ISSN: 2237-826X
ARTIGO | PAPER

Geração automática de resumo em português a partir de conteúdo audiovisual em inglês: método, validação e aplicativo "caseiro"

Automatic generation of Portuguese summary from audiovisual content in English: method, validation, and "homemade" application

Keomas da Silva Monteiro[1], Hendrik Teixeira Macedo[2], Leonardo Nogueira Matos[3], Kalil Araújo Bispo[4]

[1] *Universidade Federal de Sergipe (UFS), Aracaju, SE, Brasil. ORCID: https://orcid.org/0009-0007-9656-9478*

[2] *Universidade Federal de Sergipe (UFS), Aracaju, SE, Brasil. ORCID: https://orcid.org/0000-0002-6477-756X*

[3] *Universidade Federal de Sergipe (UFS), Aracaju, SE, Brasil. ORCID: https://orcid.org/ 0000-0002-6302-3299*

[4] *Universidade Federal de Sergipe (UFS), Aracaju, SE, Brasil. ORCID: https://orcid.org/0000-0002-4587-732X*

Autor para correspondência/*Mail to:* Hendrik Teixeira Macedo, hendrik@dcomp.ufs.br

Recebido/*Submitted*: 28 de fevereiro de 2024; **Aceito/*Approved*:** 25 de setembro de 2025

Resumo

Introdução: a síntese audiovisual é essencial para democratizar o conhecimento, facilitar pesquisa e aprendizado, aprimorar a experiência do usuário e promover inclusão digital, mas a produção manual de resumos é trabalhosa e não é escalável. A IA automatiza esse processo, mas ainda não há uma solução automatizada completa, de baixo custo e de fácil uso. Este trabalho propõe um roteiro para criar um pipeline automatizado "caseiro" para gerar resumos em português a partir de vídeos com áudio em inglês. **Método**: para atingir o objetivo proposto, implementamos quatro algoritmos em pipeline para: (1) extrair o áudio do vídeo, (2) transcrevê-lo para texto, (3) resumir o texto na língua original e (4) traduzir o resumo para português. Os algoritmos usam modelos de aprendizado de máquina e são validados com métricas específicas para cada etapa: WER, CER, ROUGE, BLEU. **Resultados**: o trabalho apresenta o "Smart Summy", uma arquitetura e solução integrada para geração automática de resumos em português de vídeos em inglês, com execução em nuvem, sem necessidade de instalação ou entendimento de tecnologias por parte do usuário, e de interface leve, simples e intuitiva. Avaliações quantitativas das etapas do pipeline com uso das métricas estabelecidas demonstram altíssima qualidade na transcrição, boa qualidade do resumo em inglês e ótimo poder de tradução para o português. **Conclusão**: o "Smart Summy" e seu roteiro de uso orientado demonstra poder preencher uma lacuna ainda existente no que concerne à integração de ferramentas (ou modelos) de Inteligência Artificial para automatização da produtividade do usuário "comum".

Palavras-chave: Síntese audiovisual; Tradução inglês-português; Inteligência Artificial; Modelos de aprendizado de máquina; Aplicação Web.

Abstract

Introduction: *audiovisual synthesis is essential to democratize knowledge, facilitate research and learning, enhance user experience, and promote digital inclusion. However, manual summarization is laborious and not scalable. AI automates this process, but the lack of a complete, low-cost, and user-friendly automated solution remains. This work proposes a roadmap to create a "home-made" automated pipeline to generate summaries in Portuguese from videos in English.* ***Method:*** *to achieve the proposed goal, we implemented four algorithms in a pipeline to: (1) extract audio from the video, (2) transcribe it into text, (3) summarize the text in the original language, and (4) translate the summary into Portuguese. The algorithms use machine learning models and are validated with specific metrics for each step: WER, CER, ROUGE, BLEU* ***Results:*** *the work presents the "Smart Summy," an architecture and integrated solution for automatic generation of Portuguese summaries from videos in English, with cloud execution, no need for installation or understanding of technologies from the user's part, and a lightweight, simple, and intuitive interface. Quantitative evaluations of pipeline stages using established metrics demonstrate very high transcription quality, good quality of the English summary, and excellent translation power to Portuguese.* ***Conclusions:*** *the "Smart Summy" and its guided usage roadmap demonstrate the ability to fill an existing gap regarding the integration of Artificial Intelligence tools (or models) for automating the productivity of the "average" user.*

Keywords: *Audiovisual synthesis; English-Portuguese translation; Artificial Intelligence; Machine learning models; Web application.*

INTRODUÇÃO

Com o advento da era digital e a proliferação de conteúdos audiovisuais, a necessidade de ferramentas eficazes para sintetizar e extrair informações de vídeos tornou-se cada vez mais importante. É possível observar ao menos quatro dimensões de abrangência da relevância da síntese audiovisual. A primeira delas diz respeito à democratização do conhecimento. A produção massiva de vídeos, em diversos idiomas e sobre diferentes temas, nem sempre acompanha a capacidade em assisti-los e compreendê-los. Resumos textuais podem tornar esses conteúdos acessíveis a um público mais amplo, incluindo pessoas com deficiência auditiva ou com tempo limitado. A segunda trata da facilitação da pesquisa e do aprendizado. A vastidão de informações disponíveis em formato de vídeo exige ferramentas que auxiliem na busca rápida e eficiente por conteúdos relevantes. Resumos textuais permitem que usuários identifiquem o tema central de um vídeo e sua pertinência em relação à pesquisa

ou necessidade de aprendizado. Outra dimensão é o aprimoramento da experiência do usuário. A navegação por plataformas de conteúdo audiovisual pode ser otimizada com a presença de resumos textuais já que a pré-visualização do conteúdo facilita a escolha de vídeos relevantes e aumenta a satisfação do usuário. Finalmente, a geração de resumos textuais em diferentes idiomas contribui para a inclusão digital de indivíduos que não dominam o idioma original do vídeo, ampliando o acesso à informação e promovendo diversidade cultural.

Mas como se fazer valer desses benefícios? Sabemos que o procedimento manual é extremamente trabalhoso devido à necessidade de compreensão profunda do conteúdo audiovisual, tradução precisa, geração de um resumo conciso e informativo na língua de destino e, principalmente, a falta de escalabilidade do processo. Em quanto tempo conseguiríamos resumir dezenas ou centenas de vídeos seguindo esse procedimento conservador?

O advento de ferramentas construídas sob a ótica da Inteligência Artificial (IA), particularmente desenvolvidas no subcampo do Aprendizado de Máquina (ML), revolucionaram esse procedimento tradicional de compreensão de áudio e vídeo (Zhang, Haddow, & Sennrich, 2022), tradução automática (Rivera-Trigueros, 2022) e geração automática de resumos (El-Kassas, Salama, Rafea, & Mohamed, 2021), tarefas típicas da subárea de Processamento de Linguagem Natural (PLN). Modelos como o Google Cloud Speech-to-Text[1] e o Amazon Transcribe[2], por exemplo, convertem áudio em texto com alta precisão, mesmo em ambientes ruidosos. Serviços como Google Translate[3] e o DeepL[4] oferecem tradução instantânea entre idiomas com alta fidelidade e chatbots como o OpenAI ChatGPT[5], Microsoft Copilot[6] e o Google Gemini[7] geram resumos automáticos de textos e documentos com grande qualidade e diversidade. Outras ferramentas gratuitas e comerciais são criadas com grande frequência para atender a uma demanda cada vez maior.

