UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JOÃO PAULO ABDALA BOHACZK

RECONHECIMENTO ÓPTICO DE CARACTERES PARA LEITURA E TRADUÇÃO DE DOCUMENTOS EM FORMATO PDF

GUARAPUAVA

2025

JOÃO PAULO ABDALA BOHACZK

RECONHECIMENTO ÓPTICO DE CARACTERES PARA LEITURA E TRADUÇÃO DE DOCUMENTOS EM FORMATO PDF

Optical Character Recognition for Reading and Translating PDF Documents

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Kelly Lais Wiggers

GUARAPUAVA

2025

RESUMO

Diante da inacessibilidade de determinados livros e outros textos publicados antes da era digital, bem como do tratamento secundário da comunidade de OCR para com o tratamento dos documentos analisados em texto digital próprio para a leitura casual, determina-se a necessidade de uma ferramenta que, trabalhando com OCR, dedique-se na extração de textos de imagens ou arquivos PDF, visando facilitar a leitura e disseminação desses textos. Para atingir esse objetivo será realizada uma pesquisa comparativa entre ferramentas OCR. Será construída uma API em Python para realizar o trabalho de processamento do OCR e a formatação do texto. Isso será integrado por uma interface web desenvolvida com o framework PHP Laravel, com o objetivo de disponibilizar o projeto para a maior quantidade de pessoas possível. O presente trabalho possui tanto uma via científica e experimental como prática, buscando comparar as ferramentas OCR e solucionar algo que é até então ignorado pelas ferramentas, a devida formatação do texto para leitura, também tendo como objetivo disponibilizar os resultados dessa pesquisa como uma ferramenta de fácil uso e acesso para a comunidade.

Palavras-chave: ocr; pdf; revistas.

ABSTRACT

Given the inaccessibility of certain books and other texts published before the digital era, as well as the secondary treatment by the OCR community toward the processing of analyzed documents into digital text suitable for casual reading, there is a need for a tool that, working with OCR, focuses on extracting text from images or PDF files, aiming to facilitate the reading and dissemination of these texts. To achieve this objective, comparative research between OCR tools will be conducted. A Python API will be built to perform the OCR processing work and text formatting. This will be integrated through a web interface developed with the PHP Laravel framework, with the goal of making the project available to as many people as possible. The present work has both a scientific and experimental aspect as well as a practical one, seeking to compare OCR tools and solve something that has been ignored by this tools until now: the proper formatting of text for reading, also aiming to make the results of this research available as an easy-to-use and accessible tool for the community.

Keywords: ocr; pdf; magazines.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A escrita é uma grande invenção da humanidade, sendo ela primariamente a representação dos fonemas sonoros, isto é, representação da fala através de símbolos (SCHMANDT-BESSERAT; ERARD, 2009, p. 7). É o salto tecnológico que permitiu que a humanidade se desenvolvesse intelectualmente através do compartilhamento de informações. Durante longo tempo a escrita foi empregada em diferentes materiais e superfícies com o objetivo de transmitir informações, passando por pedras e papiros até finalmente chegar ao papel que é conhecido atualmente. (GABRIAL, 2009)

Grande parte dos escritos da humanidade estão presos em mídia física, como livros e revistas, o que dificulta a sua transmissão por meios digitais. Uma das formas de compartilhamento mais eficientes é através de fotografias que podem ser juntadas em, por exemplo, um arquivo PDF. Isso preserva a sua estrutura original, mas impede a análise de dados, pesquisa e capacidade de compartilhamento que poderia ser obtida se o texto fosse devidamente convertido em formato digital.

A humanidade conta com séculos de história escrita inacessíveis, ou com acesso dificultado e inapropriado, para o formato digital. É de grande interesse obter esses textos digitalizados propriamente para o seu compartilhamento, de forma que se preserve ao máximo a sua estrutura original, facilitando a sua leitura e disseminação nos meios digitais.

Algumas tecnologias atuais se destacam na extração de textos em imagens para o formato digital, como por exemplo, Optical Character Recognition (OCR), que é muito popular nos dias de hoje, estando presente na maioria dos smartphones na função da câmera ou na galeria de fotos, fazendo com que seja fácil extrair textos de fotos. Pesquisas recentes têm utilizado OCR para análise de documentos históricos (BOENIG *et al.*, 2019), análise de jornais antigos (DROBAC, 2019) e extração de texto de fotografias de forma geral (LI *et al.*, 2022, p. 9).

Uma das dificuldades encontradas na extração de texto via ferramentas de OCR é a falta de formatação dos textos extraídos. O objetivo do OCR é a extração do texto puro para processamento computacional, contudo, a sua reconstrução para a leitura humana é algo secundário e que muitas vezes deixa a desejar. Outro problema é que essas ferramentas tem de ser treinadas usando Ground Truth (GT) para se tornarem mais eficientes, uma técnica de treinamento em que se mantêm um arquivo que corresponde ao resultado final da análise de OCR, como espécie de prova real do resultado final (BOENIG *et al.*, 2019).

Visto que o acesso a essas ferramentas e sua otimização exige um olhar especializado e dedicado, esse trabalho busca solucionar esse problema, focando na extração e compartilhamento de textos. Além disso, visa obter uma solução genérica o suficiente para ser usada em diferentes situações.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo deste trabalho é extrair e transcrever textos completos provenientes de arquivos, os quais podem ser imagens ou arquivos em formato PDF, mediante técnicas de processamento de imagens e OCR.

1.1.2 Objetivos específicos

- Definir a base de dados para realização dos experimentos.
- Efetuar pré-processamento na base de dados, como realce, redimensionamento, recortes, dentre outros.
- Implementar experimentos comparativos com as tecnologias de OCR disponíveis.
- Construir uma API para permitir a inserção e formatação de arquivos
- Criar um protocolo experimental robusto para validação dos resultados.

1.2 Justificativa

Boenig *et al.* (2019), ao comentar a digitalização de textos por bibliotecas, mostra a distinção entre "digitalização de imagem"(*image digitalization*) e o processamento dessas imagens por OCR. Os avanços na tecnologia de OCR permitiram que a maioria desses textos fossem processados, resultando em *dirty full text*, um texto sujo, com erros e sem formatação, que tem validade de processamento para alguns casos científicos, mas que pecam em sua precisão e falta de estrutura.

Isso faz com que a conversão de uma página de, por exemplo, uma revista ou um livro, seja uma fotografia ou um arquivo PDF composto por essas fotografias, seja frustrante, caso o objetivo seja a leitura por humanos, devido as imprecisões no resultado final. Por vezes o texto digitalizado final acaba sendo ilegível.

Portanto, ao avaliar-se que a tecnologia de OCR está mais direcionada para a análise de dados do que necessariamente para o consumo humano desses dados, o desenvolvimento de tecnologias que auxiliem na leitura e transcrição do conteúdo para versões em formato digital pode ser visto como complemento e optimização da tecnologia, focando no consumo humano desses textos.

Como contribuição, tem-se o desenvolvimento de uma ferramenta que visa facilitar a formatação de textos digitalizados com OCR, preservando sua estrutura original, o que torna

sua leitura mais agradável e aumenta seu potencial de difusão por meios digitais, como redes sociais e sites dedicados a disseminação cultural.

A principal motivação para o desenvolvimento deste projeto é a dificuldade que se encontra no compartilhamento, anotação e localização de textos que tiveram seu principal período de publicação anterior à popularização da internet. E, consequentemente, esses textos podem não ter sua republicação disponível em formato digital.

