UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

Aluno: Mário Sérgio Maduro Santana
Orientador: João Batista Florindo

Modelo de inteligência artificial para transição entre imagens: uma aplicação no tratamento da afasia (Parte II)

Campinas
Junho de 2024

Mário Sérgio Maduro Santana

Modelo de inteligência artificial para transição entre imagens: uma aplicação no tratamento da afasia (Parte II)

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do Prof Dr João Batista Florindo.

Contents

1 Introdução

Este projeto é uma continuação do estudo desenvolvido na discipina MS777 (veja [1]), o qual explorou a utilização da rede neural FILM (veja [5])- Frame Interpolation For Large Motion - para a realização de uma transição suave entre os visemas (imagens dos fonemas). No presente trabalho, a proposta é utilizar os vídeos gerados e armazenados na parte 1 do projeto para desenvolver um aplicativo que, realmente, possa auxiliar a reabilitação de pessoas com Afasia.

O nome do aplicativo que esse projeto visou desenvolver é SpeakRehab e, embora não esteja 100% finalizado, já existe uma versão para demonstração utilizável.Nas próximas seções serão apresentados os passos realizados para o desenvolvimento do aplicativo.

2 SpeakRehab

A idea do SpeakRehab é ser um aplicativo onde o usuário possa digitar uma palavra em português do Brasil e, após um processamento interno, seja gerado um vídeo com os movimentos labiais da palavra em questão[1]. Para a criação dessa lógica, foi necessário dividir o projeto em etapas menores que, quando utilizadas em conjunto, resultam no resultado final desejado.

As etapas para o desenvolvimento do aplicativo são as seguintes:

- **Conseguir os vídeos de todas as sílabas do Português do Brasil**: Essa etapa foi concluída na parte I do projeto por meio da utilização da rede FILM - Frame Interpolation For Large Motion.
- **Fazer uma separação silábica das palavras do Português do Brasil**: Esse passo é importante, pois, dada uma sílaba, é possível encontrar o vídeo que reproduz os movimentos labiais dessa sílaba.
- **Concatenadar vídeos**:A ideia desse passo é, após a realização da separação silábica e da obtenção dos vídeos de cada sílaba da palavra, conseguir concatenar cada um desses vídeos e formar a palavra completa.
- **Implementar a aceleração de vídeos**: Como foi discutido na Parte I deste projeto (veja [1]), ao processar uma palavra como CAVALO, é necessário concatenar os vídeos dos pares de letras correspondentes: **CA**, **AV**, **VA**, **AL** e **LO**. Para tornar a transição entre os pares mais natural e dinâmica, certos vídeos de transição – como **AV** e **AL** – precisam ser acelerados. Assim, a capacidade de ajustar a velocidade de reprodução dos vídeos é fundamental para o bom funcionamento do aplicativo.
- **Desenvolver uma lógica de mapeamento entre sílabas e vídeos correspondentes**: Essa etapa é crucial, pois integra as três fases anteriores do processo. Após identificar os vídeos de cada sílaba de uma palavra e aplicar a aceleração onde necessário, o próximo passo é concatenar esses vídeos de forma que representem a palavra como um todo.

Nas próximas subseções será explicado como as etapas da separação silábica, da concatenação de vídeos e do mapeamento de uma sílaba qualquer em seu vídeo correspondente estão sendo realizadas.

[1] Aplicativo SpeakRehab está sendo desenvolvido em *C#* com auxílio do *framework .NET MAUI* da Microsoft (confira [3] para mais informações).

2.1 Separação Silábica

A princípio, realizar a separação silábica de uma palavra do português do Brasil pode não parecer um grande desafio. No entanto, alcançar esse objetivo mostrou-se uma tarefa complexa.

Após extensas discussões com o orientador, surgiu a ideia de utilizar os dicionários de hifenização do LibreOffice como solução para o problema. A partir dessa proposta, os dicionários de hifenização do LibreOffice foram baixados e a biblioteca *NHunspell* — que permite o uso desses dicionários para verificação ortográfica, hifenização e outras funcionalidades linguísticas (veja [4]) — foi incorporada ao projeto.

Com a estratégia e as ferramentas definidas, restou apenas o desafio de implementação. Para realizar a separação silábica, foi desenvolvido o objeto *SyllabicSeparator*, que reúne todos os atributos e métodos necessários para efetuar a hifenização de uma palavra. Dessa forma, dada a palavra de entrada do usuário, foi possível realizar a separação silábica corretamente.

