UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII - GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS VINICIUS ALVES MINERVINO

UM MODELO PARA EXTRAÇÃO AUTOMÁTICA DE INFORMAÇÕES EM FORMULÁRIOS DE REQUERIMENTO ACADÊMICO DA UEPB

PATOS - PB
2021

CARLOS VINICIUS ALVES MINERVINO

UM MODELO PARA EXTRAÇÃO AUTOMÁTICA DE INFORMAÇÕES EM FORMULÁRIOS DE REQUERIMENTO ACADÊMICO DA UEPB

Trabalho de Conclusão de Curso, apresentado como requisito parcial à obtenção do grau de bacharel em Ciência da Computação pela Universidade Estadual da Paraíba.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. RICARDO SANTOS DE OLIVEIRA.

PATOS - PB
2021

M664m Minervino, Carlos Vinicius Alves.

Um modelo para extração automática de informações em formulários de requerimento acadêmico da UEPB [manuscrito] / Carlos Vinicius Alves Minervino. - 2021.

61 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas , 2021.

"Orientação : Prof. Dr. Ricardo Santos de Oliveira , Coordenação do Curso de Computação - CCEA."

1. OCR. 2. Reconhecimento de documentos. 3. Reconhecimento de formulários. 4. C. 5. CER. I. Título

21. ed. CDD 658

Elaborada por Kléber T. de Sousa - CRB - 15/472 BSC5/UEPB

Carlos Vinícius Alves Minervino

UM MODELO PARA EXTRAÇÃO AUTOMÁTICA DE INFORMAÇÕES EM FORMULÁRIOS DE REQUERIMENTO ACADÊMICO DA UEPB

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 13/10/2021

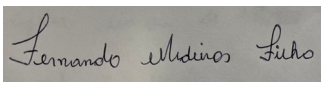
BANCA EXAMINADORA

Ricardo Santos de Oliveira

Prof. Ricardo Santos de Oliveira
(Orientador)

Jannayna Domingues Barros Filgueira

Profª. Jannayna Domingues Barros Filgueira
(Examinadora)



Prof. Fernando Medeiros Filho
(Examinador)

AGRADECIMENTOS

Primeiramente à Deus, que me criou e tem sido minha fiel fortaleza, e meu protetor, ajudador, companheiro e encorajador.

À minha mãe, e aos meus irmãos e irmã, pela paciência e apoio, pois possivelmente mais difícil do que fazer um TCC é morar com alguém que está fazendo um.

Ao meu pai, que não está mais aqui, mas me deixou um legado de sabedoria, paciência, humildade e pacificidade.

Aos meus professores da UEPB, dos quais não citarei nomes, para não correr o risco de esquecer algum, por toda a paciência no lecionar das disciplinas.

À professora e coordenadora adjunta Jannayna, por apoiar de perto, não só a mim mas a todos os estudantes de TCC, orientando com relação às dúvidas.

À Damião da secretaria, pelo imensurável apoio aos estudantes concluintes.

Ao meu orientador Ricardo, que teve não pouca paciência com todos os meus questionamentos, e recomendou dicas valiosas.

À toda a equipe docente, secretaria, coordenação, diretoria e aos demais servidores do campus VII pelo profissionalismo e apoio prestado durante minha convivência no campus.

E, finalmente, à Julyanne Rodrigues, por ter me animado todas as vezes que desanimei, e por acreditar em mim, e ser o meu braço direito, e meu apoio psicológico, e minha incessante fonte de alegria e inspiração.

RESUMO

Um problema muito recorrente em praticamente todos os tipos de empresas ou instituições públicas é a necessidade de tratamento e gestão de documentos físicos impressos, em fotos ou escaneados, a verificação, de forma manual por um agente humano, da consistência, completude e validade das informações preenchidas, e muitas das vezes, a leitura com posterior inserção no computador, por digitação manual, tarefa esta que poderia ser muito mais rápida se automatizada. Tendo em vista este contexto, o objetivo deste trabalho é propor um modelo automático de extração de informações preenchidas à mão nos formulários de requerimento acadêmico da UEPB, que possa fazer a extração automática das informações preenchidas e o devido armazenamento no computador, gerando um arquivo JSON representativo de um requerimento acadêmico. Tal modelo, foi proposto a partir da concepção de um *pipeline* especificamente voltado para esta tarefa, consistindo de: padronização das imagens dos formulários, aplicação de um algoritmo de alinhamento das imagens com o *template*, repartição dos campos dos formulários nas imagens, aplicação de OCR nos campos particionados, e associação do formulário com o aluno, de uma base de dados de alunos sintética, que possuir os dados mais similares ao resultado do OCR. O modelo proposto foi testado e executado numa base de trinta formulários reais dos estudantes, e para o problema de detecção do aluno pelo OCR no formulário de requerimento, pelo *pipeline* proposto, foi obtida uma acurácia de 86,67%. Ao final, foram sugeridas, para trabalhos futuros, possíveis medidas para aperfeiçoar a acurácia do modelo.

Palavras-chave: OCR. Reconhecimento de documentos. Reconhecimento de formulários. Alinhamento de imagens. CER.

ABSTRACT

A very common problem that occurs in practically all types of public and private organizations is the need for handling and management of physical documents, printed, and in image format, whether as a picture or scanned document, and also the verification of integrity, completeness, and validity of the information filled in the document, performed by some person manually, and still very frequently, the reading and subsequent manual retyping of the information in the document back in the computer, a step that could be much faster if automated. With respect to this issue, the objective of this work is to propose an automatic model for extracting the information filled, even by hand, in an academic request form from UEPB, that can automatically extract and store the information in the computer in a form of a JSON file, with the data that fully describe the academic request. Such a model was proposed to implementation through the conception of a pipeline specifically designed to this task, consisting of: standardization of the images of the forms, execution of an image alignment algorithm based on template matching, separation of the fields of the forms in the images, performing of OCR in the fields, and association of each form with the student, from a sintetic student database, that has the data that is most similar to the OCR output. The proposed model was tested and executed in a dataset of 30 real forms filled by UEPB students, and an accuracy of 86.7% was obtained for the problem of detecting the student that filled each academic request form, by using the proposed pipeline. Finally, some different methodologies were suggested, for future works, that could possibly improve the accuracy of the model.

Key-words: OCR. Document recognition. Form recognition. Image alignment. CER.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo introdutório será apresentado o cenário da problemática na qual se situa o objetivo deste trabalho, bem como a justificativa e os objetivos geral e específicos da proposta a ser apresentada, e por último uma descrição da estrutura a qual seguem os capítulos seguintes.

1.1 Cenário

A evolução dos algoritmos de Inteligência Artificial (IA) e do poder computacional nos últimos tempos tem contribuído para uma evolução tecnológica rápida e abrangente. Tais avanços têm possibilitado o desenvolvimento de modelos capazes de realizar tarefas que requerem inteligência, de forma automática, trazendo inúmeras aplicações interessantes. Hoje existem melhorias na forma de analisar dados, e usar as informações obtidas para melhores tomadas de decisão e para realizar predições, que constituem parte do objetivo da Ciência de Dados.

O surgimento e aprimoramento do *deep learning* tem possibilitado a concepção de modelos muito poderosos. Alguns exemplos de aplicações desta área são o reconhecimento facial, motorista autônomo, reconhecimento de voz, processamento de linguagem natural, reconhecimento de objetos em movimento em vídeos, reconhecimento óptico de caracteres dentro de imagens, além de uma vasta gama de outras aplicações.

Trazendo especificamente a aplicação de reconhecimento óptico de caracteres dentro de imagens, problema popularmente conhecido como *Optical Character Recognition* (OCR), existem diversas subaplicações dentro dessa área. Um exemplo dentro do problema de OCR é a aplicação que possibilita a uma pessoa cega saber quais palavras ou textos estão à sua frente, capturadas por uma câmera e “lidas” pelo modelo de Inteligência Artificial.

Outro exemplo de aplicação de OCR desejável e altamente útil nos dias atuais é a concepção de modelos capazes de extrair informações a partir de imagens de documentos escaneados, uma vez que o avanço da tecnologia, dos computadores portáteis e da internet até hoje não implicou a inutilização de documentos impressos, visto que muitas vezes documentos eletrônicos são impressos em papel, assim como

papéis impressos são digitalizados (escaneados) de volta em um formato eletrônico (O'GORMAN; KASTURI, 1997).

Porém, após o documento passar por tais transformações, tem-se uma imagem ou foto do mesmo, que é um formato legível e compreensível, a priori, apenas para humanos, não sendo interpretável para o computador que apenas lida com as posições e cores dos pixels da imagem, em vez de interpretar uma estrutura de um documento, seus campos, *layout*, significado, etc.

Para acrescentar ainda mais a dificuldade de tais arquivos serem interpretáveis por computador (*machine-readable*), muitas das vezes estes acumulam imperfeições chamadas *noises* (ruídos, em uma tradução literal), que podem ser pontos pretos em fundo branco, pontos brancos em fundo preto, manchas, inclinação errada da folha, dentre outros problemas, sendo imperfeições geradas durante os processos de impressão ou digitalização (O'GORMAN; KASTURI, 1997; FARAHMAND *et al*., 2013), fazendo aparecer essas anomalias indesejáveis, dificultando ainda mais a compreensão da estrutura fundamental do documento por um modelo de IA.

1.2 Justificativa

Parte das demoras referentes à burocracia dos processos das empresas se dá pelo tempo de espera de agentes humanos lerem informações contidas em documentos escaneados. Assim, uma ferramenta capaz de extrair informações de tais imagens é capaz de trazer melhorias ao fluxo de trabalho que envolve documentos (TENSMEYER, 2019) e acelerar o atendimento em processos burocráticos de empresas ou órgãos públicos . A pandemia que ocorre no mundo recentemente tornou ainda mais óbvia a necessidade por ferramentas e mecanismos de automatização de processos comunicativos e que deem apoio à comunicação virtual/eletrônica, que é claramente vista hoje como um importante processo a ser aprimorado e aperfeiçoado na sociedade. Nesse contexto, a aplicação de extração de informações a partir de imagens de documentos escaneados é um potente acelerador na direção desse objetivo.

Existem muitas tarefas cotidianas onde a extração de informação a partir de documentos é necessária. Seguem alguns exemplos: Um banco virtual, ou operadora de cartão de crédito geralmente necessita que um cliente, que esteja se cadastrando, anexe comprovante de documentos, os quais serão analisados por essas empresas,

que farão a extração de informações de forma visual/humana/manual. A análise e extração de informação de um comprovante de endereço é um bom exemplo de uma tarefa apropriada para automação. O reconhecimento de documentos de identificação escaneados e a extração de informações a partir dos mesmos é outro exemplo interessante.

Um exemplo palpável nesse sentido ocorre no caso da UEPB, onde, por exemplo, os requerimentos estudantis se dão pelo preenchimento manual de formulários, que são posteriormente escaneados, e enviados aos respectivos setores responsáveis pela tratativa. A extração das informações advindas dos formulários escaneados é feita manualmente (pelo agente humano), o que pode significar um maior tempo de duração do atendimento.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo deste trabalho é propor um modelo capaz de extrair a informação contida nos formulários escaneados de requerimentos acadêmicos da UEPB e que retorne um arquivo contendo os dados do requerimento específico, conforme foram preenchidos nos formulários, para que esses dados obtidos possam ser armazenados, e acessados por sistemas de *software* de forma automática, acelerando os processos comunicativos acadêmicos.

1.3.2 Objetivos específicos

Os objetivos específicos consistem em:

- Propor e especificar as etapas que compõem um *pipeline* concebido especialmente para a tarefa de extração automática de informações dos formulários de requerimentos escaneados;
- Codificar, em linguagem Python, uma versão parcial do *pipeline* proposto;
- Coletar um conjunto de dados composto de formulários de requerimentos reais preenchidos pelos estudantes da UEPB, em formato de imagem, em foto ou escaneado;

- Submeter os formulários do conjunto de dados ao *pipeline* proposto;
- Avaliar a *performance* deste *pipeline* no conjunto de formulários reais;
- Avaliar a *performance* deste *pipeline* em um conjunto de dados sintéticos, composto por formulários preenchidos a mão com dados fictícios, para adicionar mais ruídos e verificar se o pipeline proposto consegue lidar com estas imperfeições;
- Testar a concepção deste *pipeline* com diferentes metodologias, comparar os resultados e analisar possíveis propostas de melhoria;

1.4 Estrutura do trabalho

O trabalho está organizado em 5 capítulos. Neste capítulo foi discutido sobre o cenário, a justificativa e os objetivos do atual trabalho. O capítulo 2 traz uma revisão das subáreas de inteligência artificial que possuem relação com a proposta deste trabalho, conceituando OCR, Métricas de similaridade entre cadeias de caracteres, alinhamento de imagens, análise e reconhecimento de documentos, e reconhecimento de tabelas e formulários, citando alguns dos mais relevantes trabalhos publicados que dissertam sobre estes tópicos. O capítulo 3 apresenta a metodologia utilizada para concepção da solução proposta, especificando explicitamente quais etapas compõem o *pipeline* proposto para a tarefa de extração automática de informações contidas nos formulários de requerimentos em foto ou escaneados, e explicando como é feita a construção de tal modelo. O capítulo 4 apresenta os resultados obtidos nesta pesquisa e traz uma discussão sobre estes. Por fim, o capítulo 5 conclui o trabalho, enfatizando o resumo das conclusões obtidas e apresentando sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Os tópicos a seguir trarão uma exposição das principais áreas relacionadas com a pesquisa deste trabalho. Serão apresentados brevemente os temas Inteligência Artificial e Ciência de Dados. A seguir será discutido sobre *machine learning*, que é considerada a principal subárea destas duas grandes áreas, e portanto representa a intersecção de ambas. Posteriormente será discutido sobre uma subárea específica de *machine learning*, chamada *deep learning*.

As apresentações são extremamente superficiais, pois estes campos constituem um universo vasto de assuntos, que não caberia neste trabalho. Será introduzida também a noção de Visão Computacional, como uma das grandes aplicações de Inteligência Artificial, e uma subárea especial desta, a qual é uma base para o atual trabalho: Reconhecimento Óptico de Caracteres, em inglês, *Optical Character Recognition* (OCR). A seguir será definido o escopo do problema de reconhecimento de documentos, um subproblema de Visão Computacional altamente dependente de OCR. Por fim, o capítulo será concluído com uma discussão sobre trabalhos no campo de reconhecimento de documentos semelhantes à proposta deste trabalho, em particular, serão discutidos reconhecimento de tabelas e reconhecimento de formulários.

2.1 Áreas relacionadas

Serão brevemente apresentadas as áreas de interesse deste trabalho e suas relações entre si, em particular, Inteligência Artificial, Ciência de Dados, *machine learning*, *deep learning*, Visão Computacional e Reconhecimento Óptico de Caracteres.