Ao tempo que o correto manuseio dessas ferramentas resolve grande parte das dificuldades citadas com respeito à qualidade do resumo a ser produzido, não resolve completamente o problema da escalabilidade, já que não há um pipeline automatizado estabelecido que conecte as saídas de ferramentas distintas (ou de propriedade distinta). As ferramentas "tudo-em-uma" (*all-in-one*) que mais se aproximam de fornecer esse tipo de pipeline não são gratuitas e o custo de aquisição ou de uso do serviço podem ser impeditivos para o usuário comum.

O objetivo deste trabalho é propor e validar uma solução "caseira" de criação de um pipeline automatizado para geração de resumos em texto a partir de vídeos; o *pipeline* faz, naturalmente, uso de modelos e soluções do campo do aprendizado de máquina (ML). Algumas diretrizes foram estabelecidas:

a) A reprodução da solução deve ser uma tarefa simples de ser executada por qualquer interessado, ainda que não especialista em Computação ou programação de computadores, bastando para isso um computador pessoal com acesso à Internet;

b) O aplicativo criado deve ter interface simples, leve e intuitiva;

c) Os componentes computacionais do *pipeline* devem ser todos de uso gratuito;

d) A parametrização dos modelos de ML deve ser estabelecida de tal forma que o *pipeline* final não tome tempo consideravelmente grande de execução.

REFERENCIAL TEÓRICO

No contexto de um processo computacional, *pipeline* refere-se a uma abordagem em que uma sequência de etapas ou processos são organizados em um fluxo contínuo. Cada etapa do *pipeline* realiza uma tarefa específica e entrega seus resultados diretamente à próxima etapa, formando um fluxo de processamento eficiente. Esta seção descreve as principais técnicas e desafios de cada etapa do *pipeline* proposto à luz do Estado da Arte técnico-científico relacionado à etapa.

A extração de áudio de um vídeo, atualmente, é um processo bem estabelecido. Bibliotecas como FFmpeg, Libav e MoviePy oferecem funcionalidades para a manipulação de arquivos multimídia e a extração de faixas de áudio. A escolha da biblioteca depende da linguagem de programação utilizada e dos requisitos específicos da implementação utilizada.

A transcrição de áudio, ou reconhecimento automático de fala (Automatic Speech Recognition (ASR)), tem apresentado avanços significativos nos últimos anos, impulsionados pelo desenvolvimento de modelos de aprendizado profundo, como as Redes Neurais Recorrentes (RNNs), Redes Neurais Convolucionais (CNNs) e *Transformers*. Modelos como LSTM e GRU são capazes de capturar dependências temporais na fala, o que é crucial para a transcrição precisa (Graves, Mohamed, & Hinton, 2013). Já as CNNs são eficientes na extração de características

[1]Google Cloud Speech-to-Text: https://cloud.google.com/speech-to-text/
[2]Amazon Transcribe: https://aws.amazon.com/pt/transcribe/
[3]Google Translate: https://translate.google.com
[4]DeepL: https://www.deepl.com/pt-BR/translator
[5]OpenAI ChatGPT: https://chat.openai.com/
[6]Microsoft Copilot: https://copilot.microsoft.com
[7]Google Gemini: https://gemini.google.com/

acústicas relevantes do áudio (Hinton, Deng, Yu, et al., 2012). Modelos como o *Transformer* e suas variantes, como o *Conformer* (Gulati, Qin, Chiu, et al., 2020), têm alcançado resultados excepcionais em tarefas de ASR, superando as RNNs e CNNs em termos de precisão.

A tradução automática também tem se beneficiado do avanço dos modelos de aprendizado profundo, especialmente o *Transformers* (Vaswani, Shazeer, Parmar, et al., 2017). O Google Translate e o DeepL, por exemplo, utilizam o *Transformer* e suas variantes para realizar a tradução entre diferentes idiomas. Técnicas como o aprendizado multitarefa e o aprendizado por transferência têm sido utilizadas para melhorar a qualidade da tradução em diferentes domínios e para idiomas com poucos recursos (Raffel, Shazeer, Roberts, et al., 2020; Wu, Schuster, Chen, et al., 2016).

O resumo automático de texto pode ser dividido em duas abordagens principais: extração e abstração. A extração consiste em selecionar as frases mais importantes do texto original para formar o resumo. A abstração consiste em gerar um novo texto que resume o conteúdo original, utilizando paráfrases e generalizações (Nallapati, Zhou, Gulcehre, et al., 2016). Modelos de aprendizado profundo, como o *Transformer*, têm sido utilizados com sucesso em ambas as abordagens. Técnicas como o aprendizado por reforço e o aprendizado baseado em grafos têm sido exploradas para melhorar a qualidade do resumo (See, Liu, & Manning, 2017; Yasunaga, Kasai, Zhang, Liu, & Miyao, 2021).

MÉTODO

Para que o objetivo de gerar um resumo em português do conteúdo de um vídeo com áudio em inglês pudesse ser atingido, um *pipeline* foi definido com as seguintes etapas sequenciais:

1. Extração do áudio original do arquivo de vídeo;
2. Transcrição do áudio para texto na língua original (inglês);
3. Resumo automático do texto na língua original (inglês);
4. Tradução automática do resumo para a língua alvo (português).

Para cada etapa, um algoritmo foi criado, que faz uso de tecnologias de processamento de audiovisual e modelos de aprendizado de máquina pré-treinados e parametrizados de acordo com a necessidade do problema. Cada algoritmo foi então implementado em linguagem de programação e uso de bibliotecas mais adequadas para refletir a proposta. Uma vez implementadas, cada etapa passou por validação intrínseca via métricas estabelecidas na literatura. Cabe aqui ressaltar que em todas as representações algorítmicas focamos apenas no fluxo funcional e evitamos modelar explicitamente o funcionamento interno das redes neurais dentro do modelo; funções matemáticas específicas para extração de características, codificação e decodificação também foram omitidas, mantendo um nível de abstração um pouco mais elevado, priorizando clareza.

Extração do áudio original do arquivo de vídeo

A entrada de dados inicial de todo o procedimento é o caminho de localização do arquivo de vídeo de interesse. A saída dessa etapa de extração é, naturalmente, o áudio desse arquivo. Assim, criamos o seguinte algoritmo-solução para a etapa (Algoritmo 1).

Algorithm 1 Extração do áudio original do arquivo de vídeo

1: **Entrada:** V (*string* representando o caminho do arquivo de vídeo)
2: **Saída:** A (dados de áudio como função do tempo)
3: $L(V) \rightarrow (V_1, A_1, V_2)$
4: $f(A_1) \rightarrow A_s$
5: $OP(V) \rightarrow P$
6: $W(A_s, P) \rightarrow A$

No algoritmo, L representa a função de carregamento de vídeo (linha 1), responsável por gerar informações do vídeo como resolução, quadros etc. (V'), dados brutos de áudio (A') e os dados de vídeo (V"), que seria opcional para o nosso propósito principal. A função f processa os dados de áudio brutos, gerando função de sinal de áudio extraído (As) (linha 2). OP é a função de geração de caminho de saída com base no vídeo de entrada (V), com P representando o caminho do arquivo (ex: .mp3) de saída (linha 3). Finalmente, W consiste na função de gravação de áudio que salva os dados de áudio final (linha 4).

Transcrição do áudio para texto na língua original

A etapa anterior gera como saída um arquivo de áudio na língua original. O caminho de localização desse arquivo de áudio é então passado para a etapa seguinte do *pipeline* que procederá com a extração do texto a partir da

compreensão dos dados brutos de áudio. Sistemas com essa capacidade são conhecidos como sistemas ASR. O seguinte algoritmo-solução para esta etapa foi elaborado (Algoritmo 2).

