UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIANA MARCELA DA SILVA

PROVENDO ACESSO AO CONTEÚDO DE DOCUMENTOS CENTENÁRIOS: Um processo de correção e melhoria do texto extraído de imagens utilizando técnicas de Processamento de Linguagem Natural

Recife
2023

DIANA MARCELA DA SILVA

PROVENDO ACESSO AO CONTEÚDO DE DOCUMENTOS CENTENÁRIOS: Um processo de correção e melhoria do texto extraído de imagens utilizando técnicas de Processamento de Linguagem Natural

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de mestra em Ciência da Computação. Área de concentração: Inteligência Computacional

Orientadora: Flávia de Almeida Barros

Recife

2023

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586p Silva, Diana Marcela da
Provendo acesso ao conteúdo de documentos centenários: um processo de correção e melhoria do texto extraído de imagens utilizando técnicas de processamento de linguagem natural / Diana Marcela da Silva. – 2023.
118 f.: il., fig.

Orientador: Flávia de Almeida Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Inteligência computacional. 2. Linguagem natural - processamento. I. Barros, Flávia de Almeida (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2023-187

Diana Marcela da Silva

“Provendo acesso ao conteúdo de documentos centenários: Um processo de correção e melhoria do texto extraído de imagens utilizando técnicas de Processamento de Linguagem Natural”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 28 de agosto de 2023.

BANCA EXAMINADORA

_____

Profa. Dra. Flávia de Almeida Barros
Centro de Informática/UFPE
(**Orientadora)**

_____

Profa. Dra. Sandra de Albuquerque Siebra
Departamento de Ciência da Informação / UFPE

_____

Prof.Dr. Marcos Galindo Lima

Departamento de Ciência da Informação / UFPE

Dedico esse trabalho à minha filha e ao meu esposo pelo incentivo e paciência. Vocês foram fundamentais nessa caminhada.

AGRADECIMENTOS

À Deus, por me proporcionar essa oportunidade. Em meio aos momentos mais desafiadores me sustentar e me capacitar para superar as dificuldades ao longo desta jornada. Sem Ele, nada disso teria sido possível.

Aos meus amados pais, Maria e Murilo, pelo exemplo de vida e incentivo em toda minha trajetória.

Ao meu esposo, Raphael, por toda sua paciência e apoio incondicional foram fundamentais. Mesmo nos dias mais difíceis da sua vida, você me fez acreditar que tudo seria possível.

À minha filha Sofia, cujo amor e compreensão, mesmo limitados a sua tenra idade de 2 anos, fortaleceram-me todos os dias. A sua existência é um presente Deus.

Aos meus irmãos, pelas palavras de incentivo.

À Professora Flávia de Almeida Barros, sou eternamente grata por sua orientação, competência, profissionalismo e dedicação que desempenharam um papel tão importante nessa trajetória. Suas orientações e cuidados foram inestimáveis.

À banca examinadora, representada pela Profª Sandra de Albuquerque Siebra e pelo Prof. Marcos Galindo Lima, agradeço por aceitarem participar e por suas contribuições que enriqueceram esta versão final.

Ao projeto LIBER, minha gratidão pela oportunidade de estabelecer essa parceria e colaboração durante esse trabalho.

À instituição, a secretaria da pós-graduação-CIN, em especial Maria Lilia e aos professores que tive o privilégio de aprender tanto.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação e do meu mestrado, o meu sincero agradecimento.

RESUMO

Este estudo de mestrado foi desenvolvido em colaboração com o Laboratório LIBER (DCI-UFPE) dentro do contexto de um projeto mais amplo. O objetivo geral desse projeto é *resgatar, preservar e disponibilizar* livremente jornais centenários em língua portuguesa. O foco principal desta pesquisa de mestrado foi contribuir para o acesso a documentos centenários digitalizados (imagens) de modo a facilitar a realização de consultas baseadas em palavras-chave. Foi realizada a extração do texto dos documentos com o auxílio de algoritmos de *Optical Chacater Recognition* (OCR). No entanto, devido à idade e ao estado de conservação precário dos documentos, o desempenho do OCR nem sempre atingia níveis satisfatórios. Além disso, algumas palavras tinham grafia diferente da atual, tornando difícil *a* correção automática com base nos dicionários modernos das ferramentas de OCR e Processamento de Linguagem Natural (PLN) disponíveis. Assim, foi necessário desenvolver uma solução própria. Nesse contexto, a correção automática dos textos *extraídos*, a fim de identificar e dirimir erros de OCR, é a contribuição central deste trabalho. O corretor ortográfico PyEnchant serviu como base para essa solução devido ao seu alto desempenho. Foi necessário adaptar o dicionário de 1913 ao formato da PyEnchant, uma tarefa que não poderia ser executada manualmente devido ao grande número de termos presentes no dicionário (um total de 125.244 termos). Para superar esse desafio, foi implementado um processo com múltiplas etapas para automatizar a adaptação do dicionário de 1913 ao formato da PyEnchant, fazendo uso de técnicas oriundas da área de PLN. Testes iniciais realizados com algumas imagens disponíveis mostraram que essa abordagem obteve uma boa taxa de cobertura na correção dos erros do OCR. No entanto, alguns desafios persistem, como palavras que foram drasticamente modificadas pelo OCR devido à adição de letras. Essas questões serão abordadas em trabalhos futuros. Por fim, os textos corrigidos foram utilizados para indexar as imagens correspondentes, criando assim um repositório de livre acesso que permite consultas por palavras-chave. Essa etapa foi implementada com a biblioteca *PySolr*. Utilizamos aqui teorias e técnicas oriundas da área de Recuperação de Informação.

Palavras-chave: extração e correção de textos centenários; processamento de linguagem natural; recuperação de informação.

ABSTRACT

This master's study was developed in collaboration with the LIBER Laboratory (DCI-UFPE) within the context of a broader project. The overall goal of this project is to retrieve, preserve, and freely provide access to centenary newspapers in the Portuguese language. The primary focus of this master's research was to contribute to the accessibility of digitized centenary documents (images) to facilitate keyword-based queries. Text extraction from the documents was performed using Optical Character Recognition (OCR) algorithms. However, due to the age and poor condition of the documents, the OCR performance did not always reach satisfactory levels. Additionally, some words had different spellings from contemporary language, making automatic correction based on modern dictionaries from OCR and Natural Language Processing (NLP) tools challenging. Hence, it was necessary to develop a custom solution. In this context, the automatic correction of extracted texts to identify and rectify OCR errors is the central contribution of this work. The PyEnchant spell checker served as the foundation for this solution due to its high performance. Adapting the 1913 dictionary to the PyEnchant format was essential, but this task could not be done manually due to the large number of terms in the dictionary (a total of 125,244 terms). To overcome this challenge, a multi-step process was implemented to automate the adaptation of the 1913 dictionary to the PyEnchant format, utilizing techniques from the field of NLP. Initial tests with some available images demonstrated that this approach achieved a good coverage rate in correcting OCR errors. However, some challenges persisted, such as words that were significantly altered by the OCR due to the addition of extra letters. These issues will be addressed in future work. Finally, the corrected texts were used to index the corresponding images, creating a repository with open access for keyword-based queries. This stage was implemented using the PySolr library and incorporated theories and techniques from the field of Information Retrieval.

Keywords: extraction and correction of centennial texts; natural language processing; information retrieval.

.

LISTA DE IMAGENS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O Brasil já ultrapassou 200 anos da criação da imprensa nacional, e o estado de Pernambuco foi um dos pioneiros nessa área com a fundação do periódico *Diário de Pernambuco*. Segundo (GASPAR, 2009), "... jornal mais antigo em circulação na América Latina, o *Diário de Pernambuco* foi fundado no dia 7 de novembro de 1825, pelo tipógrafo Antonino José de Miranda Falcão, no Recife.".

A preservação e a disponibilização desse acervo inestimável é tema de um projeto de curadoria digital mais amplo, coordenado pelo professor Marcos Galindo Lima (Laboratório LIBER[1], DCI-UFPE) em colaboração com a Associação de Imprensa de Pernambuco (AIP).[2,3] O projeto "Programa de Curadoria do patrimônio memorial de Jornais centenários de Pernambuco" objetiva *resgatar*, *preservar* e *prover livre acesso* a jornais periódicos centenários em língua portuguesa. O resultado desse projeto será de grande utilidade para jornalistas, historiadores e outros cidadãos interessados nesse acervo.

Os processos iniciais foram executados pelos pesquisadores do LIBER, que resgataram e digitalizaram (escanearam) diversos exemplares do "Diário de Pernambuco" datados do início do século XIX. Já o terceiro processo configura o objetivo geral desta pesquisa de mestrado, visando contribuir para o acesso a documentos antigos digitalizados (imagens) de modo a facilitar a realização de consultas baseadas em palavras-chave.

Acreditamos que este trabalho trará uma grande contribuição para o resgate da história cotidiana contada nesse acervo, libertando a informação guardada nesses jornais há 200 anos, para que se torne de fácil e livre acesso. Essas memórias certamente irão contribuir para a criação de novo conhecimento histórico, derivado de uma visão atual dos acontecimentos passados.

Considerando o acervo já digitalizado, o próximo passo para criação do repositório indexado foi a seleção de palavras/termos (tokens) para indexar automaticamente cada imagem. A opção de seleção manual dos tokens foi

---

[1] LIBER - www.liber.ufpe.br
[2] https://www.diariodepernambuco.com.br/noticia/vidaurbana/2021/09/parceria-vai-preservar-o-rico-acervo-do-diario-de-pernambuco.html
[3] https://www.ufpe.br/agencia/noticias/-/asset_publisher/dlhi8nsrz4hK/content/ufpe-fara-restauracao-e-digitalizacao-do-acervo-historico-do-diario-de-pernambuco/40615

descartada devido ao grande volume do acervo a ser preservado. Assim, este trabalho optou pela seleção automática desses termos através da extração de texto utilizando algoritmos de Reconhecimento Óptico de Caracteres (do inglês, *Optical Chacater Recognition* - OCR[4]), que têm demostrado ótimo desempenho.

Contudo, como estamos lidando com documentos muito antigos, que podem apresentar dobras e deterioração causada por fungos e maus cuidados, às vezes as palavras extraídas estão incompletas ou incorretas. Além disso, o texto nesses jornais periódicos não era organizado em coluna única, tendo notícias separadas por traços verticais e horizontais, e tendo também separação de sílabas. Esses traços e outros caracteres especiais muitas vezes atrapalham a extração correta das palavras. Por fim, vale lembrar que algumas palavras tinham grafia diferente da atual (por exemplo, *pharmácia - farmácia*), dificultando a correção automática do texto com base nos dicionários modernos contidos nas ferramentas de OCR e de Processamento de Linguagem Natural disponíveis. A correção manual dos termos extraídos foi descartada, pela mesma razão já apresentada anteriormente, sendo, portanto, necessário desenvolver uma solução própria.

Nesse contexto, este trabalho de mestrado teve como objetivos específicos a melhoria do texto extraído via OCR a partir das imagens digitalizadas, através da (1) *extração e correção automática dos termos contidos nas imagens*, (2) *indexação das imagens* a partir dos termos corrigidos, permitindo consultas baseadas em palavras-chave que cada imagem contém a partir da (3) *criação de um repositório*.