Isso é um caso comum em, por exemplo: cursos de faculdades da área de humanas, em que há um desinteresse por parte das editoras na republicação de determinados livros que ainda são muito lidos internamente na academia, fazendo com que um dos parcos meios de acesso desse material seja por escaneamento (VARELLA; FERREIRA, 2012)

O processamento desses textos é custoso e tecnicamente desafiador, mas os avanços recentes de reconhecimento textual utilizando redes neurais oferece maior precisão e eficácia para a área. Isso possibilita visualizar o avanço do uso de redes neurais na tecnologia de OCR, bem como suas possíveis limitações e avaliar sua eficiência. Existem várias ferramentas de OCR disponíveis no mercado que vão desde o nível industrial (PaddleOCR) até o nível acadêmico (OCR-D), o que faz disso um bom momento para testes e comparações entre elas.

Foram selecionadas cinco ferramentas de OCR para testes. PaddleOCR (PaddlePaddle Community, 2020) é uma ferramenta chinesa que tem foco em nível industrial de escalonamento; OCR-D (OCR-D Project, 2020) faz parte de um programa alemão de preservação histórica, tendo foco em documentos históricos dos séculos XVI ao XVIII; Docling (Docling Team, 2024) tem como foco a comunidade de IA, especializando-se em extrair texto em arquivos *json* e *markdown*; Calimari-OCR (WICK; REUL; PUPPE, 2020) tem foco tanto em fontes históricas como modernas e busca ser mais simples e modular; Tesseract (TEAM, 2024) é a ferramenta mais tradicional da comunidade, contando com boa documentação e vastos exemplos de implementação devido a sua popularidade.

Como resultado do projeto desenvolvido será possível disponibilizar uma ferramenta que digitalize texto a partir de imagens ou arquivos PDF facilmente, bem como contribuir para uma pesquisa comparativa de ferramentas OCR disponíveis no mercado.

2 FUNDAMENTAÇÃO TEÓRICA

/Nessa seção serão apresentados as principais definições das técnicas que envolvem visão computacional aplicada na detecção e reconhecimento de textos.

2.1 Visão computacional

A visão computacional é um ramo da inteligência artificial que permite aos computadores interpretar e compreender informações visuais de imagens e vídeos. Muitas vezes, supera-se a percepção visual humana devido o uso de algoritmos especializados (KRISHNA, 2017).

Uma aplicação fundamental de visão computacional é a detecção de texto em imagens, nos quais os sistemas podem identificar e extrair automaticamente o conteúdo textual incorporado em fotos, documentos digitalizados ou *frames* de vídeos. Esse processo, frequentemente chamado de OCR, envolve a detecção de regiões que contém texto e o reconhecimento dos caracteres e palavras para convertê-los em formatos legíveis por máquinas.

O pipeline típico de uma aplicação OCR se divide em pré-processamento, processamento e pós-processamento. Alguns modelos OCR possuem processos de pré e pós-processamento integrados. No pré-processamento, geralmente se tem a optimização da imagem para facilitar a detecção da região do texto e a própria detecção de caracteres, em seguida é realizada detecção de texto, classificando as regiões de interesse em que o a detecção de caracteres será realizada (SUBRAMANI *et al.*, 2020).

Para resolver problemas de visão computacional, é considerado o formato de reconstrução da visão humana. Um computador não tem toda a contextualização que um ser humano possui para, por exemplo, distinguir os planos de uma imagem, localizando seu centro e seu fundo. Portanto, uma parte do estudo da área consiste em um esforço por conseguir contextualizar as imagens através de algum algoritmo para que operações sejam possíveis. (SZELISKI, 2010)

A partir dessas questões, tecnologias como *cadeias de markov* e *machine learning* surgem de forma associada ao ramo da visão computacional. Essa evolução em torno da contextualização são os primeiros passos para a formulação de IA's (SZELISKI, 2010). Portanto, realizando esse contexto de evolução histórica, o presente artigo também se propõe a explorar técnicas de transcrição utilizando mLLMs (*Multimodal Large Language Models*), pois os modelos multimodais possuem capacidade contextual maior, sendo capazes de ler imagens que possuem agravantes que podem ser ignorados por humanos, mas que representam dificuldades para uma aplicação OCR típica, necessitando de correções.

2.1.1 Processamento de imagens

Segundo Gonzalez e Woods (2017) uma imagem pode ser definida como uma função bidimensional, representada por *f(x,y)*, onde *x* e *y* são coordenadas espaciais, respectivamente horizontal e vertical, e o valor da função *f* indica a intensidade ou nível de cinza da imagem, que pode variar de 0 a 255. Quando os valores dessa função são finitos e discretos eles resultam em uma imagem digital.

Gonzalez e Woods (2017) elenca as principais etapas para processamento de imagens digitais como sendo: aquisição de imagem, filtragem, aprimoramento, processamento de cores, *wavelets* [1], compressão, manipulação morfológica, segmentação, extração de características e classificação de padrões.

Considerando as etapas de processamento levantadas por Gonzalez e Woods (2017), o foco do presente trabalho são pré-processamento, morfologia, segmentação e extração de características das imagens.

2.1.2 Pré-processamento

Bieniecki, Grabowski e Rozenberg (2007), chama atenção para o pré-processamento de imagens que foram adquiridas por aparelhos celulares (em contraposição a documentos escaneados), as deformações correspondentes a esse tipo de captura, os seus efeitos no processamento de OCR e como amenizá-los.

Os principais pontos levantados por Bieniecki, Grabowski e Rozenberg (2007) são a distorção geométrica, perca de foco e iluminação não uniforme. Para amenizar isso são aplicadas técnicas de rotação de imagem, correção de perspectiva e transformação de imagem não linear. Um dado interessante ressaltado das tabelas que o trabalho produziu é que imagens com 600dpi resultam em menor acurácia do que imagens com 300dpi de resolução.

Os métodos de correção usados na experimentação proposta por Bieniecki, Grabowski e Rozenberg (2007) são de rotação, correção de perspectiva e transformação não linear. Para o experimento foi utilizada uma imagem de uma página com rotação em 90°. Os resultados foram uma imagem sem área de texto detectável para a imagem original e para a correção de rotação, para 2,72% de erros com a correção de perspectiva e 0,96% de erros com a transformação não linear.

Um dado valioso desse artigo é a perspectiva histórica de ferramentas OCR serem optimizadas para a leitura de documentos escaneados, fazendo com que eles tenham iluminação uniforme, sem distorção espacial. Esse trabalho se propõe em aumentar a eficiência de documentos adquiridos através de fotografias, sendo necessário o pré-processamento aplicado por Bieniecki, Grabowski e Rozenberg (2007).

[1] Termo em inglês que indica a representação da imagem em várias etapas de resolução

2.1.3 Morfologia Matemática

É um processo que pode ser utilizado em imagens binarizadas para aprimorar o reconhecimento de caracteres. Ela se concentra na análise da forma, geometria e estrutura dos objetos presentes em imagens, utilizando operadores não lineares e elementos estruturantes para transformar e extrair informações relevantes(GIROD, 2018).

A morfologia pode ser aplicada tanto no sentido de, por exemplo, expandir as sessões em preto (as linhas de uma letra), para torná-las mais legíveis, ou diminuir as mesmas linhas, com o mesmo objetivo, a depender da mancha da fonte. Em termos gerais é um processo que modifica a forma da imagem (SZELISKI, 2010). Duas operações comuns de morfologia matemática são:

- Abertura: Suaviza contornos, elimina pequenos objetos e ruídos, sendo composta por erosão seguida de dilatação.
- Fechamento: Preenche pequenas lacunas e conecta regiões próximas, sendo composta por dilatação seguida de erosão.

2.1.4 Segmentação de imagens

A segmentação de imagens é o processo de dividir uma imagem em suas partes constituintes ou objetos, de modo que cada parte seja mais simples e mais significativa para análise. A segmentação visa identificar regiões homogêneas com base em propriedades como intensidade, cor, textura ou forma, facilitando a extração de informações relevantes para tarefas posteriores, como reconhecimento ou análise de cena (GONZALEZ; WOODS, 2017).

