplit` foca especificamente na segmentação de frases, apresentando vantagens como:

- **Baixa latência**: A segmentação ocorre de maneira rápida e eficiente.
- **Preservação do conteúdo original**: O modelo não adiciona ou remove palavras arbitrariamente, garantindo fidelidade ao texto original.
- **Facilidade de implementação**: A integração do `wtpsplit` ao pipeline de processamento foi direta, proporcionando bons resultados sem necessidade de ajustes complexos nos prompts.





5.1.4 Comparação e Resultados

Para avaliar o desempenho das abordagens, foram realizadas comparações entre os textos segmentados pelos LLMs e pelo `wtpsplit`. Os critérios analisados incluíram:

- Tempo de processamento
- Fidelidade ao texto original
- Legibilidade subjetiva (avaliação qualitativa)

Os resultados mostraram que `wtpsplit` proporcionou uma segmentação mais precisa e eficiente, superando os LLMs em termos de velocidade e fidelidade ao conteúdo original. Assim, a implementação final optou pelo uso de `wtpsplit` como solução principal para a melhoria da legibilidade das legendas.

5.2 Geração de Capítulos

A geração automática de capítulos em vídeos do YouTube é um processo essencial para melhorar a navegabilidade e compreensão do conteúdo. Para isso, adotamos um método baseado na estruturação do texto e na identificação de mudanças de tópico através de um modelo de linguagem de grande escala (LLM).

5.2.1 Algoritmo e Implementação

O algoritmo segue um fluxo de processamento em três etapas principais:

1. **Estruturação do texto:** O conteúdo transcrito do vídeo é segmentado em parágrafos coerentes, respeitando pausas naturais e mudanças de contexto no discurso. Essa segmentação melhora a compreensão e a análise subsequente do texto.

2. **Numeração dos parágrafos:** Cada parágrafo recebe um identificador numérico no início, o que permite um referenciamento direto na etapa seguinte.

3. **Identificação de mudanças de tópico:** Utilizamos um LLM para analisar o texto e determinar os parágrafos que marcam a transição entre tópicos distintos. O modelo retorna a lista de números dos parágrafos que representam pontos de mudança relevantes no conteúdo.

5.2.2 Integração com LLMs

Para a identificação das transições de tópico, utilizamos function calling para interagir com o LLM. Essa abordagem permite enviar a transcrição segmentada e numerada, solicitando ao modelo que indique os pontos de mudança mais relevantes. A função chamada retorna uma lista de parágrafos que representam as mudanças de tópico, possibilitando a geração automática de capítulos estruturados.

Previamente, foi tentado um método em que a function calling solicitava apenas o nome dos tópicos de transição e seu tempo correspondente. No entanto, observou-se um aumento significativo nas alucinações do modelo. Ao utilizar a numeração dos parágrafos, conseguimos reduzir esse problema, pois conhecendo a posição do parágrafo na transcrição, podemos determinar com maior precisão a sua posição temporal no vídeo.

Essa abordagem garante uma segmentação eficiente e adaptável, permitindo uma melhor compreensão do conteúdo dos vídeos e melhorando a experiência do usuário ao consumir informações em formato de vídeo.

5.3 Transcrição

A transcrição precisa de vídeos do YouTube é essencial para gerar textos mais confiáveis do que as legendas automáticas, que apresentam qualidade inferior especialmente para línguas que não são o inglês.

5.3.1 Processo Inicial com Whisper

No fluxo original utilizando o Whisper, o processo consistia em:

1. **Download do áudio** (comum a ambos os métodos): - Uso da biblioteca `pytubefix` para download - Progresso reportado ao usuário via SSE (Server-Sent Events) - Arquivo de áudio com qualidade balanceada entre precisão e velocidade

2. **Pré-processamento específico para Whisper**: - Divisão do áudio em segmentos de 10 minutos - Cortes preferencialmente em momentos de silêncio - Envio paralelo dos segmentos para transcrição - Necessidade decorrente das limitações do Whisper com arquivos longos

5.3.2 Migração para o Deepgram

Com a adoção do Deepgram, o fluxo foi otimizado:

1. **Download do áudio mantido**: - Mesmo processo via `pytubefix` com feedback via SSE - Eliminação da etapa de divisão do áudio

2. **Transcrição direta**: - Envio do arquivo de áudio completo - Processamento único sem necessidade de paralelização - Interface exibe estimativa de tempo restante - Capacidade nativa de lidar com longas durações