Embora existam muitas plataformas e ferramentas de Processamento de Linguagem Natural que realizam a correção automática de textos, não encontramos nenhuma opção de livre uso adequada para o nosso contexto. Assim, foi necessário desenvolver uma solução com base em um dicionário de 1913 (FIGUEIREDO, 1913), que se aproxima mais da grafia utilizada nos documentos centenários do que os dicionários atuais. A seguir, a indexação das imagens foi realizada usando-se a ferramenta PySolr. A seção a seguir descreve melhor o trabalho realizado e suas principais contribuições.

[4] OCR - https://en.wikipedia.org/wiki/Optical_character_recognition

1.1 TRABALHO REALIZADO

Este trabalho é uma pesquisa descritiva, qualitativa e aplicada que foi

desenvolvida em foi realizada em quatro etapas (módulos), visando o reuso da solução oferecida: (a) Pré-processamento das imagens de entrada (b) Extração do texto via OCR; (c) Processamento (tratamento) do texto extraído, a fim de identificar e dirimir erros de OCR; e (d) Indexação das imagens a partir dos termos tratados, resultando em um repositório para livre acesso a esses documentos através de interface de consultas baseadas em palavras-chave.

A etapa (a) consiste na aplicação de diversas técnicas de pré-processamento às imagens digitalizadas, com o intuito de melhorar as imagens e, assim, otimizar a eficiência da extração de texto via OCR (seção 4.2). Utilizamos aqui algumas técnicas disponíveis via *OpenCV* (*Open Source Computer Vision*[5]), uma biblioteca popular para processamento de imagens e visão computacional em Python.

A etapa (b), voltada à extração de texto a partir das imagens digitalizadas, foi realizada utilizando-se o *PyTesseract*[6], uma biblioteca em Python que disponibiliza a tecnologia OCR para reconhecer e extrair texto a partir de imagens (seção 4.3). Essa biblioteca é baseada no *Tesseract*[7], que é um mecanismo de OCR de código aberto. Depois de testarmos algumas opções, essa opção foi escolhida por ter apresentado o melhor resultado para imagens tão antigas e com falhas devidas às avarias no papel original dos jornais escaneados.

A etapa (c) é responsável pelo processamento do texto pós-OCR, com o objetivo de corrigir o texto extraído, reduzindo assim as imprecisões do OCR e melhorando a qualidade do texto final (seção 4.4). *A correção automática dos textos extraídos* é a principal contribuição deste trabalho. Depois de investigar várias técnicas e ferramentas de correção ortográfica automática, optamos por utilizar a biblioteca Python *PyEnchant*[8] como base da solução desenvolvida, pelo seu alto desempenho. Contudo, essa biblioteca oferece um corretor ortográfico que utiliza um dicionário atual da língua portuguesa. Como estamos trabalhando com jornais do

[5] OpenCV - https://opencv.org/
[6] PyTesseract - https://pypi.org/project/pytesseract/
[7] Tesseract - https://github.com/tesseract-ocr/tesseract
[8] PyEnchant - https://pypi.org/project/pyenchant/

século XIX, foi necessário buscar um dicionário que se aproximasse mais da grafia utilizada nesses documentos.

Inicialmente, consideramos usar um dicionário[9] de 1832, que só está disponível em formato de imagem. A extração via OCR dos termos desse dicionário não foi bem-sucedida, pois caímos em um problema semelhante ao que estamos tentando resolver (extração de textos a partir de documentos centenários). Assim, optamos por utilizar um dicionário de 1913 que já se encontrava em formato digital. Porém, o padrão de dicionário utilizado pela *PyEnchant* não se baseia apenas em uma lista de vocábulo, utilizando metadados associados às entradas do dicionário para realizar a flexão dos termos (e.g., gênero e número, flexão verbal, aumentativo e diminutivo etc.). Então foi necessário adaptar o dicionário de 1913 para o formato da *PyEnchant*.

Não seria viável fazer essa adaptação de modo manual, devido ao grande volume de entradas do dicionário escolhido (125.244 termos). Assim, foi necessário implementar um processo com vários passos para automatizar a adaptação do dicionário de 1913 ao padrão da *PyEnchant*. Utilizamos aqui técnicas oriundas da área de Processamento de Linguagem Natural. Testes iniciais realizados com imagens disponíveis mostraram uma boa taxa de cobertura na correção dos erros do OCR. Foi observado que algumas palavras escaparam da correção por terem letras adicionadas pelo OCR que modificaram muito a palavra (radical) original. Esse problema será tratado em trabalhos futuros.

Por fim, na etapa (d), os textos já corrigidos foram utilizados para indexar as imagens correspondentes, criando assim um repositório para livre acesso através de consultas via palavras-chave (seção 4.5). Essa etapa foi implementada com apoio da biblioteca *PySolr*[10]. Utilizamos aqui teorias e técnicas oriundas da área de Recuperação de Informação.

Assim, consideramos que os objetivos iniciais deste trabalho foram alcançados, tendo sido materializados no dicionário adaptado e na base de imagens indexadas automaticamente através de termos que ocorrem em cada imagem. Devido a restrições de tempo, não foi possível implementar todas as melhorias

[9] Dicionário 1832 - https://digital.bbm.usp.br/handle/bbm/5414
[10] PySolr - https://pypi.org/project/pysolr/

inicialmente planejadas, estando indicadas como trabalhos futuros no Capítulo 6. Destacamos aqui a modernização vocabular para facilitar as buscas via interface.

1.2 ORGANIZAÇÃO DO DOCUMENTO

Além da Introdução, este documento está organizado em mais 5 capítulos.

O **Capítulo 2** aborda a extração de textos a partir de imagens com base em OCR. Esse tema é de importância central para o nosso trabalho de mestrado, incluindo duas grandes áreas de pesquisa: o processamento de imagens e o processamento de texto – processamento de linguagem natural.

O **Capítulo 3** se dedica a apresentar os corretores ortográficos automáticos em mais detalhe, aprofundando a discussão sobre os dicionários/léxicos e as técnicas utilizadas por essas ferramentas para realizar a correção dos textos. Por fim, temos um abreve discussão sobre trabalhos relacionados ao nosso.

O **Capítulo 4** apresenta em detalhes o processo proposto para promoção da acessibilidade ao conteúdo de documentos históricos, que recebe como entrada imagens escaneadas, realiza a correção e a melhoria dos textos extraídos via OCR e, por fim, indexa as imagens de entrada a partir das palavras que ocorrem em cada imagem. Esse capítulo destaca a criação/adaptação do dicionário de 1913 para o padrão *PyEnchant*.

O **Capítulo 5** apresenta os estudos de caso realizados com o protótipo desenvolvido, que foi utilizado para processar imagens de dois jornais centenários diferentes, Diário de Pernambuco de 1840, e Diário da Manhã de 1930. Comentaremos aqui sobre os resultados obtidos com o processo de correção pós-OCR, e veremos exemplos de uso da interface de consulta ao repositório de imagens indexados criado neste trabalho.

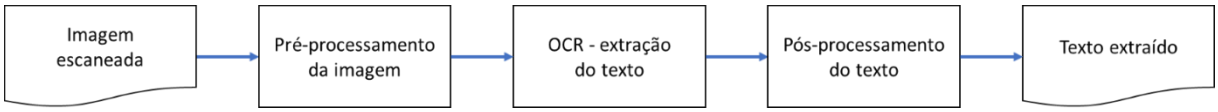
O **Capítulo 6** conclui este documento com uma breve revisão das principais contribuições e com indicação de trabalhos futuros para melhorar e estender os resultados obtidos até o momento.

2 EXTRAÇÃO DE TEXTOS A PARTIR DE IMAGENS VIA OCR

Este capítulo aborda a extração de textos a partir de imagens com base no reconhecimento óticos de caracteres (*Optical Characater Recognition* – OCR). Esse tema é de importância central para o nosso trabalho de mestrado, incluindo duas grandes áreas de pesquisa: o processamento de imagens e processamento de texto.

A Figura *1* resume as etapas geralmente presentes na extração de texto a partir de imagens com base no OCR. As etapas 1 e 2 estão relacionadas ao processamento de imagens (seções 2.1 e 2.2), enquanto a etapa 3 utiliza de técnicas de processamento de texto (seção 2.3).

Figura 1 - Etapas presentes na extração de textos com base em OCR.



Fonte: A autora (2023).

O processamento de imagens pode ser definido de forma simples como a utilização de algoritmos computacionais para a análise e manipulação de imagens digitais (GONZALEZ, 2018). Devido ao grande volume de documentos digitalizados atualmente, uma tarefa de grande relevância é a extração de textos a partir de imagens, geralmente utilizando técnicas de reconhecimento óptico de caracteres (do inglês, *Optical Character Recognition* - OCR) (SMITH, 2007). As técnicas de OCR conseguem extrair com eficiência textos a partir de imagens de documentos legíveis e com vocabulário atual. Contudo, quando estamos tratando de documentos muito antigos, a precisão do OCR pode ser consideravelmente reduzida.

A extração de texto a partir de documentos centenários demanda cuidados especiais, uma vez que eles usualmente apresentam certo grau de degradação devido a diversos fatores, tais como oxidação e envelhecimento do papel e da tinta e desgaste causado pela manipulação. Além disso, existem questões cruciais como a linguagem utilizada e à forma como o texto é representado graficamente no papel. As variações linguísticas são resultado da evolução da língua ao longo do tempo, o que influencia o léxico eletrônico usado no OCR. Já as diferenças gráficas surgem

devido às mudanças na tecnologia de impressão, nas fontes utilizadas e na antiguidade dos documentos originais (HAUSER, 2007).

Nesse contexto, é de fundamental importância a realização de melhorias das imagens, com o objetivo de remover os efeitos indesejados contidos na imagem escaneada dos documentos originais. Essa etapa é comumente denominada de *pré-processamento da imagem*, pois antecede a etapa de OCR (seção 2.1). A etapa de pré-processamento geralmente retorna uma imagem mais limpa, otimizando assim a eficiência da etapa de OCR. A seguir, é realizada a *extração do texto*, sendo esse o núcleo do processo de OCR (seção 2.2).

Por fim, o texto extraído pode ainda passar por um pós-processamento, a fim de aprimorar a saída da etapa de OCR (seção 2.3). No caso específico de documentos históricos, essa etapa é de importância central, devido à maior quantidade de erros oriundos do OCR. Como mencionado acima, as imagens desses documentos apresentam muitas falhas, e as variações linguísticas (mudanças ortográficas, palavras em desuso etc.) dificultam a correta extração do texto original utilizando-se as ferramentas de OCR atuais. O *pós-processamento do texto* pode usar um simples corretor ortográfico, ou pode também buscar reconhecer palavras com base no seu contexto de ocorrência. Ainda é possível usar um corretor gramatical, em casos especiais.

Esses temas serão brevemente apresentados nas seções a seguir. Dedicamos mais atenção ao *pós-processamento de texto*, por ser a etapa desenvolvida neste trabalho de mestrado para melhorar a qualidade da extração de textos centenários. Por se tratar de uma área de pesquisa muito ampla, vamos manter o foco nos conceitos e técnicas mais relacionados ao nosso trabalho. Por fim, teremos uma breve apresentação de algumas ferramentas de OCR (seção 2.4).

2.1 PRÉ-PROCESSAMENTO DE IMAGENS PARA OCR

A etapa de pré-processamento das imagens para OCR se baseia em técnicas que têm por objetivo aprimorar as imagens digitalizadas de forma a melhorar as chances de reconhecimento bem-sucedido do texto. Essas técnicas desempenham um papel crucial no processo de OCR, impactando significativamente a precisão e qualidade do texto extraído.