O processo de binarização da imagem, também conhecido como *thresholding*, é um tipo de segmentação que tem por objetivo destacar áreas de interesse das imagens. É uma técnica simples que separa objetos do fundo com base em um ou mais valores de intensidade.

No caso do presente artigo o interesse na segmentação de imagem está aliado com a detecção de textos. Ao mesmo tempo que a imagem deve ser segmentada na área de texto detectada, como o objeto de interesse são imagens fotografadas por celulares, as imagens podem conter resíduos de outras páginas em suas bordas, sendo necessária uma segmentação da área de interesse principal. Essa segmentação pode ser obtida por algoritmos de análise de *layout* de documentos ou através de *bounding boxes*, que contornam as principais áreas de interesse da imagem, podendo ser feita a eliminação das outras áreas que não a principal (SZELISKI, 2010).

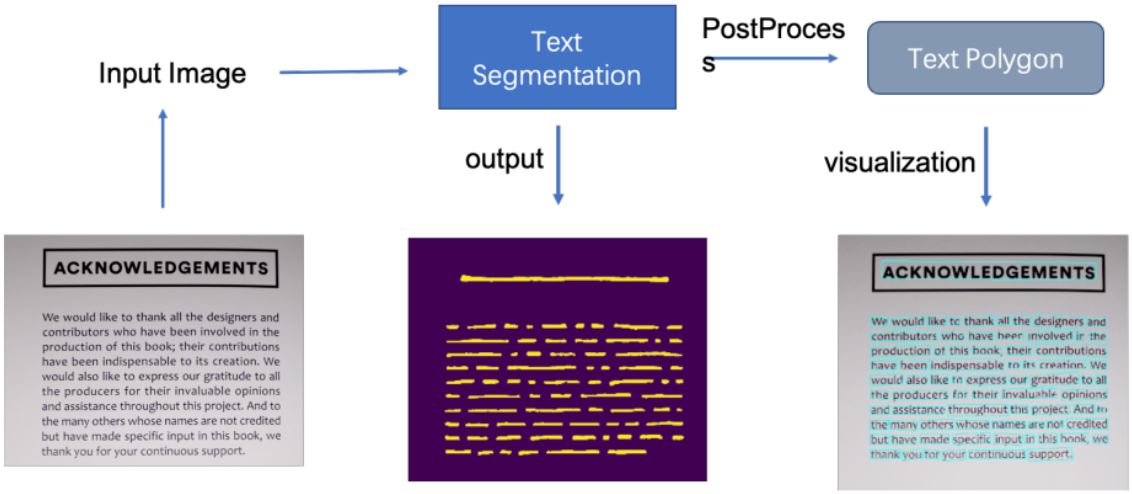


Figura 1 – Demonstração de região de interesse processada pelo paddleOCR, retirado de Li *et al.* (2022)

2.1.5 Extração de características

As técnicas de visão computacional podem permitir o reconhecimento de uma gama de características, que vão de objetos, faces, textos, logos, pontos de referência etc... (KRISHNA, 2017). Desta forma, a extração de características é a etapa na qual se obtêm informações quantitativas relevantes a partir das regiões ou objetos segmentados na imagem.

Para o presente trabalho um dos interesses está na extração de *layout* e de diagramação das páginas, podendo detectar títulos e subtítulos, bem como classificar regiões que possam representar a contagem de páginas ou o cabeçalho corrente. Alguns OCRs como o paddleOCR (PaddlePaddle Community, 2020) possuem ferramentas nativas para extração de layout.

2.2 Detecção e reconhecimento de textos

A detecção de textos é uma técnica de visão computacional que identifica e localiza regiões que contêm texto em imagens ou documentos digitalizados, permitindo a extração desse texto para posterior processamento, pesquisa ou análise. A detecção de texto normalmente é a primeira etapa nos fluxos de trabalho de OCR. Assim que as regiões de texto são detectadas, os algoritmos de OCR reconhecem e convertem os caracteres visuais em texto legível por máquina, permitindo edição, pesquisa e extração de dados.

Nesse sentido, o objetivo da fase de detecção de textos é encontrar a posição do texto dentro de uma imagem. A área detectada como área de texto pode ser tanto uma linha de texto como somente uma letra ou palavra, como pode ser visto na Figura 1.

Uma abordagem interessante foi apresentada por Li *et al.* (2022), os quais dividiram os métodos de detecção de texto em duas categorias, baseadas em regressão e baseadas em segmentação. Sendo:

- No método regressivo a imagem tem duas propriedades, o texto detectado e o restante é tido como *background*. A imagem é segmentada por uma rede neural, depois é feito um calculo de probabilidade daquele segmento conter um texto.
- O método de segmentação se baseia em uma análise pixel-a-pixel da imagem, também por uma rede neural, para então determinar se esses pixels formam um bloco textual. Devido a esse método se torna mais eficiente na detecção de textos curvados e tem mais precisão dessa segmentação.

Por sua vez, o reconhecimento de texto é uma subárea do OCR, que se realiza após a detecção do texto. Tem como objetivo converter a imagem segmentada em texto. De forma tradicional o reconhecimento de texto se divide em três etapas: 1) pré-processamento de imagem, 2) segmentação de caracteres e 3) reconhecimento de caracteres.

No trabalho de Li *et al.* (2022), os autores dividiram o reconhecimento de texto em duas categorias: reconhecimento de texto regular e reconhecimento de texto irregular. Sendo:

- O método regular é utilizado em imagens mais horizontais, como imagens documentos escaneados.
- O método irregular é usado para cenários naturais, em que o texto não está horizontalmente posicionado na imagem, também podendo estar distorcida, borrada ou com algum outro tipo de irregularidade.

Avanços recentes combinam OCR com *Large Language Models* (LLM) para detectar textos, criando sistemas híbridos para detectar textos e também compreender o contexto e estrutura. Uma LMM é um modelo de linguagem capaz de interpretar e gerar linguagem humana, tendo capacidade de aprendizado a partir de uma entrada, assim como é capaz de resolver tarefas, interpreta e organiza informações. LLMs são treinadas a partir de uma grande quantidade de dados, e conseguem construir texto de forma probabilística, determinando as palavras seguintes a partir dos dados com que foi treinada (MINAEE *et al.*, 2024).

Greif, Griesshaber e Greif (2025) faz uma comparação entre transcrições feitas com OCR e *Multimodal Large Language Models* (MLLM). Enquanto as tecnologias OCR exigem um pipeline complexo e de treinamento em suas múltiplas etapas, sendo elas, de forma geral, pré-processamento, processamento e pós-processamento, as mLLMs propõem uma solução de um passo a partir do prompt implementado.

Segundo Greif, Griesshaber e Greif (2025) o processamento por mLLMs é mais eficiente do que o estado de arte atual das tecnologias OCR para textos impressos no alfabeto latino,

chegando a ser comparado a precisão de uma transcrição manual, mas possuem desempenho inferior para textos históricos escritos.

As mLLMs são mais eficientes do que soluções OCR e não precisam de qualquer pré-processamento de imagem. Contudo, seu custo de processamento é mais elevado em comparação com o OCR. Outro fator a se levar em consideração é o *black box* típico de uma LLM, ou seja, temos entradas e saídas dessa LLM, mas o modo como o processamento da entrada resulta em determinada saída pode ser difícil de precisar, podendo ocorrer resultados imprevistos.