5.3.3 Comparativo Técnico

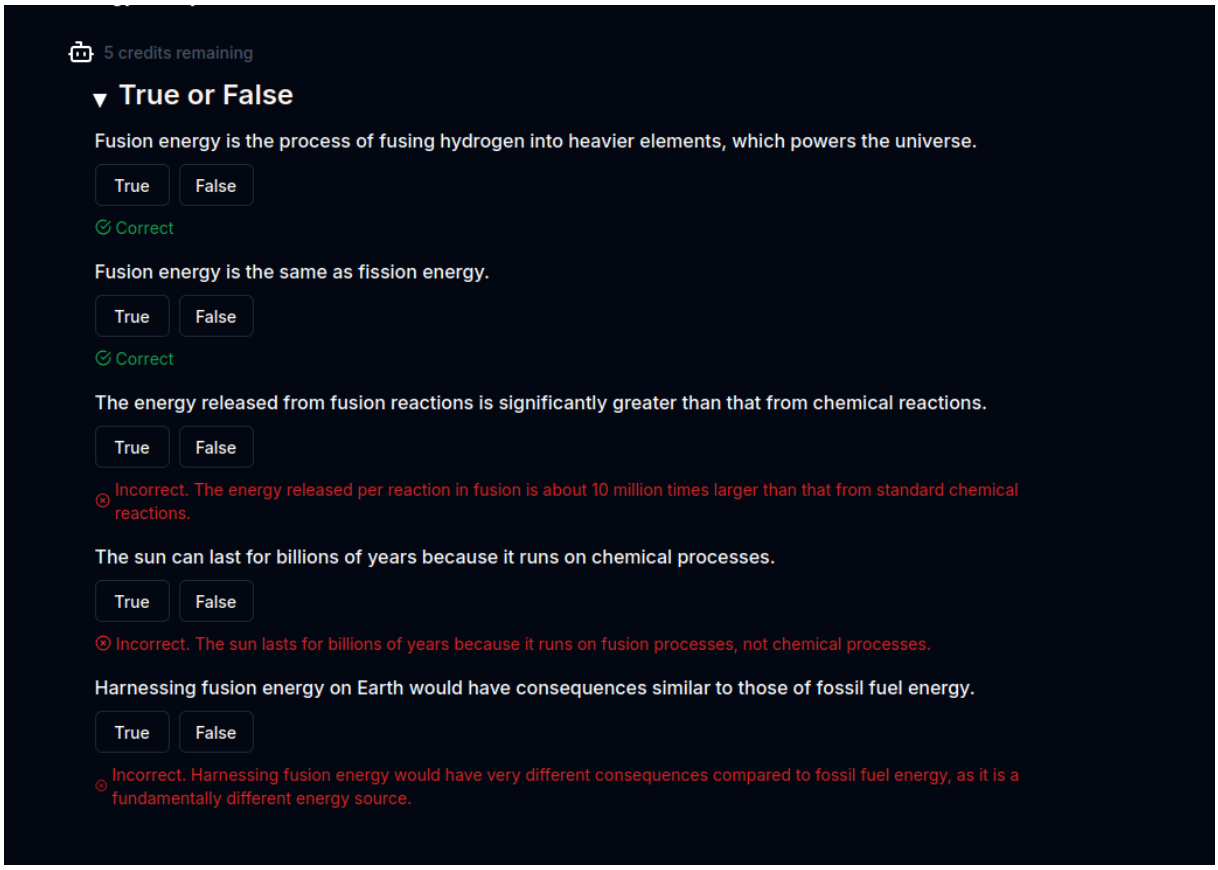
Característica	Whisper	Deepgram
Requer divisão do áudio	Sim	Não
Processamento paralelo	Necessário	Opcional
Feedback em tempo real	Apenas no download	Download + estimativa de transcrição
Limite de duração	10min/segmento	Até 4h/arquivo