Algorithm 2 Transcrição do áudio para texto na língua original

Entrada: A (*string* representando o caminho para o arquivo de áudio)
Saída: T (*string* contendo o texto transcrito do arquivo de áudio)
$L(base) \rightarrow M$
$L(A) \rightarrow A'$
$fTr(M, A') \rightarrow T$

Função: $fTr(modelo, audio) \rightarrow texto$
Entradas:
modelo: modelo pré-treinado de fala para texto
audio: dados de áudio de entrada como sequência de vetores de características
Saída:
texto: *string* de texto transcrito na língua original
$Seg(A) \rightarrow S$
$EC(S) \rightarrow F$
$Norm(F) \rightarrow F'$
$encode(M, F') \rightarrow H$
$decode(M, H) \rightarrow texto$
$RG(texto) \rightarrow texto'$

Primeiramente, o modelo de aprendizado pré-treinado é carregado na memória (linha 1) e os dados de áudio que serão transcritos também são carregados (linha 2). A função fTR de transcrição de dados de áudio para texto (linha 3) funciona da seguinte maneira. O áudio A é dividido em segmentos mais curtos, chamados de S (linha 4). De cada segmento em S, características de áudio, a exemplo dos Mel-Frequency Cepstral Coefficients (MFCCs) (Abdul & Al-Talabani, 2022), são extraídas (linha 5). As características extraídas (F) são então normalizadas (F') e preparadas para serem utilizadas como entrada para o modelo de aprendizado (linha 6). O modelo consiste em uma rede neural do tipo *Encoder-Decoder* (Yusuf, Gandhe, & Sokolov, 2022), que inicialmente processa as características sequenciais em uma representação oculta (H) (linha 7) e a partir de H, no mesmo modelo, realiza a previsão de palavras iterativamente, com base em previsões anteriores e utilizando técnicas como busca em feixe local (*local beam Search*) (linha 8). Finalmente, um modelo de linguagem[8] é aplicado à transcrição bruta inicial para refiná-la gramaticalmente (RG), resultando na transcrição final do áudio (linha 9).

Resumo automático do texto na língua original

A etapa anterior gera como saída um texto em língua inglesa correspondente ao áudio do vídeo original. O Algoritmo 3 descreve os passos para geração do resumo automático desse texto.

Algorithm 3 Resumo automático do texto na língua original

Entrada: T (*string* contendo o texto a ser resumido)
Saída: R (*string*contendo o resumo do texto)
$L(modelo) \rightarrow S$
$S(T, \{\mu_1, \mu_2, \rho_1, \rho_4, \phi, \beta, \varsigma\}) \rightarrow \bar{e}$
$eTransf(\bar{e}) \rightarrow \mathcal{C}$
$dTransf(A_t, \mathcal{C}) \rightarrow R$

Descrição dos parâmetros do modelo:
μ_2: limite máximo de *tokens* para o resumo
μ_1: limite mínimo de *tokens* para o resumo
ρ_1: tamanho máximo de n-gramas não repetidos no resumo
ρ_4: tamanho máximo de n-gramas não repetidos no codificador
ϕ: penalidade para repetição de n-gramas
β: número de feixes usados na decodificação
ς: critério de parada antecipada da decodificação

Inicialmente, o modelo de aprendizado a ser utilizado é carregado na memória (linha 1). O modelo é então aplicado ao texto de entrada T (transcrição completa resultante da etapa anterior), fazendo uso de um conjunto

[8]Um modelo de linguagem, no contexto do aprendizado de máquina, é um sistema que aprende a processar e gerar linguagem natural através de grandes conjuntos de dados que representam a linguagem humana em diferentes contextos.

de parâmetros que permitem ingerência no formato final do resumo a ser obtido. Incialmente, T é transformado em *tokens* (palavras) e convertido em um vetor de embeddings, ē (linha 2). Este vetor é então passado por um codificador da arquitetura neural, que aprende a gerar uma representação contextualizada de cada *token* $\mathcal{C}$ C(linha 3). Finalmente, um decodificador usa um modelo de atenção (At) para focar em diferentes partes da representação codificada, aprendendo a identificar quais palavras são mais relevantes para compor o resumo final R.

Tradução automática do resumo para a língua alvo

A etapa anterior gera como saída um resumo na língua inglesa. A última etapa do processo, então, é aplicar um modelo de tradução automática para a língua portuguesa. O Algoritmo 4 descreve os passos para tradução desse resumo.

Algorithm 4 Tradução automática do resumo para a língua alvo

Entrada: T (*string* contendo o texto a ser traduzido)
Saída: R_f (*string* contendo o resumo traduzido)
$L(modelo) \rightarrow Tr$
$TrN(T) \rightarrow T'$
$TrS(T') \rightarrow \{f\}$
4: $Tr(\{f\}) \rightarrow R_{temp}$
$Pos(R_{temp}) \rightarrow R_f$

Inicialmente, o modelo de tradução a ser utilizado é carregado na memória a fim de gerar o objeto tradutor Tr (linha 1). Na sequência, duas etapas de pré-processamento da tradução são aplicadas: a normalização do texto, com remoção de caracteres especiais, conversão para minúsculas etc. (linha 2) e a segmentação do texto em frases (linha 3). Na sequência, o processamento efetivo da tradução é realizado, agregando coerentemente o conjunto de frases geradas no passo anterior f para uma versão inicial do resumo na língua alvo R_{temp} (linha 4). Por fim, uma etapa de pós processamento é aplicada sobre essa versão inicial a fim de corrigir erros gramaticais e prover boa formatação. O *pipeline* de processamento finaliza com a geração do resumo final na língua alvo, Rf. A Figura 1 ilustra o *pipeline* produzido com destaque dos artefatos gerados após cada etapa.

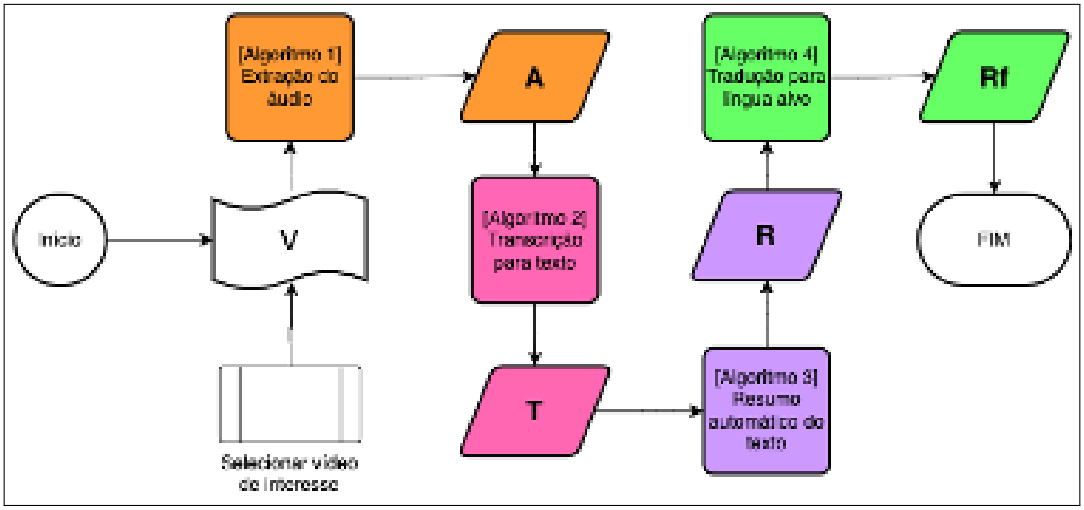

Figura 1. Pipeline de geração do resumo na língua alvo.