Ao receber uma imagem do usuário, algumas ferramentas de OCR realizam internamente técnicas (ou fases) de pré-processamento antes de executar as funções de extração do texto. A depender da ferramenta escolhida, estas funções de pré-processamento podem ser inclusive configuradas pelo usuário. Contudo, diversos trabalhos sugerem que é vantajoso implementar funções de pré-processamento personalizadas para o contexto da imagem, antes do uso de alguma ferramenta de OCR (MITTAL; GARG, 2020; MURSARI, 2021). A seguir são descritas as técnicas usualmente empregadas na melhoria das imagens.

Alinhamento do documento: É preciso alinhar o documento corretamente para a digitalização. Se o documento não foi alinhado corretamente quando digitalizado pode ser necessário corrigir sua inclinação rotacionando o documento em alguns graus no sentido horário ou anti-horário, a fim de tornar as linhas do texto perfeitamente horizontais ou verticais.

Conversão para escala de cinza: As imagens coloridas geralmente são construídas com vários canais de cores, cada um deles representando os níveis de valor de um determinado canal. Por exemplo, as imagens RGB (do inglês, *Red-Green-Blue*) são compostas por três canais independentes para componentes de cores primárias vermelho, verde e azul. Uma imagem em tons de cinza é aquela constituída de apenas um canal de cores que representa a informação da intensidade luminosa em cada pixel. Para realizar a conversão da imagem RGB obtida do scanner para a imagem em escala de cinza utiliza-se usualmente uma média ponderada em cada pixel dos valores de cada canal da imagem.

Binarização: Esta é uma das fases mais cruciais do pré-processamento. A binarização converte uma imagem colorida em uma imagem que consiste apenas em pixels pretos e brancos (valor de pixel preto = 0 e valor de pixel branco = 255). Como regra básica, a conversão pode ser feita fixando um limite (geralmente o limite é 127, pois é exatamente metade do intervalo de pixels de 0 a 255). Se o valor do pixel for maior que o limite, ele será considerado um pixel branco, caso contrário, será considerado um pixel preto. Todavia, nos casos em que as condições de iluminação não são uniformes na imagem, este método falha. Existem diversas técnicas diferentes para encontrar o valor limite, sendo o método de limiarização adaptativa um dos mais eficientes. Esse método fornece um limiar de forma adaptativa para uma pequena parte da imagem que depende das características de

sua localidade e vizinhos, ou seja, não há um limite fixo único para a imagem toda, e sim cada pequena parte da imagem tem um valor limite diferente.

Remoção de ruído: O principal objetivo da fase de remoção de ruído é suavizar a imagem removendo pequenos pontos/manchas que possuem intensidade mais alta do que o restante da imagem. Novamente, diversos métodos estão disponíveis para realização dessa tarefa.

Operações morfológicas: São um conjunto de operações que processam imagens com base em formas. As operações morfológicas aplicam um elemento estruturante a uma imagem de entrada e geram uma imagem de saída. Destacam-se duas operações morfológicas mais básicas: Erosão e Dilatação. Essas técnicas buscam aprimorar ainda mais a remoção de ruído, isolar elementos individuais e juntar elementos díspares em uma imagem, além de encontrar saliências ou buracos de intensidade em uma imagem. O borrão de tinta em documentos históricos pode ser compensado usando uma técnica de erosão. A erosão pode ser usada para reduzir os caracteres para sua estrutura de glifo normal. A posterior dilatação da imagem permite restaurar possíveis degradações indesejadas que foram impostas aos caracteres de interesse juntamente com as partículas ruidosas na etapa de erosão.

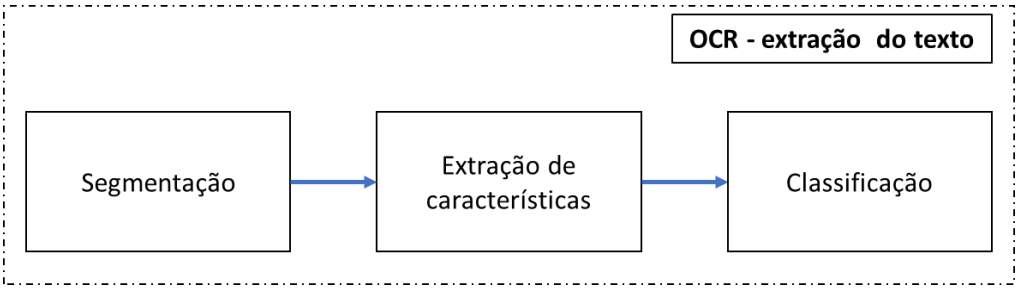
As ferramentas de OCR geralmente se diferenciam pela integração e personalização de suas técnicas de pré-processamento de imagem (ver seção 2.4). Algumas ferramentas fornecem um conjunto abrangente de opções de pré-processamento, permitindo que os usuários ajustem parâmetros como redução de ruído, binarização e correção de inclinação com base em seus requisitos específicos. Outras podem se concentrar em algoritmos automatizados que ajustam de forma inteligente as configurações de imagem para resultados de OCR ideais. Além disso, como já mencionado, é possível aplicar técnicas de pré-processamento personalizadas ao cenário de aplicação antes do uso da ferramenta de OCR, possibilitando alcançar maior precisão na extração do texto.

2.2 RECONHECIMENTO ÓPTICO DE CARACTERES (OCR)

Uma vez que a imagem de entrada foi adequadamente pré-processada, a etapa de reconhecimento de texto é executada. Essa etapa é composta de algumas

fases (processos) necessárias para a obtenção de um arquivo de texto editável por computador. A Figura *2*, a seguir, ilustra as fases presentes no núcleo OCR.

Figura 2 - Fases presentes na etapa extração do texto com OCR.



Fonte: A autora (2023).

A **Segmentação** objetiva isolar o elemento de interesse do resto da imagem e é constituída de diversos subprocessos. Na segmentação de página, isola-se o conteúdo do restante da imagem, resultando em uma imagem contendo apenas texto. Na segmentação de linhas, o objetivo é identificar e segmentar a imagem em linhas de texto.

A segmentação de palavras objetiva segmentar uma imagem contendo uma única linha (resultado da etapa anterior) em uma sequência de palavras. A segmentação de caracteres, por fim, processa a imagem de cada palavra (resultado da etapa anterior) segmentando-a em caracteres individuais. Se o OCR estiver sendo aplicado a um texto no qual os caracteres das palavras estão separados por espaços, a segmentação em nível de caractere é facilitada, uma vez que há uma lacuna uniforme entre os caracteres dentro da palavra.

A segmentação de caracteres torna-se particularmente importante ao lidar com cenários desafiadores, como texto cursivo ou fontes complexas, em que as formas e a conectividade dos caracteres podem variar significativamente.

A **Extração de Característica** é o módulo de extração de características é um componente crucial dos sistemas OCR (reconhecimento óptico de caracteres) que desempenha um papel fundamental na conversão de representações visuais de caracteres em características significativas e reconhecíveis. Este módulo extrai informações relevantes dos caracteres segmentados ou regiões de texto, possibilitando as etapas posteriores de classificação e reconhecimento. A extração de características envolve a transformação dos dados brutos de pixel dos caracteres

em um conjunto de recursos representativos que capturam as características distintivas de cada caractere. Esses recursos devem ser discriminativos o suficiente para diferenciar um caractere do outro, mesmo ao lidar com variações de fontes, tamanhos e estilos.

A etapa de **classificação** nos OCR é um componente fundamental do processo de extração do texto. Ela envolve a identificação e rotulação de caracteres ou símbolos individuais em uma imagem. Durante a classificação, as características extraídas de cada caractere, como a distribuição de intensidade de pixels, propriedades de forma e informações estruturais, são utilizadas como entrada para um modelo de classificação treinado. Esse modelo, geralmente desenvolvido utilizando algoritmos de aprendizado de máquina, como redes neurais convolucionais (CNNs) ou máquinas de vetor de suporte (SVMs), aprendeu com exemplos rotulados de caracteres durante a fase de treinamento. Reconhecendo padrões e relacionamentos nas características extraídas, o modelo de classificação prevê a classe ou rótulo de cada caractere. Técnicas de pós-processamento podem ser aplicadas para refinar ainda mais os resultados. A classificação precisa dos caracteres é crucial para um OCR bem-sucedido, permitindo a conversão de texto baseado em imagem em formatos digitais editáveis e pesquisáveis.

Por fim, as ferramentas atuais de OCR realizam de uma etapa de pós-processamento interno para juntar os caracteres reconhecidos, a fim de formar as palavras do texto. Algumas ferramentas aceitam dicionários da língua em foco para auxiliar nesse processo. Porém, cada ferramenta poderá implementar essa fase usando técnicas diferentes (ver seção 2.4).

Além desse pós-processamento interno, pode ser necessário realizar mais ajustes nos textos de saída, uma vez que o texto extraído via OCR pode apresentar erros, principalmente quando a imagem de entrada tem baixa qualidade. Nesses casos, o texto extraído pode ser melhorado por uma etapa externa de pós-processamento do texto, que pode contar com vários processos independentes, implementados com técnicas variadas. A seção 2.3 apresenta técnicas e abordagens para o pós-processamento dos textos extraídos via OCR.

2.3 PÓS-PROCESSAMENTO DE TEXTO

Como já mencionado, é comum que o texto extraído pelas ferramentas de OCR contenha erros, especialmente quando a qualidade da imagem de entrada é baixa. Além dessa questão, ressaltamos também outros problemas mais específicos do nosso contexto de trabalho: jornais periódicos centenários. O texto nesses documentos históricos não era organizado em coluna única, tendo notícias separadas por trações verticais e horizontais, e tendo também separação de sílabas. Além disso, ocorreram mudanças na ortografia da língua (por exemplo, *pharmácia* - *farmácia*), dificultando a correção automática do texto com base nos dicionários modernos. Para refinar o texto extraído, garantindo maior precisão e legibilidade, é importante realizar uma etapa adicional de pós-processamento externo.

Historicamente, a motivação para o desenvolvimento de métodos de correção pós-OCR advém da ineficiência das ferramentas de OCR na obtenção de resultados suficientemente precisos para documentos históricos (ENGLMEIER; FINK; SCHULZ, 2019). Neste sentido, o pós-processamento desempenha um papel crítico na preservação e na interpretação dessa categoria de documentos. Por meio desse processo, é possível aprimorar a qualidade e a legibilidade do conteúdo extraído, garantindo a precisão e a acessibilidade desses documentos, e permitindo que a riqueza da história seja compartilhada e apreciada por gerações futuras.

Esta seção descreve algumas das técnicas mais usadas no pós-processamento dos textos extraídos via OCR. A seguir, a seção 2.3.1 dá uma visão geral da etapa de pós-processamento de texto. A seção 2.3.2 traz uma breve apresentação sobre técnicas de Processamento de Linguagem Natural (PLN) que podem ser usados nesse pós-processamento, e a seção 2.3.3 descreve outras técnicas importantes usadas nessa etapa de extração de texto.

2.3.1 Visão geral

A etapa de pós-processamento de texto, seja interna ou externa às ferramentas, pode incluir técnicas como correção ortográfica, detecção e correção de erros de formatação, entre outros processos independentes, que podem utilizar técnicas variadas. Veremos aqui algumas das técnicas mais utilizadas nessa tarefa.

Na literatura relacionada, as abordagens/técnicas de pós-processamento do texto são categorizadas entre manual, semiautomática ou totalmente automática, de acordo com o nível de dependência humana (DROBAC, 2020).