Este parágrafo apresenta um resumo da utilização da biblioteca *NHunspell* na tarefa de hifenização de palavras. Abaixo, é descrito o passo a passo para hifenizar uma palavra armazenada, por exemplo, em uma variável do tipo *string* chamada *word*:

- Criar um objeto do tipo *Hyphen* da seguinte forma:

```
1 private Hyphen hyphen;
```

Algoritmo 1: Objeto responsável pela Hifenização

- Inicializar o objeto *hyphen*:

```
1       string hyphenFile =
           Path.Combine(AppContext.BaseDirectory,"hyph_pt_BR.dic");
           //"hyph_pt_BR.dic" eh o nome do arquivo de hifenizacao
2       hyphen = new Hyphen(hyphenFile);
```

Algoritmo 2: Inicialização do objeto hyphen

- Hifenizar a palavra:

```
1       var hyphenatedWord = hyphen.Hyphenate(word);// faz a
           hifenizacao da palavra
2       string texto = hyphenatedWord.HyphenatedWord; //Obtem a
           palavra hifenizada no formato de uma string
```

Algoritmo 3: Hifenizando a palavra

2.2 Concatenação de Vídeos

Assim como na separação silábica, a concatenação de vídeos também apresenta desafios. Após pesquisar sobre o tema, encontrou-se o FFmpeg, uma ferramenta de código aberto que permite manipular arquivos de áudio e vídeo. Com ele, é possível unir vídeos, ajustar formatos, aplicar filtros e realizar diversas outras operações multimídia. No SpeakRehab, o uso do FFmpeg é viabilizado pela biblioteca Xabe.FFmpeg (veja a documentação em [6]), que facilita a integração do FFmpeg ao projeto.

Vale destacar que, para resolver os desafios de concatenação e aceleração de vídeos, foi desenvolvido o objeto *VideoManager*, que centraliza todos os métodos e atributos necessários para essa tarefa. A seguir, será demonstrado como utilizar a biblioteca *Xabe.FFmpeg* para concatenar 2 vídeos cujos caminhos estão armazenados nas variáveis *video1Path* e *video2Path*:

- Criar um arquivo de texto *videolist.txt* com os caminhos dos 2 vídeos:

```
1       string videolistPath = Path.Combine(Path.GetTempPath(),
           "videolist.txt");
2       using (StreamWriter sw = new StreamWriter( videolistPath,
           false, new UTF8Encoding(false))) // UTF-8 sem BOM
3       {
4       string formattedVideo1Path = video1Path.Replace("\\", "/");
5       string formattedVideo2Path = video2Path.Replace("\\", "/");
6       sw.WriteLine($"file '{formattedVideo1Path}'");
7       sw.WriteLine($"file '{formattedVideo2Path}'");
8       }
```

Algoritmo 4: Criando o documento de texto *videolist.txt*

- Concatenar os 2 vídeos e salvar o resultado no caminho armazenado na variável *outputFilePath*:

```
1       // Concatenar os videos
2       var conversion = FFmpeg.Conversions.New()
3       .AddParameter($"-f concat -safe 0 -i \"{videolistPath}\"
           -c copy \"{outputFilePath}\"");
4       await conversion.Start();
```

Algoritmo 5: Comando de concatenação dos vídeos

Explicando mais sobre o comando acima:

- **var conversion = FFmpeg.Conversions.New()**: cria uma nova instância de *Conversion*, que representa um processo de conversão ou manipulação de mídia com o FFmpeg;
- **AddParameter()**: permite adicionar parâmetros personalizados ao comando do FFmpeg;
- **-f concat**: Esse comando instrui o FFmpeg a juntar os vídeos listados em sequência;
- **-safe 0**: Permite o uso de caminhos absolutos (ou relativos) no arquivo *videolistPath*. Por padrão, o FFmpeg rejeita caminhos absolutos por razões de segurança, mas *-safe 0* desativa essa verificação;
- **-i \“{videolistPath}\”**:Define o arquivo de entrada (-i significa “input”), que no caso é videolistPath;
- **-c copy**: Copia o áudio e o vídeo para o arquivo de saída sem recodificação, mantendo a qualidade e acelerando o processo;
- **\“outputFilePath\”**: Define o caminho do arquivo de saída, onde o vídeo concatenado será salvo;
- **await conversion.Start()**: Esse comando inicia o processo de conversão de forma assíncrona, ou seja, sem bloquear a execução do restante do código.