2.3 Pós-processamento

Nguyen *et al.* (2021) classifica o trabalho de pós-processamento em duas etapas: detecção de erros e correção de erros. Para a correção de erros de forma automática o artigo explora as seguintes abordagens:

- **Misturar múltiplos outputs:** Consiste em várias leituras OCR do mesmo *input*, podendo ser feitas várias condições diferentes, até mesmo com múltiplas ferramentas, fazendo a junção dessas leituras a partir de algum critério;
- **Abordagem lexical:** A correção é realizada a partir de um dicionário verificando o nível de assertividade das palavras. Esse dicionário pode ser customizado a depender da área de especificidade do texto, algumas abordagens criativas para realizar a correção são exemplificadas, como realizar uma pesquisa no Google com as palavras erradas para que o motor de busca sugira a correção;
- **Modelos de erros:** Baseiam-se no número de mudanças de caracteres por palavra para realizar correções de forma descontextualizada, podendo ajustar palavras que possuem caracteres a mais ou a menos.

Greif, Griesshaber e Greif (2025) chama a atenção para o uso de mLLMs no pós-processamento dos resultados de OCR como meio de correção de erros, mas destaca sua maior eficiência para um processo *end-to-end* em si.

Como o esperado de um artigo que é focado em um *output* para processamento de dados, a formatação do texto não é levada em conta. Para isso é possível trabalhar a partir dos metadados que a ferramenta de OCR fornece, como ocorre, por exemplo, com o PaddleOCR (PaddlePaddle Community, 2020) e o Tesseract (TEAM, 2024).

O Tesseract, por exemplo, pode extrair dados de linha de texto, bloco de texto posição do conteúdo e grau de confiança da transcrição, entre outros (TEAM, 2024).

Esse tipo de dado será importante em uma etapa de reconstrução do texto, como é o maior objetivo deste trabalho, mantendo a sua formatação original. Também é importante para sinalizar o grau de confiança da transcrição.

Com a devida separação do que pode ser parágrafo, título, imagem e outros componentes do texto é possível construir *outputs* de forma adaptada para diversos formatos. No caso do uso de uma mLLM para a extração do texto Greif, Griesshaber e Greif (2025) utiliza do próprio *input* no *prompt* para enviar instruções da formatação esperada no retorno seu retorno.

2.4 Trabalhos relacionados

A Tabela 1 apresenta os trabalhos relacionados encontrados na literatura que são similares ao presente projeto. Verifica-se que os autores utilizam o OCR integrado com outras técnicas, tais como LLM e DNN para treinamento das bases de dados. Contudo, considerando que é necessário o treinamento, muitos documentos em diferentes idiomas foram requisitados para a realização dos experimentos.

Tabela 1 – Trabalhos relacionados

Artigo	**Técnica**	**Vantagens**	**Limitações**
Greif, Griesshaber e Greif (2025)	comparação entre mLLM e OCR.	Não precisa de treinamento, consegue por vezes ser mais eficiente que abordagens OCR tradicionais.	Custo computacional elevado. Tem de lidar com *black boxes*.
Drobac (2019)	OCR com multiplas linguagens e análise de layout.	Desenvolvimento de um modelo OCR capaz de processar mais de uma língua, extração da formatação de jornais em múltiplas colunas.	Precisa de dados em várias línguas para ser treinado.
Kumar, Tanwar e Tiwari (2025)	Comparativo entre técnicas OCR e usando DNN.	Solução end-to-end, não é necessário saber exatamente o que acontece na execução dos testes, sendo fácil e rapidamente aplicável. É mais malevável do que uma aplicação OCR típica.	Precisa de um corpus maior para treinamento.

3 MATERIAIS E MÉTODOS

Para o desenvolvimento deste projeto, serão realizadas as etapas apresentadas na Figura 2. Primeiramente, será organizada a base de dados. Após isso, um aprimoramento com técnicas de processamento de imagens será executado. Na sequência, serão realizados experimentos comparativos das tecnologias OCR disponíveis, bem como LLMs recentes. Após, os resultados serão comparados no processo de digitalização da base de dados escolhida.

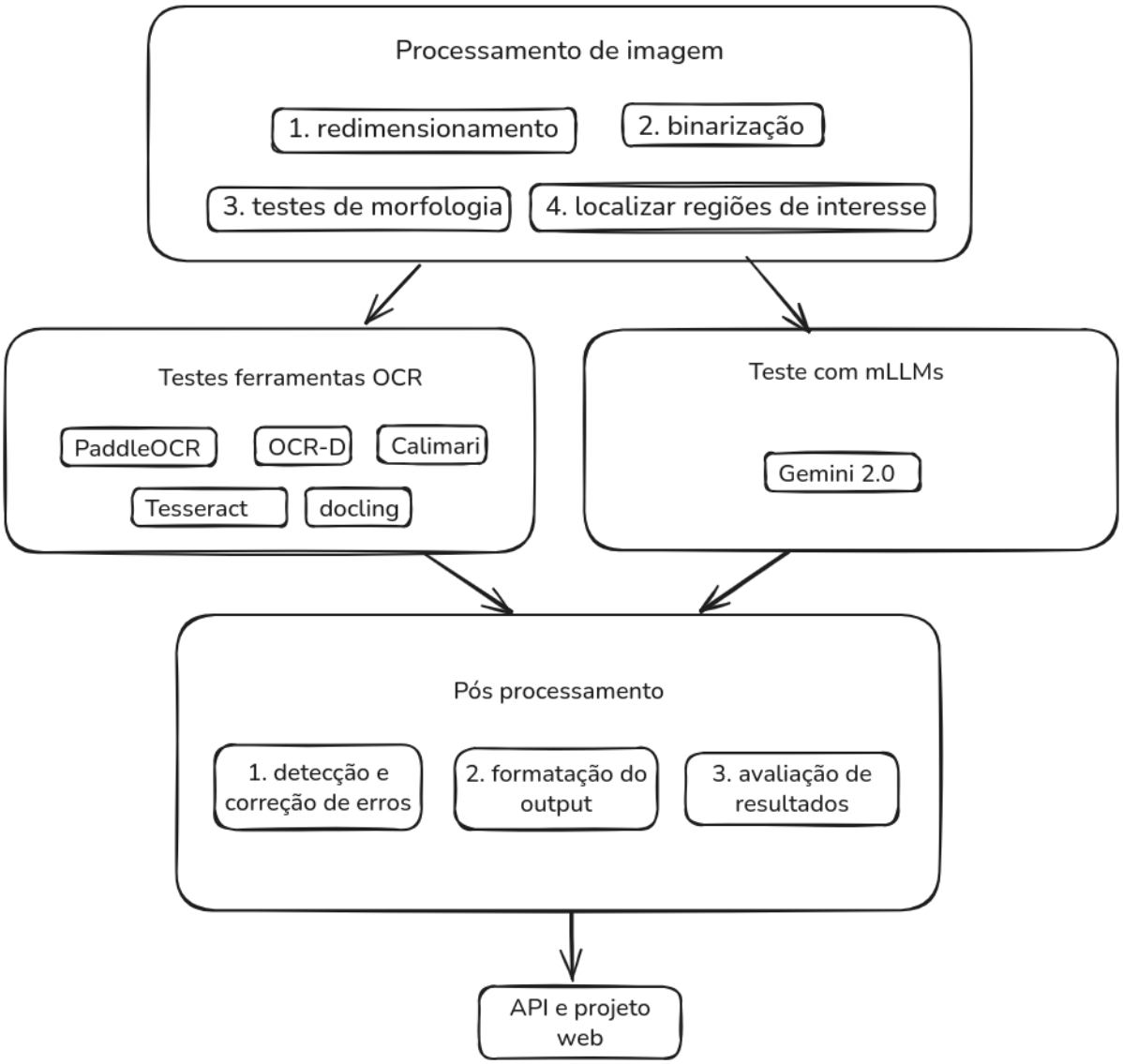
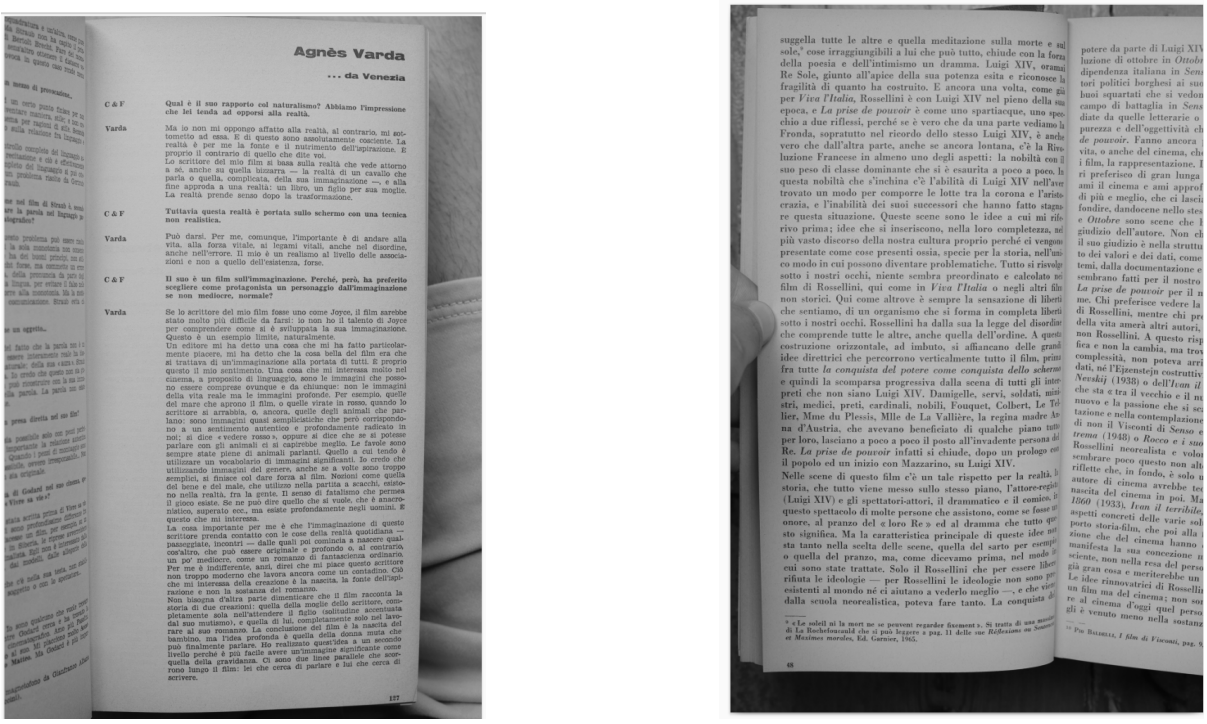