A *abordagem manual* envolve intervenção humana para melhorar a precisão e a qualidade do texto reconhecido pelo OCR. Nesse caso, a imagem original é consultada pelo responsável pela correção, buscando identificar possíveis erros de OCR. Essa abordagem pode incluir várias etapas, como revisão, detecção de erros e correção. Essas tarefas requerem experiência humana e podem ser demoradas e dispendiosas. Por fim, vale lembrar que os humanos também podem confundir caracteres graficamente semelhantes quando seu contexto não é considerado (e.g., i, l, I, 1, 0, O, o, e, c) (EVERSHED e FITCH, 2014). Contudo, em geral essa abordagem oferece um alto nível de controle e precisão no refinamento dos resultados de OCR.

Apesar dos resultados precisos, nem sempre é viável realizar essa verificação de modo manual, por falta de tempo ou falta de pessoas capacitadas. Assim, torna-se fundamental utilizar alguma técnica (semi)automática de verificação de texto, como um corretor ortográfico automático capaz de corrigir palavras com pequenos erros, e de destacar os termos não encontrados no léxico/dicionário em uso.

A *abordagem semiautomática* geralmente utiliza alguma ferramenta que auxilia o humano na verificação e correção do texto. É comum a interface oferecer duas telas, uma mostrando a imagem original e outra tela de edição que ajuda o humano na correção do texto. A ferramenta destaca prováveis erros e pode também fazer sugestões para correção. Em seguida, a experiência humana e a compreensão da língua são usadas na tomada de decisão. Essa abordagem faz uso de técnicas automáticas de correção de texto (seção 2.3.2), semelhante ao que temos nos editores de texto atuais.

Por fim, a *abordagem totalmente automática* dispensa intervenção humana, trabalhando apenas com o texto obtido do OCR, sem consultar a imagem original. As abordagens automática e semiautomática foram implementadas no nosso trabalho de mestrado.

2.3.1.1 Pós-processamento automático de texto

Essa abordagem utiliza métodos mais complexos e especializados, podendo

contar com vários processos independentes, implementados com técnicas variadas. A grande maioria das técnicas de pós-processamento automático se baseiam em métodos da Inteligência Artificial ou da Estatística. A IA tem duas vertentes predominantes: a *Abordagem Simbólica*, que se baseia na representação do conhecimento em bases de conhecimento, utilizando regras e um mecanismo de raciocínio; e a *Aprendizagem de Máquina* (AM), baseada em algoritmos capazes de induzir regras a partir de exemplos (RUSSEL e NORVIG, 2003). Já a abordagem estatística utilizada nessa aplicação se baseia na análise de grandes corpora de texto (JURAFSKY e MARTIN, 2008).

Apesar das dificuldades de implementar esses métodos automáticos, esta tem sido a abordagem mais adotada atualmente, devido ao grande volume de documentos textuais a serem analisados e pós-processados. Além disso, esta é a única opção para o pós-processamento interno às ferramentas, que não depende do usuário.

As técnicas utilizadas no pós-processamento de texto podem ser classificadas em duas categorias principais, dependo das informações usadas no processo realizado: técnicas baseadas em *palavras isoladas* ou técnicas *dependentes do contexto*. As técnicas baseadas em palavras isoladas consideram apenas os recursos do próprio token extraído pelo OCR, como sua presença em um dicionário, a similaridade entre o token e uma entrada no léxico, sua frequência, pontuação de confiança de reconhecimento fornecida pelo software OCR, entre outros (NGUYEN, 2022).

Já as técnicas dependentes do contexto no pós-processamento de OCR visam melhorar a precisão e a correção do texto reconhecido considerando o contexto circundante de cada palavra. Essa abordagem vai além da análise de palavras individuais e leva em consideração fatores como relacionamentos entre palavras, modelos de linguagem e estruturas (morfo)sintáticas das frases do texto. Ao considerar o contexto, essas abordagens podem lidar com erros que não se baseiam apenas no reconhecimento de palavras individuais, mas também na coerência e consistência de todo o texto (NGUYEN, 2022).

A seguir, vamos apresentar brevemente algumas das técnicas mais utilizadas dentro da abordagem automática de pós-processamento. A seção 2.3.2 apresenta brevemente alguns conceitos da área de Processamento de Linguagem Natural, que

está intimamente relacionada ao processamento de texto. Por fim, a seção 2.3.3 destaca outras abordagens relevantes para correção do texto.

2.3.2 PLN para Pós-processamento de texto

Esta seção tem por objetivo dar uma visão geral do Processamento de Linguagem Natural (seção 2.3.2.1), apresentando brevemente os processos de PLN mais usados no pós-processamento de textos (seção 2.3.2.2). Nosso foco será na análise léxica/morfológica, que tem mais relação com o trabalho desenvolvido neste mestrado. Por fim, a seção 2.3.2.3 cita algumas ferramentas que auxiliam na construção de sistemas de PLN.

2.3.2.1 Visão geral do PLN

O Processamento de Linguagem Natural tem por objetivo principal criar sistemas capazes de interpretar e/ou gerar automaticamente documentos/dados em alguma língua natural. O PLN é considerado uma área de pesquisa interdisciplinar, combinando métodos e técnicas de Inteligência Artificial (IA) com a Linguística (JURAFSKY e MARTIN, 2008). As subáreas tradicionais de estudo/atuação do PLN são a *Interpretação de LN*, que busca compreender o texto de entrada e processá-lo com algum objetivo final, e a *Geração de LN*, que objetiva gerar texto a partir de outros tipos de dados (HIRSCHBERG e MANNING, 2015).

Aplicações de PLN comuns na atualidade são (HIRSCHBERG e MANNING, 2015): Corretores ortográficos e gramaticais, Preditores de texto, Sistemas de pergunta-resposta (semelhantes a FAQs – *frequent asked questions*), Tradução automática, Chatbots com fala e/ou imagem associada, Mineração de opinião, Extração de informação a partir de textos livres, Detecção de tópico/tema do texto, Sumarização de texto, dentre outras. Destacam-se ainda áreas de estudo que dão base ao desenvolvimento dessas e de outras aplicações do PLN na atualidade: Processamento da fala[11]; Reconhecimento óptico de caracteres (OCR) a partir de textos manuscritos ou tipografados; Classificação e agrupamento de documentos, entre outras.

[11] Processamento da fala - https://en.wikipedia.org/wiki/Speech_processing

As abordagens usadas no desenvolvimento de sistemas de PLN variam ao longo do tempo, combinando técnicas de IA e Estatística com a teoria Linguística. É possível destacar três grandes fases das pesquisas em PLN[12], descritas a seguir. Essas abordagens não são excludentes, e suas técnicas ainda estão em uso, isoladamente ou combinadas em sistemas híbridos.

Inicialmente (1950–1980), os trabalhos de PLN eram desenvolvidos dentro da *abordagem simbólica* da IA (ALLEN, 1994; BARROS e ROBIN, 2001), empregando técnicas de casamento de padrões (ELIZA[13]) e informação linguística. Na década de 1980 surgiram os parsers baseados em gramáticas linguísticas, modelos de interpretação semântica, análise do discurso/diálogo e interpretação pragmática. A *abordagem estatística* (1990–2010) marca o surgimento da Linguística de *corpus,* que trabalha com grandes bases de dados anotados. Nos anos 2000, com enorme crescimento de dados textuais na Web, essa abordagem tornou-se a mais adotada.

A partir da década de 2010, a *Aprendizagem de Máquina* passou a ser a abordagem predominante no PLN (JURAFSKY e MARTIN, 2008). Além das técnicas tradicionais de AM supervisionada e não supervisionada, também surgiram trabalhos baseados em Redes neurais profundas (*Deep learning*). Essas técnicas são utilizadas na construção de *parsers*, na indução de gramáticas, classificação e agrupamento de texto, no OCR, no reconhecimento de escrita cursiva, reconhecimento da fala e síntese de voz, entre outras.

Nosso trabalho de mestrado foi desenvolvido dentro da abordagem simbólica, por ter se mostrado a mais adequada ao nosso objetivo. Assim, essa será a abordagem detalhada a seguir.

2.3.2.2 PLN com foco na Análise Léxica/Morfológica

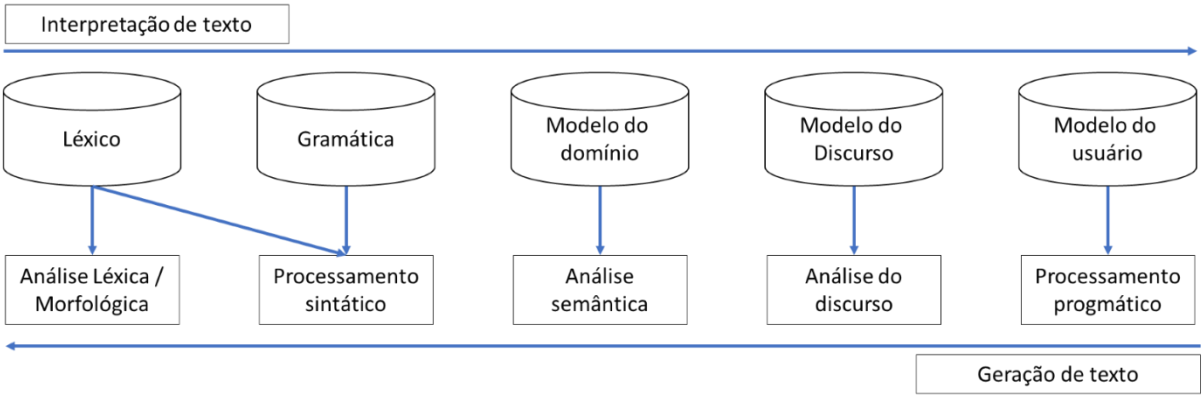
Esta seção apresenta brevemente as principais *fases/etapas de processamento* de sistemas de PLN baseados na Abordagem Simbólica (Figura *3*) (adaptada de Barros e Robin, 2001). Contudo, é importante frisar que essas fases também podem ser implementadas usando técnicas das outras abordagens mencionadas acima (Estatística e Aprendizagem de máquina).

---

[12] PLN - https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=1168924527
[13] ELIZA - https://www.masswerk.at/elizabot/

A Figura 3 destaca as cinco grandes fases do PLN, sugerindo ainda que os sistemas de interpretação e de geração de texto passam por essas mesmas fases. Contudo, dependendo da direção do PLN, há diferenças no processo que cada módulo executa, bem como na forma de representar conhecimento em cada base. Como nosso trabalho é de interpretação de texto (correção ortográfica no pós-processamento de texto), vamos discutir brevemente os processos de *interpretação de LN*, para não fugir do nosso escopo. Vamos ressaltar aqui o que foi abordado no nosso trabalho, e o Capítulo 4 irá detalhar como a solução foi construída.

Figura 3 - Fases do PLN



Fonte: Adaptada de (Barros e Robin, 2001).

Como mostra a Figura 3, um sistema de PLN simbólico completo pode contar com 5 **bases de conhecimento**: o Léxico e a Gramática da língua em uso, além do modelo do domínio da aplicação, modelo do discurso (para relacionar elementos das frases do texto) e modelo do usuário (para sistemas com perfil de usuário). Nosso trabalho conta apenas com o Léxico, devido ao seu foco atual. Assim, daremos mais destaque a essa base de conhecimento. Vamos ainda mencionar brevemente a gramática, por ter indicação de uso em trabalhos futuros.