2.3 Acelerando os Vídeos

O FFmpeg oferece funcionalidades para aumentar a velocidade de vídeos, o que inicialmente fez a implementação da aceleração dos vídeos no SpeakRehab parecer simples. No entanto, ao aplicar essa funcionalidade nos vídeos das sílabas do português brasileiro, notou-se que a

velocidade permanecia inalterada, sem qualquer erro aparente. Após extensiva investigação e diversas tentativas de ajuste, identificou-se a raiz do problema: cada vídeo de sílaba possuía apenas 3 *frames*, o que tornava a mudança de velocidade imperceptível.

Para resolver o problema identificado, ficou claro que era necessário encontrar uma forma de aumentar a quantidade de *frames* em cada vídeo. E, felizmente, o FFmpeg já oferece uma funcionalidade que regula a taxa de *frames* de um vídeo, permitindo que, mantendo aproximadamente o tempo original do vídeo, um mesmo quadro seja repetido mais vezes ao longo do tempo e, consequentemente, que o ajuste de velocidade dos vídeos das sílabas seja possível.

A seguir, será demonstrado como utilizar a biblioteca *Xabe.FFmpeg* para acelerar um vídeo com poucos *frames* cujo caminho está armazenado na variável *inputFilePath*:

- Obtendo informações sobre o vídeo de entrada com auxílio do método *GetMediaInfo()* :

```
1       // Obtem as informacoes do arquivo de midia
2       IMediaInfo inputFile = await
           FFmpeg.GetMediaInfo(inputFilePath);
```

Algoritmo 6: Obtendo informações sobre o vídeo

- Definindo a nova taxa de *frames* do vídeo e aplicando o fator de velocidade desejado (*speedFactor*):

```
1       IVideoStream videoStream = inputFile.VideoStreams.First()
2                   .SetFramerate(60)  // Aumenta a taxa de quadros
3                   .ChangeSpeed(speedFactor);  // Aplica a
                       aceleracao
```

Algoritmo 7: Definindo a nova taxa de quadros e a velocidade desejada

Explicando mais sobre o comando acima:

  - **inputFile.VideoStreams.First()**: seleciona o primeiro fluxo de vídeo detectado no arquivo;
  - **SetFramerate(60)**:Define a taxa de quadros do vídeo para 60 frames por segundo (fps);
  - **ChangeSpeed(speedFactor)**: Aplica o *speedFactor* ao fluxo de vídeo, ajustando sua velocidade. Por exemplo, se *speedFactor* for 2.0, o vídeo será reproduzido duas vezes mais rápido.

- Definindo o processo de conversão e iniciando a aceleração do vídeo:

```
1       IConversionResult conversionResult = await
           FFmpeg.Conversions.New()
2       .AddStream(videoStream)
3       .SetOutput(outputFilePath) //outputFilePath eh o caminho
           onde o video acelerado sera salvo
4       .Start();
```

Algoritmo 8: Executando o processo de conversão e acelerando o vídeo

Explicando mais sobre o comando acima:

  - **FFmpeg.Conversions.New()**: cria uma nova instância de um processo de conversão de mídia usando o FFmpeg;

- **AddStream(videoStream)**: Adiciona o fluxo de vídeo *videoStream* à conversão, já configurado com a nova taxa de quadros e o fator de velocidade ajustado;
- **SetOutput(outputFilePath)**: Define o caminho do arquivo de saída onde o vídeo resultante será salvo;
- **Start()**: Inicia o processo de conversão de forma assíncrona.

2.4 Mapeamento das sílabas para os caminhos dos vídeos

Para gerenciar o mapeamento das sílabas aos caminhos dos vídeos correspondentes, foi criado o objeto *RequiredVideos*, que reúne os métodos e atributos necessários para essa tarefa. No entanto, a implementação dessa parte ainda não está totalmente concluída, com algumas funcionalidades ainda pendentes. Mesmo assim, os recursos já implementados permitem a obtenção de resultados bastante promissores.