Modelos de aprendizado utilizados

Os algoritmos foram elaborados para serem aplicados indistintamente no que diz respeito aos modelos de aprendizado de máquina envolvidos. São três os momentos de envolvimento de modelos inteligentes no *pipeline* proposto: (1) no momento da transcrição (Algoritmo 2), fazemos uso de um modelo ASR, (2) ao resumir o texto na língua original (Algoritmo 3), fazemos uso de um modelo de Resumo automático e (3) no momento de compor o resumo final na língua alvo (Algoritmo 4), fazemos uso de um modelo de Tradução.

Para a tarefa de ASR, usamos o modelo base.en[9], que é um modelo ASR de última geração que se destaca por sua alta precisão e eficiência na transcrição de áudio. O modelo se baseia na arquitetura *Transformer* (Vaswani et al., 2017), uma rede neural profunda que se tornou a base para muitos modelos de linguagem de última geração. A arquitetura do modelo inclui inovações como um codificador híbrido, que combina um codificador convolucional

[9]Biblioteca Whisper.ai (https://openai.com/research/whisper)

(Mohamed, Okhonko, & Zettlemoyer, 2019) com um *Transformer*, para extrair informações acústicas e linguísticas do áudio de entrada e um decodificador auto atentivo, que usa um mecanismo de atenção (*attention*) para se concentrar nas partes mais relevantes do codificador durante a decodificação. O modelo incorpora ainda conhecimento linguístico através de um modelo de linguagem pré-treinado de grande porte (Large Language Model (LLM)) (Shanahan, 2024).

Para a tarefa de resumo automático, utilizamos o modelo *led-large-book-summary*. Trata-se de um modelo LLM que foi criado para resumir textos longos de forma informativa e concisa. O modelo também se baseia na arquitetura neural *Transformer*, dispondo de um codificador e um decodificador. O codificador processa o texto de entrada e o transforma em uma sequência de representações vetoriais. O decodificador, também auto atentivo, usa essas representações vetoriais para gerar o texto de saída, uma palavra de cada vez.

Por fim, a tarefa de tradução faz uso do Google Translate, que também se baseia no *Transformer*. Os componentes do modelo são um codificador, que processa o texto de entrada e o transforma em uma sequência de representações vetoriais, um decodificador que usa essas representações vetoriais para gerar o texto de saída, uma palavra de cada vez, e o mecanismo de atenção, que permite ao decodificador se concentrar nas partes mais relevantes do codificador durante a tradução do texto. O modelo também incorpora conhecimento linguístico via LLM para melhorar a qualidade da tradução.

Avaliação das etapas do *pipeline*

Para medir a qualidade do *pipeline*, definimos o seguinte roteiro de avaliação:
1. **Q(T):** medição da qualidade da transcrição gerada pelo Algoritmo 2;
2. **Q(R):** medição da qualidade do resumo gerado pelo Algoritmo 3;
3. **Q(Rf):** medição da qualidade da tradução gerada pelo Algoritmo.

Cada medição contou com uma linha de referência (*baseline*) e uma métrica estabelecida pela literatura. Para a avaliação $Q(T)$, estabelecemos como texto de referência a transcrição gerada pela ferramenta Happy Scribe (Happy Scribe, 2022), uma vez que a qualidade de sua transcrição para as mais variadas línguas tem sido atestada em trabalhos de comparação (Eser, 2022; Wollin-Giering, Hoffmann, Hofting, & Ventzke, 2023). As métricas de avaliação utilizadas foram a WER (*Word Error Rate*) e a CER (*Character Error Rate*). A WER calcula a taxa de erro por palavra, considerando os limites das palavras e permitindo o alinhamento entre o texto de referência e o texto gerado pela transcrição (Eq. 1). O numerador da equação calcula o total de substituições, inserções ou deleções que precisariam ser feitas nas palavras P do texto transcrito T para que ele se tornasse igual ao texto de referência $Tref$.

$$WER = \frac{\sum_{P=1}^{|T|} \text{Error}(P)}{|T_{\text{ref}}|} \quad \text{(Eq. 1)}$$

A CER calcula a taxa de erro por caractere, ignorando os limites das palavras e tratando cada caractere de forma independente (Eq. 2). O numerador da equação calcula o total de substituições, inserções ou deleções que precisariam ser feitas nos caracteres C do texto transcrito T para que ele se tornasse igual ao texto de referência $Tref$.

$$CER = \frac{\sum_{C=1}^{|T|} Error(C)}{|T_{\text{ref}}|} \quad \text{(Eq. 2)}$$

Portanto, tanto para o WER quanto para o CER, valores mais próximos à 0 (zero) são melhores enquanto valores próximos à 1 (um) são piores.

Para a avaliação $Q(R)$, estabelecemos como texto de referência o resumo gerado pelo ChatGPT que, notadamente, tem obtido resultados expressivos nessa atividade (Soni & Wade, 2023). Nessa avaliação, utilizamos a família de métricas ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) (Eq. 3, Eq. 4 e Eq. 5). A ROUGE avalia a qualidade de resumos automáticos, com diferentes variações que medem a similaridade entre o resumo gerado e o texto original. A ROUGE-N (n-grama) mede a correspondência entre o texto gerado e o texto de referência usando n-gramas. A ROUGE-L mede a correspondência da subsequência comum mais longa (LCS) de palavras entre os dois textos. A ROUGE-Lsum é uma variação da ROUGE-L que divide o texto em sentenças e calcula a média dos escores de LCS para cada par de sentenças.

$$\text{ROUGE-N} = \frac{\sum_{\text{grama}_n \in T_{\text{ref}}} \min\left(\text{Count}(\text{grama}_n, T_{\text{ref}}), \text{Count}(\text{grama}_n, R)\right)}{\sum_{\text{grama}_n \in T_{\text{ref}}} \text{Count}(\text{grama}_n, T_{\text{ref}})} \quad \text{(Eq. 3)}$$

$$\text{ROUGE-L} = \frac{\text{LCS}(T_{\text{ref}}, R)}{|T_{\text{ref}}|} \quad \text{(Eq. 4)}$$

$$\text{ROUGE-Lsum} = \frac{\sum_{i=1}^{n} \text{LCS}(T_{\text{ref},i}, R_i)}{\sum_{i=1}^{n} |T_{\text{ref},i}|} \quad \text{(Eq. 5)}$$

Onde T_{ref_i} é a i-ésima sentença do texto de referência e R_i é a i-ésima sentença do texto gerado. No caso da família ROUGE, valores mais próximos a 1 (um) são os melhores.

Para a avaliação $Q(Rf)$, estabelecemos como referência a tradução fornecida pelo DeepL[10]. A qualidade comparativa da ferramenta de tradução DeepL tem sido cientificamente atestada (Rescigno, Vanmassenhove, Monti, & Way, 2020; Yulianto & Supriatnaningsih, 2021). Para essa avaliação utilizamos a métrica BLEU (*Bilingual Evaluation Understudy*), que busca medir a similaridade entre o texto gerado e o texto de referência usando n-gramas, que capturam a ordem e a frequência das palavras. Quanto maior a precisão modificada dos n-gramas, maior será a pontuação BLEU. No entanto, a precisão modificada pode favorecer textos muito curtos, que podem ter uma alta correspondência de n-gramas, mas não são boas traduções. Por isso, o fator de penalização por brevidade (BP) é usado para reduzir a pontuação BLEU de textos que são muito menores que os textos de referência. O fator BP é igual a 1 se o texto gerado for maior ou igual ao texto de referência e é menor que 1 se o texto gerado for menor. Assim, a métrica BLEU busca balancear a precisão e a brevidade dos textos gerados (Eq. 6).