O **Léxico** é a lista de palavras/termos de alguma língua aceitos pelo sistema em questão. As entradas do léxico podem indicar a classe gramatical, o gênero da palavra e outras informações úteis ao sistema. A classe gramatical é útil para distinguir palavras que podem ser associadas a mais de uma classe gramatical (e.g., mata (verbo flexionado) e mata (substantivo)). Porém, o léxico nem sempre traz o significado das palavras – não sendo, assim, considerado um dicionário. Isso poderá dificultar a distinção entre palavras ambíguas: palavras com mais de um significado (e.g., *banco* (acento ou instituição financeira)).

Neste trabalho de mestrado, construímos um Léxico com base em um dicionário de 1913, que conta com informações sobre classe gramatical e flexões das palavras, porém sem o significado das entradas. Contudo, talvez a utilização dos significados das palavras pode ser útil em alguns contextos de correção, a fim de eliminar ambiguidades. Essa tarefa ficou indicada como trabalho futuro (ver seção 6.2).

A **Gramática** traz as regras que definem as cadeias de palavras válidas na LN sendo processada. Geralmente, os sistemas de PLN realizam apenas análise morfossintática, com regras que se baseiam na classe gramatical das palavras (e.g., subst+verbo (João chegou)). Porém existem também as gramáticas que analisam a função de cada termo da frase (sujeito, predicado, etc). Nosso trabalho não incorporou uma gramática, tendo focado apenas no tratamento morfológico de palavras e termos compostos resultantes do OCR. Contudo, a construção e utilização de tal gramática pode representar uma melhoria na análise do texto extraído. Essa extensão foi indicada como continuação desta pesquisa.

Como visto na Figura 3, as **fases de processamento**[14] dos sistemas de PLN dentro da abordagem simbólica podem ser divididas em: análise léxica/morfológica, processamento sintático, análise semântica, análise do discurso, e processamento pragmático. Como já dito, cada etapa dessas pode ser implementada utilizando-se técnicas/abordagens diferentes.

A fase de **Análise Léxica/Morfológica** inicia o processo de interpretação de LN, sendo a mais relevante para nosso trabalho. Por isso, sua apresentação será mais detalhada do que as etapas seguintes. Essa primeira fase analisa o texto no nível da palavra e dos seus morfemas (suas unidades menores). Seu objetivo é reconhecer as palavras do texto e analisar sua estrutura, a fim de possibilitar outras análises e processos mais sofisticados a seguir.

O primeiro passo é a *tokenização* (do inglês, *tokenization*), que objetiva separar o texto de entrada em palavras ou termos compostos, fornecendo como saída uma lista de tokens que também pode conter caracteres especiais encontrados no texto. As técnicas utilizadas geralmente consideram espaços e sinais de pontuação como separadores. Contudo, é necessário tratar alguns

[14] Fases de processamento PLN - https://algoscale.com/blog/a-guide-to-the-5-nlp-phases/

caracteres especiais, como o hífen, que também é usado para ligar duas palavras formando um termo composto (e.g., guarda-chuva).

Se necessário, as palavras do texto podem ser reduzidas à sua forma normalizada (*lematização*) ou ao seu radical (*stemming*), com base nos seus morfemas constituintes.

Segundo (ARAÚJO, [20-?]), “Morfemas são as menores unidades de significação que formam as palavras e são classificados em desinência, raiz, radical, afixo, tema e vogal temática.” Por exemplo, a palavras “relemos” é constituída pelos morfemas “re” (prefixo), “le” (raiz) e “mos” (desinência verbal). Mais detalhes sobre esse assunto serão vistos no Capítulo 3, seção 3.2.3.

A análise morfológica completa, que separa as palavras em todos os seus morfemas constituintes, é um pouco mais complexa de ser implementada, além de consumir mais tempo de computação. Assim, as operações morfológicas geralmente realizadas de modo automático objetivam apenas reduzir as palavras à sua forma normalizada pela retirada de sufixos. Apesar da simplicidade, essas operações podem ser muito úteis para várias atividades de PLN (como nos corretores ortográficos – Capítulo 3) e de Recuperação de Informação (a fim de facilitar a identificação de termos relevantes para responder a uma busca por documentos – Capítulo 4, seção 4.5). Vale frisar ainda que a retirada dos prefixos modifica a semântica (significado) das palavras (e.g., fazer, desfazer), podendo trazer consequências negativas em algumas aplicações do PLN e da Recuperação de Informação.

As operações comumente realizadas nessa etapa do PLN são:

- Lematização busca reduzir a palavra ao seu lema retirando as terminações flexionadas (que são morfemas), a fim de retornar a palavra como aparece nos dicionários; por exemplo, “meninos -> menino”; “casinha -> casa”. Esse processo pode ser realizado usando-se regras de flexão da língua, ou consultando um léxico. Essa segunda opção geralmente utiliza alguma medida de similaridade entre palavras para identificar no léxico o termo correspondente à palavra sendo processada.
- *Stemming* (termo em inglês) reduz palavras flexionadas ou derivadas a uma forma base, não necessariamente igual ao lema, que encontramos nos dicionários. Essa operação pode eliminar todos os

afixos (por exemplo, desfazendo -> faze - elimina o prefixo “des” e o sufixo “ndo”), ou apenas os sufixos (a opção mais adotada, pois não modifica a semântica do termo restante). Essa operação é mais simples de ser realizada automaticamente do que a lematização, pois depende apenas da aplicação de poucas regras de redução. Assim, muitos sistemas de PLN e de RI optam por esse tipo de redução.

Essas operações podem ser úteis na construção de corretores ortográficos (o foco principal do nosso trabalho). De modo simples, um corretor ortográfico pode verificar se uma dada palavra está correta apenas buscando essa palavra no Léxico. Contudo, o corretor precisa considerar as formas flexionadas da palavra, como plural, diminutivo e aumentativo, flexões verbais etc. Assim, um caminho seria reduzir a palavra ao seu radical ou lema, e só então realizar a busca no Léxico.

A etapa de sugestão de correção geralmente se baseia em uma função que calcula a distância entre a palavra de entrada e as palavras contidas no Léxico. O exemplo mais conhecido é a “Distância de Levenshtein” (YUJIAN, 2007), que será apresentada na seção 3.2.5. Por exemplo: “cazar ~ casar (d=1)”, “apartemeto ~apartamento (d=2)”. Essa distância é usada para sugerir ao usuário palavras que podem ser a versão correta da palavra digitada, por exemplo.

Os corretores ortográficos geralmente são baseados na *análise de palavras isoladas*, não considerando informação de contexto de ocorrência dos termos.

Por fim, é importante destacar o *processamento morfossintático,* que objetiva realizar a etiquetagem da classe gramatical dos termos da frase. Um exemplo de frase etiqueta é: $\text{Pedro}_{\text{subst}}$ $\text{comprou}_{\text{verbo}}$ um_{art} $\text{livro}_{\text{subs}}$. Essa etiquetagem geralmente é realizada a partir das palavras originais, sem redução ao seu radical, uma vez que os morfemas de flexão são úteis para determinar a classe gramatical da palavra.

A etiquetagem das palavras pode ser realizada apenas consultando-se um Léxico que informe a classe gramatical das palavras, adotando assim uma técnica baseada em *palavra isoladas*. Contudo, existem palavras homógrafas que pertencem a classes gramaticais distintas (e.g., “mata” pode ser um substantivo (floresta) ou o verbo matar flexionado). Assim, o ideal é utilizar um parser morfossintático (do inglês, *Parts-Of-Speech tagger - POS-tagger*), que considera o *contexto de ocorrência* das palavras para realizar a etiquetagem correta da classe gramatical de cada token (NGUYEN, 2022). *POS-taggers* podem ser construídos

utilizando-se qualquer das abordagens e técnicas da IA mencionadas na seção 2.3.1, porém atualmente a abordagem da AM tem prevalecido.

Essa fase de processamento é muito importante para algumas tarefas de PLN, tais como corretor gramatical, análise de sentimento baseada na polaridade de adjetivos e advérbios e muitas outras aplicações de PLN. Devido à sua importância para a interpretação de texto, geralmente esse processo é realizado como uma etapa separada da análise morfológica.

Essa etapa de processamento não foi implementada na versão atual do nosso trabalho por restrições de tempo. Porém, trata-se de uma informação útil para a melhoria dos nossos resultados, e será considerada como trabalho futuro de grande relevância (ver seção 6.2). As outras etapas de análise apresentadas a seguir não foram consideradas para inclusão no nosso trabalho, pois tratamos com notícias de jornais, e esses textos nem sempre seguem regras sintáticas e semânticas rigorosas. Ainda assim, as etapas serão brevemente apresentadas aqui, para dar uma visão geral da área.

As outras fases do PLN citadas acima não são consideradas relevantes para o pós-processamento de texto. Assim, a fim de manter o foco, não iremos apresentá-las nesse texto.

2.3.2.3 Ferramentas para PLN

Muitas das tarefas/etapas do PLN já dispõem de ferramentas/bibliotecas implementadas, e muitas delas podem ser usadas livremente. A seguir citamos algumas ferramentas que são usadas com maior frequência.

Tesauros (dicionários de sinônimos): este projeto de mestrado poderia se beneficiar muito de um dicionário de sinônimos automático na etapa de indexação e busca dos jornais digitalizados. Esses termos iriam auxiliar na recuperação de mais documentos relevantes para as consultas. Contudo, por falta de tempo, esta etapa foi sugerida como trabalho futuro. Citamos abaixo dois tesauros online muito usados.

- **WordNet[15]** – oferece uma rede de palavras com mais recursos do que apenas sinônimos (incluindo, antônimos, hiperônimos, hipônimos etc). O WordNet original foi construído para a língua inglesa. Porém, existe também uma versão reduzida para a língua portuguesa[16].
- **BabelNet[17]** – é um tesauro que também oferece tradução automática.

NLTK - Natural Language Toolkit[18] é uma plataforma para construção de programas em Python para PLN em inglês. O NLTK oferece bibliotecas para Tokenização (análise léxica), Stemming e lematização, POS-tagging, Parsing sintático e Raciocínio semântico, Classificação de texto, entre outras. Existe ainda um site com exemplos para processamento de Português[19].

Stanford CoreNLP[20] oferece várias ferramentas para PLN na língua inglesa. Como exemplo, citamos: Tokenização (análise léxica), Separação de sentenças, Stemming e lematização, POS-tagging e Parsing sintático, Resolução de correferência, Reconhecimento de entidades nomeadas etc. Destacamos por fim o Framework **RegEx[21]** para definir expressões regulares para extração de informação textual, e a Extração de Informação aberta, livre de domínio.

Além dessas, existem ainda muitas outras plataformas e ferramentas para o PLN, porém citamos aqui as mais utilizadas no momento.

2.3.3 Outras Abordagens e Técnicas de Pós-processamento de Texto Automático

Assim como no caso do PLN, as técnicas apresentadas aqui nesta seção também podem ser categorizadas entre baseadas em palavras isoladas ou abordagens dependentes do contexto. Porém, utilizam outras técnicas diferentes das já mencionadas na seção anterior.

---

[15] WordNet - https://wordnet.princeton.edu/
[16] WordNet em português - http://wordnet.pt
[17] BabelNet - http://babelnet.org/
[18] NLTK Toolkit - http://www.nltk.org/
[19] NLTK Toolkit em português - https://www.nltk.org/howto/portuguese_en.html
[20] Stanford CoreNLP - http://stanfordnlp.github.io/CoreNLP/
[21] RegEx - https://stanfordnlp.github.io/CoreNLP/regexner.html

2.3.3.1 Abordagem baseada em palavras isoladas

As técnicas de correção baseadas em Léxicos desempenham um papel central no pós-processamento de OCR para melhorar a precisão do texto reconhecido (NGUYEN, 2022). Essas abordagens baseiam-se na utilização de dicionários, léxicos e informações no nível de palavra para detectar e corrigir erros (ver seção 2.3.2). Os verificadores ortográficos podem utilizar técnicas como medidas de similaridade entre *strings* ou cálculos de distância de edição (por exemplo, Levenshtein ou Damerau-Levenshtein) para classificar e sugerir candidatos a correção. Abordagens lexicais priorizam alta cobertura e dicionários específicos de domínio para garantir correções precisas.