2.1.1 Inteligência Artificial

Turing (1950) propôs a seguinte pergunta: “máquinas podem pensar?”. Em seu artigo submetido ao jornal britânico Mind, Turing propôs um teste, que leva o seu nome, que define uma metodologia para determinar se uma máquina conseguiu ou não adquirir comportamento inteligente. Poucos anos depois, em 1956, um novo

campo de estudo, chamado de Inteligência Artificial, foi proposto numa conferência na universidade Dartmouth, Estados Unidos.

Para McCarthy (1956), um dos escritores da proposta da conferência de Dartmouth, a Inteligência Artificial é o ramo da Ciência da Computação preocupado com fazer computadores “agirem” como humanos. Minsky (1968) define como a ciência de fazer máquinas fazerem coisas que requerem inteligência quando feitas por humanos.

A definição de artificial é simples, significa: criado pelo homem. Já a definição de inteligência é algo muito mais difícil (ROSA, 2011). Muitos livros-texto de Inteligência Artificial atuais consideram a teoria das múltiplas inteligências de Howard (1987), que enumera uma lista de habilidades humanas chamadas de inteligências: verbal/linguística; musical/rítmica; lógica/matemática; interpessoal/social; visual/espacial; intrapessoal/introspectiva e corporal/cinestésica.

2.1.2 Ciência de dados

Ciência de dados é uma disciplina multidisciplinar que envolve ciência da computação, estatística e matemática com foco em descobertas úteis e valiosas, a partir de estudos em bases de dados brutos, estruturados ou não, isto é, a busca pela descoberta de informação e conhecimento a partir do estudo dos dados (ETIF, 2020).

2.1.3 Machine learning

Michel (1997) define *machine learning* da seguinte forma:

> *“Dizemos que um programa de computador aprende da experiência E a respeito de alguma tarefa T, com medida de desempenho P, se seu desempenho na tarefa T, medido pela métrica P, melhora com a experiência E.”* (MITCHEL, 1997, p. 2, tradução nossa).

Aprendizagem de máquina, ou *machine learning*, é simultaneamente subárea da Inteligência Artificial e da Ciência de Dados, e pode ser vista como um conjunto de ferramentas estatísticas que visam possibilitar a um modelo computacional aprender a partir dos dados. Egbuna (2018) defende que *machine learning* consiste em ensinar

computadores como aprender, para que possam fazer predições, a partir do conhecimento adquirido com os dados que experienciou.

A divisão mais aceita de *machine learning*, divide esta área em 3 subáreas: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço, porém uma outra classificação considera ainda uma outra subárea chamada aprendizagem semi-supervisionada.

Na aprendizagem supervisionada, queremos prever o valor de uma variável alvo, com o nosso modelo preditivo, a partir de um conjunto de várias outras variáveis conhecidas, chamadas *features*. O modelo dispõe de um grande conjunto de "respostas certas" para realizar um treinamento do modelo, por isso o termo "supervisionado".

Na aprendizagem não supervisionada, deseja-se que o modelo encontre padrões nos dados, porém sem dispor de "respostas certas" para treiná-lo, assim ele deve encontrar padrões "por si só". A aprendizagem por reforço consiste em um modelo que age em um ambiente, e suas ações ocasionam um *feedback* do ambiente, que o entrega uma recompensa, caso a ação tenha o feito se aproximar do objetivo, ou uma penalidade, caso sua ação o tenha distanciado do seu objetivo, e o objetivo de modelo é maximizar as recompensas, ou seja, errar cada vez menos.

Em todos os casos, *machine learning* visa treinar modelos a partir de muitos dados, através de estatística. Há algoritmos de Inteligência Artificial que não usam esta metodologia, como os algoritmos de busca, assim não se classificam como algoritmos de *machine learning*.

2.1.4 Deep learning

Aprendizagem profunda, ou *deep learning*, é uma subárea de *machine learning* que consiste em modelos algorítmicos com uma grande complexidade de camadas, chamados redes neurais profundas, que visam aprender por um processo que foi projetado inspirado pela forma como os neurônios do nosso cérebro processam informações (BROWNLEE, 2019b). Consiste em modelos mais poderosos e autossuficientes, ou seja, mais capazes de aprender por si só. Enquanto que nos algoritmos de *machine learning* tradicional é indispensável limpar os dados e saber gerar e selecionar as melhores *features* manualmente para poder treinar um modelo com boa acurácia, *deep learning* visa desenvolver modelos capazes de lidar com os

ruídos nos dados e saber criar e escolher as melhores *features* automaticamente, ou seja, modelos capazes de aprenderem sozinhos (BENGIO, 2009).

2.1.5 Visão computacional

Brownlee (2019a) conceitua visão computacional como a aplicação de Inteligência Artificial preocupada com o desenvolvimento de técnicas e algoritmos que visam habilitar um modelo computacional a “ver” e “entender” o conteúdo de imagens e vídeos digitais. Esta é uma aplicação vasta, e pode ser aplicada em muitos cenários, como detecção de pessoas em câmeras de CFTV, reconhecimento facial, gerenciamento inteligente do trânsito em rodovias, reconhecimento de impressão digital e reconhecimento óptico de caracteres.

2.1.6 Reconhecimento Óptico de Caracteres (OCR)

Reconhecimento Óptico de Caracteres, em inglês, O*ptical Character Recognition* (OCR), é uma subárea de Visão computacional especificamente interessada em desenvolver modelos capazes de reconhecer textos dentro de imagens e vídeos. As primeiras tentativas de concepção de máquinas capazes de realizar leitura visual de caracteres fazem referência ao final do século XIX, no entanto a primeira máquina de OCR verdadeira foi concebida para a revista Reader’s Digest apenas em 1954, e convertia automaticamente relatórios de vendas datilografados em cartões perfurados, assim a empresa podia transmitir a informação datilografada para o computador (EIKVIL, 1993).

Os surgimento dos computadores eletrônicos na década de 1950 fortificou a busca pelo desenvolvimento de sistemas que realizassem a tarefa de OCR, e o primeiro sistema OCR comercial foi o IBM 1418, lançado na década de 1960, porém apenas capaz de “ler” uma fonte especial da IBM e sob diversas restrições (MORI *et al*., 1992).

Com a evolução do *deep learning*, hoje existem diversos sistemas de OCR disponíveis na internet. Desde aplicativos para *smartphones* até API ’s em nuvem que disponibilizam OCR como um serviço. Há algumas soluções gratuitas e também algumas pagas. Como modelo de OCR gratuito e *open-source* pode-se citar o Tesseract, que é treinado através de redes neurais e dispõe de uma interface de linha

de comando (CLI) que possibilita converter arquivos de imagens em texto, e é executado localmente no computador, ou seja, não necessita de conexão com a Internet.

Existem também API 's que oferecem OCR como um serviço em nuvem, como o Microsoft Computer Vision, Cloudmersive e Google Cloud Vision, que apesar de não serem *open-source*, disponibilizam de modelos mais bem treinados e melhores na tarefa de reconhecimento de escrita a mão (GINO, 2021).

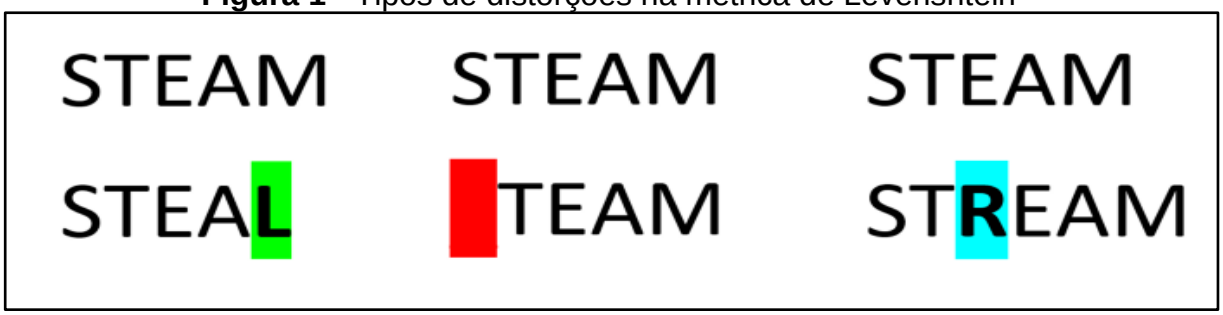
O Google Cloud Vision é disponibilizado como API dentro da cesta de serviços em nuvem Google Cloud Platform, e possui um limite de 1000 chamadas ao sistema de OCR gratuitas por mês (GOOGLE, [2021?]).

2.2 Métricas de similaridade entre cadeias de caracteres

Após a aplicação de um OCR, obtém-se uma transcrição em texto, conforme foi reconhecida dentro da imagem, mas que, pode ser um pouco diferente da transcrição correta esperada. A forma de avaliar a precisão com que o texto retornado pelo OCR se aproxima do texto correto que deveria ser reconhecido se dá através das métricas de similaridade entre duas cadeias de caracteres.

As principais métricas de similaridade baseiam-se em uma métrica chamada distância de Levenshtein, que procura quantificar a distância ou grau de diferença entre duas cadeias de caracteres, considerando três tipos de distorções: inserção, deleção e substituição de caractere (LEUNG, 2021). A Figura 1 ilustra esses tipos de distorções.

Figura 1 - Tipos de distorções na métrica de Levenshtein



Fonte: Kenneth Leung (2021).

A distância de Levenshtein entre duas cadeias de caracteres (*strings*) é definida como o número mínimo de distorções dos tipos inserção/deleção/substituição

necessárias para se transformar uma das *strings* na outra. Três importantes métricas de similaridade entre cadeias de caracteres são o *Character Error Rate* (CER), o *Word Error Rate* (WER), e a Similaridade de Levenshtein, todas estas três, utilizam o conceito da distância de Levenshtein.

A taxa de erro de caractere (CER) entre uma cadeia de caracteres de referência w_1 e uma outra cadeia w_2 interpretada como uma distorção da anterior é definida como (LEUNG, 2021):

$$CER(w_1, w_2) = \frac{I + D + S}{N}$$

Onde I, D, S representam, respectivamente, o número de inserções, o número de deleções e o número de substituições de caractere necessárias para se chegar da cadeia w_1 à cadeia w_2, e N representa o número de caracteres na cadeia de referência w_1. Sendo assim, a distorção entre duas *strings* pela métrica CER pode atingir valores de 0 a infinito, onde 0 representa que as *strings* são idênticas, e quanto maior for o valor do CER, mais diferente são as *strings*.

A taxa de erro de palavra (WER) entre uma cadeia de referência w_1 e uma outra cadeia w_2 interpretada como uma distorção da anterior é definida, de forma bastante similar ao CER, como (LEUNG, 2021):

$$WER(w_1, w_2) = \frac{I_W + D_W + S_W}{N_W}$$

Onde I_w, D_w, S_w representam respectivamente o número de inserções, deleções e substituições de palavras na cadeia, necessárias para se chegar da cadeia w_1 até a cadeia w_2, e N_w representa o número de palavras na cadeia de referência. Sendo assim, da mesma forma, a distorção entre duas *strings* pela métrica WER pode atingir valores de 0 a infinito, onde 0 representa que as *strings* são idênticas, e quanto maior for o valor do WER, mais diferente são as *strings*. A diferença entre o CER e o WER é que o CER conta por caracteres, e o WER conta por palavras.

Para exemplificar, podemos tomar a *string* de referência $w_1 = "ABC\ DE"$ e a variação $w_2 = "ABCDF"$. Para se chegar de w_1 a w_2 são necessárias uma deleção

(remover o caractere de espaço em branco) e uma substituição (substituir o E pelo F), e portanto, a distância de Levenshtein entre estas *strings* é 2, e como o tamanho da *string* de referência w_1 é 6, o CER entre elas é $2/6 = 0{,}3333 = 33{,}33\%$. No mesmo exemplo, pelo WER, é necessário deletar uma palavra inteira, e substituir a outra palavra inteira, para se chegar de w_1 a w_2, e como o número de palavras na string w_1 é 2, temos que o WER entre w_1 e w_2 é $2/2 = 1 = 100\%$.

Citando outro exemplo, se uma *string* é distorcida de forma a ter o dobro do comprimento original, o CER entre a distorcida e a original é de 1 (100%). O mesmo valor de CER ocorre entre duas *strings* com mesmo comprimento, porém sem caracteres em comum.

A Similaridade de Levenshtein entre duas *strings* w_1 e w_2 (JAUME; EKENEL; THIRAN, 2019) é por sua vez definida por

$$S(w_1, w_2) = 1 - \frac{L(w_1, w_2)}{max(|w_1|, |w_2|)}$$

Onde o L é a distância de Levenhstein entre w_1 e w_2, e a notação $|W_1|$ representa o tamanho da cadeia, isto é, o número de caracteres. Por exemplo, retomando o exemplo anterior com $w_1 = "ABC\ DE"$ e $w_2 = "ABCDF"$, tem-se que a similaridade de Levenshtein entre w_1 e w_2 é $1 - 2/6 = 66{,}67\%$.

Estas três métricas são de grande utilidade no processo de avaliação da qualidade do resultado de um processo de OCR.

2.3 Tópicos em Processamento de Imagens e Visão Computacional

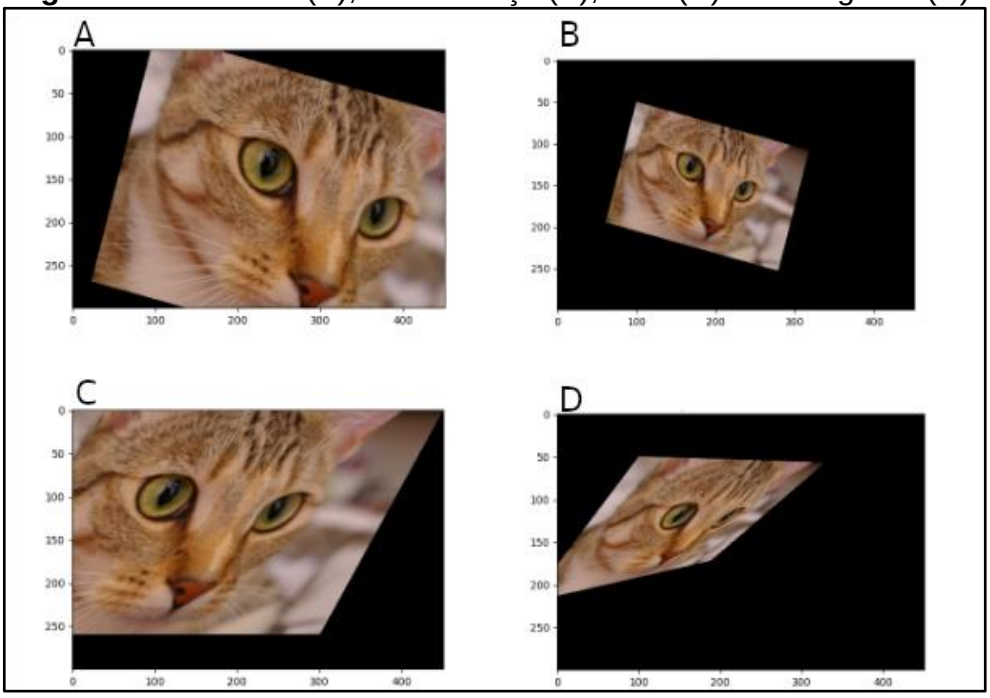
Processamento de imagens é um vasto campo de estudo, e uma de suas subáreas estuda as transformações bidimensionais que podem ser aplicadas em imagens, algumas das quais serão citadas a seguir. Os algoritmos de Processamento de Imagens constituem um alicerce necessário para Visão Computacional, e um dos tópicos que unem estas duas áreas é o problema de correspondência e alinhamento de imagens, que também é brevemente apresentado nesta seção.