Figura 2 – Diagramas de Etapas do desenvolvimento do Sistema

3.1 Materiais

3.1.1 Base de dados

A base de dados escolhida para os experimentos é a primeira edição da revista italiana de cinema, *Cinema & Film*, que teve suas atividades entre 1967-1970. Essa revista foi escolhida por ser um material de difícil acesso, disponível somente em bibliotecas especializadas de cinematecas, fazendo com que a maior parte do público atual da revista tenha acesso somente através de fotografias que foram transformadas em arquivos PDF. A formatação da revista em texto digital visa tanto facilitar a leitura da revista como a maior autonomia do usuário para ferramentas de tradução. Exemplos das imagens digitalizadas da base de dados são apresentados na Figura 3.



(a) **(b)**

Figura 3 – Exemplos de páginas do livro Cinema & Film

3.1.2 Ferramentas

- OpenCV para processamento de imagens;
- Ferramentas OCR: PaddleOCR, Tesseract, Calimari, Docling e OCR-D;
- Gemini 2.0 para realizar teste com mLLMs;
- Python para criar uma API usando as ferramentas OCR;

- Laravel para fazer uma aplicação web intuitiva e de fácil acesso para a ferramenta;

3.2 Métodos

3.2.1 Processamento de imagens

Na fase de pré-processamento, quando o arquivo de entrada for um PDF, ele deve primeiramente ser convertido em imagens de alta resolução, garantindo a preservação dos detalhes visuais essenciais para a detecção de texto.

Em seguida, cada imagem gerada passa por uma série de etapas fundamentais de processamento, sendo elas:

- Aplicar o redimensionamento para padronizar as dimensões das imagens, otimizando o desempenho e a precisão dos algoritmos subsequentes.
- Realizar a binarização para converter a imagem em um formato de alto contraste, facilitando a separação do texto do fundo.
- Testar processos de morfologia matemática, como abertura e fechamento, para melhorar a qualidade dos caracteres.
- Realizar a localização das regiões de interesse (ROIs), onde o texto está presente, utilizando técnicas de seleção manual ou então baseadas em análise de contornos, detecção de bordas ou modelos de aprendizado profundo, conforme descrito em (LI *et al.*, 2022). Essas etapas combinadas aumentam significativamente a eficiência e a acurácia da ferramenta OCR durante o reconhecimento textual.

3.2.2 Detecção de textos

Nessa etapa serão comparadas algumas ferramentas de OCR disponíveis na literatura, tais como: PaddleOCR (PaddlePaddle Community, 2020), Docling (Docling Team, 2024), OCR-D (OCR-D Project, 2020), Calamari-OCR (WICK; REUL; PUPPE, 2020) e Tesseract (TEAM, 2024).

Os experimentos utilizando mLLM, serão baseados no artigo de Greif, Griesshaber e Greif (2025). Nesse artigo é utilizado o Gemini em sua versão gratuíta atual, o 2.0-flash.

Além de uma solução end-to-end com Gemini, ele também pode ser aplicado na fase de pós-processamento para correções.

É importante destacar que cada ferramenta possui tratamentos específicos. Uma ferramenta pode conseguir extrair a formatação da página e categorizá-la, enquanto outra pode não ter essa função. Portanto, o desenvolvimento das capacidades de pré e pós processamento serão determinadas, em partes, pela ferramenta selecionada.

3.2.3 Avaliação dos resultados

A avaliação das ferramentas terá como critério a maior quantidade de acertos de palavras em conjunto com a facilidade que consegue-se extrair a formatação da página original, isto é, a detecção de linhas, parágrafos e imagens do texto original e a maleabilidade do *output* gerado por ela dessas.

Será seguido o mesmo método de avaliação disponibilizado em Greif, Griesshaber e Greif (2025), em que é feito um cálculo de distância de Levenshtein e a partir disso é verificada a quantidade de erros por caractere e por palavra, conforme a equação 1:

$$\frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{N} \tag{1}$$

O cálculo de Levenshtein consiste em uma contagem de inserções, deleções e substituições de letras, para o caso de *Word Error Rating* (WER), ou palavras, para o caso de *Character Error Rating* (CER) do resultado final da ferramenta OCR em comparação com um arquivo *Ground Truth*. Essa contagem de Levenshtein é então dividida pelo total de caracteres ou palavras do arquivo *Ground Truth*, representado na equação por *N*, para se obter a porcentagem de cada CER e WER. (GREIF; GRIESSHABER; GREIF, 2025).

3.2.4 Desenvolvimento da API e Sistema Web

Será construída uma API em Python que realize a digitalização e formatação do texto extraído de imagens e arquivos PDF, visando melhorar os erros cometidos na fase de digitalização, bem como no pós-processamento de texto encontrados no processamento de ferramentas de OCR. A API será integrada a um projeto *web* desenvolvido com o *framework* Laravel, visando facilitar a disponibilização do projeto a um público abrangente.

Como resultado, espera-se criar uma ferramenta que possa digitalizar e formatar o texto digitalizado de forma eficiente, baseado em sua formatação original. A ferramenta irá possibilitar a inserção de um documento, seja em PDF ou imagem, e trará como resultado o texto digitalizado. Além disso, permitir a disponibilização dessa ferramenta ao público que deseje digitalizar um texto, (mais potencialmente um público acadêmico).