Abordagens lexicais são fáceis de aplicar, no entanto, a falta de léxicos de alta cobertura é o problema mais desafiador dessa abordagem, especialmente com documentos históricos, cujos textos não seguem a ortografia padrão. Dessa forma, no tratamento de documentos históricos é crucial garantir um léxico alta cobertura e específico ao vocabulário próprio do período considerado. Esta abordagem foi a opção adotada neste trabalho de mestrado, como já mencionado.

Além da correção baseada em Léxicos, outra técnica usada na abordagem baseada em palavras isoladas é a mesclagem de saídas de OCRs (NGUYEN, 2022). Essa técnica combina várias saídas de OCR e geralmente inclui três etapas. Inicialmente, os textos são extraídos a partir de várias ferramentas de OCR diferentes com base na mesma imagem de entrada, ou utilizando-se da mesma ferramenta de OCR em diferentes versões digitais (imagens) do documento original. A seguir, os algoritmos de mesclagem são aplicados para alinhar os resultados do OCR. Por fim, diferentes métodos de decisão são explorados para selecionar o resultado final.

Também existem as abordagens que se concentram em modelos de erros para corrigir palavras de OCR incorretas. Os modelos de erro são criados analisando um grande corpus de texto corretamente reconhecido e comparando-o com a saída de OCR correspondente. Ao comparar os dois, os modelos de erro podem identificar padrões e características de erros cometidos pelo sistema OCR. Os primeiros modelos de erros utilizados dependiam de medidas de distância, como distância de *Levenshtein*, ponderando igualmente todas as edições ocorridas na palavra. Recentemente, diversas outras técnicas têm sido propostas como modelos

generativos probabilísticos e modelo oculto de Markov de forma a identificar padrões e características de erros cometidos pelo sistema OCR.

2.3.3.2 Abordagem dependente do contexto

Os métodos de correção pós-OCR baseados em contexto empregam diferentes abordagens para aprimorar a precisão do reconhecimento óptico de caracteres, levando em consideração o contexto linguístico.

Algoritmos baseados em aprendizagem de máquina como árvores de decisão, florestas aleatórias (*Random Forest*) ou máquinas de vetor de suporte (*Support Vector Machine* - SVM), são usados para criar modelos de padrões e relacionamentos entre características de entrada e correções correspondentes (KARTHIKEYAN, 2021). As características usadas podem incluir informações lexicais (como resultados de busca em dicionários), medidas de similaridade entre palavras candidatas, pontuações de confiança do software de OCR e vários atributos linguísticos ou estatísticos. Ao treinar os algoritmos com conjuntos de dados rotulados, os modelos de AM baseados em características podem aprender a prever a correção mais provável para erros de OCR com base nas características fornecidas.

Outra estratégia de correção baseada em contexto envolve Modelos de linguagem. Esses métodos exploram modelos de linguagem para gerar e classificar os candidatos a correção. Esses modelos podem ser divididos em dois grupos principais, de acordo com a abordagem usada para gerar o modelo: os modelos estatísticos e os baseados em redes neurais (NGUYEN, 2022).

Nos modelos de linguagem estatísticos, estima-se a distribuição de probabilidade de sequências de palavras com base na análise estatística de grandes *corpora* de texto. Esses modelos utilizam técnicas como modelagem baseada em *n-grams*, que capturam as frequências e probabilidades de sequências de palavras de comprimentos variados. Ao considerar a probabilidade de combinações de palavras específicas, os modelos estatísticos podem aprimorar o processo de correção, sugerindo alternativas mais plausíveis para resultados de OCR errôneos ou incertos (NGUYEN, 2022).

Já os modelos de linguagem baseados em Redes Neurais (*Neural Networks* – NN) oferecem uma abordagem poderosa para o pós-processamento de OCR,

usufruindo dos avanços nas técnicas de *deep learning*. Esses modelos usualmente empregam redes neurais recorrentes (RNNs) para capturar padrões complexos e dependências de linguagem (NGUYEN, 2022).

Outra técnica muito utilizada na correção de textos baseia-se em modelos de linguagem baseados em tópicos (BLEI, D., NG, A.; JORDAN, M., 2001) para melhor a qualidade da saída de OCR combinando um modelo de erro e um modelo de linguagem de n-gram das palavras (NGUYEN, 2022).

Os modelos de linguagem baseados em tópicos são um tipo especializado de modelo de linguagem usado no pós-processamento de OCR para melhorar a precisão e a compreensão contextual do texto reconhecido (NGUYEN, 2022). Ao contrário dos modelos de linguagem tradicionais, que se concentram apenas nas frequências de palavras e nas probabilidades de n-gram, os modelos de linguagem baseados em tópicos incorporam informações de tópicos para aprimorar o processo de modelagem de linguagem. Esses modelos utilizam técnicas como alocação Latent Dirichlet (LDA) (MO; KONTONATSIOS; ANANIADOU, 2015) ou modelagem de tópicos baseada em redes neurais artificiais para identificar tópicos ou temas subjacentes em um determinado corpus de texto. Ao incorporar informações de tópico, os modelos de linguagem podem capturar as relações semânticas e coocorrências de palavras, melhorando a compreensão baseada no contexto do texto reconhecido. Isso permite previsão mais precisa de palavras e sugestões de correção durante o pós-processamento de OCR, principalmente ao lidar com palavras ambíguas ou fora do vocabulário. Os modelos de linguagem baseados em tópicos oferecem uma ferramenta poderosa para aprimorar os resultados de OCR, aproveitando as informações contextuais derivadas dos tópicos presentes nos dados de treinamento.

2.4 FERRAMENTAS DE EXTRAÇÃO DE TEXTO COM OCR

Existe uma grande variedade de ferramentas disponíveis para realização do processo de OCR, tanto comerciais quanto *open source* (*FUGGETTA*,2003). Atualmente a difusão desse tipo de ferramenta é tal que muitos softwares nativos em câmeras de smartphones possuem um módulo de OCR integrado. Todavia, ao lidar com documentos históricos digitalizados, maiores são os desafios para obter precisão e eficiência do OCR. Assim, é importante investigar bem as diferentes

ferramentas disponíveis e suas particularidades, a fim de escolher a que ofereça melhor desempenho no contexto do nosso trabalho. Descrevemos a seguir algumas das ferramentas de OCR mais populares.

OCRopy[22] é um framework que oferece ferramentas em Python para otimizar várias etapas no fluxo de trabalho de OCR. Ele inclui recursos como binarização de imagens, segmentação de páginas, criação de texto GT, treinamento de modelos de OCR, reconhecimento de caracteres, avaliação de resultados e exportação para o formato hOCR. As versões iniciais do OCRopy apresentaram problemas de desempenho, mas a versão 3 resolveu essas questões utilizando a biblioteca *PyTorch* para suporte a unidades de processamento gráfico (*Graphics Processing Unit* - GPUs).

O **ABBYY FineReader**[23] é um software comercial de OCR reconhecido como líder. Ele oferece modelos pré-treinados para diversos idiomas, possui uma ferramenta de segmentação poderosa e recursos de pós-processamento semiautomático. Estudos mostraram que, para documentos históricos, é mais vantajoso treinar modelos personalizados usando ferramentas de código aberto, como Calamari e Tesseract (descrito a seguir), em vez de utilizar os modelos pré-treinados do ABBYY (DROBAC, 2020).

Calamari[24] é uma ferramenta avançada de OCR baseada no TensorFlow[25]. Ela foi desenvolvida para treinar, reconhecer e avaliar modelos de OCR inspirada em resultados experimentais anteriores. Calamari utiliza para reconhecimento de caracteres uma abordagem híbrida de redes convolucionais profundas (*Convolutional Neural Networks* - CNN) e redes neurais *Long short-term memory* (LSTM) para *deep learning*.

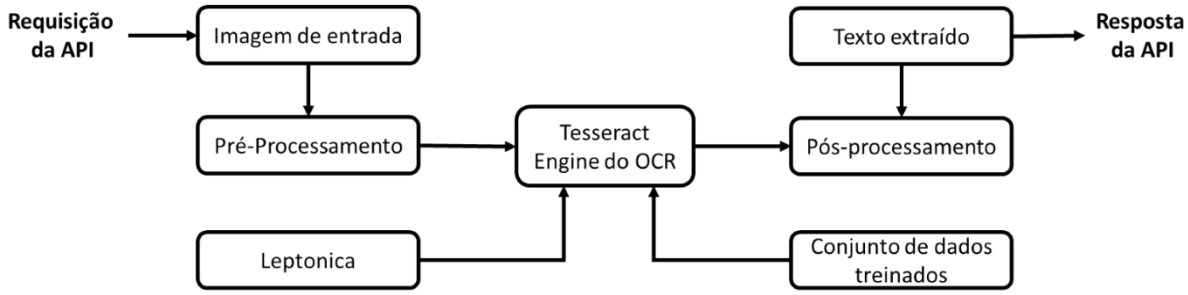
Por fim, vamos apresentar o **Tesseract**[26], que é a ferramenta utilizada neste trabalho de mestrado. Tesseract é uma ferramenta de OCR que possui um amplo uso no mercado e na academia. Inicialmente, essa ferramenta foi desenvolvida pela HP como um software proprietário em 1985, mas foi disponibilizada como código aberto em 2005. Após sua aquisição pelo Google em 2006, ocorreram melhorias

---

[22] OCRopy - https://github.com/ocropus/ocropy
[23] ABBY FineReader - https://pdf.abbyy.com/
[24] Calamari - https://github.com/Calamari-OCR/calamari
[25] TensorFlow - https://github.com/tensorflow
[26] Tesseract - https://tesseract-ocr.github.io/tessdoc/

significativas nas versões 3 e 4, introduzindo recursos avançados de reconhecimento de caracteres. A versão mais recente, Tesseract 5, lançada em 2021, apresenta melhorias no treinamento e reconhecimento de caracteres, proporcionando maior velocidade e consumo de memória reduzido. Além disso, essa versão oferece suporte a arquiteturas de redes neurais profundas, e disponibiliza opções adicionais de binarização[27]. A Figura *4* resume o fluxo de processos do Tesseract.

Figura 4 - Processos presentes na etapa de extração do texto com Tesseract



Fonte: Adaptado de Nanonets[28].

O **Tesseract** inicia o processo de reconhecimento óptico de caracteres realizando o pré-processamento da imagem de entrada. Nessa etapa, são aplicadas técnicas como binarização, que converte a imagem em preto e branco, segmentação para identificar regiões de texto e filtragem para remover ruídos e melhorar a qualidade da imagem. Conforme discutido na seção 2.1, essas técnicas são essenciais para tornar os caracteres mais distintos e facilitar o processo de reconhecimento. O Tesseract realiza o pré-processamento das imagens através da Leptonica[29] que é uma biblioteca de código aberto, contendo software amplamente útil para aplicações de processamento de imagens e análise de imagens.