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log(p_n)\right) \quad \text{(Eq. 6)}$$

Onde

$$BP = \begin{cases} 1, & \text{se } c > r \\ e^{(1-\frac{r}{c})}, & \text{se } c \leq r \end{cases}$$

onde c é o comprimento do texto gerado, r é o comprimento do texto de referência e p_n representa a precisão de n-gramas. Uma pontuação BLEU mais alta indica uma tradução mais próxima dos textos de referência.

RESULTADOS

Esta seção está organizada em três subseções para acomodar de forma mais organizada os resultados do trabalho. Na primeira subseção, tratamos da descrição das tecnologias envolvidas na implementação do pipeline e como foram combinadas na implantação da solução. A segunda subseção detalha como a solução deve ser executada pelo usuário interessado e mostra a rotina de utilização do aplicativo final gerado para um vídeo de estudo de caso, ilustrando a composição dos subprodutos gerados ao final de cada etapa: transcrição do áudio do vídeo original (T), o resumo dessa transcrição (R) e a tradução desse resumo para a língua alvo (Rf). A terceira e última subseção traz resultados quantitativos da avaliação de qualidade de cada subproduto: $Q(T)$, $Q(R)$ e $Q(Rf)$, conforme especificadas anteriormente.

Arquitetura da solução

Os algoritmos descritos foram implementados na linguagem de programação Python com auxílio de diversas bibliotecas e pacotes. A Tabela 4 do Apêndice A traz a lista de bibliotecas, pacotes e classes utilizadas e respectivas descrições de seus propósitos. Tanto a linguagem utilizada, quanto as referidas bibliotecas são de uso gratuito, conforme prega uma das diretrizes que estabelecemos para esse trabalho.

Uma outra diretriz estabelecida dizia respeito a viabilizar o uso da solução implementada por qualquer interessado não especialista em Computação ou programação de computadores. Para que essa diretriz pudesse ser atendida, definimos uma infraestrutura de implantação da solução que evita a necessidade de instalação de dependências computacionais específicas no computador do interessado. Para esse fim, utilizamos o serviço de nuvem gratuito chamado Google Colaboratory (Colab), uma máquina remota que permite a execução de programas escritos na linguagem Python diretamente do navegador, incluindo a possibilidade de instalação de todas as dependências necessárias que, no nosso caso, são as já citadas na Tabela 4.

Para garantir que o aplicativo criado tivesse interface simples, leve e intuitiva (outra diretriz), foi feito uso do Streamlit, um *framework* Python que torna a criação de aplicativos da *web* interativos e visualizações de dados mais fácil e rápida. Por fim, utilizamos a ferramenta gratuita LocalTunnel, que permite expor um servidor local na *web* pública através de um "túnel seguro". Ou seja, é possível executar uma aplicação em seu computador pessoal, por exemplo, e ele permite que essa aplicação seja acessada de forma segura de qualquer lugar na Internet. Em nossa proposta de implantação, estabelecemos o serviço Colab como servidor da aplicação ao invés

[10]DeepL: https://www.deepl.com/pt-BR/translator

de uma máquina local, livrando o usuário das preocupações com instalações e configurações específicas da rotina de programação de computadores.

A Figura 2 ilustra como todas essas tecnologias se relacionam no diagrama de implantação da solução que, temporariamente, estamos denominando de "Smart Summy". Os algoritmos descritos foram implementados na linguagem de programação Python com auxílio de diversas bibliotecas e pacotes. A Tabela A.1 do apêndice A traz a lista de bibliotecas, pacotes e classes utilizadas e respectivas descrições de seus propósitos. Tanto a linguagem utilizada, quanto as referidas bibliotecas são de uso gratuito, conforme prega uma das diretrizes que estabelecemos para esse trabalho.

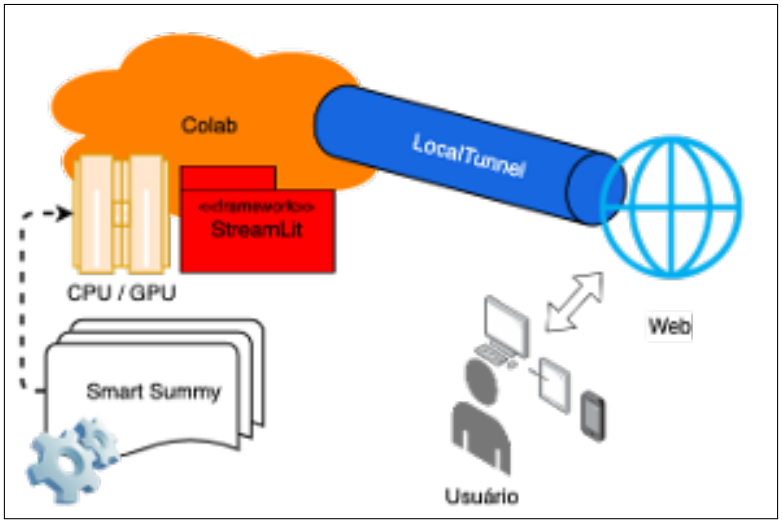

Figura 2. Diagrama de implantação da solução "Smart Summy".

Execução da solução "Smart Summy"

O "Smart Summy" pode ser acessado livremente a partir de qualquer navegador[11]. Uma vez acessado, o usuário terá acesso a um portal web, mais especificamente, a um notebook Colab (Figura 3). Esse *notebook* está organizado em trechos, conforme descrição em tela. Aos interessados em conhecer toda a implementação dos algoritmos criados, basta expandir o trecho intitulado "Código da aplicação". Entretanto, para execução da solução, a única ação necessária é acessar o menu "Ambiente de execução" e clicar em "Executar tudo". Cada um desses trechos irá executar em sequência, sendo necessário aguardar alguns poucos instantes. Quando apenas o último trecho estiver em execução (este em específico nunca encerra), o usuário pode expandir os dois últimos trechos, conforme Figura 4, copiar o código gerado (senha) e clicar na *url* que aparece em azul ao final.

[11] Solução "SmartSummy" (Colab): https://colab.research.google.com/drive/1aIpRtSa7H6wp7D5wBzJ1_2LuYYds9qMT?usp=sharing