4 RESULTADOS PARCIAIS

4.1 Preparação da base de dados

A base de dados consiste de um PDF composto por imagens da revista Cinema & Film que foram fotografadas. O PDF contêm 162 páginas e ocupa espaço de 771MB. Portanto, a base de dados é composta de imagens de alta resolução, cada página ocupando em torno de 4,8MB. A capa da revista está colorida, mas o restante passou por um processo de *grayscaling*.

4.2 Desenvolvimento WEB

Foi feita uma prototipação da parte web do projeto, focando em deixar o ambiente já configurado para a fase de testes com as ferramentas OCR. Para a prototipação a implementação foi feita usando o Gemini 2.0-flash por ser a ferramenta de mais fácil acesso e que exige menos configurações, não sendo necessário treinamentos para conseguir um resultado satisfatório, como notado por Greif, Griesshaber e Greif (2025) .

Para a fase de prototipação focou-se nos elementos principais do projeto como maneira de deixar clara as intenções do trabalho. A tela principal, que pode ser vista na Figura 4, possui um *input* que recebe tanto imagens como arquivos PDF. Os arquivos PDF são convertidos em PNG antes de serem enviados ao Gemini.

O *prompt* instrui o Gemini a extrair os principais dados da página, como título, subtítulo, parágrafos e número da página em um formato *json*, que é então processado para a visualização do usuário. Foi construída uma versão inicial do *prompt*, que pode ser vista no apêndice A, e