2.3.1 Transformações geométricas bidimensionais em imagens

Existem diversas transformações 2D que podem ser aplicadas em uma imagem. Dentre as mais importantes, citam-se aqui quatro: isometria, semelhança, transformação afim e homografia. Dentre essas, a homografia é a que permite o maior grau de liberdade para distorções (DUBROFSKY, 2009).

A isometria, ou transformação euclidiana, preserva as distâncias entre cada par de pontos na figura, e suporta translação e rotação; a semelhança preserva as proporções e a forma das figuras, e suporta escala além das transformações euclidianas anteriores; transformações afins preservam linhas e paralelismo, e suportam cisalhamento além das transformações por semelhança; homografias, podem modelar todas as anteriores e ainda distorções de perspectiva, e é caracterizada por preservar linhas retas mas não necessariamente paralelismos (SCHÖNBERGER *et al*., 2019). Transformações afins são capazes de transformar um paralelogramo em um quadrado, enquanto que transformações projetivas (homografias) são capazes de transformar um quadrilátero qualquer em um quadrado, e portanto, é capaz de modelar a distorção de perspectiva sofrida por um objeto em uma foto (REDZUWAN *et al*., 2015). A Figura 2 abaixo ilustra resumidamente esta ideia.

Figura 2 - Isometria (A), semelhança (B), afim (C) e homografia (D).



Fonte: Site do scikit-image (com adaptações).

2.3.2 Alinhamento de imagens

Alinhamento de imagens, em inglês, *image alignment*, é um problema importante dentro de Visão Computacional, que consiste em alinhar duas imagens que contém o mesmo objeto, porém sob proporções, ângulos e perspectivas diferentes. A abordagem mais comum para o problema de alinhamento de imagens, consiste em quatro etapas: detecção de pontos-chave (*keypoints*) em ambas as imagens; obtenção de descritores de *features* binárias para cada um dos pontos-chave; associação dos pontos-chave (*keypoint matching*) das duas imagens; e obtenção da matriz de homografia capaz de gerar esta associação. (ROSEBROCK, 2020a).

A detecção de pontos-chave consiste na seleção de pontos de destaque dentro das imagens. A obtenção dos descritores de *features* binárias consiste em gerar, para cada ponto-chave, um vetor de *features* binárias que guarda informações interessantes sobre a vizinhança de cada ponto-chave. Por exemplo, um descritor de *features* binárias 128D para cada ponto-chave, consistirá em um vetor de 128 números binários, isto é, 0 ou 1. Para gerar este vetor a partir de um dado ponto-chave, analisa-se a sua vizinhança, e aplica-se alguns testes binários, isto é, testes do tipo verdadeiro ou falso. Nos testes onde o resultado for falso, ou seja, a vizinhança do ponto-chave não satisfazer uma determinada característica, a respectiva entrada no vetor de *features* é 0, e nos testes onde o resultado for verdadeiro, ou seja, a vizinhança do ponto-chave satisfazer alguma determinada característica, a respectiva entrada no vetor de *features* é 1. Assim o vetor de *features* servirá como um conjunto de informações sobre a vizinhança do ponto (TYAGI, 2019).

Alguns exemplos de algoritmos que executam a detecção de pontos-chave e a obtenção de descritores são SIFT, SURF e ORB (RUBLEE, 2011; KARAMI; PRASAD; SHEHATA, 2017). Os algoritmos SIFT e SURF são patenteados, já o ORB é livre e *open-source*, e possui implementação disponível na biblioteca OpenCV, que é uma extensa biblioteca *open-source* com algoritmos de Visão Computacional e processamento de imagens.

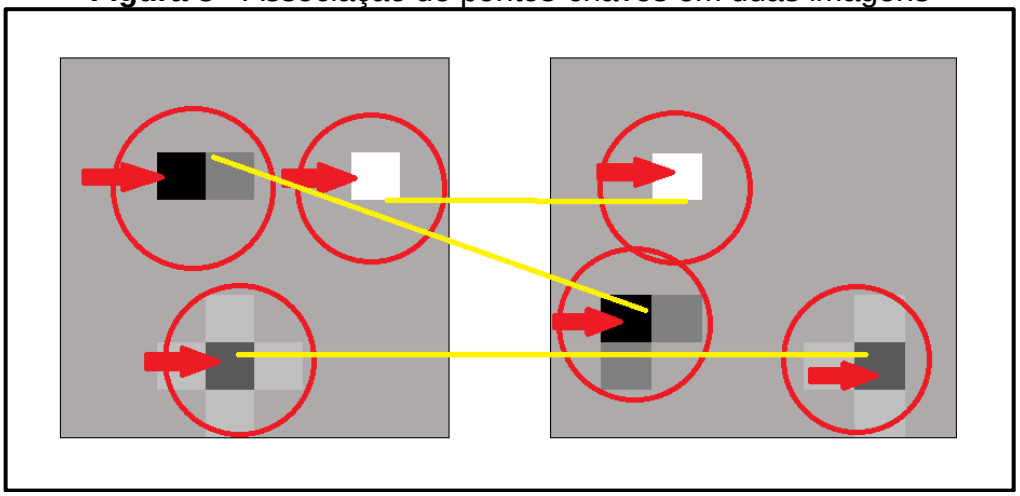
O nome algoritmo ORB vem de “Oriented FAST and Rotated BRIEF”, por ser uma metodologia que se baseia na combinação de dois outros algoritmos, sendo a etapa de detecção de pontos-chaves baseada no algoritmo FAST (Features from Accelerated Segment Test), e a etapa de geração de descritores de *features* binárias

para cada ponto-chave baseada no algoritmo BRIEF (Binary Robust Independent Elementary Features) (RUBLEE, 2011; KARAMI; PRASAD; SHEHATA, 2017).

A deteção de pontos-chave da imagem e a geração dos descritores de *features* destes pontos-chave é o alicerce para o alinhamento de imagens, e para o reconhecimento de um objeto em comum em duas imagens diferentes, em posições diferentes, pois a partir dos pontos-chave extraídos de ambas as imagens, e dos descritores de *features* destes pontos, pode-se comparar todas as possíveis associações entre os pontos-chave da primeira imagem com os pontos-chave da segunda imagem, e selecionar a associação ótima como a que minimiza a diferença entre os descritores de *features* correspondentes. Uma forma comum de quantificar a diferença entre dois vetores de *features* de igual dimensão é através da distância de Hamming (ROSEBROCK, 2020a), que conta o número de posições nas quais os vetores têm entradas diferentes.

A Figura 3 abaixo foi elaborada de forma meramente ilustrativa para ilustrar a ideia de detecção de pontos-chave (marcados com as setas) em duas imagens , delimitação da vizinhança destes pontos-chave (círculos), e obtenção da associação ideal entre os pontos-chave ideal das duas imagens como sendo a que minimiza a diferença entre as vizinhanças correspondentes destes pontos. Tal metodologia permite o reconhecimento de um mesmo objeto em duas imagens, e se necessário, o alinhamento de uma imagem com a outra, através da transformação de homografia, que leva os pontos-chave da primeira imagem nos seus respectivos correspondentes ótimos na segunda imagem.

Figura 3 - Associação de pontos-chaves em duas imagens



Fonte: Autoria própria.

Encontrando-se a correspondência ótima entre os pontos-chave das duas imagens, pode-se construir uma transformação por homografia que leva os pontos-chave da primeira imagem nos pontos-chave da segunda imagem, o que resolve o problema de alinhamento de imagens.

2.4 Trabalhos relacionados

Nesta seção são apresentadas algumas das pesquisas nas áreas de reconhecimento de documentos, reconhecimento de tabelas e reconhecimento de formulários.

2.4.1 Reconhecimento de documentos

O'Gorman e Kasturi (1997) definem o escopo do problema de extração de informações de imagens de documentos e a sua importância para as empresas e apresentam um compilado das técnicas desenvolvidas na época, mostrando as diferenças nos termos técnicos utilizados nos trabalhos de diferentes autores, e unificando a terminologia nesse campo de estudo. Porém as metodologias principais não destacam o uso de *deep learning*, mas de vários outros algoritmos, como *K-nearest neighbors*.

Os autores definem um *pipeline* de quatro etapas para a tarefa de extração de informações de imagens de documentos, que nomeiam de *document image analysis*: preparação (limpeza), extração de *features* apropriadas, reconhecimento de palavras e reconhecimento de componentes gráficos e *layouts*.

No início da década de 90, a criação da *International Conference On Document Analysis and Recognition* (ICDAR) reuniu cientistas e profissionais do mundo inteiro interessados na análise, reconhecimento e extração de informações a partir de documentos em formato de imagem. Desde a sua criação, as publicações submetidas às conferências ICDAR trouxeram grandes avanços no campo de Reconhecimento de documentos.

Na literatura científica pode-se encontrar pesquisas relacionadas a diversos problemas distintos dentro do contexto de reconhecimento de documentos. Um exemplo é o problema de classificação de documentos, onde o objetivo é categorizar

documentos por tipos, como carta, formulário, e-mail, artigo científico, etc. Como importante exemplo de pesquisa na área de classificação de documentos, podemos citar o trabalho de Harley *et al* (2015) que tornou público o maior *dataset* voltado para o problema de classificação de documentos do mundo, o RVL-CDIP *dataset*.

Existe também o problema de OCR em documentos escaneados ou fotografados, que pode envolver a tarefa de reconhecimento da estrutura de *layout* de um documento, e a hierarquia e semântica dos componentes presentes no documento, problema este que possui ainda duas subáreas mais específicas: reconhecimento de tabelas e reconhecimento de formulários, que são estruturas muito comuns em documentos, e serão abordadas a seguir.

2.4.2 Reconhecimento de tabelas

O processamento de tabelas embutidas em documentos digitais é algo tão antigo quanto o próprio campo de Reconhecimento de documentos (COUASNON; LEMAITRE, 2014 apud SCHREIBER *et al*., 2017). A maioria dos trabalhos voltados para o problema de Reconhecimento de tabelas separa dois objetivos: detectar uma tabela num documento, e reconhecer as linhas, colunas e células dessa tabela corretamente, problemas conhecidos na comunidade científica como *table detection* e *table structure recognition*.

Apesar da variedade de métodos disponíveis para detecção de tabelas em imagens e decomposição das mesmas em seus blocos estruturais, essas tarefas ainda provam ser difíceis, mesmo para sistemas de processamento de documentos modernos, por causa da imensa variedade de diferentes *layouts* e conteúdos nos documentos (SCHREIBER *et al*., 2017).

A maior parte dos trabalhos voltados para *table recognition* costuma dividir esta tarefa em duas subtarefas diferentes: *table detection* e *table structure recognition*, onde a primeira tarefa visa detectar onde ocorrem tabelas numa determinada imagem de documento, e uma vez detectadas, a segunda tarefa é invocada, que é a de reconhecer a estrutura dessas tabelas, isto é, a separação de linhas, colunas e células.

Shreiber *et al* (2017) propõem um modelo para detecção e reconhecimento de estrutura de tabelas dentro de imagens, chamado de DeepDeSRT baseado em *deep learning,* que utiliza os conceitos de *transfer learning* e *domain adaptation*, que são

conceitos avançados de *deep learning* que consistem em reaproveitar modelos já treinados, para alocá-los em outras tarefas semelhantes, através de um treinamento complementar para alinhá-los à nova perspectiva, metodologias úteis quando se tem uma pouca quantidade de dados para treinamento de um modelo. Paliwal (2019) e Prazad (2020) também trazem contribuições para o problema de *table recognition*, apresentando outras abordagens diferentes para a construção e treinamento.

2.4.3 Reconhecimento de formulários

Os termos Reconhecimento de formulários, *form recognition* e *form understanding* se referem a tarefa de extrair informações estruturadas automaticamente a partir de formulários escaneados ou contidos em uma foto . Há poucos esforços voltados diretamente para o reconhecimento de formulários, que são um tipo muito específico de documento, que em geral pode ser interpretado como um conjunto de perguntas e respostas. O formato em que as entidades semânticas ocorrem num formulário pode ser de diversas maneiras, como uma campo de texto, área de texto livre, múltipla escolha, sim ou não, escala de Likert, etc. Todas essas possibilidades contribuem para dificultar a tarefa de reconhecimento de formulários, no entanto existem algumas poucas estratégias publicadas, com interesse nessa tarefa (JAUME *et al*., 2019).

Jaume *et al* (2019) traz uma modelagem na qual, uma pergunta e sua respectiva resposta dentro de um formulário constituem um par de elementos semânticos interligados (relação chave-valor), e classifica os possíveis elementos semânticos dentro de um formulário em quatro categorias: cabeçalho, pergunta, resposta e outros, muito embora a estrutura e *layout* destes elementos possam vir de diversas formas diferentes. Em seu trabalho, os autores constroem e tornam público um *dataset* chamado FUNSD (*Form understanding in noisy scanned documents*), totalmente voltado para a tarefa de *form understanding*, e afirmam acreditar que este seja o primeiro *dataset* público voltado especificamente para esta tarefa. Para construir o *dataset* os autores tomaram um subconjunto de 199 imagens do *dataset* RVL-CDIP que havia sido anteriormente criado para endereçar o problema de *document classification*, e fizeram anotações manuais, construindo cada imagem, um correspondente arquivo JSON que contém a lista das entidades semânticas do formulário, e a informação de como elas estão interligadas.