Após o pré-processamento, o OCR segmenta[30] a imagem em regiões que contêm texto. Essa fase é fundamental para identificar corretamente as áreas relevantes a serem reconhecidas. O algoritmo de segmentação utiliza informações como espaçamento entre palavras, tamanhos de fonte e orientação dos caracteres

---

[27] Tesseract - https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html
[28] Nanontes - https://nanonets.com/blog/ocr-with-tesseract
[29] Leptonica - http://www.leptonica.org/
[30] Segmentação da iamgem - https://tesseract-ocr.github.io/tessdoc/Command-Line-Usage.html

para determinar os limites das regiões de texto. Uma segmentação precisa garantir que apenas as áreas relevantes sejam processadas no reconhecimento de caracteres.

O reconhecimento de caracteres é realizado pelo Engine, sendo a etapa central do Tesseract. Utilizando técnicas de aprendizado de máquina, o algoritmo analisa cada região segmentada e realiza a identificação e classificação dos caracteres individuais presentes. Ele utiliza redes neurais convolucionais treinadas com grandes conjuntos de dados para executar essa tarefa. Os modelos pré-treinados para diferentes idiomas permitem que o OCR seja aplicado em várias línguas, aumentando sua versatilidade.

Após o reconhecimento dos caracteres, o Tesseract OCR realiza uma etapa de pós-processamento interno, na qual os caracteres reconhecidos são combinados para formar palavras e frases. O algoritmo implementa técnicas específicas para esse processo, incluindo o uso de dicionários da língua em questão. Esses dicionários auxiliam na reconstrução correta das palavras, aumentando a precisão do texto extraído. O Tesseract possui sua própria abordagem de pós-processamento interno, que pode variar entre diferentes versões e configurações da ferramenta.

O Tesseract foi escolhido para ser aplicada no nosso trabalho por ter apresentado bons resultados de extração e por sua facilidade de uso, quando comparada com outras opções de livre uso (ver Capítulo 4).

2.5 CONSIDERAÇÕES FINAIS

Este capítulo descreveu brevemente as etapas de processamento realizadas para extração de texto a partir de imagens. Demos mais ênfase às técnicas e abordagens adotadas na solução desenvolvida neste trabalho de mestrado: pós-processamento de texto baseado na abordagem léxica.

O Capítulo 3, a seguir, trata de corretores ortográficos automáticos, que é o tema central do nosso trabalho. Além de descrever esses sistemas, esse capítulo também irá apresentar alguns trabalhos relacionados ao nosso tema, destacando suas características principais.

3 CORRETORES ORTOGRÁFICOS AUTOMÁTICOS PARA PÓS-PROCESSAMENTO DE TEXTO

Este capítulo é dedicado a apresentar os corretores ortográficos em mais detalhe, por se tratar de um tema central para o nosso trabalho de mestrado. A seção 3.1 traz uma visão geral sobre o tema, apresentando um breve histórico do seu desenvolvimento. A seguir, a seção 3.2 trata do uso de OCR para extração de textos a partir de documentos históricos.

3.1 PROBLEMÁTICAS DA APLICAÇÃO DE OCR EM DOCUMENTOS HISTÓRICOS

A aplicação de OCR em documentos e textos históricos revela uma série de desafios que não são tão evidentes ao lidar com documentos contemporâneos. Alguns desses problemas podem ser abordados por meio de melhorias ou ajustes no software de OCR, como a adição de recursos de reconhecimento para caracteres especiais utilizados nesses documentos. Outros problemas podem ser tratados por meio de intervenção na etapa de correção pós-OCR. Essas dificuldades podem ser classificadas em duas categorias principais: problemas gráficos e problemas lexicais (HAUSER, 2007).

A categoria de *problemas gráficos* aborda questões e peculiaridades relacionadas a documentos e textos históricos, tanto em nível de documento quanto de caracteres. Isso inclui o estado de conservação do documento e a representação gráfica dos caracteres (tipografia). Existem ainda elementos gráficos que fazem parte do layout do documento, como bordas e linhas separadoras (como é o caso dos jornais periódicos), bem como figuras e imagens fotográficas.

Já os *problemas lexicais* referem-se a questões relacionadas ao vocabulário de palavras encontradas em textos históricos. Devido às variações nas convenções ortográficas ao longo do tempo, as palavras de documentos históricos podem ter grafias diferentes em comparação com o seu uso contemporâneo (e.g., *pharmácia - farmácia*). Isso inclui variações ortográficas, mudanças morfológicas, vocabulário histórico, palavras compostas, estrangeirismos, caracteres especiais e abreviações.

Seja qual for a categoria dos problemas presentes, a qualidade final dos resultados de OCR depende fortemente da eficiência do tratamento aplicado. Os

problemas gráficos por serem baseados nas imagens sendo analisadas, podem ser minorados com a aplicação de técnicas refinadas de processamento de imagens. Algumas dessas técnicas já foram mencionadas no capítulo anterior (seção 2.1). Como este não é o foco do nosso trabalho, não vamos mais aprofundar esse tema aqui. Para maiores informações, ver (HAUSER, 2007).

Como já visto, o foco do nosso trabalho está na correção de *problemas lexicais*, sendo esse o assunto principal deste capítulo. Esses problemas são geralmente tratados por corretores ortográficos, que podem ser implementados com técnicas variadas. Assim, as seções a seguir irão apresentar as técnicas e soluções mais usadas por corretores ortográficos automáticos, apresentando também algumas ferramentas de livre uso.

3.2 CORRETORES ORTOGRÁFICOS

Corretores ortográficos são uma aplicação já bem consolidada e muito difundida na computação, tendo por objetivo verificar e corrigir a ortografia das palavras de um texto (ARAUJO, 2021). Trata-se de uma ferramenta de apoio que valida a ortografia de cada palavra no texto, independentemente do seu contexto ou semântica.

Os corretores ortográficos representam uma evolução técnica dos verificadores ortográficos, podendo apenas oferecer um conjunto de sugestões similares à palavra com erros, ou mesmo substituir automaticamente palavras digitadas pelo usuário. Eles podem ser utilizados como aplicações independentes ou, mais comumente, como módulos integrados em sistemas como editores de texto, editores de correio eletrônico, navegadores de internet, programas de conversação e outras aplicações que processam texto tanto para o usuário final quanto como ferramenta intermediária em sistemas de processamento de linguagem natural.

O Capítulo 2 apresentou duas possíveis abordagens de construção de corretores ortográficos (seção 2.3.3): a abordagem baseada em palavras isoladas e a abordagem dependente de contexto. Dentro de cada abordagens, é possível utilizar técnicas variadas de IA e PLN, como já apresentado. Este capítulo aprofunda a abordagem baseada em palavras isoladas com o uso de léxicos, por ser este o foco do nosso trabalho. Outras abordagens já foram discutidas no Capítulo 2.

Como já mencionado, corretores ortográficos lexicais geralmente se baseiam em dicionários/léxicos da língua em questão. Um dicionário tradicional de uma língua é uma obra que estabelece a ortografia e os significados atribuídos a cada palavra. Além disso, ele pode conter a representação fonética da palavra, bem como informações sobre sua origem e sua classificação gramatical. Ao longo do tempo, os dicionários têm desempenhado um papel fundamental na padronização linguística (DE ALMEIDA, 2003).

No campo da Computação, geralmente um dicionário é tido como um arquivo que contém uma lista (possivelmente limitada) de termos associados a informações específicas. Nesse contexto, os *Dicionários* são geralmente chamados de *Léxicos* ou *Vocabulários*, uma vez que não contêm todas as informações que os dicionários apresentam (como mencionado acima). Assim, neste texto, usaremos essas três denominações como sendo sinônimas.

Por fim, é importante observar que os corretores ortográficos lexicais podem operar em duas direções opostas. Uma opção é utilizar regras de derivação morfológica para detectar os afixos de uma palavra dada e removê-los, preservando a raiz da palavra. Em seguida, verifica-se se a raiz (ou radical) está presente no dicionário. A outra opção, na direção oposta, busca gerar o vocabulário (quase) completo de uma língua pela afixação das raízes das palavras. A seguir, as palavras do texto a ser corrigido são comparadas às entradas desse vocabulário expandido. Quando a palavra é encontrada com a mesma grafia do vocabulário, será considerada correta.

As seções a seguir apresentam conceitos e técnicas relacionados aos corretores ortográficos lexicais. A seção 3.2.1 traz um breve histórico do desenvolvimento de corretores ortográficos automáticos. As três seções seguintes detalham conceitos relacionados aos dicionários, à formação das palavras e à análise morfológica. A seguir, a seção 3.2.5 trata de medidas usadas para avaliar a similaridade entre duas cadeias de caracteres, sendo muito usadas na busca de correspondências entre as palavras do texto e o dicionário. Por fim, a seção 3.2.6 apresenta a ferramenta para correção ortográfica *PyEnchant*, usada neste trabalho de mestrado.

3.2.1 Breve Histórico e Visão Geral

Antes mesmo da criação dos corretores ortográficos automáticos, algoritmos de busca tinham o desafio de encontrar registros em bancos de dados considerando erros ortográficos. Como exemplo, citamos o *Soundex*, um algoritmo precursor tinha como propósito simplificar a busca por registros que tenham representação fonética similar, quer representem nomes ou palavras, quer estejam grafados correta ou erroneamente. (RUSSELL, 1918, 1922; ODELL, 1956; KNUTH, 1973).

O desenvolvimento de ferramentas específicas para correção ortográfica começou de fato com o *Spell*, criado por Stephen Curtis Johnson em 1975 e aperfeiçoado por McIlroy em 1982 como parte do sistema UNIX (GORIN, 1974). A ideia inicial do Spell era simples: verificar se as palavras de um texto estavam presentes em uma lista prévia de palavras corretas, ou seja, em um dicionário. Johnson utilizou uma lista de palavras do *Corpus Brown* para implementar o Spell (KUČERA; FRANCIS, 1967).

Contudo, a simples comparação entre listas, como um dicionário e o corpus em análise, não é suficiente para abranger diversos casos de derivação ou afixação de termos. Esses casos podem não estar presentes no dicionário devido às múltiplas possibilidades existentes. Para lidar com essas situações, e assim aumentar a abrangência do dicionário, Johnson optou por remover sufixos e prefixos, utilizando regras simples (KUČERA; FRANCIS, 1967). Basicamente, se uma determinada palavra, como "civilidade", não estiver no dicionário, a remoção do sufixo "-idade" resulta na palavra "civil", o que aumenta a possibilidade de encontrá-la no dicionário. No entanto, essa abordagem pode levar à aceitação de alguns erros. É importante observar que, até o momento, a funcionalidade do Spell estava limitada a localizar apenas erros ortográficos.

Neste sentido, Church e Gale (1991) propuseram uma ferramenta chamada *Correct*, que complementa o *Spell*. O *Correct* gera uma lista ordenada por probabilidade de sugestões para corrigir possíveis erros de digitação, utilizando um modelo de canal ruidoso baseado no trabalho de Kernighan, Church e Gale (1990) e Shannon (1948). Esse modelo pressupõe que, em um processo de comunicação, uma sequência recebida é o resultado da transmissão de uma mensagem gerada pela fonte através de um canal suscetível a ruídos (KERNIGHAN; CHURCH; GALE, 1990; SHANNON, 1948).

Em 1971, também integrado ao sistema UNIX, foi desenvolvido o *Ispell*, com o objetivo de ser um corretor interativo. Ele sugere correções para palavras escritas incorretamente considerando uma distância de edição de 1 (um), conforme a métrica de Damerau-Levenshtein (YUJIAN, 2007), conhecida como estratégia de quase erro (GORIN; WILLISSON; KUENNING, 1971). O *Ispell* também utiliza uma lista de regras de afixos para abranger uma variedade maior de palavras, evitando a necessidade de um dicionário extenso que inclua todas as variações resultantes dos processos de afixação.