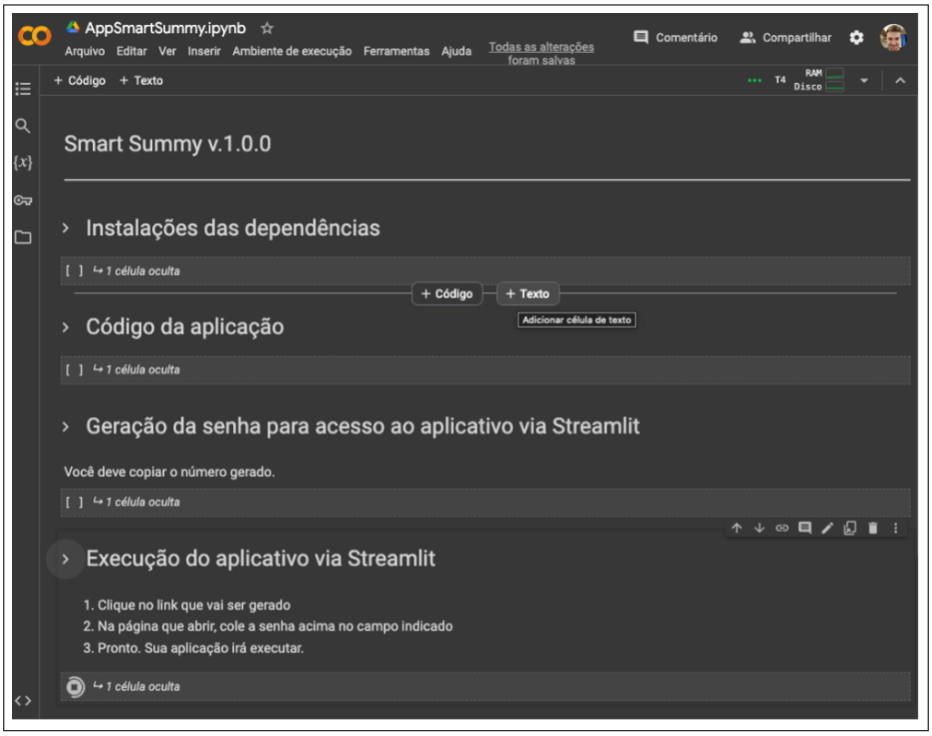

Figura 3. Página inicial da solução "Smart Summy" em um *notebook* Colab.

Figura 4. Parte do *notebook* Colab que exibe senha e *hiperlink* do aplicativo final.

Essa ação levará a uma outra página *web* que pedirá a senha copiada (Figura 5). Essa "entre etapas" é uma exigência de segurança da ferramenta LocalTunnel.

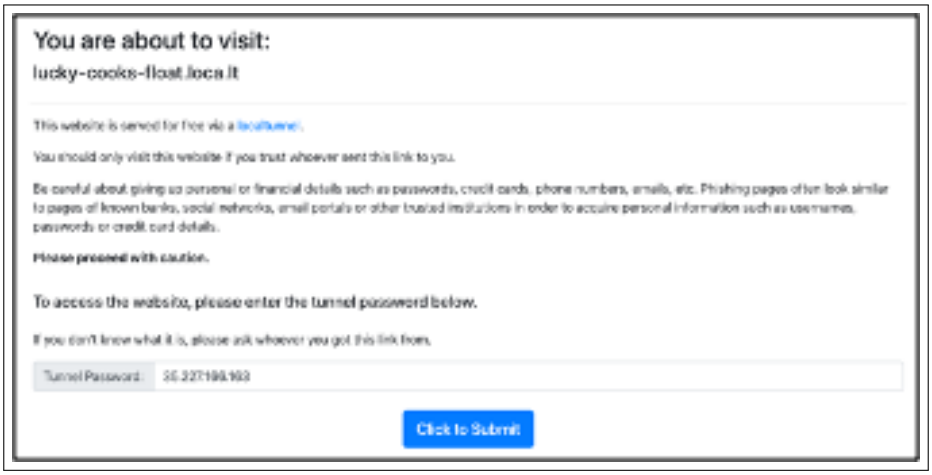

Figura 5. Tela do LocalTunnel onde se deve colar a senha e clicar no botão de submissão.

Ao pressionar o botão de submissão, o *frontend* do aplicativo de interesse será executado. A partir daí, trata-se de interação usual com uma aplicação *web* intuitiva, iniciando-se com a seleção do vídeo de interesse (Figura 6a).

Figura 6. Figura 6.(a) Tela inicial do *frontend* do "Smart Summy" e (b) Tela que ilustra ações de reprodução do vídeo (V) e extração do áudio (A). O usuário arrasta o vídeo de interesse[a] (cujo áudio deve estar em língua inglesa) e, automaticamente, o *pipeline* descrito será executado até a geração do resumo final na língua portuguesa. Caso tenha interesse, o usuário poderá executar o vídeo na própria interface do aplicativo.

[a]Nessa versão do SmartSummy, o formato do vídeo de interesse deve obrigatoriamente ser .mp4

Para ilustrar a rotina de utilização, selecionamos um trecho do vídeo do acervo The Internet Archive[12], intitulado "How the Eye Functions"[13](por Karl Kurt Bosse, 1941), que traz uma explicação animada do olho humano e da visão. A tela que ilustra o carregamento do vídeo (V) pode ser observada na Figura 6b. A extração do áudio original (A) e geração automática do resumo em inglês (R) podem ser observadas na Figura 7a e a tradução do resumo para o português (Rf), na Figura 6b.

[12]"The Internet Archive", https://archive.org/, é uma organização sem fins lucrativos que visa preservar o conhecimento e a informação digital. Foi fundada em 1996 por Brewster Kahle e está sediada em São Francisco, Califórnia. O acervo contempla vários tipos de conteúdos, entre eles, vídeos com licenças livres.

[13]"How the Eye Functions" (Public Domain, cc): https://archive.org/details/HowtheEy1941

Figura 7. (a) Tela que ilustra o resultado da transcrição do áudio original (T) e (b) Tela que ilustra o resultado da geração automática do resumo na língua original inglesa (R) e a respectiva tradução para a língua portuguesa (Rf).

Avaliação de qualidade dos subprodutos gerados

Para atestar a qualidade dos subprodutos produzidos ao longo do *pipeline* de execução, definimos um cenário de experimentação, configurado com os seguintes requisitos:

1. Todos os vídeos utilizados nos experimentos deveriam possuir garantias de acesso e manipulação livre. Além disso, naturalmente, todos os vídeos deveriam possuir áudio na língua inglesa e formato .mp4. Por essa razão, optamos pela coleção de vídeos do Prelinger Archives[14], um grande acervo de produções visuais, com vários filtros de busca avançada, disponíveis em vários formatos, com *download* livre, sob o Domínio Público da Creative Commons;
2. O áudio dos vídeos deveria variar ao longo de algumas dimensões, para garantir diversidade, a exemplo do gênero da voz, idade da voz, sotaque do inglês, mais ou menos ruído no áudio, existência (ou não) de som ao fundo da transmissão, entre outros;
3. O quantitativo total de vídeos utilizado deveria garantir minimamente conclusões assertivas sobre as medidas de qualidade. A duração do vídeo, por outro lado, não seria fator preponderante para essa assertividade e, para reduzir o tempo total de experimentação, utilizamos apenas trechos de aproximadamente 1 minuto (com áudio) de cada vídeo. Ao todo, utilizamos 17 trechos de vídeos baixados do acervo.

Resultados dos experimentos

Para medição da qualidade da transcrição gerada, $Q(T)$, utilizamos a ferramenta Happy Scribe como *baseline*, e as métricas WER e CER. Estabelecemos um valor limiar de 20% para o que pode ser considerado uma boa transcrição. Foram calculados valores médios globais de WER e CER, além dos valores respectivos de cobertura (*recall*), precisão (*precision*) e medida-F (F1-score). A Tabela 1 traz todos esses resultados obtidos ao longo das sessões de experimentação.

Métrica	Valor médio	Cobertura	Precisão	Medida-F
WER	$0{,}11 \pm 0{,}10$	0,85	1,00	0,92
CER	$0{,}03 \pm 0{,}03$	1,00	0,87	0,93

Tabela 1. Resultado da análise de qualidade $Q(T)$.

Os valores de cobertura mostram que o modelo identifica corretamente 85% das palavras e 100% dos caracteres da transcrição de referência. Os valores de precisão mostram que 100% dos elementos de texto que foram identificados como sendo palavras reais da transcrição realmente o são, enquanto esse valor foi de 87% para caracteres. Finalmente, a medida-F, que representa a média harmônica entre esses dois valores, mostrou valores de pelo menos 92%, um excelente desempenho na identificação tanto de palavras quanto de caracteres corretos.