Jaume *et al* (2019) propõem ainda um *pipeline* de três partes para a tarefa de *form understanding*: agrupamento de palavras, rotulamento de entidades e interligação de entidades. Os autores também implementam um modelo para a tarefa de *form understanding* utilizando o *pipeline* por eles definido e o *dataset* FUNSD, porém resultados obtidos não são tão promissores, mostrando a dificuldade de construção de um modelo preciso para reconhecimento de formulários em geral, principalmente com o *dataset* pequeno, e chamando a atenção dos pesquisadores para a necessidade de pesquisas futuras voltadas para *form understanding* que possam trazer metodologias que aperfeiçoem a acurácia do modelo.

Aggarwal *et al* (2020) utilizam uma abordagem *Seq2Seq*, que é uma técnica de *deep learning* muito útil na área de processamento de linguagem natural (*Natural Language Processing* - NLP), e adapta esta técnica para endereçar a tarefa de *form recognition* que chamam de Form2Seq, e divide duas tarefas: classificação de elementos de baixo nível (bloco de texto, caixa de seleção, etc), e agrupamento dos elementos de baixo nível formando construções de alto nível (campos de texto, grupos de seleção, etc).Os autores também tornam público um *dataset* de 400 imagens e seus respectivos JSON ’s que guardam a interpretação de sua estrutura, porém o *dataset* de Form2Seq contém imagens sem imperfeições (inclinação da folha, manchas, diferenças de claridade, pontos brancos ou pretos, etc), o que o torna provavelmente não tão adequado para conseguir reconhecer formulários escaneados reais utilizando um modelo treinado apenas com formulários sem imperfeições.

2.4.4 Reconhecimento de documentos com auxílio de templates

Jaume *et al* (2019) e Aggarwal (2020) trazem abordagens para treinamento de um extrator genérico de informações de formulários escaneados, isto é, um extrator projetado para analisar qualquer tipo de formulário. No entanto, os atuais extratores genéricos de informações de formulários disponíveis publicamente ainda possuem uma acurácia não tão satisfatória, devido à dificuldade da tarefa de *form recognition*, dadas as imperfeições geradas durante os processos de impressão e digitalização dos documentos, e também pela grande variedade de *layouts*, fontes, resolução da imagem e outros problemas. Por outro lado, as organizações requerem uma acurácia muito precisa na tarefa de extração de informações de documentos escaneados, o que enfraquece a motivação pelo treinamento de extratores genéricos.

Uma solução que pode ajustar melhor os objetivos de uma organização específica seria a construção de um extrator específico, isto é, um extrator que funcione bem com os modelos (*templates*) de documentos específicos daquela organização com grande acurácia, o que provavelmente seria melhor para esta do que o extrator genérico com acurácia pouco satisfatória.

Existe portanto a abordagem de concepção de um extrator de informações de formulários escaneados baseado em *templates*. Dessa forma uma tarefa nova surge: detectar qual *template* melhor encaixa a dada imagem dentro de um conjunto de *templates* pré-armazenados e conhecidos. Peng *et al* (2003) divide a tarefa de reconhecimento de documentos em duas subtarefas: encontrar o *template* correto da imagem, dentro um conjunto de *templates* pré-armazenados, e extração do conteúdo da imagem, uma vez conhecido seu *template*. Dessa forma os trabalhos de Jaume e Aggarwal são considerados estratégias independentes de *template*, enquanto que a abordagem de Peng utiliza *templates* como uma heurística.

É reconhecível que mesmo com a expansão do *deep learning*, e com os atuais algoritmos estado-da-arte, a tarefa de reconhecimento de documentos ainda prova ser difícil, principalmente quando espera-se um modelo de alta acurácia. Sendo assim estratégias apropriadamente voltadas para algum tipo específico de documento podem ser a melhor opção para solucionar um determinado problema específico.

Uma determinada organização geralmente possui apenas um pequeno conjunto de *templates* de documentos. Desta forma, no intuito de treinar um extrator de informações de documentos, é necessário apenas que este extrator faça a tarefa de extração com boa acurácia nos modelos de *templates* necessários àquela organização, e não necessariamente em modelos quaisquer de documentos escaneados.

Rosebrock (2020b) cita um método para realização desta tarefa, onde ele expõe uma metodologia prática para treinamento de um extrator de informações de documentos que utiliza como heurística o conhecimento do *template* do documento a ser analisado, e traz exemplos de trechos de códigos em Python e utilizando recursos da biblioteca OpenCV.

2.5 Lei Geral de Proteção de Dados Pessoais

A lei Nº 13.709/2018, conhecida como Lei Geral de Proteção de Dados (LGPD) é a atual legislação brasileira que regula o tratamento de dados pessoais, principalmente no aspecto da privacidade. Desta forma, pesquisas científicas que envolvam coleta e análise de dados pessoais, precisam estar de acordo com todas as especificações da LGPD. Este é um tópico que merece considerável atenção nas áreas da Ciência de Dados, e *machine learning*, por serem áreas que trabalham com bases de dados frequentemente contendo dados pessoais.

2.6 Considerações finais

Neste capítulo foram apresentadas as áreas que possuem relação com a proposta deste trabalho, e foi também introduzida a fundamentação teórica necessária para o perfeito entendimento do capítulo subsequente, no qual será detalhada a metodologia utilizada na concepção do modelo proposto neste trabalho. Em particular, foi discutido sobre métricas de similaridade entre cadeias de caracteres, e sobre algoritmos de alinhamento de imagens. Além disso, foi exposto um resumo sobre alguns dos trabalhos publicados com conteúdo semelhante à proposta aqui trazida.

3 METODOLOGIA

A metodologia proposta para o desenvolvimento do modelo de extração de informações contidas nos formulários de requerimento acadêmico da UEPB foi construída a partir da concepção de um *pipeline* projetado especialmente para esta tarefa, ou seja, um modelo formado por um conjunto de etapas sequenciais e interdependentes, onde os dados de saída de cada uma das etapas internas se tornam os dados de entrada da próxima etapa. Tal *pipeline* foi formado a partir de uma adaptação e extensão da metodologia proposta por Rosebrock (2020b), acrescentando uma etapa que ficou chamada de associação, que foi crucial para trazer uma boa acurácia. Todo código-fonte desenvolvido foi escrito em Python[1]. O *pipeline* concebido é constituído das seguintes etapas:

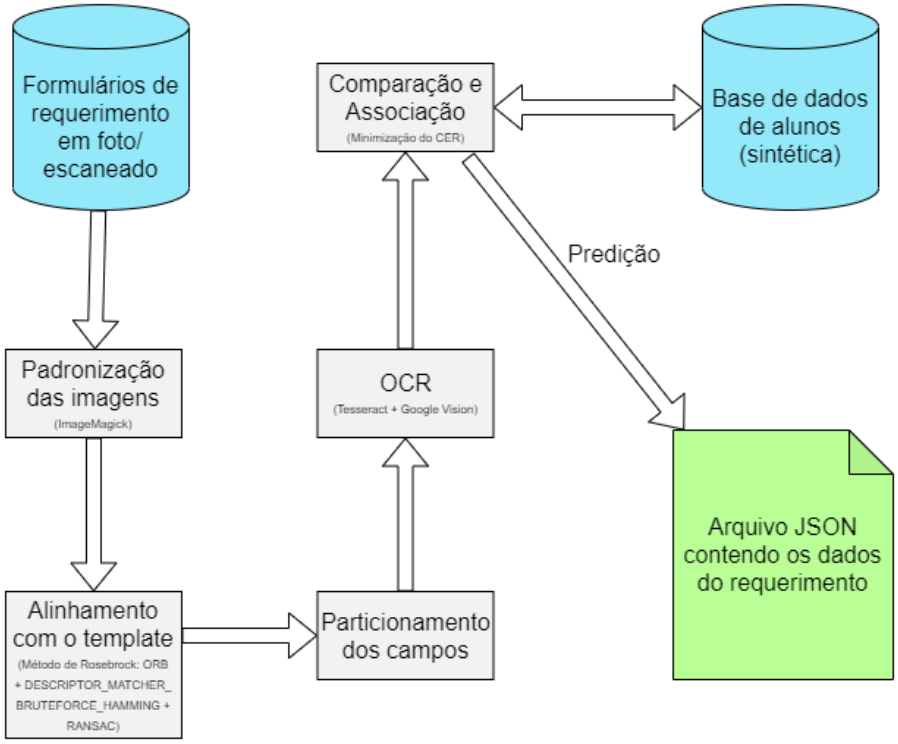
- Padronização de formato e resolução dos arquivos dos requerimentos em foto ou escaneados, utilizando comando *convert* do *software* ImageMagick;
- Alinhamento das imagens com o *template* de requerimento em branco pelo algoritmo de alinhamento proposto por Rosebrock (2020a), que combina três algoritmos: ORB, DESCRIPTOR_MATCHER_BRUTEFORCE_HAMMING e RANSAC.
- Separação dos campos do requerimento (nome, data, matrícula, telefone, e-mail, esclarecimento);
- Aplicação do OCR nos campos, com os modelos Tesseract e Google Vision;
- Comparação do resultado do OCR de cada campo com os valores existentes em uma base de dados de estudantes, e associar àquele que for mais similar, pela minimização da métrica CER;
- Geração de objetos JSON contendo os dados dos requerimentos extraídos com esta metodologia.

A etapa de associação consiste em comparar o resultado do OCR de cada campo com os valores que constam na base de dados dos estudantes, e associar o preenchimento do campo pelo estudante ao valor mais similar encontrado na base de dados. Após estas etapas, foram aplicadas métricas de avaliação de acurácia do

[1] O código-fonte encontra-se disponível em https://github.com/viniciuscva/extrator_automatico_python

modelo proposto. O fluxograma na Figura 4 descreve de forma mais clara o *pipeline* proposto.

Figura 4 - Fluxograma representativo do pipeline proposto



Fonte: Autoria própria.

3.1 Conjunto de dados utilizado

Foi utilizado para treinamento um conjunto de dados formado por 30 formulários de requerimentos acadêmicos da UEPB preenchidos, em sua maioria à mão, em foto ou escaneado, contendo dados reais de requerimentos dos estudantes. Os formulários escaneados foram obtidos com o setor da Secretaria Integrada de Cursos, com a autorização da coordenação do curso de Computação da UEPB campus VII, para o uso com fins exclusivamente voltados para a pesquisa, permanecendo protegidas todas as informações sensíveis do conjunto de dados, conforme as diretrizes da LGPD.

3.2 Construção de ground-truth

Em *machine learning*, costuma-se chamar dados *ground-truth* (verdade fundamental), a um conjunto de dados construído com registros reais, ou seja, fatos

já ocorridos, dados verdadeiros já registrados, que podem ser usados para treinar um modelo de predição ou para avaliar a acurácia de um modelo já treinado, comparando a predição do modelo com os valores corretos que constam no conjunto de dados *ground-truth*.

Para este trabalho também foi construído um conjunto de *ground-truth*, porém não para treinamento de modelo, mas apenas para a avaliação. Desta forma, os valores preditos pelo modelo puderam ser comparados aos dados corretos.

Foram feitas anotações manuais para cada formulário do conjunto de dados, de maneira que, para cada formulário foi criado manualmente um arquivo JSON com o esquema conforme a Figura 5, contendo os dados do formulário em questão, para que, a partir deste conjunto de dados com os valores corretos do preenchimento dos formulários (dados *ground-truth*), pudesse ser avaliada a acurácia do *pipeline* proposto, pela comparação do resultado predito pelo *pipeline* com o valor correto do conjunto de *ground-truth*.

As anotações manuais foram feitas para os campos nome, matrícula, data, telefone, e-mail, tipo de formulário e esclarecimento de cada requerimento do conjunto de dados, e salvas em arquivos JSON com o esquema representado na Figura 5, para servir como forma de avaliar a acurácia do *pipeline* aplicado. Como há 30 formulários no conjunto de dados, foram criados 30 arquivos JSON contendo as informações *ground-truth* de cada requerimento, e estes JSON ‘s foram nomeados de 1 a 30, obtendo assim, cada um, um número identificador único. O formato desses JSON’ s também é o formato desejável para o JSON final contendo as informações extraídas do formulário pelo modelo de automação proposto.

Figura 5 - Esquema de arquivo JSON ground-truth de um requerimento

```
{
   "id": "identificador único para o requerimento",
   "data": "data do requerimento",
   "nome":"Nome do estudante",
   "matrícula":"Matrícula do estudante",
   "curso":"curso do estudante",
   "telefone":"telefone do estudante",
   "email":"email do estudante",
   "código_opção":"código do tipo de requerimento",
   "esclarecimento":"texto do estudante esclarecendo o motivo do requerimento"
}
```

Fonte: Autoria própria.

Devido à impossibilidade, no âmbito deste trabalho, de se ter acesso à base de alunos da UEPB, foi criada uma base sintética, em arquivo JSON, contendo apenas informações dos alunos constantes no conjunto dos formulários, para que assim pudesse ser demonstrado o funcionamento da etapa de associação, que é uma etapa que necessita da base de alunos para execução.

Assim foi criada uma base de dados de alunos, em um único arquivo JSON, concebida apenas com os alunos que constavam no conjunto de dados, ou seja, com os alunos que fizeram requerimentos recentemente, no intuito de simular a base de todos os alunos. O arquivo *base_alunos.json* foi construído como um *array* de alunos, cada aluno com o formato descrito na Figura 6. Esta base não foi construída manualmente, mas através de um *script* Python que varreu todos os arquivos JSON com os dados *ground-truth* dos formulários, e localizou os diferentes alunos requerentes, criando com estes, a base de dados de alunos sintética.

Importante acrescentar que a abordagem aqui proposta torna-se uma possível ameaça a confiabilidade da acurácia aqui medida nos resultados, pois com uma base de alunos bem menor que a real, não é possível garantir que a acurácia por esta medida seja de fato igual a acurácia quando se usa toda a base real de alunos da universidade, base esta que, conforme explicado, não foi acessível no âmbito deste trabalho.

Figura 6 - Esquema da representação de aluno no arquivo base_alunos.json.

```
{
    "id": "identificador único do aluno na base",
    "matricula":"número de matrícula do aluno",
    "nome":"nome completo do aluno",
    "telefone":"telefone do aluno",
    "email":"email do aluno"
}
```

Fonte: Autoria própria.

Também foi criado um arquivo JSON, chamado *ground_truth_associacao.json* contendo a correta correspondência entre cada requerimento e seu respectivo aluno requerente, no formato como na Figura 7. Os números à esquerda são os números identificadores dos formulários, enquanto que os à direita, são os respectivos id ‘s dos alunos que os preencheram.

Figura 7 - Esquema do arquivo ground_truth_associacao.json.

```
{
    "1": "id do aluno que preencheu o requerimento 1",
    "2": "id do aluno que preencher o requerimento 2",
    "..."
}
```

Fonte: Autoria própria.