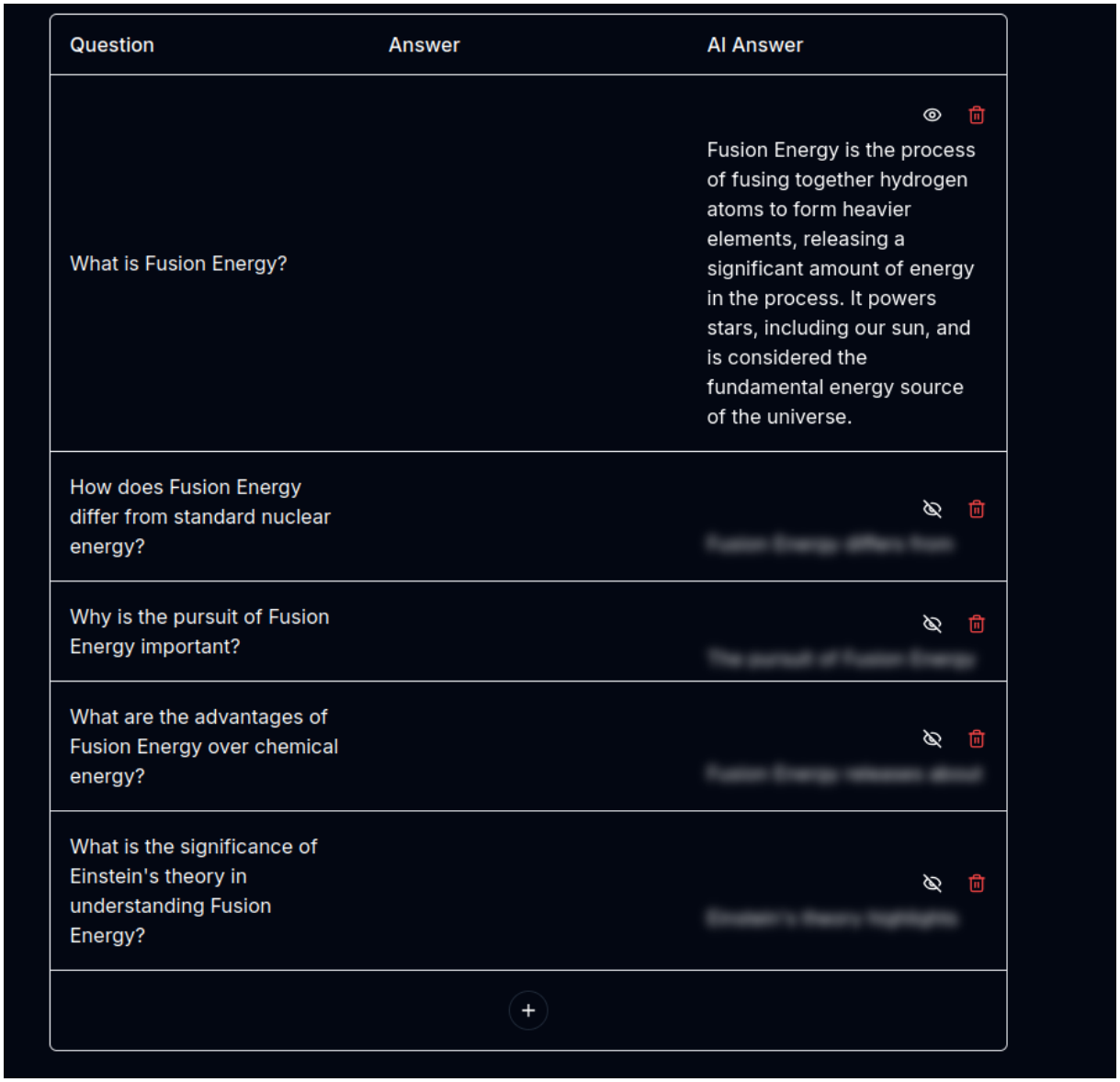
Tabela 1 – Diferenças na abordagem de transcrição

5.4 Geração de Quizzes

5.4.1 Extração de Conceitos-Chave

5.4.2 Geração via LLM





5.5 Bate-Papo com Vídeo

5.5.1 Processamento de Perguntas

5.5.2 Contextualização com Conteúdo

Durante a interação com o sistema, foi implementada uma funcionalidade que permite ao usuário selecionar capítulos específicos do vídeo que são relevantes para sua pergunta. Esta característica é especialmente importante considerando que, frequentemente, o conteúdo completo do vídeo pode exceder os limites da janela de contexto do modelo de linguagem.

Para garantir uma experiência transparente, o sistema exibe notificações em formato toast quando há risco de exceder a janela de contexto. Embora seja possível selecionar a legenda completa do vídeo, o usuário é alertado quando parte do conteúdo precisa ser truncada devido às limitações técnicas.

A geração de capítulos tornou-se um elemento fundamental neste processo, pois permite que o usuário faça uma seleção precisa do conteúdo pertinente à sua questão, otimizando assim a qualidade das respostas e o uso do contexto disponível.

Em perspectivas futuras, considera-se desenvolver uma abordagem mais agêntica, onde a própria LLM seria responsável por identificar e selecionar os capítulos relevantes para cada pergunta específica. No entanto, nesta primeira implementação, optou-se por manter o controle da seleção nas mãos do usuário.

5.5.3 Geração de Respostas

5.6 Desafios e Soluções

5.6.1 Obtenção dos dados do youtube

5.6.2 Utilização de GPU's em produção

6 RESULTADOS

6.1 Análise de Desempenho

6.2 Feedback dos Usuários

7 CONCLUSÃO

7.1 Objetivos Alcançados

7.2 Trabalhos Futuros

8 REFERÊNCIAS

REFERÊNCIAS

Apêndices

APÊNDICE A – Um Apêndice

Anexos

ANEXO A – Um Anexo

Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla.

Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla.

ANEXO B – Outro Anexo

Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla.

Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla. Bla blabla blablabla bla.