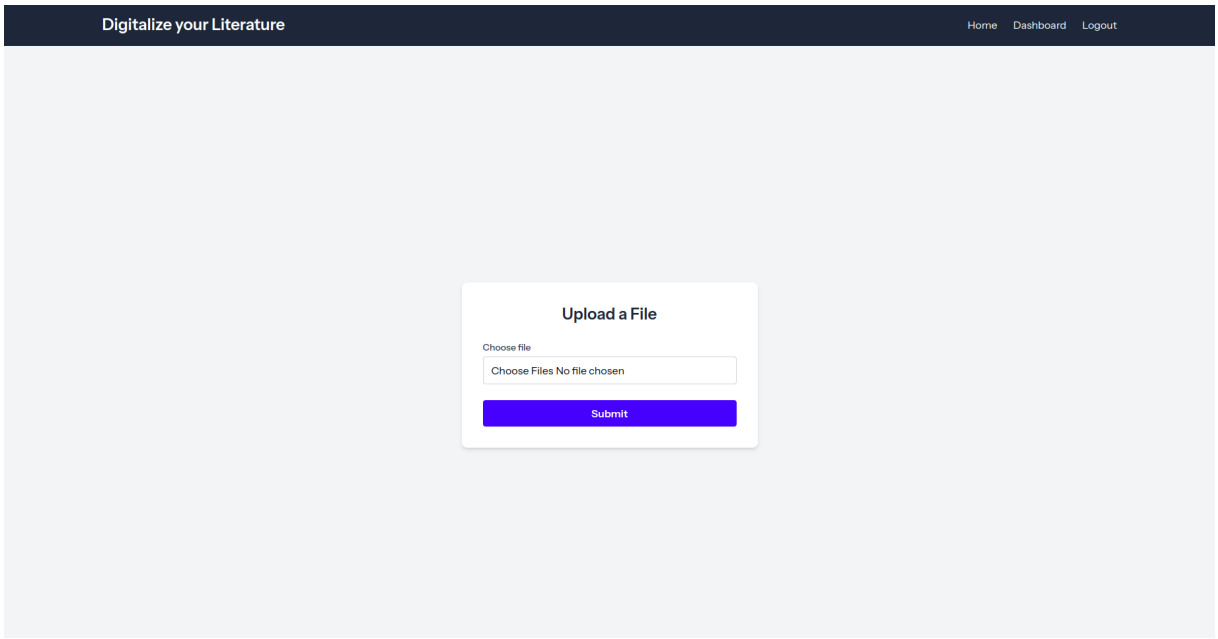


Figura 4 – Tela principal

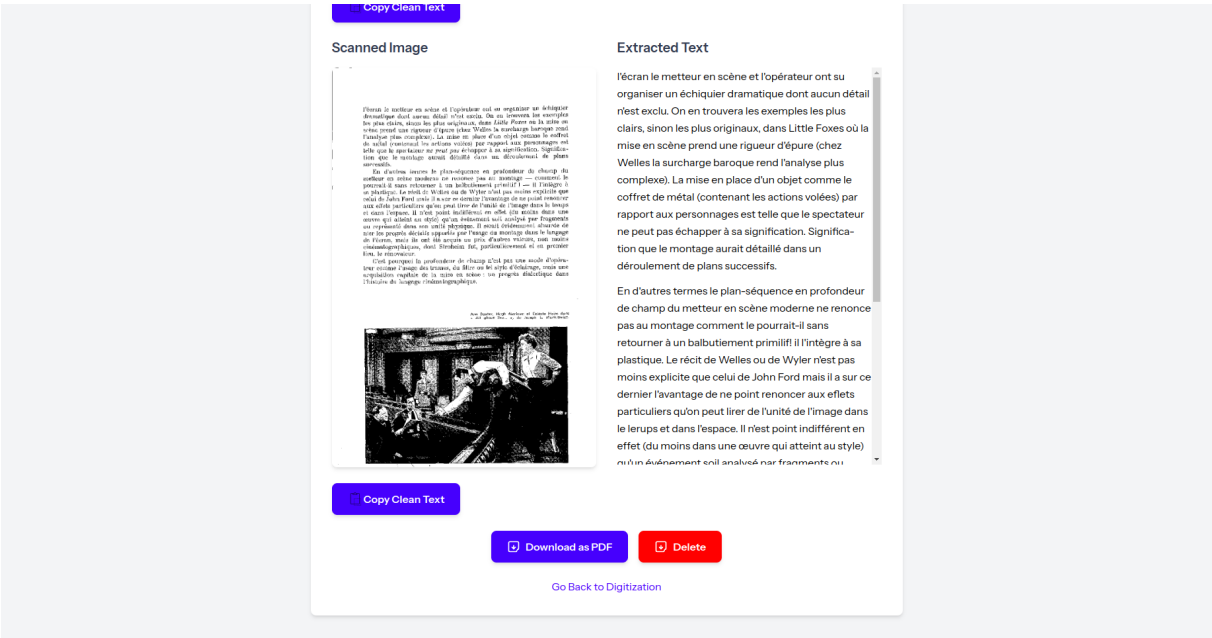


Figura 5 – Resultado Digitalização

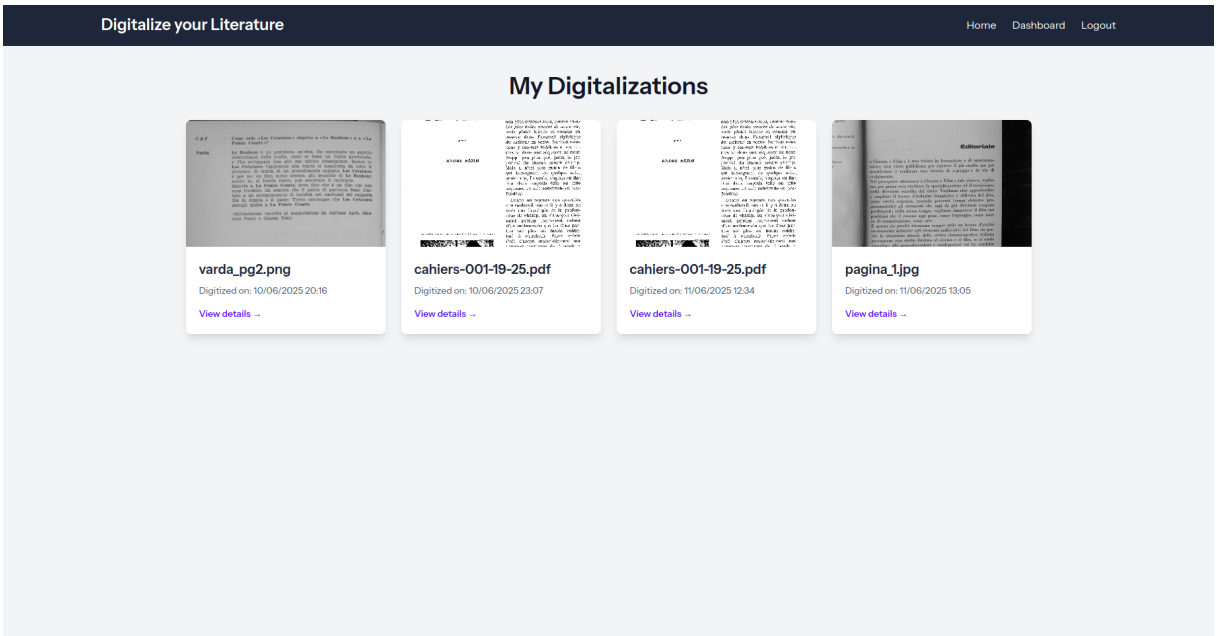


Figura 6 – Resultados das digitalizações

que posteriormente foi aprimorada utilizando da ajuda do próprio Gemini, como instruído por Greif, Griesshaber e Greif (2025) e que também pode ser visto em Apêndice A. O texto então pode ser baixado em formato PDF e cada página tem a opção de ser copiada para a área de transferência, como pode ser visto na Figura 5.

Caso o usuário não esteja logado as transcrições serão apagadas de tempo em tempo por um *job* em uma *schedule*. Já os usuários logados tem a possibilidade de observar suas transcrições em um *dashboard*.

Como o trabalho foca no teste de várias ferramentas OCR e mLLMs foi adotado um padrão de *Factories* com *Adapters*, para que na mudança do *provider* OCR não seja necessário

```php
class DigitalizesFactory
{
    João Paulo Abdala Bohaczk *
    public static function make()
    {
        $provider = config( key: 'ocr-service.provider');

        return match ($provider) {
            'gemini' => new GeminiAdapter(),
            default => throw new \Exception( message: "Provider '{$provider}' not supported")
        };
    }
}
```

Figura 7 – Factory do provedor OCR

demais alterações no código geral da aplicação, somente a reconstrução do *service* desse *provider*.

4.3 Testes iniciais com as ferramentas

Um teste foi realizado com a Figura 3a, apresentada na seção de Materiais. Foram avaliadas algumas ferramentas já para ambientação de como utilizá-las nos próximos experimentos. Para o primeiro teste foi utilizada a ferramenta Gemini e foram obtidos resultados satisfatórios quanto a transcrição. O tempo de processamento do Gemini foi de 9.6 segundos. Esse tempo é consideravelmente maior do que o de uma ferramenta OCR, como o Tesseract, que processou a mesma imagem em 1.4 segundos.

Um problema encontrado nos testes manuais da aplicação é a inconsistência de *outputs* do Gemini, mesmo que o prompt instrua que, por exemplo: não devem ser retornados caracteres como '\n' ou '\t', por vezes os mesmos acabam sendo retornados. Também houveram problemas com o envio de *jsons* com a formatação quebrada.

Foi feito uma breve comparação com as transcrições utilizando o Tesseract, Paddle, Docling e Gemini com a biblioteca Jiwer no Python. Essas transcrições foram feitas com a mesma imagem da figura 3a, sem nenhum tratamento na imagem.

A comparação foi feita com um arquivo *Ground Truth*, uma transcrição feita à mão da página. O texto resultante de cada ferramenta passou por uma normalização para não levar em conta aspectos de formatação, somente a sequência de palavras. O resultado de *WER* e *CER* de cada ferramenta pode ser visto na Tabela 2 .

É possível observar que a ferramenta que levou mais tempo de processamento foi a Docling, e consequentemente, ocorreram mais erros em caracteres e palavras. Já o mais eficiente, em tempo de processamento foi o Tesseract, contudo, os menores erros foram observados utilizando o Gemini. Deve-se levar em consideração que esses são resultados preliminares, as

Tabela 2 – Trabalhos relacionados

Ferramenta	**Tempo de processamento (ms)**	**WER (%)**	**CER(%)**
Gemini	9681	0.77	0.20
Tesseract	1490	69.60	40.65
Docling	33502	85.96	74.21
Paddle	14499	33.33	16.91

imagens não passaram por nenhum pré-processamento ou pós-correção, as ferramentas não foram optimizadas ou treinadas.

5 CONCLUSÃO

Diante da inacessibilidade de determinados livros (VARELLA; FERREIRA, 2012) e outros textos publicados antes da era digital, bem como do tratamento secundário da comunidade de OCR para com o uso dos documentos analisados em texto digital próprio para a leitura casual, determinamos a necessidade de uma ferramenta que, trabalhando com OCR, dedique-se na extração de textos de imagens ou arquivos PDF, visando facilitar a leitura e disseminação desses textos.

Para atingir esse objetivo será realizada uma pesquisa comparativa entre ferramentas OCR, sendo selecionada aquela que mais se destacar no acerto de palavras e no retorno da formatação original do texto. A ferramenta selecionada será treinada para atingir maior nível de precisão.

Será construída uma API em Python para realizar o trabalho de processamento do OCR e a formatação do texto. Isso será integrado por uma interface web desenvolvida com o framework PHP Laravel, com o objetivo de disponibilizar o projeto para a maior quantidade de pessoas possível.

Portanto, o presente trabalho possui tanto uma via científica e experimental como prática, buscando comparar as ferramentas OCR e solucionar algo que é até então ignorado pelas ferramentas, a devida formatação do texto para leitura, também tendo como objetivo disponibilizar os resultados dessa pesquisa como uma ferramenta de fácil uso e acesso para a comunidade.

REFERÊNCIAS

BIENIECKI, W.; GRABOWSKI, S.; ROZENBERG, W. Image preprocessing for improving ocr accuracy. *In*: **2007 International Conference on Perspective Technologies and Methods in MEMS Design**. [*S.l.*: *s.n.*], 2007. p. 75–80.

BOENIG, M. *et al.* Labelling ocr ground truth for usage in repositories. *In*: **Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage (DATeCH2019)**. New York, NY, USA: ACM, 2019. p. 3–8. Disponível em: https://doi.org/10.1145/3322905.3322916.

Docling Team. **Docling**. 2024. https://github.com/docling-project/docling. Acesso em: 17 abr. 2025. Veja também: https://arxiv.org/abs/2408.09869. Disponível em: https://github.com/docling-project/docling.

DROBAC, S. **DATeCH2019 - Session 4. Senka Drobac**. 2019. Apresentação do artigo "Improving OCR of Historical Newspapers and Journals Published in Finland". Disponível em: https://www.youtube.com/watch?v=3ZhHPx00wjg.

GABRIAL, B. History of writing technologies. *In*: BAZERMAN, C. (Ed.). **Handbook of Research on Writing: History, Society, School, Individual, Text**. [*S.l.*]: Taylor & Francis Group, 2009. p. 27–39.

GIROD, B. **Digital Image Processing: Morphological Image Processing**. 2018. Course material, Stanford University. Available from EE368/CS232 Digital Image Processing course, Stanford University. Disponível em: https://stanford.edu/class/ee368/.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing, Global Edition**. 4. ed. London, England: Pearson Education, 2017.