3.3 Ferramentas de apoio utilizadas

Foram utilizadas, as seguintes ferramentas de apoio:

- Python: Linguagem de programação de alto nível, multi plataforma, *open-source* e muito utilizada para Ciência de dados e *machine learning*;
- OpenCV: biblioteca com recursos de visão computacional, multiplataforma e *open-source*, escrita em C/C++ e com suporte a Python;
- Google Vision API: uma API do Google Cloud Platform para realizar OCR, com suporte a reconhecimento de escrita à mão;
- Tesseract OCR: um modelo de OCR que é gratuito e *open-source*;
- Jupyter Lab: Ambiente de desenvolvimento;
- Visual Studio Code: Editor de código;
- ImageMagick: Uma ferramenta de edição de imagens de código aberto que dispõe de uma interface de linha de comando.

A linguagem Python foi escolhida por ter uma sintaxe bastante amigável, ser *open-source* e contar com inúmeras bibliotecas úteis, gratuitas e publicamente disponíveis, e também por suportar todo o processo de desenvolvimento em Ciência de Dados e *machine learning*. A biblioteca OpenCV foi escolhida por ser uma das mais completas bibliotecas de Visão Computacional do mundo, se não a mais completa, além de ser gratuita, *open-source* e possuir extensa documentação disponível.

O ambiente de desenvolvimento Jupyter Lab foi adotado por facilitar a implementação de código Python, pois dispõe de uma interface que permite separar o código em trechos chamados de *notebooks*, e permite executar cada um separadamente, enquanto mantém o resultado das variáveis em memória. Porém o objetivo foi desenvolver no Jupyter Lab, e à medida que os códigos foram sendo

validados, os mesmos foram sendo escritos em *scripts* Python, utilizando-se o Visual Studio Code, para que assim pudessem se transformar em *software*.

Como a intenção deste trabalho foi mostrar o processo de construção de uma ferramenta automática que apoiasse o processo de processamento e gestão de documentos preenchidos manualmente, procurou-se incluir no *pipeline* apenas software que possuísse uma interface de linha de comando (CLI), pois se fosse incluída no *pipeline*, por exemplo, uma ferramenta de edição de imagem que funcionasse apenas com interface gráfica, ou um aplicativo OCR também permitindo uso apenas pela interface gráfica, ocasionaria maior dificuldade em criar um sistema automático sem necessidade de trabalho humano manual. Assim, a ferramenta de edição de imagens e o próprio modelo OCR foram escolhidos pela condição de possuírem interface de comando, em vez de interface gráfica, pois assim foi facilitado o processo de criação de um *pipeline* automatizado, controlado por *software*, e que dispensa intervenção humana, para que a partir da foto de um formulário preenchido, pudesse gerar um objeto JSON contendo os dados do requerimento do estudante, de forma totalmente automatizada.

Assim foram descartados todos os sistemas de OCR e ferramentas de processamento de imagens que possuíssem apenas interface gráfica, e focou-se nas que pudessem ser invocadas com CLI, principal motivo pelo qual foi escolhido o software ImageMagick para os pré-processamentos nas imagens, e o Google Cloud Vision API para o OCR.

A principal razão de ser escolhida a API de OCR Google Vision, sendo esta não *open-source*, em vez do Tesseract OCR que é *open-source*, foi o fato deste último não vir treinado para tarefa de reconhecimento de escrita a mão, e necessitar de treinamento para se especializar, inclusive treinamento de idioma, o que iria demandar necessidade de mais dados de treinamento, e de tempo, recursos esses bastante escassos durante a implementação deste trabalho. Sendo assim procurou-se um OCR já preparado para a tarefa de reconhecimento de textos escritos à mão.

3.4 Especificação das etapas do pipeline

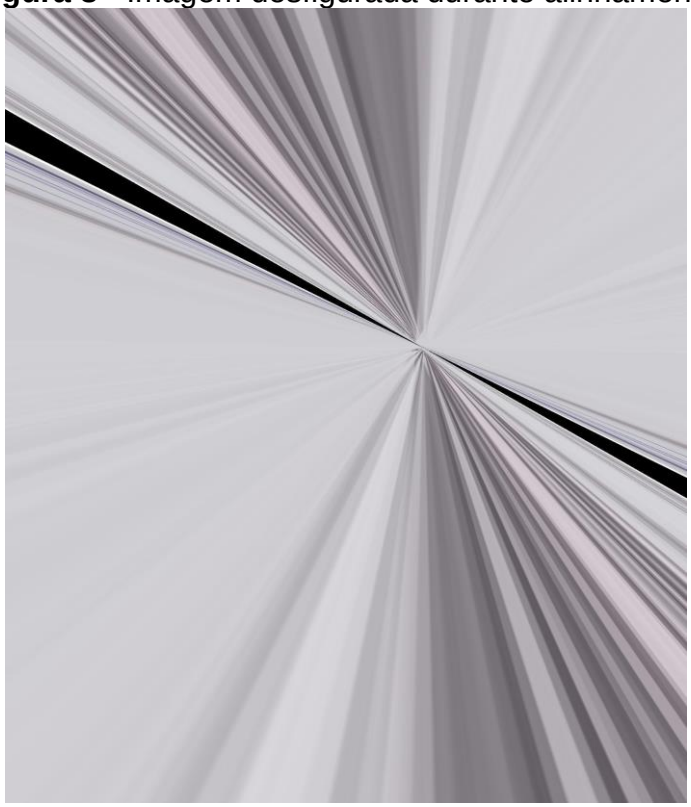
A seguir descreve-se em detalhes, os passos que compõem o *pipeline* concebido para a tarefa de extração automática de informações dos formulários de requerimento.

3.4.1 Conversão de tipos e padronização de resolução das imagens

O conjunto de dados contém formulários em diversos formatos como PDF/texto, PDF/imagem, JPEG, JPG, PNG, e em diversas proporções e resoluções diferentes, assim, foi realizado um pré-processamento utilizando o *software* ImageMagick para converter todos os arquivos para o padrão JPG e com a resolução limitada a 744 x 1052, sem mudar as proporções das imagens. A resolução padrão de 744 x 1052 para o *template* foi escolhida manualmente após alguns testes trabalhando o *template* em resoluções superiores a esta terem gerado dificuldades no algoritmo de alinhamento, gerando algumas vezes um *bug*, desfigurando a imagem do formulário, ao invés de alinhá-la, como ilustrado na Figura 8. Além disso, a grande maioria das imagens no conjunto de dados não ultrapassaram esta qualidade.

Foi então escrito um *script* Python para invocar o *software* ImageMagick sequencialmente, e converter, em *batch*, todas as imagens (formulários) contidas no diretório em que foram alocadas.

Figura 8 - Imagem desfigurada durante alinhamento



Fonte: Autoria própria.

3.4.2 Alinhamento com o template

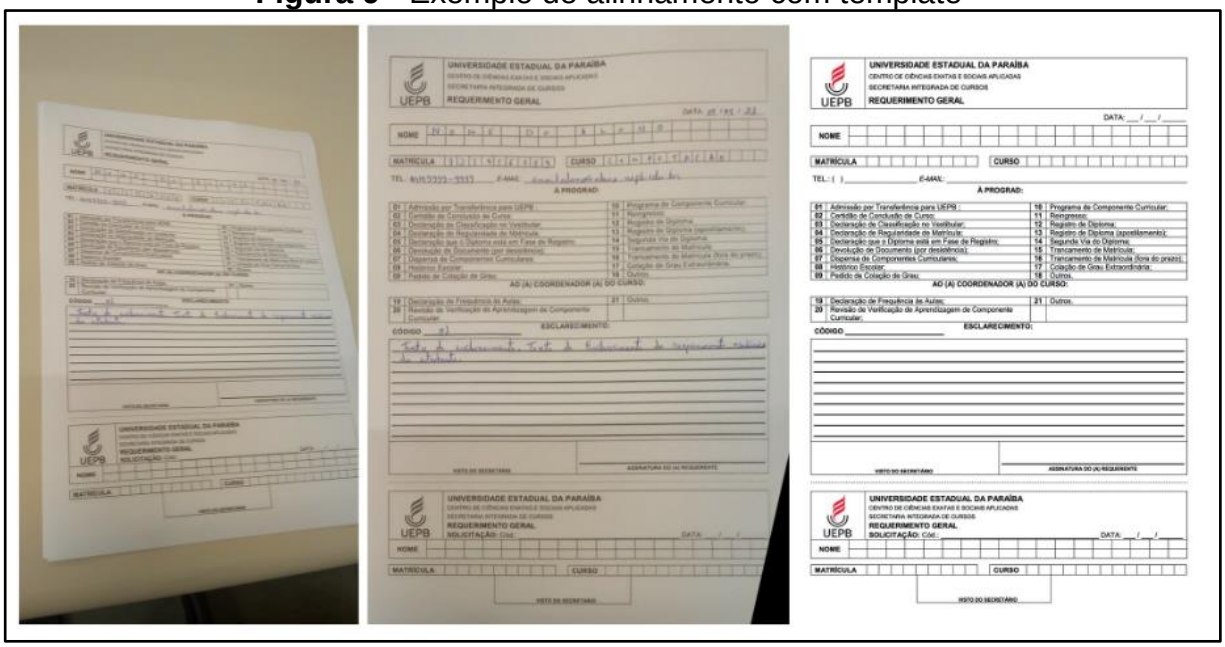
A etapa de alinhar o formulário escaneado com o *template* de formulário é um caso especial do problema de alinhamento de imagens, onde tem-se duas imagens contendo os mesmos objetos, porém sob diferentes pontos de vista. Resumidamente, a abordagem feita para o alinhamento de imagens foi baseada na implementação de um algoritmo de alinhamento baseado em *template* proposto por Rosebrock (2020a), escrito em Python, que é baseado no algoritmo ORB, citado no capítulo 2. Essa metodologia consiste em encontrar pontos-chave (regiões salientes) em ambas, gerar os descritores de *features* binárias invariantes locais destes pontos-chave, que são números que guardam informações sobre a vizinhança destes pontos-chave, encontrar a correspondência ideal entre os pontos-chave das duas imagens a partir da minimização de uma função de custo, e gerar a matriz de homografia que modela esta correspondência, isto é, a matriz necessária para se transformar uma imagem na outra.

A transformação adotada para o problema aqui tratado é a homografia, pois esta é capaz de modelar distorções de perspectiva (ROSEBROCK, 2020a), e será apropriada para este contexto, visto que muitos alunos tiram a foto do formulário, e dado o fato de que a foto contém o formulário com uma distorção de perspectiva, e não apenas uma distorção simples 2D euclidiana (DUBROFSKY, 2009), como aconteceria em um processo de *scanner*.

Após a seleção dos pontos-chave e da seleção da melhor correspondência entre eles, obtém-se a matriz de homografia pelo método cv2.findHomography do OpenCV, e a partir desta, obtém-se o alinhamento do formulário da imagem com o *template* a partir de uma perspectiva de distorção feita com a matriz de homografia obtida. Assim, cada um dos 30 formulários em JPG ficou alinhado com o *template* vazio de requerimento com a resolução 744 x 1052, e após o processo de alinhamento, cada um dos formulários teve a mesma resolução exata do *template*, e também ficou com cada um dos componentes da imagem nas mesmas posições coordenadas absolutas que o *template* (Figura 9), facilitando a próxima etapa, que consiste na separação dos campos dos formulários.

Foi então utilizado, com adaptações, o algoritmo proposto por Rosebrock (2020a) e escrito em Python, para realizar o alinhamento de imagem baseado em *template*.

Figura 9 - Exemplo de alinhamento com template

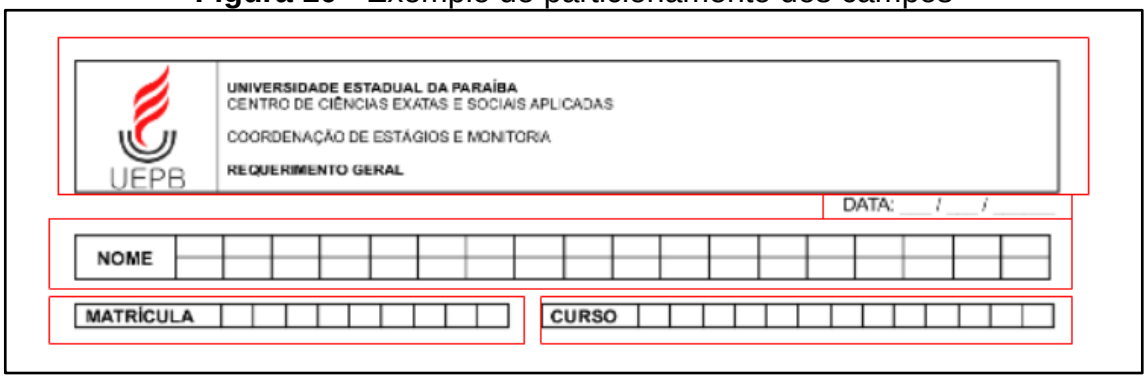


Fonte: Autoria própria.

3.4.3 Separação dos campos do formulário

Uma vez obtido o alinhamento do formulário escaneado ou fotografado com o *template*, a próxima etapa foi quebrá-lo em seus campos internos: Nome, Matrícula, Curso, Data, Telefone, E-mail, Código e Esclarecimento. A Figura 10 ilustra em que consiste este particionamento.

Figura 10 - Exemplo de particionamento dos campos



Fonte: Autoria própria.

O delineamento dos campos foi feito manualmente com anotações de coordenadas a partir do *template* original, verificando e anotando as coordenadas ao se passar o *mouse* sobre a imagem do *template*, aberta pelo método cv2.imshow da

biblioteca OpenCV, como ilustra a Figura 11. Estas coordenadas serviram como base para particionar todos os 30 formulários do conjunto de dados após o alinhamento.

Conforme ilustra a Figura 11, utilizando-se o *mouse*, foram anotadas as coordenadas para o caso da imagem plotada na resolução 372 x 526, e a partir desses valores, foi feita a divisão de cada coordenada anotada pelo tamanho do comprimento da imagem no sentido da respectiva coordenada, isto é, coordenadas x divididas pela largura da imagem, e coordenadas y divididas pela altura da imagem, a fim de se obter coordenadas relativas em um formato percentual. No exemplo, cada coordenada x anotada, foi dividida por 372 (o tamanho da largura da imagem), e cada coordenada y anotada foi dividida por 526 (o tamanho da altura da imagem), obtendo-se assim coordenadas relativas, que podem ser aplicadas em qualquer outra resolução, bastando multiplicá-las pela largura e altura da resolução desejada.

Figura 11 - Anotações manuais das coordenadas

Fonte: Autoria própria.