Para ilustrar a lógica adotada, será utilizado como exemplo o mapeamento da palavra **COLA**:

- Suponha que a separação silábica já tenha sido realizada, resultando nas sílabas **CO** e **LA**. Após um processamento interno, identifica-se a necessidade dos vídeos associados aos pares **CO**, **OL** e **LA**.
- Com os pares **CO**, **OL** e **LA** identificados, o próximo passo é iterar sobre cada um deles para determinar os caminhos dos vídeos correspondentes.Veja abaixo a iteração sendo realizada:
  - **CO**: Ao iterar sobre essa sílaba, é essencial verificar qual vogal está acompanhando a consoante **C**. Por exemplo:
    - Para as sílabas **CA**, **CO** e **CU** é utilizado o visema nomeado de "CACOCUK".
    - Já para as sílabas **CE** e **CI**, o visema utilizado é o "CECI".

    Assim, no caso de **CO**, determina-se o visema correto como "CACOCUK" e, com isso, identifica-se o caminho do vídeo apropriado.
  - **OL**: Esse par de letras representa a transição entre as sílabas **CO** e **LA**, com a letra **L** aparecendo em uma posição intermediária na palavra, sem ser a última letra. Essa observação é essencial para a escolha do vídeo associado ao visema correto, pois o visema da letra **L** em uma posição intermediária, como em **LÁPIS**, é diferente daquele apresentado quando está no final da palavra, como em **SOL**. Essa distinção garante que a representação visual seja adequada e condizente com o contexto fonético da palavra.
  - **LA**: Essa sílaba segue uma lógica semelhante ao caso do par **OL**, pois também requer atenção à posição da letra **L** dentro da palavra.

3 FFmpeg

O objetivo desta seção é explicar de uma forma um pouco mais aprofundada como o FFmpeg consegue realizar a concatenação, aceleração e aumento da taxa de *frames* de vídeos, recursos os quais são fortemente utilizados no SpeakRehab. Para maiores explicações consulte a documentação do ffmpeg em [2].

3.1 Concatenação de Vídeos

O processo de concatenação de vídeos com o FFmpeg envolve vários passos, veja-os a seguir:

- **Leitura dos Arquivos de Entrada**
  - Para concatenar vídeos, o FFmpeg precisa de uma "lista de reprodução" que indique a sequência dos arquivos a serem unidos. O usuário deve criar essa lista em um arquivo de texto (por exemplo, *videoslist.txt*) com o seguinte formato:

```
1 file 'video1.mp4'
2 file 'video2.mp4'
3 file 'video3.mp4'
```

Algoritmo 9: Arquivo de texto com os caminhos dos vídeos

  - Após o passo anterior, o FFmpeg lê os arquivos de vídeo presentes no *videoslist.txt*. Isso é feito analisando os "metadados" do vídeo, como resolução, taxa de quadros e formato de compressão.

- **Compatibilidade dos Vídeos:** Para que dois vídeos possam ser concatenados diretamente (sem reprocessamento), eles precisam ser compatíveis. Isso significa que os vídeos devem possuir:
  - Mesma resolução (por exemplo, 1920x1080);
  - Mesma taxa de quadros (por exemplo, 30 fps);
  - Mesmo codec de vídeo (como H.264) e codec de áudio (como AAC);

  Se os vídeos não forem compatíveis - como é o caso nesse projeto - o FFmpeg faz uma "conversão" (reencode) para ajustá-los ao mesmo formato.

- **Concatenação:**
  - O FFmpeg toma os dados de cada vídeo ("frames") e os organiza na sequência correta;
  - Se os vídeos forem compatíveis, ele basicamente copia os dados diretamente, sem fazer alterações significativas (o que é bastante rápido);
  - Se for necessário reprocessar (reencode), ele lê cada *frame* dos vídeos, ajusta ao formato desejado e depois escreve tudo em um novo arquivo.

- **Escrita do Arquivo de Saída:** O FFmpeg salva o novo vídeo concatenado em um arquivo de saída. Ele também gera os metadados atualizados, como a duração total do vídeo.

3.2 Aceleração de Vídeos

Assim como na concatenação, a aceleração de vídeos contém uma série de etapas, são elas:

- **Leitura do Vídeo Original:** O FFmpeg começa lendo o vídeo, assim como faria para qualquer outro processamento. Ele analisa os "metadados" do vídeo, incluindo a taxa de quadros, que indica quantos quadros são exibidos por segundo;

- **Ajuste do *Framerate*:**

  - Para acelerar o vídeo, o FFmpeg altera o tempo de exibição de cada quadro;
  - Existem duas maneiras principais de fazer isso:
    * **Alterando a taxa de quadros de reprodução:** Isso faz com que o vídeo seja reproduzido mais rapidamente sem descartar nenhum quadro;
    * **Pulando quadros:** Em vez de mostrar cada quadro, o FFmpeg pode "pular" alguns quadros. Isso também acelera o vídeo, mas com uma leve perda de suavidade.