GREIF, G.; GRIESSHABER, N.; GREIF, R. **Multimodal LLMs for OCR, OCR Post-Correction, and Named Entity Recognition in Historical Documents**. arXiv, 2025. Disponível em: https://arxiv.org/abs/2504.00414.

KRISHNA, R. **Computer Vision: Foundations and Applications**. Stanford University, 2017. First printing, December 2017. Licensed under the Apache License, Version 2.0. Disponível em: http://www.apache.org/licenses/LICENSE-2.0.

KUMAR, B.; TANWAR, S.; TIWARI, S. Ocr using traditional and dnn approach. *In*: ____. **Mechatronics**. CRC Press, 2025. p. 196–210. ISBN 9781003494478. Disponível em: http://dx.doi.org/10.1201/9781003494478-11.

LI, C. *et al.* **Dive into OCR**. PaddleOCR Community, 2022. Disponível em: https://github.com/PaddleOCR-Community/Dive-into-OCR.

MINAEE, S. *et al.* **Large Language Models: A Survey**. arXiv, 2024. Disponível em: https://arxiv.org/abs/2402.06196.

NGUYEN, T. T. H. *et al.* Survey of post-ocr processing approaches. **ACM Computing Surveys**, Association for Computing Machinery (ACM), v. 54, n. 6, p. 1–37, jul. 2021. ISSN 1557-7341. Disponível em: http://dx.doi.org/10.1145/3453476.

OCR-D Project. **OCR-D: Integrated Workflow for OCR in Historical Documents**. 2020. https://ocr-d.de/en. Acesso em: 17 abr. 2025. Disponível em: https://ocr-d.de/en.

PaddlePaddle Community. **PaddleOCR**. 2020. https://github.com/PaddlePaddle/PaddleOCR. Acesso em: 17 abr. 2025. Disponível em: https://github.com/PaddlePaddle/PaddleOCR.

SCHMANDT-BESSERAT, D.; ERARD, M. Origins and forms of writing. *In*: BAZERMAN, C. (Ed.). **Handbook of Research on Writing: History, Society, School, Individual, Text**. [*S.l.*]: Taylor & Francis Group, 2009. p. 7–24.

SUBRAMANI, N. *et al.* **A Survey of Deep Learning Approaches for OCR and Document Understanding**. arXiv, 2020. Disponível em: https://arxiv.org/abs/2011.13534.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. Springer, 2010. August 18, 2010 draft. For non-commercial personal use only. Disponível em: http://szeliski.org/Book/.

TEAM, T. T. O. **Tesseract OCR**. 2024. https://github.com/tesseract-ocr/tesseract. Acesso em: maio de 2025.

VARELLA, G.; FERREIRA, I. M. **Livros inacessíveis, lei antiquada**. 2012. Acesso em: 28 abr. 2025. Disponível em: https://idec.org.br/em-acao/artigo/livros-inacessiveis-lei-antiquada.

WICK, C.; REUL, C.; PUPPE, F. Calamari - a high-performance tensorflow-based deep learning package for optical character recognition. **Digital Humanities Quarterly**, v. 14, n. 2, 2020. Disponível em: https://digitalhumanities.org/dhq/vol/14/2/000451/000451.html.

APÊNDICE A – Prompts Utilizados no Gemini

Neste apêndice, estão listados os prompts fornecidos ao modelo Gemini durante a fase de extração e organização de texto a partir de imagens digitalizadas.

Prompt 1

> Transcreva o conteúdo da imagem mantendo a formatação. A imagem é uma página de livro ou de uma revista. Foque somente na página principal, ignore o conteúdo das bordas. O conteúdo deve ser retornado em json classificando o título, subtítulo, o número da página, qualquer header com o título do livro/-capítulo. O conteúdo principal deve ser retornado por parágrafos, mantendo ao máximo a estrutura original do texto, mas as quebras de linha originais não precisam ser especificadas como quebras, o importante é saber o início e fim dos parágrafos, ""e "" ̂ não devem ser retornados. No caso de tabulações no formato de entrevistas separar o nome do falante como se fosse um novo parágrafo. Caso alguma dessas informações esteja faltando deve ser retornado null. Caso a imagem tenha mais de uma página o conteúdo deve retornar várias páginas. Exemplo de construção do json: "page": "headerTitle": null "title": null, "subtitle": null, "paragraphs": ["parágrafo1", "parágrafo2", "parágrafo3",] "pageNumber": null

Prompt 2

> Transcreva o conteúdo textual da imagem. A imagem representa uma página de livro ou revista. Concentre-se **exclusivamente no conteúdo principal da página**, ignorando elementos das margens, rodapés ou cabeçalhos que não sejam o título do livro/capítulo e imagens.
>
> O resultado deve ser um objeto JSON formatado rigorosamente conforme o exemplo fornecido. Para cada página detectada, o JSON deve conter um objeto "page"com as seguintes chaves:
>
> - **"headerTitle"**: (string ou null) O título do livro, capítulo ou seção que aparece no cabeçalho da página (se existir). - **"title"**: (string ou null) O título principal da página (se houver). - **"subtitle"**: (string ou null) O subtítulo da página (se houver). - **"pageNumber"**: (inteiro ou null) O número da página (se detectável). - **"paragraphs"**: (array de strings) Uma lista de parágrafos. Cada string no array deve representar um parágrafo completo.
>
> **Instruções detalhadas para "paragraphs":**
>
> 1. **Preservação da Estrutura**: Mantenha a estrutura original do texto o máximo possível, respeitando a separação lógica dos parágrafos. 2. **Remoção

de Caracteres Especiais**: **Remova estritamente** quaisquer caracteres de nova linha (“) e tabulação (“͡) do texto dos parágrafos. Eles não devem aparecer no JSON final. Esse tipo de será lidada na separação por parágrafos. 3. **Diálogos/Entrevistas**: Se houver diálogos ou entrevistas com nomes de falantes (ex: "Nome do Falante: Texto"), o nome do falante deve ser tratado como um novo parágrafo. 4. **Conteúdo Faltante**: Se alguma das informações solicitadas (título, subtítulo, etc.) não for encontrada na imagem, seu valor correspondente no JSON deve ser **null**.

Estrutura do JSON (obrigatório):

Se a imagem contiver múltiplas páginas, o JSON deve ser um array de objetos "page". Se contiver apenas uma página, será um único objeto "page".

Exemplo de JSON esperado:

“‘json "page": "headerTitle": "Introdução à Inteligência Artificial", "title": "A Evolução das Máquinas Pensantes", "subtitle": "Desde a Lógica Simbólica à Aprendizagem Profunda", "paragraphs": ["A inteligência artificial (IA) tem sido um campo de estudo fascinante e em constante evolução, buscando replicar e aprimorar capacidades cognitivas humanas em sistemas computacionais.", "Desde os primórdios da lógica simbólica, que tentava codificar o conhecimento de forma explícita, até as abordagens modernas de aprendizado de máquina, que permitem aos sistemas aprender com dados, a IA tem percorrido um longo caminho.", "O impacto da IA é vasto e crescente, influenciando setores como saúde, finanças e transporte, e prometendo transformar radicalmente a forma como interagimos com a tecnologia."], "pageNumber": 15 , "headerTitle": "Introdução à Inteligência Artificial", "title": null, "subtitle": "Desafios Atuais e Futuros", "paragraphs": ["Apesar dos avanços notáveis, a IA enfrenta desafios significativos, como a interpretabilidade de modelos complexos e a ética no uso de algoritmos preditivos.", "A garantia de que os sistemas de IA sejam justos, transparentes e responsáveis é crucial para a sua aceitação e integração na sociedade.", "Pesquisas futuras se concentram em áreas como a IA explicável (XAI) e o aprendizado federado, buscando superar essas barreiras e expandir as fronteiras do que é possível."], "pageNumber": 16