Na imagem observa-se que, posicionando a seta do mouse no ponto superior esquerdo da caixa delimitadora que queremos para o campo nome, encontramos no rodapé da janela no OpenCV que as coordenadas deste ponto são x = 15 e y = 84. Desta mesma forma foi feito com todos os campos, anotando-se assim as coordenadas delimitadoras dos campos nome, data, matrícula, telefone, email e esclarecimento.

A partir das coordenadas relativas (coordenadas absolutas dividido pelo tamanho da imagem na qual foram medidas), foi possível separar os campos de todos os formulários do conjunto de dados após o alinhamento com *template*. Os valores anotados das coordenadas das caixas delimitadoras dos campos do formulário, plotado na resolução 372 x 526, são exibidos na Figura 12.

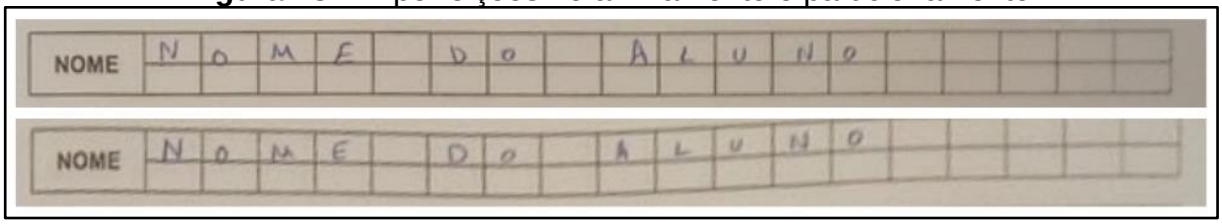
Figura 12 - Resultado das anotações manuais de coordenadas

```
#372 526
data = [265, 65, 360, 88]
nome = [15, 84, 360, 110]
matricula = [15, 110, 170, 126]
telefone = [15, 126, 115, 140]
email = [110, 125, 360, 140]
esclarecimento = [15, 260, 360, 367]
```

Fonte: Autoria própria.

No código ilustrado na Figura 12, os primeiros dois números de cada lista representam as coordenadas x e y do ponto superior esquerdo da caixa delimitadora do respectivo campo, e os últimos dois números representam as coordenadas x e y do ponto inferior direito da caixa delimitadora do mesmo. Para cada campo separado foi considerada uma caixa delimitadora que contivesse todo o espaço do campo e uma sobra de espaço a mais, para que, mesmo após alinhamentos imperfeitos, todo o espaço do campo pudesse estar contido dentro da respectiva caixa delimitadora. Esta imperfeição está ilustrada na Figura 13.

Figura 13 - Imperfeições no alinhamento e particionamento



Fonte: Autoria própria.

3.4.4 OCR

Com os campos dos formulários separados, foram aplicados cinco métodos de OCR diferentes, que são descritos a seguir. Porém, devido a limitações de tempo e recursos, foi feita a aplicação do OCR, como forma de exemplificação, apenas nos campos NOME e MATRÍCULA dos formulários já alinhados e repartidos em seus respectivos campos.

Outra limitação, foi o fato de que o Google Vision permite apenas 1000 chamadas à função de OCR gratuitas por mês, assim o *pipeline* construído para a concepção da solução foi projetado de forma a economizar consultas à API. Em direção desse objetivo, pensou-se em realizar, a princípio, o OCR, apenas no nome, na matrícula, reforçado também pelo motivo de que, basta uma dessas duas informações sobre um aluno, para que se possa realizar uma consulta numa base de dados de alunos, e encontrar os seus demais dados, como telefone e email.

Outro motivo para não se aplicar o OCR nos campos telefone e email é o fato de que estes dados são críticos e intoleráveis a erros, sendo assim, para criar um *pipeline* seguro de extração automática de informações, a provável melhor opção é detectar corretamente o aluno requerente apenas pelo OCR nos campos NOME e MATRÍCULA, e posterior obtenção dos dados restantes, email e telefone, a partir da localização do aluno na base de dados.

Os outros campos essenciais para serem reconhecidos no formulário são a data, o tipo de requerimento, e o texto de esclarecimento. No entanto, pelos fatores limitantes já mencionados, estes não foram incluídos no atual trabalho.

Os resultados das 5 metodologias foram comparados e os detalhes destes, bem como a análise dos mesmos, são apresentados no capítulo 4.

Método Tesseract

O Tesseract OCR, modelo já citado no capítulo 2, foi aplicado para o reconhecimento nos campos NOME e MATRÍCULA. No entanto, pelo fato deste modelo não vir treinado para a tarefa de reconhecimento de escrita à mão, e necessitar de treinamento, inclusive para o idioma alvo, não teve um bom desempenho em nossa aplicação, conforme será detalhado no capítulo 4.

Método TEXT_DETECTION do Google Vision

Ao notar-se o insatisfatório desempenho do Tesseract, foi procurado um modelo de OCR gratuito e mais bem preparado para a tarefa de reconhecimento de escrita a mão, e que possuísse interface de CLI (seja local ou por API), contando que pudesse ser codificada em *software*. Estas características desejáveis então, conduziram ao Google Vision API, e a primeira função do Google Vision OCR testada foi a função TEXT_DETECTION.

Método TEXT_DETECTION com ordenação de palavras

Observou-se que a API Google Vision retorna um objeto contendo uma estrutura organizada, que guarda uma lista de todas as palavras reconhecidas pelo OCR, e para cada uma delas, sua respectiva caixa delimitadora, descrita através de coordenadas x, y da posição na qual o texto foi localizado (Figura 14).

Figura 14 - Estrutura do objeto retornado pelo OCR

```
>>> dt
[locale: "und"
description: "M.\nA.\nNOME\n"
bounding_poly {
  vertices {
    x: 39
    y: 20
  }
  vertices {
    x: 648
    y: 20
  }
  vertices {
    x: 648
    y: 54
  }
  vertices {
    x: 39
    y: 54
  }
}
, description: "M."
bounding_poly {
  vertices {
    x: 261
    y: 22
```

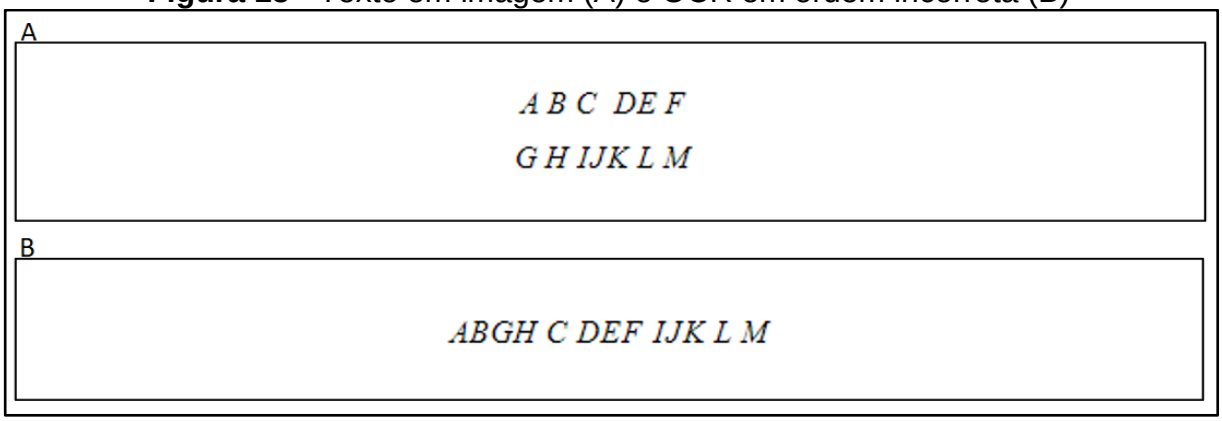
Fonte: Autoria própria.

Porém, foi observado que se for feita uma mera concatenação das palavras retornadas pelo OCR na ordem em que vêm, obtêm-se um texto numa ordem diferente da desejada, pois espera-se que o OCR leia da esquerda até a direita, até o final da

linha, e após, passe para a linha inferior, e faça o mesmo, e assim sucessivamente. Porém, as palavras retornadas muitas das vezes não ficam nessa ordem, pois seguem uma estrutura de blocos, onde a API muita das vezes detecta um bloco à esquerda do outro, quando na verdade não há tal estrutura na imagem.

Por exemplo, um texto como na Figura 15-A poderia ser acidentalmente transcrito, se seguindo a ordem das palavras retornadas pelo OCR, como algo semelhante ao exemplo na Figura 15-B.

Figura 15 - Texto em imagem (A) e OCR em ordem incorreta (B)



Fonte: Autoria própria.

Assim, observou-se que isto prejudicou muito a corretude dos resultados, e também a acurácia, em especial nos campos NOME, pois no formulário, o campo NOME possui duas linhas para preenchimento, o que pode ter facilitado ao OCR detectar blocos de forma incorreta, mudando a ordem dos sobrenomes da pessoa, e assim, enfraquecendo a acurácia do modelo.

Assim, no intuito de construir uma transcrição do OCR fiel à ordem das palavras, da esquerda para a direita, implementou-se um algoritmo que a partir das palavras retornadas pelo OCR, e das coordenadas das caixas delimitadoras dessas, que também são providas pela API, retorna uma *string* única contínua, contendo toda as palavras na ordem certa. De forma resumida, o algoritmo toma todas as palavras, e ordena as mesmas por coordenadas y de suas caixas, e em seguida, ordena-as em cada linha y pelas suas coordenadas x, da menor para a maior.

Contudo, as letras de uma palavra numa mesma linha, nem sempre possuem a mesma altura y, podendo variar poucos *pixels*, pela imperfeição da escrita à mão do estudante. Sendo assim, observou-se que a ordenação das linhas puramente pela

coordenada y, cria na verdade, várias linhas que inexistem, gerando praticamente uma linha para cada palavra ou letra. Desta forma, foi ajustado um parâmetro de 6 *pixels*, para que duas palavras, com coordenadas y próximas, pudessem ser consideradas escritas na mesma linha. O algoritmo na íntegra, é apresentado no Apêndice A.

Método DOCUMENT_DETECTION do Google Vision

Após uma melhor exploração da documentação do Google Vision, encontrou-se uma segunda possível função dentro da API chamada DOCUMENT_TEXT_DETECTION, que também realiza OCR, porém, sendo mais especializada para o reconhecimento de textos dentro de imagens contendo documentos, sendo assim, a função TEXT_DETECTION é mais genérica, e pode ser aplicada para reconhecimento de texto dentro de imagens quaisquer, como uma foto de uma cena natural, por exemplo, enquanto que a função DOCUMENT_TEXT_DETECTION é mais especializada no reconhecimento de imagens contendo documentos escritos, e também mais bem preparada para o reconhecimento de escrita à mão, conforme explica a própria documentação da API.

Método DOCUMENT_DETECTION com ordenação de palavras

Pelo mesmo motivo explicado na seção 3.4.4, a função DOCUMENT_TEXT_DETECTION também retorna as palavras numa ordem indesejada. Assim também foi utilizado a metodologia de DOCUMENT_TEXT_DETECTION + ordenação de palavras, e comparados os resultados com todas as metodologias anteriores conforme mostrado no capítulo 4.

3.4.5 Associação

Após a extração, via OCR com cada uma das metodologias citadas na seção 3.4.4, o *script* Python construído salvou os resultados de OCR em arquivos TXT para posterior análise, e para dar continuidade às próximas etapas do *pipeline* estabelecido.

Foram implementados códigos para comparar o resultado do OCR com os dados *ground-truth*, e foi importado o arquivo JSON da base de alunos, e calculada, para cada formulário, a distorção, pela métrica CER, entre a matrícula retornada pelo OCR, e a respectiva matrícula correta, e o mesmo feito para os campos NOME.

Após o cálculo das estatísticas com a métrica CER, que avalia o grau de diferença entre duas *strings*, neste caso, a matrícula e o nome obtidos com o OCR e a matrícula e o nome corretos, foi feita a próxima etapa do *pipeline* proposto, chamada Associação. Esta etapa de associação foi feita de maneira separada para os campos MATRÍCULA e NOME, a partir da minimização da métrica CER.

Desta maneira, foi feito como a seguir: Para cada formulário i ($1 \leq i \leq 30$), sendo $m_{ocr}(i)$ a matrícula retornada pelo OCR no formulário i, e sendo $m_{gt}(j)$ a matrícula correta (*ground-truth*) do aluno com id j na base de alunos B, cada formulário i foi associado ao id de aluno $\hat{\jmath}$ de maneira que:

$$\hat{\jmath} = arg\,min_{j \in B} \left(CER(m_{gt}(j), m_{ocr}(i)) \right) \quad ,$$

formando a associação de formulário-aluno $(i, \hat{\jmath})$, ou seja, cada formulário i foi associado ao id de aluno cuja distorção CER entre sua matrícula e a matrícula retornada pelo OCR no formulário i fosse a mínima dentre todos os alunos da base, em outras palavras, ao aluno com matrícula mais similar ao resultado do OCR, mediante a minimização da métrica CER. Da mesma forma foi feita a associação pela minimização do CER para os campos NOME.

Cada uma dessas associações foram comparadas com a associação correta, que foi escrita no arquivo *ground_truth_associacao.json* que guarda as associações corretas dos pares (nº do formulário, ID do aluno que preencheu), arquivo este construído conforme descrito na seção 3.2. A partir da comparação entre a associação feita pela minimização do CER e a associação *ground-truth* foi então medida a acurácia do *pipeline* concebido. Os resultados constam no capítulo 4.

3.4.6 Geração do JSON

Após a etapa de associação ter revelado que, a associação pela minimização da métrica CER quando feita com base nos campos matrícula pode ser diferente da

associação quando feita com base nos campos nome, foi observado que uma possível melhor abordagem para se encontrar a associação correta do par (formulário, aluno) seria combinar as diferentes associações ideais obtidas pelo OCR nos diferentes campos.

A estratégia de minimização do CER realizada em cada campo do formulário: nome, matrícula, telefone e email, gera uma associação otimizada (Formulário, Aluno). Porém fazendo esta minimização em cada campo do formulário, obtém-se uma associação ótima diferente. Sendo assim, se for considerado não apenas o aluno com o dado mais similar possível ao OCR, mas os três alunos com dados mais similares ao resultado do OCR com relação ao OCR em cada campo, poderia ser feita uma associação mais segura, de forma a evitar predições incorretas. Poderia-se então, a partir das 3 propostas de associação ótima geradas pelo OCR em cada campo, realizar uma contagem e obter-se a associação ótima para todos os campos como aquela formada pelos dados mais frequente dentre as associações ótimas para cada campo do formulário.