- **Escrita do Vídeo de Saída**: Após ajustar o tempo dos quadros, o FFmpeg cria um novo arquivo de vídeo com os quadros reorganizados para serem exibidos mais rapidamente.

3.3 Ajuste na Taxa de Quadros

A taxa de quadros de um vídeo indica quantas "imagens" são exibidas por segundo. Por exemplo, um vídeo com 30 quadros por segundo (fps) exibe 30 imagens em cada segundo de reprodução, criando a ilusão de movimento contínuo. Ajustar a taxa de quadros de um vídeo pode ser necessário para adequar o vídeo a uma nova configuração ou para efeitos específicos, como suavizar a reprodução ou prepará-lo para outras operações, como a aceleração.

No projeto, foi utilizado o método `SetFramerate` do Xabe.FFmpeg para modificar a taxa de quadros dos vídeos. Esse método permite definir quantos quadros por segundo queremos que o vídeo final exiba. Por exemplo, ao aplicar `SetFramerate(60)` em um vídeo com taxa de 3 fps, configuramos o vídeo de saída para exibir 60 quadros por segundo.

Como o SetFramerate Funciona: No entanto, é importante entender que o `SetFramerate` não cria novas imagens ou quadros entre os quadros originais do vídeo. Em vez disso, ele reorganiza o tempo de exibição dos quadros existentes para preencher o número desejado de quadros por segundo. Isso significa que os quadros do vídeo original podem ser exibidos várias vezes para alcançar a nova taxa de especificada.

Exemplo: Imagine que temos um vídeo de 10 segundos com uma taxa de 3 fps, o que significa que ele contém 30 quadros no total (3 quadros por segundo $\times$ 10 segundos = 30 quadros). Se usarmos o `SetFramerate(60)`, o FFmpeg configura o vídeo para 60 fps, mas como não há novos quadros sendo criados, ele exibe cada quadro original 20 vezes para preencher os 60 quadros por segundo, mantendo aproximadamente a mesma duração total do vídeo.

4 Resultados

Neste projeto, o desenvolvimento do aplicativo *SpeakRehab* resultou em avanços significativos, especialmente no que diz respeito às funcionalidades implementadas e à criação de um protótipo funcional. A Figura (1) apresenta a interface gráfica do aplicativo, evidenciando os elementos principais que o compõem e sua estrutura intuitiva.

Como é possível verificar na Figura (1), embora ainda em fase de protótipo, a interface foi projetada para ser intuitiva e funcional. Ela é composta por um campo de entrada, onde o usuário pode digitar a palavra desejada, e por um botão "Verificar", que executa automaticamente as etapas de verificação ortográfica, separação silábica e geração do vídeo correspondente. Após o processamento, o aplicativo exibe o texto processado, incluindo a palavra separada em sílabas, e apresenta na tela o vídeo gerado com os movimentos labiais correspondentes à palavra digitada.

Apesar dos avanços apresentados, é importante destacar que algumas funcionalidades ainda estão em fase de desenvolvimento. Um exemplo notável é o caso de palavras que possuem sílabas com mais de duas letras, como "FRA" ou "TRE". Atualmente, o aplicativo não é capaz