Para medição da qualidade do resumo gerado na língua original inglesa, $Q(R)$, utilizamos a geração de resumo pelo ChatGPT como referência, e a família de métricas ROUGE. A Tabela 2 traz os valores médios obtidos para cada uma das métricas da família sobre os resumos gerados a partir das 20 transcrições anteriormente geradas.

Em linhas gerais, a tabela mostra que 36% dos termos (palavras isoladas) do resumo de referência estão presentes no resumo gerado pelo "Smart Summy". Esse valor cai bastante para bigramas, pares de palavras consecutivas, chegando a 8% com alto desvio. Essa redução de ROUGE-1 para ROUGE-2 é, obviamente, sempre esperada.

[14]Prelinger Archives: https://archive.org/details/prelinger

Métrica	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
Valor médio	$0{,}36 \pm 0{,}08$	$0{,}08 \pm 0{,}05$	$0{,}20 \pm 0{,}05$	$0{,}20 \pm 0{,}05$

Tabela 2. Resultado da análise de qualidade $Q(T)$.

A tabela também mostra que houve 15% a 25% de sobreposição de n-gramas de diferentes tamanhos, com média em 20%; é o que diz a ROUGE-L, que considera a sobreposição de n-gramas de diferentes tamanhos e pondera n-gramas mais longos. Por fim, a pontuação média e o desvio padrão da ROUGE-Lsum são idênticos ao ROUGE-L, o que indica que a sobreposição de n-gramas não é significativamente diferente em resumos concisos.

Ainda que a família ROUGE seja a referência de métricas de avaliação para geração automática de resumos, ela apresenta algumas limitações muito importantes que devem ser consideradas ao interpretar os resultados. A primeira delas é que por se basear na sobreposição de n-gramas (sequências de n palavras) entre o resumo gerado e o resumo de referência, ela recompensa resumos que copiam diretamente frases do resumo de referência, mesmo que não capturem o significado completo do texto original. Dois efeitos colaterais ruins desse comportamento são: (1) resumos que parafraseiam ou reestruturam o conteúdo original são naturalmente penalizados, ainda que transmitam a mesma informação e, talvez, de forma mais concisa ou elegante e (2) resumos que contêm palavras-chave aleatórias do resumo de referência podem obter pontuações altas, mesmo que não façam qualquer sentido. Um segundo ponto é que as métricas ROUGE são sensíveis à ordem das palavras, o que pode levar a resultados inconsistentes; pequenas mudanças na ordem das palavras no resumo gerado podem resultar em uma queda significativa na pontuação, mesmo que o significado geral do resumo seja preservado. Um terceiro ponto é que a ROUGE possui um viés para resumos longos, pois há maior probabilidade de que contenham mais n-gramas do resumo de referência. Isso pode levar à avaliação inadequada de resumos concisos que capturam os pontos principais do texto original. Em trabalhos futuros, pretendemos adaptar e incluir o uso de métricas de avaliação semântica de resumos a exemplo da proposta por Lin, Li, Zhang, et al. (2022).

Para medição da qualidade da tradução do resumo para língua portuguesa, $Q(Rf)$, utilizamos o DeepL, como referência de tradução correta, e a métrica BLEU. Para mitigar o problema da "contagem zero", que pode ocorrer quando um n-grama não está presente na tradução gerada, mas está presente na tradução de referência, aplicamos uma técnica de suavização em nível de sentença baseada no trabalho de Chen (2014). A Tabela 3 traz os resultados do BLEU médio para três diferentes configurações de combinação de n-gramas: (1) BLEU(100): unigrama (100%), (2) BLEU(50,50): unigrama (50%) e bigrama (50%) e (3) BLEU(25,25,25,25): unigrama (25%), bigrama (25%), 3-grama (25%) e 4-grama (25%).

Métrica	BLEU(100)	BLEU(50,50)	BLEU(25,25,25,25)
Valor médio	$0{,}99 \pm 0{,}10$	$0{,}82 \pm 0{,}11$	$0{,}63 \pm 0{,}13$

Tabela 3. Resultado da análise de qualidade $Q(R_f)$.

A tabela mostra que 99% das palavras presentes na tradução de referência estavam presentes na tradução gerada pelo "Smart Summy" para um dado texto qualquer dentre os 17 usados no experimento. Naturalmente, à medida que aumentamos a ponderação dos n-gramas maiores, esse valor cai. Entretanto, mesmo considerando 3-gramas e 4-gramas na contagem, a média BLEU obtida é de 63%, que é um resultado muito bom.

Todos os textos gerados automaticamente pelo "Smart Summy" para cada etapa de avaliação e respectivos textos de referência, usados nos experimentos, estão organizados em subpastas e disponíveis para acesso[15]. O código fonte das etapas de experimentação também segue disponível[16]

CONCLUSÃO

Este trabalho teve como objetivo principal prover um roteiro para a geração de resumos em português a partir de vídeos com áudio em inglês, utilizando para isso modelos de ML pré-treinados, com a restrição de que o usuário interessado em utilizar a solução não precisaria ter qualquer conhecimento prévio de computação ou programação de computadores.

Os resultados obtidos mostraram como esse objetivo pode ser atingido a partir da construção de um *pipeline* de quatro etapas: extração do áudio original do arquivo de vídeo, transcrição do áudio para texto na língua original (inglês), resumo automático do texto na língua original (inglês) e tradução automática do resumo para a língua alvo (português). Para cada etapa do *pipeline*, um algoritmo foi criado, implementado em linguagem de programação Python e validado com métricas adequadas. Para que o usuário final não tivesse que instalar qualquer programa ou biblioteca, e pudesse executar a solução de qualquer computador com acesso à internet, o aplicativo final foi implantado em um ambiente de nuvem. A execução da solução mostrou-se bastante simples,

[15] Repositório com textos usados nos experimentos: https://1drv.ms/f/c/7b93c0afe0da13d0/Elq6JLas40VCtw6Y7udXwAEBlK3vEV8uVWf5GKKm
[16] Código fonte dos experimentos: https://colab.research.google.com/drive/1593sUABfcPTLMfU9YOg3xpoiozmPHCzy?usp=sharing

bastando que o usuário acesse o aplicativo, selecione o vídeo de interesse em seu computador e aguarde a execução do *pipeline*.

As avaliações quantitativas das etapas do *pipeline* com uso das métricas estabelecidas demonstraram alta qualidade na transcrição, boa qualidade do resumo em inglês e ótimo poder de tradução para o português.

Este trabalho explora soluções na Ciência da Informação para organizar e acessar informações em ambientes digitais, focando na crescente demanda por ferramentas para análise e recuperação de conteúdos audiovisuais. A proposta apresenta um método automatizado de sumarização de vídeos, que pode ser útil nas tarefas de:

- Indexação e classificação: Criação de metadados descritivos para facilitar recuperação em sistemas digitais.
- Organização de coleções: Agrupamento de vídeos por temas ou critérios.
- Análise de conteúdo: Identificação de tendências e padrões em vídeos.
- Análise de conteúdo: Identificação de tendências e padrões em vídeos.
- Acessibilidade: Melhoria do acesso para pessoas com deficiência auditiva ou conectividade limitada.
- Pesquisa e aprendizado: Facilitação na identificação de conteúdos relevantes.