Por exemplo, se a partir da minimização do CER nos campos matrícula, fosse constatado que os três mais prováveis alunos a terem preenchido o formulário 1 fossem, digamos, os alunos 4,1,7, já a partir da minimização do CER nos campos nome, fosse encontrado que os 3 alunos mais similares ao OCR fossem os alunos 1,8,9, de maneira que também para a análise dos campos telefone e email pudesse ainda ser encontrado outro resultado. Assim poderia considerar como a predição o id de aluno que mais aparecesse em comum nas listas dos 3 alunos mais similares ao OCR pela matrícula, pelo nome, pelo telefone e pelo email.

Tal metodologia não foi implementada neste trabalho, e a título de exemplificação foi implementada uma forma de gerar um JSON contendo os dados do requerimento, porém, é claro, apenas contendo o nome, matrícula, telefone e email, pois são as informações alcançáveis na base de dados através da detecção do aluno, pela associação por meio do NOME mais similar ao OCR, enquanto que as informações data do formulário, tipo de requerimento e esclarecimento não constam na base de alunos e precisariam ser extraídas por OCR, campos estes que não foram submetidos ao OCR neste trabalho. Portanto, o aluno predito foi simplesmente selecionado como o aluno mais similar ao OCR com relação ao campo NOME, e o JSON gerado com os dados do requerimento, contemplou nesta pesquisa apenas a matrícula, nome, telefone e email de cada requerimento.

3.5 Considerações finais

Neste capítulo foi apresentada a metodologia utilizada durante a concepção do modelo proposto por este trabalho. Foi explicado o motivo da criação da base de alunos sintética, pela necessidade desta para uso na etapa de associação, e tendo em vista a impossibilidade do acesso à base real completa dos estudantes da instituição no âmbito deste trabalho. Foi ainda sugerida uma técnica de combinação das diferentes associações ideais obtidas a partir da minimização do CER na comparação do OCR em cada um dos campos do formulário com os dados *ground-truth*. A combinação das melhores associações candidatas para o OCR nos diferentes campos, poderia resultar na escolha de uma associação ótima que melhor se ajusta ao OCR de todos os diferentes campos, e poderia aumentar a acurácia do modelo. Porém esta proposta ficou como sugestão para trabalhos futuros.

O capítulo a seguir traz, em detalhes, os resultados obtidos na execução dos passos da metodologia aqui apresentada, e acrescenta uma análise explicativa sobre os mesmos.

4 RESULTADOS E DISCUSSÕES

Este capítulo traz os resultados obtidos na execução da metodologia proposta no capítulo 3, bem como, uma análise sobre os mesmos.

4.1 Resultado e discussão da padronização e do alinhamento

A etapa de padronização das imagens em formato JPG limitadas a 744 x 1052 executada em *batch* pelo *script* Python através do ImageMagick foi bem sucedida, e partir desta padronização foi possível prosseguir para a etapa de alinhamento das imagens com o *template*, que também foi bem-sucedida, e ao final da execução em *batch*, obtiveram-se 30 imagens JPG de resolução 744 x 1052, cada uma já alinhada com o *template* original vazio de requerimento acadêmico. Assim, a execução do algoritmo de alinhamento pôde corrigir as rotações e diferenças de perspectivas contidas nas imagens dos formulários dos alunos devido a orientação e posicionamento do documento escaneado ou em foto.

Foram também preenchidos alguns formulários para teste, com diversas distorções propositais, como rotação de 90º, foto em paisagem e mudanças de perspectiva. Em todos estes casos o algoritmo utilizado foi capaz de fazer o alinhamento com o *template* de forma bem-sucedida.

4.2 Resultado e discussão do particionamento

Na execução da etapa de repartição dos campos dos formulários após alinhamento, observou-se que a transformação adotada, homografia, não é capaz de ajustar o problema de papel curvado, o que ocorre frequentemente após o documento ser dobrado, e em seguida, desdobrado. Surgiram assim, imperfeições mínimas na posição na qual ficaram os campos extraídos após o alinhamento, devido a estas curvaturas no papel, no entanto, uma diferença desprezível para os objetivos deste trabalho.

Foi justamente devido a essa diferença mínima no alinhamento, que foram determinadas caixas delimitadoras levemente maiores do que o campo em si, no intuito de capturar os campos por completo, mesmo em documentos onde após o alinhamento o campo ficou levemente acima, levemente a baixo, ou levemente

curvado de cima a baixo ou vice-versa. A Figura 13, no capítulo 3, ilustra a ideia desta diferença.

4.3 Resultado e discussão do OCR

A etapa de OCR foi executada apenas nos campos NOME e MATRÍCULA, utilizando-se as cinco metodologias diferentes descritas na seção 3.4.4: Tesseract, Google Vision TEXT_DETECTION, TEXT_DETECTION com ordenação de palavras, Google Vision DOCUMENT_DETECTION, e por fim, DOCUMENT_DETECTION com a ordenação de palavras. Os resultados destes processos de OCR foram avaliados pela métrica CER, introduzida no capítulo 2, que vai de 0 a infinito, e avalia o grau de distorção entre uma *string* e outra, no caso, entre o resultado do OCR e o valor correto *ground-truth*.

Foi aplicado o OCR nos 30 campos MATRÍCULA, e para cada método de OCR, os 30 resultados de OCR foram comparados às 30 matrículas corretas, constantes na base de dados *ground-truth*. Assim foi calculado o CER entre a matrícula extraída pelo OCR e a respectiva matrícula correta para cada um dos 30 campos, obtendo-se assim 30 valores de CER para cada método de OCR, que, também para cada um destes métodos, foram ordenados do menor para o maior, e deles aferido o mínimo, o máximo, a média, a mediana e também os primeiro e terceiro quartis, para servir com forma de avaliar cada método. O mesmo foi feito com relação ao OCR nos campos NOME, e os resultados são exibidos nas Tabelas 1 e 2.

Pela Tabela 1, observa-se que o pior CER em cada método de OCR aplicado no campo MATRÍCULA foi igual a infinito. Isso ocorreu por que em alguns formulários do conjunto de dados o campo MATRÍCULA não foi preenchido pelo aluno, assim o cálculo do CER teve em seu denominador, o tamanho de uma *string* vazia, isto é, zero, e nessa divisão por 0, foi atingido o valor infinito. Uma forma fácil de interpretar isso é a seguinte: a distorção necessária para se transformar uma *string* vazia em uma outra *string* não vazia qualquer é infinita, pois o número de caracteres a serem adicionados, que é um inteiro positivo, é infinitamente maior do que zero, o tamanho da *string* original. Por este mesmo motivo, as médias também ficaram iguais a infinito.

Tabela 1 - Avaliação do CER (%) no OCR das matrículas

Método de OCR	Melhor CER	CER 1º quartil	CER médio	CER mediano	CER 3º quartil	Pior CER
Tesseract	77,8	100,0	inf	138,9	286,1	inf
TEXT_DETECTION	0,0	14,9	inf	38,9	136,1	inf
TEXT_DETECTION com ordenação de palavras	0,0	22,2	inf	44,4	144,4	inf
DOCUMENT_DETECTION	0,0	22,2	inf	55,6	186,1	inf
DOCUMENT_DETECTION com ordenação de palavras	0,0	33,3	inf	55,6	186,1	inf

Fonte: Autoria própria.

Tabela 2 - Avaliação do CER (%) no OCR dos nomes

Método de OCR	Melhor CER	CER 1º quartil	CER médio	CER mediano	CER 3º quartil	Pior CER
Tesseract	80,0	87,1	94,7	98,6	100,0	126,1
TEXT_DETECTION	31,4	66,0	71,8	77,1	82,6	96,4
TEXT_DETECTION com ordenação de palavras	31,4	47,2	66,7	67,7	80,7	96,4
DOCUMENT_DETECTION	38,1	61,1	68,3	69,1	79,6	109,4
DOCUMENT_DETECTION com ordenação de palavras	28,6	38,9	59,6	55,8	79,0	109,4

Fonte: Autoria própria.

Ainda na Tabela 1, observa-se que, com relação ao OCR no campo matrícula, o método Tesseract retornou *strings* com 138,9% de distorção, em mediana, em relação ao *ground-truth*; o método TEXT_DETECTION, retornou com 38,9% de distorção, em mediana, em relação ao *ground-truth*; o método TEXT_DETECTION com ordenação de palavras, 44,4%; o método DOCUMENT_DETECTION, 55,6%, e, por fim, o método DOCUMENT_DETECTION com ordenação de palavras, com os mesmos 55,6% do método antecedente. Percebe-se que a adição da etapa de ordenação das palavras retornadas pelo OCR não trouxe melhoria no CER para o

OCR no campo MATRÍCULA. Isto é explicado pelo fato de o campo MATRÍCULA, no modelo de requerimento, possuir apenas uma linha, e assim facilitar ao OCR detectar o texto na ordem desejada, diferente do que ocorreu no OCR dos campos NOME, conforme foi explicado na seção 3.4.4.

Para o OCR no campo NOME, comparando-se os resultados de OCR com os nomes corretos para cada requerimento, conforme exposto na Tabela 2, obteve-se que o valor mediano de CER entre um nome de OCR e seu respectivo nome correto foi de 98,6% no método Tesseract, 77,1% no método TEXT_DETECTION, 67,7% no método TEXT_DETECTION com ordenação de palavras, 69,1% no método DOCUMENT_DETECTION, e 55,8% no método DOCUMENT_DETECTION com ordenação de palavras.

Observa-se assim que a metodologia de ordenação de palavras proposta na seção 3.4.4 fez a diferença e trouxe melhores valores de CER no caso de OCR nos campos NOME, pelo fato destes campos possuírem duas linhas no *template* de requerimento, e com isso permitir ao OCR retornar um texto numa ordem diferente da desejada.

4.4 Resultado e discussão da Associação

Foi aplicada a etapa de associação conforme descrita na seção 3.4.5, para o OCR nos campos NOME e MATRÍCULA para cada um dos métodos de OCR conforme explicado na seção 3.4.4. Sendo assim, cada valor de matrícula retornado pelo OCR, para cada um dos cinco métodos de OCR, foi comparado com todas as matrículas da base de alunos, e associada àquela que possuísse menor distorção CER, obtendo-se assim, cinco configurações diferentes de associação, cada uma obtida com um método de OCR diferente, onde uma dada configuração de associação é uma lista de trinta pares (nº do formulário, id do aluno requerente).

Sendo assim, o teste de comparação do OCR com a base de alunos pela minimização da métrica CER, retornou, para cada método de OCR, uma configuração ideal de associação, e portanto, cinco configurações de associação pelo OCR nos campos MATRÍCULA, e outras cinco configurações de associação pelo OCR nos campos NOME. Cada uma destas dez configurações de associação foram comparadas com a associação correta salva no arquivo *ground_truth_associacao.json*, e aferido o número de correspondências em comum,

dividido pelo número total de correspondências, obtendo-se assim a acurácia da associação.

Somente para efeito de comparação entre o processo de OCR sem associação, e com associação, foi aferida a acurácia bruta para os campos NOME e MATRÍCULA, para cada método de OCR, obtida simplesmente pela divisão do número de ocorrências em que o método de OCR retornou um texto exatamente igual ao texto correto correspondente, pelo número total de elementos no conjunto, que é 30. Os resultados são apresentados nas Tabelas 3 e 4.

Tabela 3 - Avaliação da Associação no OCR das matrículas

Método de OCR	Acurácia bruta (%)	Acurácia com Associação (%)
Tesseract	0,0	6,7
TEXT_DETECTION	36,7	73,3
TEXT_DETECTION com ordenação de palavras	33,3	73,3
DOCUMENT_DETECTION	36,7	63,3
DOCUMENT_DETECTION com ordenação de palavras	33,3	63,3

Fonte: Autoria própria.

Tabela 4 - Avaliação da Associação no OCR dos nomes

Método de OCR	Acurácia bruta (%)	Acurácia com Associação (%)
Tesseract	0,0	3,3
TEXT_DETECTION	0,0	53,3
TEXT_DETECTION com ordenação de palavras	0,0	73,3
DOCUMENT_DETECTION	0,0	73,3
DOCUMENT_DETECTION com ordenação de palavras	0,0	86,7

Fonte: Autoria própria.

Observa-se que em nenhum dos 30 campos MATRÍCULA o Tesseract retornou a matrícula exata correta, enquanto que o método de OCR TEXT_DETECTION retornou a matrícula exata em 36,7% dos casos, o método TEXT_DETECTION com ordenação de palavras em 33,3% dos casos, o método DOCUMENT_DETECTION em 36,7%, e o método DOCUMENT_DETECTION com ordenação de palavras, em 33,3%.

Após a associação, o Tesseract obteve apenas 6,7% de acurácia, ou seja, mesmo com a comparação, pela minimização da distorção CER, do resultado do OCR com a base de alunos, ainda assim em meros 6,7% dos casos a associação foi feita corretamente, e para o OCR com Tesseract nos campos NOME, um resultado ainda pior de 3,3% de acurácia na associação.

Isto provavelmente se deve ao fato de que o Tesseract por padrão não vem treinando para reconhecimento de escrita à mão, mas apenas para textos digitados, e para que possa ser capaz de realizar OCR em escritas à mão, precisa de treinamento para estes casos, e também treinamento de idioma.

Observa-se na primeira coluna da tabela 4 que nenhum dos cinco métodos de OCR retornou ao menos um nome perfeitamente igual ao nome correto do aluno, porém, comparando-se o resultado do OCR com a base de alunos, obtiveram-se associações com boa acurácia, sendo 53,3% pelo método TEXT_DETECTION, 73,3% pelo método TEXT_DETECTION com ordenação de palavras, 73,3% pelo método DOCUMENT_DETECTION, e, o melhor resultado, 86,7% pelo método DOCUMENT_DETECTION com ordenação de palavras.

Observou-se assim, que a metodologia de associação (formulário, aluno) pela comparação do OCR com os dados da base de alunos e minimização da distorção CER trouxe melhores resultados pelo OCR nos campos NOME, em comparação ao OCR nos campos MATRÍCULA.

4.5 Considerações finais

O desempenho do OCR pelo método Tesseract foi insatisfatório, pelo fato deste modelo não vir treinado para reconhecimento de escrita à mão, e necessitar de treinamento adicional, de escrita à mão e de idioma alvo. Para o OCR nos campos MATRÍCULA, a adição do algoritmo de ordenação de palavras nos métodos TEXT_DETECTION e DOCUMENT_DETECTION não trouxe melhorias em relação

ao desempenho destes métodos sem a ordenação, devido ao fato de que o campo MATRÍCULA no *template* de requerimento possuir apenas uma linha de preenchimento, o que de certa forma já induz o OCR a extrair o texto na ordem desejada (da esquerda para a direita).