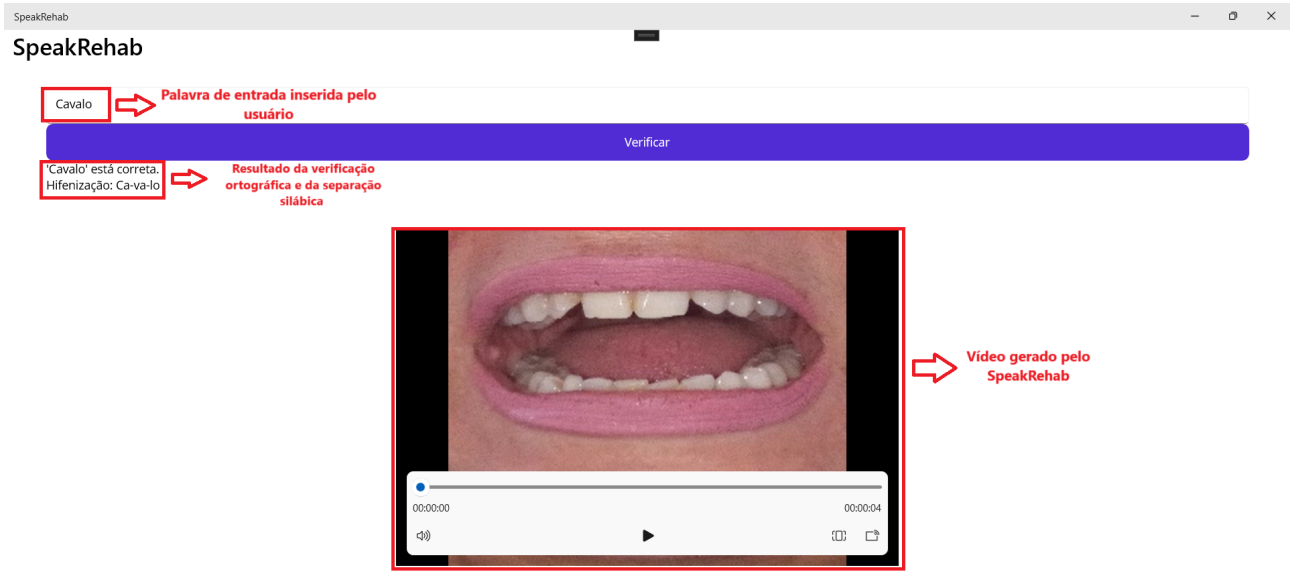

Figure 1: Interface gráfica do protótipo do SpeakRehab.

de processar essas situações de forma completa, devido à complexidade adicional que elas introduzem no mapeamento e na concatenação dos vídeos correspondentes. No entanto, essa limitação já está sendo abordada, e a implementação dessa funcionalidade está em andamento.

Para demonstrar o funcionamento do protótipo, foi produzido um vídeo que apresenta o fluxo completo do aplicativo, desde a inserção da palavra até a geração e exibição do vídeo com os movimentos labiais. Esse vídeo permite visualizar como o *SpeakRehab* processa as palavras e gera os resultados, destacando o potencial do aplicativo em seu estágio atual. O vídeo pode ser acessado no seguinte link: Vídeo Demonstrativo.

5 Conclusão

Nesta segunda fase do projeto, avançamos significativamente no desenvolvimento do *SpeakRehab*. A implementação da separação silábica com os dicionários do LibreOffice foi essencial para a correta divisão das palavras, e o uso do *FFmpeg* facilitou a manipulação de vídeos, possibilitando a concatenação e aceleração necessárias para criar transições suaves entre as sílabas. Com essas conquistas, o aplicativo está mais próximo de seu objetivo de auxiliar na reabilitação de pessoas com afasia.

6 Próximos Passos

Com a finalização da lógica de conversão de sílabas para caminhos dos vídeos e o lançamento da versão inicial do *SpeakRehab*, um próximo passo promissor é utilizar um conversor léxico fonético de *Text-to-Speech*. Esse recurso substituirá a separação silábica pela transcrição fonética, permitindo uma sincronização mais precisa dos vídeos com os fonemas. Além disso, essa abordagem possibilitará a adição de áudio aos vídeos, tornando o aplicativo ainda mais completo e eficaz para a reabilitação de pessoas com afasia.

References

[1] Mário Sérgio Maduro Santana e João Batista Florindo. Modelo de inteligência artificial para transição entre imagens: uma aplicação no tratamento da afasia, 2024. Disponível em: `https://www.ime.unicamp.br/~mac/db/2024-1S-223615.pdf`. Acesso em: novembro de 2024.

[2] FFmpeg Developers. *FFmpeg Documentation*, 2024. Accessed: 2024-11-15.

[3] Microsoft. Introduction to .net maui, 2024. Disponível em: `https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-8.0`. Acesso em: novembro de 2024.

[4] NHunspell Developers. NHunspell: Spell checking, hyphenation, word stemming, and thesaurus library for .net based on hunspell. `https://www.nuget.org/packages/nhunspell`. Accessed: 2024-11-12.

[5] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. *Proceedings of Machine Learning Research*, 2022.

[6] Xabe Software. Xabe.FFmpeg: .net wrapper for ffmpeg to convert and process multimedia files. `https://ffmpeg.xabe.net/index.html`. Accessed: 2024-11-13.