A solução contribui para a gestão de coleções audiovisuais, análise de conteúdo e democratização da informação, beneficiando estudantes, pesquisadores e o público em geral. Como trabalhos futuros, pretende-se:

- Incorporar a sumarização multimodal, que leva em conta informações visuais e sonoras do vídeo, além da transcrição do áudio.
- Melhorar a qualidade do resumo, avaliando diferentes modelos de sumarização e técnicas de pré-processamento e pós-processamento.
- Criar uma interface gráfica mais amigável, que permita ao usuário personalizar as configurações do *pipeline*.
- Disponibilizar a solução em outras plataformas, como dispositivos móveis.
- Realizar testes com usuários para avaliar a usabilidade e a efetividade da solução.

APÊNDICE A

Biblioteca/Pacote	Propósito
streamlit	Biblioteca para criar aplicativos interativos e baseados na web. Nesse caso, foi usada para exibir a saída do *script* em um navegador.
whisper	Biblioteca do pacote *transformers* da OpenAI, usada para reconhecimento de fala e tradução.
torch	Biblioteca de aprendizado profundo utilizada para construção de redes neurais.
pipeline	Parte do pacote *transformers*, usada para criação de modelos e *pipelines* de PLN.
VideoFileClip	Classe do pacote moviepy.editor, utilizada para manipulação de arquivos de vídeo e criação de clipes.
Translator	Classe do pacote googletrans, usada para tradução de texto entre diferentes idiomas.
sys	Módulo da biblioteca padrão do Python que fornece acesso a funções e variáveis do sistema.
os	Módulo da biblioteca padrão do Python que permite interação com o sistema de arquivos.
tempfile	Módulo da biblioteca padrão do Python para criação e gerenciamento de arquivos e diretórios temporários.

Tabela 4. Lista de bibliotecas e pacotes Python e respectivos propósitos

AGRADECIMENTO

Os autores agradecem à Fundação de Apoio à Pesquisa e à Inovação Tecnológica do Estado de Sergipe – FAPITEC-SE pelo suporte financeiro [EDITAL FAPITEC/SE/FUNTEC N°01/2024, Processo 019203.04225/2024-9]. Os autores agradecem ainda ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq pela bolsa de produtividade de Hendrik Macedo [DT-C, 304788/2025-2].

REFERÊNCIAS

Abdul, Z., & Al-Talabani, A. (2022). Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, *10*, 122136–122158. doi: 10.1109/ACCESS.2022.3223444

Chen, B. A. (2014). A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the ninth workshop on statistical machine translation* (p. 362–367). doi: 10.3115/v1/W14-3346

El-Kassas, W. S., Salama, C., Rafea, A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, *165*, 113679. doi: 10.1016/j.eswa.2020.113679

Eser, O. (2022). The quality of translation students' transcriptions for subtitling in healthcare settings. *The Interpreter and Translator Trainer*, *16*(4), 524–539. doi: 10.1080/1750399X.2022.2082103

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Ieee international conference on acoustics, speech and signal processing* (p. 6645–6649). doi: 10.1109/ICASSP.2013.6638947

Gulati, A., Qin, J., Chiu, C. C., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech* (p. 5036–5040). doi: 10.48550/arXiv.2005.08100

Happy Scribe. (2022). *Audio transcription and subtitles*. Recuperado de https://www.happyscribe.com/

Hinton, G., Deng, L., Yu, D., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, *29*(6). doi: 10.1109/MSP.2012.2205597

Lin, W., Li, S., Zhang, C., et al. (2022). Summscore: A comprehensive evaluation metric for summary quality. *arXiv preprint*. doi: 10.48550/arXiv.2207.04660

Mohamed, A., Okhonko, D., & Zettlemoyer, L. (2019). Transformers with convolutional context for asr. *arXiv preprint*. doi: 10.48550/arXiv.1904.11660

Nallapati, R., Zhou, B., Gulcehre, C., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns. In *Conll* (p. 280–290). doi: 10.48550/arXiv.1602.06023

Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*, 1–67. doi: 10.5555/3455716.3455856

Rescigno, A. A., Vanmassenhove, E., Monti, J., & Way, A. (2020). A case study of natural gender phenomena in translation: A comparison of google translate, bing microsoft translator and deepl for english to italian, french and spanish. In *Proceedings of clic-it (computational linguistics in italy)* (p. 257–262). doi: 10.4000/books.aaccademia.8844

Rivera-Trigueros, I. (2022). Machine translation systems and quality assessment: A systematic review. *Language Resources and Evaluation*, *56*(2), 593–619. doi: 10.1007/s10579-021-09537-5

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Acl* (p. 1073–1083). doi: 10.48550/arXiv.1704.04368

Shanahan, M. (2024). Talking about large language models. *Communications of the ACM*, *67*(2), 68–79. doi: 10.1145/3624724

Soni, M., & Wade, V. (2023). Comparing abstractive summaries generated by chatgpt. *arXiv preprint*. doi: 10.48550/arXiv.2303.17650

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. In *Neurips*. doi: 10.48550/arXiv.1706.03762

Wollin-Giering, S., Hoffmann, M., Hofting, J., & Ventzke, C. (2023). *Automatic transcription of qualitative interviews*. Sociology of Science Discussion Papers. doi: 10.13140/RG.2.2.14480.38404

Wu, Y., Schuster, M., Chen, Z., et al. (2016). Google's neural machine translation system. *arXiv preprint*. doi: 10.48550/arXiv.1609.08144

Yasunaga, M., Kasai, J., Zhang, R., Liu, Y., & Miyao, Y. (2021). Graph-based neural sentence ordering. In *Proceedings of the conference of the north american chapter of the association for computational linguistics: Human language technologies* (p. 1890–1906). doi: 10.48550/arXiv.1912.07225

Yulianto, A., & Supriatnaningsih, R. (2021). Google translate vs. DeepL: A quantitative evaluation of close-language pair translation (french to english). *Asian Journal of English Language and Pedagogy*, *9*(2), 109–127. doi: 10.37134/ajelp.vol9.2.9.2021

Yusuf, B., Gandhe, A., & Sokolov, A. (2022). Usted: Improving asr with a unified speech and text encoder-decoder. In *Ieee international conference on acoustics, speech and signal processing (icassp)* (p. 8297–8301). doi: 10.48550/arXiv.2202.06045

Zhang, B., Haddow, B., & Sennrich, R. (2022). Revisiting end-to-end speech-to-text translation from scratch. In *Proceedings of the international conference on machine learning (icml)* (p. 26193–26205). PMLR. doi: 10.48550/arXiv.2206.04571

Como citar este artigo (APA):
Monteiro, K. da S., Macedo, H. T., Matos, L. N., & Bispo, K. A. (2025). Geração automática de resumo em português a partir de conteúdo audiovisual em inglês: método, validação e aplicativo "caseiro". *AtoZ: novas práticas em informação e conhecimento, 14*, 1 – 15. Recuperado de: http://dx.doi.org/10.5380/atoz.v14.94711

NOTAS DA OBRA E CONFORMIDADE COM A CIÊNCIA ABERTA

CONTRIBUIÇÃO DE AUTORIA

Papéis e contribuições	Keomas da Silva Monteiro	Hendrik Teixeira Macedo	Leonardo Nogueira Matos	Kalil Araújo Bispo
Concepção do manuscrito	X	x		
Escrita do manuscrito	X	X		
Metodologia	X	X	X	X
Curadoria dos dados	X	X	X	
Discussão dos resultados		X	X	X
Análise dos dados		X	X	

FINANCIAMENTO

O(s) autor(es) declara(m) que esta pesquisa recebeu financiamento conforme dados indicados a seguir e o documento comprobatório foi anexado como documento suplementar: **CNPq – Processo nº 304738/2020-4 (Bolsa DT – Hendrik Macedo).**

EQUIPE EDITORIAL