Para o OCR nos campos NOME, a adição do algoritmo de ordenação de palavras retornadas pelo OCR, a partir da verificação de suas caixas delimitadoras, trouxe uma melhoria de 37,5% na acurácia da associação pelo método TEXT_DETECTION, e uma melhoria de 18,3% na acurácia da associação pelo método DOCUMENT_DETECTION. Esta melhoria se deve ao fato de que os campos NOME no formulário, possuírem duas linhas de preenchimento, e com isso o OCR frequentemente retorna em uma ordem indesejada as palavras extraídas, por causa de uma incorreta detecção de blocos por parte do OCR, conforme explicado na seção 3.4.4.

Portanto a metodologia vencedora foi a aplicação da função DOCUMENT_DETECTION do Google Vision nos campos NOME com adição do algoritmo de ordenação de palavras retornadas pelo OCR, e posterior associação ao aluno da base de alunos com nome mais similar ao resultado do OCR com a ordenação. A associação ideal (Formulário, Aluno), obtida pela minimização da métrica CER na comparação dos resultados do OCR por este método com os dados da base sintética de alunos, resultou em uma acurácia de 86,7% na tarefa de detecção do aluno.

5 CONCLUSÃO

A proposta deste trabalho foi propor um modelo capaz de extrair as informações preenchidas nos formulários escaneados de requerimentos acadêmicos da UEPB, capaz de retornar um arquivo contendo os dados do requerimento específico. O projeto e execução das etapas do *pipeline* proposto, juntamente com os resultados obtidos, onde o método que mostrou melhor desempenho foi a aplicação da função DOCUMENT_DETECTION do Google Vision nos campos NOME com adição do algoritmo de ordenação de palavras, evidenciaram que é possível realizar a construção e implantação de um tal sistema automático.

Com exceção da construção de *ground-truth*, o que serviu apenas para avaliar a acurácia do *pipeline* proposto, todas as etapas deste projeto foram executadas algoritmicamente no computador, através de *scripts* Python. Sendo assim, conclui-se que todas estas etapas podem ser reunidas em um único *script* que as realize sequencialmente e de forma automatizada.

No intuito de monitorar de perto todas as saídas de cada uma das etapas (padronização das imagens, alinhamento, repartição dos campos, OCR, associação, criação do JSON), foram escritos códigos em separado para cada uma destas etapas, para que assim, cada uma delas pudesse ser avaliada separadamente. Porém, pela forma com que foram projetadas, todas estas etapas estão aptas a serem reunidas em um único *script* automático, gerando um JSON com os dados do requerimento submetido, e armazenando este JSON em algum dispositivo de armazenamento, o que permitirá seu uso por parte de outros sistemas de *software* que englobam o panorama de processamento de requerimentos estudantis da UEPB.

Os resultados aqui publicados ainda estão longe de ser uma solução completa para o problema tratado. Ainda há muito trabalho a ser feito futuramente, e também muitas outras metodologias a serem testadas e exploradas. No entanto, o autor espera ter conseguido demonstrar que a implantação de um tal sistema, conforme foi descrito ao longo do trabalho, é possível e viável, e que as etapas aqui projetadas, podem ser aperfeiçoadas separadamente.

Sugere-se para trabalhos futuros, a implementação de uma outra metodologia, possivelmente mais segura (contra erros), de efetuar a associação (detectar o aluno requerente), que se baseie da combinação das diferentes associações ideais obtidas a partir da minimização do CER na comparação dos campos matrícula, nome, telefone

e e-mail do OCR com os dados da base de alunos, conforme foi sugerido na seção 3.4.6. Espera-se que a combinação das associações ideais com base nos vários campos do formulário possa ser mais efetiva do que a escolha de uma associação ideal obtida pela comparação do OCR com a base de alunos que seja baseada em apenas um único campo do formulário.

Modelos de OCR capazes de lidar com reconhecimento de escrita à mão, como a função DOCUMENT_TEXT_DETECTION do Google Vision, proveem de uma técnica mais especializada de OCR chamada Reconhecimento Inteligente de Caracteres, em inglês, *Intelligent Character Recognition* (ICR), capaz de reconhecer escrita à mão. Pelo fato do Google Vision não ser *open-source* e ter sido utilizado como caixa-preta neste trabalho, não foi possível avaliar, nem ajustar as técnicas de *deep learning* do processo de ICR do mesmo, o que possivelmente poderia ser feito com o Tesseract, e que fica como sugestão para trabalhos futuros.

Sugere-se também a aplicação do *pipeline* com uma padronização de imagens em outras resoluções diferentes, e também realizar etapas de limpeza nas imagens, como suavização, aumento de contraste e binarização, para avaliar possíveis melhorias no resultado do OCR.

REFERÊNCIAS

AGGARWAL, M. *et al*. Form2Seq: A Framework for Higher-Order Form Structure Extraction. *In*: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), 2020. **Proceedings** [...]. Association for Computational Linguistics, 2020. p. 3830-3840.

BENGIO, Y. Learning deep architectures for AI. *In*: BENGIO, Y. **Foundations and Trends in Machine Learning**. 1. ed. Hanover, USA: Now Publishers Inc, 2009. v. 2. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=cq5ewg7FniMC&oi=fnd&pg=PA1&ots=Kpg5OYnlIB&sig=SkPsTv7cv7k_cPGgxhzYwj7GS84#v=onepage&q&f=false>. Acesso em: 18 mai. 2021.

BROWNLEE, J. A Gentle Introduction to Computer Vision. *In*: BROWNLEE, J. **Blog Machine Learning Mastery**. 5 jul. 2019. Disponível em: <https://machinelearningmastery.com/what-is-computer-vision/>. Acesso em: 18 mai. 2021.

BROWNLEE, J. What is Deep Learning?. *In*: BROWNLEE, J. **Blog Machine Learning Mastery**. 16 ago. 2019. Disponível em: <https://machinelearningmastery.com/what-is-deep-learning/>. Acesso em: 18 mai. 2021.

DUBROFSKY, E. Homography estimation. **Diplomová práce. Vancouver: Univerzita Britské Kolumbie**, v. 5, 2009.

EGBUNA, O. P. Artificial Intelligence, Machine learning, deep learning and data science - What's the difference?. *In*: WILLIAMS, E. *et al*. **Blog Medium**. 22 out. 2018. Disponível em: <https://medium.com/fbdevclagos/artificial-intelligence-machine-learning-deep-learning-and-data-science-whats-the-difference-e82f9e7094a>. Acesso em: 18 mai. 2021.

FARAHMAND, A; SARRAFZADEH, A.; SHANBEHZADEH, J. Document Image Noises and Removal Methods. *In*: INTERNATIONAL MULTICONFERENCE OF ENGINEERS AND COMPUTER SCIENTISTS (IMECS), 2013, Hong Kong. **Proceedings** [...]. Hong Kong: PUBLONS, 2013. p. 436-440.

GARDNER, H. The Theory of Multiple Intelligences. **Annals of Dyslexia**, United States, v. 37, p. 19-35, 1987.

GINO, I. The Top 5 OCR APIs & Software by Accuracy, Price & Capabilities. *In*: GINO, I. **RapidAPI**. 23 abr. 2021. Disponível em: <https://rapidapi.com/blog/top-5-ocr-apis/>. Acesso em: 26 ago. 2021.

GOOGLE. Cloud Vision pricing. *In*: GOOGLE. **Google Cloud Platform**. [2021?]. Disponível em: <https://cloud.google.com/vision/product-search/pricing> . Acesso em: 26 ago. 2021.

GRALIŃSKI, F. *et al*. Kleister: A novel task for information extraction involving long documents with complex layout. **arXiv preprint arXiv:2003.02356**, 2020.

HARLEY, A. W.; UFKES, A.; DERPANIS, K. G. Evaluation of deep convolutional Nets for document image classification and retrieval. *In*: INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR), 13., 2015, Tunis, Tunisia. **Proceedings** [...]. Tunis, Tunisia: IEEE, 2015. p. 991-995.

JAUME, G.; EKENEL, H. K.; THIRAN, J. Funsd: A dataset for form understanding in noisy scanned documents. *In*: INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION WORKSHOPS (ICDARW), 2019, Sydney, Australia. **Proceedings** [...]. Sydney, Australia: IEEE, 2019. p. 1-6.

KARAMI, E.; PRASAD, S.; SHEHATA, M. Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. **arXiv preprint arXiv:1710.02726**, 2017.

LEUNG, K. Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER). *In*: HUBERMAN, B. *et al*. **Blog Towards Data Science**. 24 jun. 2021. Disponível em: <https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510>. Acesso em: 3 set. 2021.

MITCHELL, T. M. *et al*. **Machine Learning**. Burr Ridge, IL: McGraw Hill, 1997.

O'GORMAN, L.; KASTURI, R. **Document image analysis**. Los Alamitos: IEEE Computer Society Press, 1995. Disponível em: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.6107&rep=rep1&type=pdf>. Acesso em: 18 mai. 2021.

PALIWAL, S. *et al*. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. *In*: INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR), 15., 2019, Sydney, Australia. **Proceedings** [...]. Sydney, Australia: IEEE, 2019. p. 128-133.

PENG, H.; LONG, F.; CHI, Z. Document image recognition based on template matching of component block projections. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Toronto, v. 25, n. 9, p. 1188-1192, 2003.

PRASAD, D. *et al*. CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents. *In*: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS, 2020, Seattle, USA. **Proceedings** [...]. Seattle, USA: IEEE, 2020. p. 572-573.

ROSA, J. L. G. **Fundamentos da inteligência artificial**. Rio de Janeiro: LTC, 2011.

ROSEBROCK, A. Image alignment and registration with OpenCV. *In*: ROSEBROCK, A. **Blog PyImageSearch**. 31 ago. 2020. Disponível em: <https://www.pyimagesearch.com/2020/08/31/image-alignment-and-registration-with-opencv>. Acesso em: 15 mai. 2021.

ROSEBROCK, A. OCR a document, form, or invoice with Tesseract, OpenCV, and Python. *In*: ROSEBROCK, A. **Blog PyImageSearch**. 7 set. 2020. Disponível em: <https://www.pyimagesearch.com/2020/09/07/ocr-a-document-form-or-invoice-with-tesseract-opencv-and-python/>. Acesso em: 13 mai. 2021.

RUBLEE, E. *et al*. ORB: An efficient alternative to SIFT or SURF. *In*: INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2011, Barcelona, Espanha. **Proceedings** [...]. Barcelona, Espanha: IEEE, 2011, p. 2564-2571.

SCHÖNBERGER, J. L. *et al*. Types of homographies. *In*: SCHÖNBERGER, J. L. et al. **scikit-image website.** 2019. Disponível em: <https://scikit-image.org/docs/stable/auto_examples/transform/plot_transform_types.html>. Acesso em: 3 set. 2021.

SCHREIBER, S. *et al*. DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. *In*: INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION, 14., 2017, Kyoto, Japan. **Proceedings** [...]. Kyoto, Japan: IEEE, 2017. p. 1162-1167

TURING, A. M. Computing machinery and intelligence. **Mind**, Oxford, v. 59, n. 236, p. 433, 1950.

TYAGI, D. Introduction to BRIEF (Binary Robust Independent Elementary Features). *In*: WILLIAMS, E. *et al*. **Blog Medium**. 19 mar. 2019. Disponível em: <https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6>. Acesso em: 18 mai. 2021.

APÊNDICE A - ALGORITMO DE ORDENAÇÃO DE PALAVRAS RETORNADAS PELO OCR

```
def words_with_bounds_to_unique_string(words_with_bounds):
    y_linhas = []
    range_y_linhas = []
    for tupla in words_with_bounds:
        bounds = tupla[0]
        y = bounds[0][1]
        if any([y>=a and y<=b for (a,b) in range_y_linhas]):
            continue
        y_linhas.append(y)
        range_y_linhas.append([y-6, y+6])
    #vamos criar uma lista de vários dicionários, cada dicionário é da forma {"text":texto, "bounds": bounds}
    lista_de_textos = [{"text": texto, "bounds":bounds} for (bounds,texto) in words_with_bounds]
    def isin_range(texto, range_linha):
        #texto deve ser um dicionário da forma {"text":texto, "bounds":bounds}
        #range_linha é da forma (y_inf, y_sup)
        return texto['bounds'][0][1]>=range_linha[0] and texto['bounds'][0][1]<=range_linha[1]
    unique_string = ""
    for i,range_linha in enumerate(range_y_linhas):
        #filtrar os textos da transcrição que pertencem a este range de linha
        isin_range_partial = partial(isin_range, range_linha=range_linha)
        textos_na_linha_atual = list(filter(isin_range_partial, lista_de_textos))#contém também os bounds
        #print('Textos na linha',i,":",list(textos_na_linha_atual))
        textos_na_linha_atual.sort(key = lambda texto: texto['bounds'][0][0])
        palavras_na_linha_atual = list(map(lambda texto: texto['text'], textos_na_linha_atual))#não contém mais
        os bounds
        linha_atual = " ".join(palavras_na_linha_atual)
        unique_string += linha_atual
    return unique_string
```

ANEXO A - TEMPLATE DE FORMULÁRIO DE REQUERIMENTO DA UEPB

UNIVERSIDADE ESTADUAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
SECRETARIA INTEGRADA DE CURSOS
REQUERIMENTO GERAL

DATA: ___ / ___ / _____

NOME

MATRÍCULA **CURSO**

TEL.: ( ) _____ *E-MAIL*: _____

À PROGRAD:

01	Admissão por Transferência para UEPB ;	**10**	Programa de Componente Curricular;
02	Certidão de Conclusão de Curso;	**11**	Reingresso;
03	Declaração de Classificação no Vestibular;	**12**	Registro de Diploma;
04	Declaração de Regularidade de Matrícula;	**13**	Registro de Diploma (apostilamento);
05	Declaração que o Diploma está em Fase de Registro;	**14**	Segunda Via do Diploma;
06	Devolução de Documento (por desistência);	**15**	Trancamento de Matrícula;
07	Dispensa de Componentes Curriculares;	**16**	Trancamento de Matrícula (fora do prazo);
08	Histórico Escolar;	**17**	Colação de Grau Extraordinária;
09	Pedido de Colação de Grau;	**18**	Outros.

AO (A) COORDENADOR (A) DO CURSO:

19	Declaração de Frequência às Aulas;	**21**	Outros.
20	Revisão de Verificação de Aprendizagem de Componente Curricular;		

CÓDIGO _____ **ESCLARECIMENTO:**

_____
_____
_____
_____
_____
_____
_____
_____
_____

VISTO DO SECRETÁRIO

_____
ASSINATURA DO (A) REQUERENTE

.....

UNIVERSIDADE ESTADUAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
SECRETARIA INTEGRADA DE CURSOS
REQUERIMENTO GERAL
SOLICITAÇÃO: Cód.: _____ DATA: ___ / ___ / _____

NOME

MATRÍCULA **CURSO**

VISTO DO SECRETÁRIO