A partir de 1998, Kevin Atkinson desenvolveu o *GNU Aspell* como parte do Projeto GNU, como uma alternativa ao *Ispell* (ATKINSON, 1998). O *Aspell* oferece suporte a UTF-8[31] e a vários dicionários. Ele utiliza o algoritmo Metaphone (PHILIPS, 1990, 2000) e a estratégia de quase erro do *Ispell*. O *Aspell* converte as palavras em equivalentes sonoros usando “metaphones” (que não correspondem necessariamente a fonemas), com o objetivo de encontrar palavras que soam semelhantes quando faladas. A ideia aqui é cobrir variações de pronúncia na língua inglesa. Ele também busca candidatos com base na edição dos caracteres em sequência, semelhante ao *Ispell*. Todos os candidatos gerados recebem um valor probabilístico, que é usado para ordenar as sugestões.

Em 2011, foi criado o *MySpell*[32], seguindo uma abordagem semelhante ao *Ispell* e utilizando a lista de palavras criada por Kevin Atkinson. O *MySpell* incorporou sugestões provenientes de tabelas de substituição e mecanismos de pontuação de n-gramas. Ele foi incorporado ao OpenOffice.org e deu origem ao *Hunspell*, que é usado atualmente no Mozilla Firefox, Mozilla Thunderbird, Google Chrome, macOS, entre outros. O *Hunspell* também incorporou análise morfológica e informações de dicionários de pronúncia (ARAUJO; BENEVIDES; SANSÃO, 2021).

Por fim, destacamos o *PyEnchant*[33], que é uma implementação Python da biblioteca de verificação e correção ortográfica *Enchant*[34], originalmente escrita na linguagem C. Ela foi desenvolvida como parte do projeto AbiWord[35], e é mantido por Ryan Kelly. Ele oferece a flexibilidade de carregar vários back-ends

[31] UTF-8 - https://www.utf8-chartable.de/
[32] Myspell - https://github.com/topics/myspell
[33] PyEnchant - https://github.com/pyenchant
[34] Enchant - https://github.com/pyenchant/enchant
[35] AbiWord - https://abiword.en.uptodown.com/windows

simultaneamente, chamados na ferramenta de provedores, incluindo *Aspell*, *Hunspell* e *AppleSpell*. O *PyEnchant* garante que os usuários possam utilizar verificadores ortográficos nativos em várias plataformas, como Mac OS X e Microsoft Office. Além disso, ele fornece funcionalidades como carregar dicionários, adicionar palavras personalizadas, verificar se uma palavra está escrita corretamente e sugerir correções ortográficas para palavras com erros ortográficos (KELLY, 2021). Neste trabalho realizamos a verificação e correção ortográfica automática a partir da biblioteca *PyEnchant* utilizando o provedor *Hunspell* (para mais detalhes, ver Seção 3.2.6).

3.2.2 Dicionários/Léxicos/Vocabulários

Como já mencionado, na Computação, um dicionário (léxico ou vocabulário) é um arquivo que contém uma lista (possivelmente limitada) de termos associados a informações específicas referentes a cada palavra. Essas informações, por vezes opcionais, incluem: representação fonética, classe gramatical, gênero, significados atribuídos às palavras, entre outras. Vale ainda frisar que, para facilitar a busca por palavras, o léxico é organizado em ordem alfabética (DE ALMEIDA, 2003).

Um dicionário para correção ortográfica, em sua forma mais básica, é uma *lista de termos* (e suas informações associadas) cuja *grafia* é considerada correta. A versão inicial do corretor *Aspell* usa esse formato, que é claramente insuficiente para representar eficientemente todas as variantes da língua portuguesa.

No entanto, existem corretores ortográficos que lidam melhor com os problemas relacionados às características das línguas utilizando um dicionário mais complexo, enriquecido com informações complementares. Neste sentido, algumas soluções alternativas foram implementadas, tais como: regras de afixação (seção 3.2.3), mapas de substituição fonética e gráfica, marcação morfológica (seção 3.2.4) e semântica, metainformação e mecanismos de processamento de exceções (VILELA, 2009).

Esses mecanismos mais complexos introduziram uma dinâmica adicional aos corretores ortográficos que resolveu problemas de eficiência na verificação ortográfica de determinadas línguas. Contudo, isso resultou em dicionários estruturalmente mais complexos e, portanto, mais difíceis de manter.

A seção a seguir apresenta brevemente alguns conceitos sobre a afixação na língua portuguesa, sendo esse um mecanismo de extrema importância nos corretores ortográficos. Como já mencionado na introdução da seção 3.2, esse mecanismo pode ser usado tanto para expandir o dicionário inicial como para analisar as palavras do texto a ser corrigido. De fato, o corretor ortográfico utilizado neste trabalho utiliza esse mecanismo para corrigir automaticamente o texto de entrada (ver Capítulo 4).

3.2.3 Afixos/Afixação na língua portuguesa

Afixos são morfemas que se unem à raiz de uma palavra, seja no início (prefixo), no meio (infixo) ou no final (sufixo) da palavra, criando uma nova palavra (e.g., flor – florista) e, em alguns casos, modificando seu significado (e.g., fazer – desfazer). Segundo (ARAÚJO, [20-?]), “Morfemas são as menores unidades de significação que formam as palavras e são classificados em desinência, raiz, radical, afixo, tema e vogal temática.”

Os prefixos e sufixos são facilmente identificados, pois estão nas extremidades das palavras e possuem formas mais regulares. Exemplos de prefixos e sufixos são "desfazer", "desconfortável", "plantação".

Por outro lado, os infixos, que se intercalam entre a raiz e o sufixo, possuem formas mais variáveis, mais difíceis de serem detectadas computacionalmente (por exemplo, "pedregoso” (infixo eg), “cafezal” (infixo z)). Por isso, geralmente não são contemplados nas regras de afixação dos dicionários computacionais.

No contexto dos corretores ortográficos lexicais, é importante distinguir a *raiz* do *radical* de uma palavra[36]. O Radical pode ser definido como o elemento irredutível comum às palavras de uma mesma família” (e.g., desfazer, infeliz). Já a Raiz pode ser definida como a parte básica, a origem de uma palavra. Ao contrário do radical, a raiz não possui nenhum prefixo ou sufixo em sua composição (e.g., desfazer, infeliz).

Por fim, vale ainda definir o termo *lema*, que se refere à forma canônica de uma palavra – por exemplo, “gato” é o lema de “gatos, gata e gatas”. É o lema da

[36] Raiz e Radical - http://cursocertificado.com.br/estrutura-e-formacao-de-palavras/

palavra que encabeça as entradas nos dicionários de uso comum. Porém os dicionários computacionais para uso em corretores ortográficos geralmente trabalham com a raiz ou o radical das palavras.

Os processos de afixação, tanto derivacional quanto flexional, contribuem para a expansão do vocabulário. A partir da raiz das palavras, é possível “gerar” suas variantes pela afixação, obtendo assim a maioria das palavras da língua em questão. Essa é uma forma de aumentar virtualmente a quantidade de palavras em um dicionário computacional (ARAUJO, 2021).

Como já foi mencionado anteriormente, o mecanismo de afixação é central para corretores automáticos lexicais, a fim de aumentar a precisão e a cobertura das correções realizadas.

3.2.4 Análise Morfológica

A seção 2.3.2.2 do Capítulo 2 apresentou brevemente a etapa de análise morfológica do texto dentro da abordagem simbólica do PLN, com foco nas operações realizadas para reduzir as palavras ao seu radical pela retirada de sufixos. Vimos ainda um pouco sobre morfossintaxe. Esta seção traz alguns conceitos Linguísticos sobre Morfologia, com o objetivo de esclarecer melhor seu uso pelos corretores ortográficos lexicais.

Podemos definir a Morfologia como sendo o ramo da Linguística que estuda os padrões de formação interna das palavras a partir dos morfemas da língua (seção 3.2.3), modelando o conhecimento dos falantes (MEDEIROS, 1995).

Muitas aplicações computacionais que envolvem análise textual requerem um processo automático capaz de analisar as palavras lexicalmente para reconhecer suas características gramaticais e também identificar sua existência no léxico em uso. Um *analisador morfológico* é uma ferramenta que determina um conjunto de possíveis características de uma dada palavra, como seus morfemas constituintes, classe gramatical, gênero, número etc. Voltando à seção 2.3.2.2, esse analisador morfológico incluiria tanto a análise léxica como morfossintática.

Um *dicionário morfológico* é aquele que contém um léxico com correspondências entre o lema (forma canônica da palavra, termo base) e as formas lexicais da palavra. Ele consiste no lema, que não possui flexão, e sua classe gramatical, seguida das informações correspondentes à formação das desinências

pelas quais as formas flexionadas são obtidas (GARRIDO-ALENDA; FORCADA, 2007). Devido ao grande número de desinências que alguns lemas podem conter (principalmente verbais), e para evitar repetir informações morfológicas desnecessariamente, os lemas são associados a *regras de afixação*, mantendo a forma de composição das desinências, juntamente com suas classes gramaticais, em um arquivo separado (DE ALMEIDA, 2003).

Essa solução também é observada nos dicionários utilizados em corretores ortográficos. Como mencionado na seção anterior, a partir da raiz (ou, neste caso, do lema) das palavras, é possível “gerar” suas variantes utilizando-se regras de afixação. Com exceção do *Aspell 0.5*, todos os dicionários de corretores ortográficos utilizam um arquivo com um conjunto de regras de afixação, geralmente com a extensão *.aff*, juntamente com o arquivo do dicionário, com a extensão *.dic*. Um exemplo de entrada no dicionário utilizado neste trabalho é apresentado na seção 3.2.6.1.

As palavras são reduzidas à sua forma raiz (ou lema) e são atribuídos identificadores das regras de afixação para que as palavras possam ser regeneradas. Existem diversas razões para preferir um dicionário que possua essas regras (DE ALMEIDA, 2003). Primeiramente, o uso dessas regras diminui consideravelmente o tamanho do dicionário. Ao armazenar apenas as formas raiz das palavras, evita-se a necessidade de incluir todas as formas flexionadas, resultando em uma base de dados mais compacta.

Além disso, as regras permitem estabelecer-se uma relação entre a classificação morfológica dos lemas (formas raiz) e as palavras flexionadas. Isso contribui para uma melhor organização e compreensão das informações contidas no dicionário. Por fim, ao armazenar apenas os lemas das palavras, o dicionário se torna mais legível e fácil de usar. Elimina-se o excesso de informações derivadas das flexões, concentrando-se no elemento central que define o significado e a estrutura das palavras.

3.2.5 Distância mínima de edição

A *distância mínima de edição*[37] é uma técnica usada na Ciência da Computação, na Linguística e na Estatística e em Matemática para avaliar a diferença entre duas *cadeias* (sequências) de *tokens* quaisquer. De forma geral, essas medidas avaliam quatro categorias de edição de *tokens*: adição, remoção, substituição e transposição de caracteres adjacentes. Assim, de modo simples, podemos dizer que a distância de edição pode ser medida computando-se o número de "operações" necessárias para transformar uma cadeia na outra.

Os *tokens* das cadeias podem ser *palavras* (medindo-se então as diferenças no nível das letras), *frases* (medindo-se a diferença no nível de palavras), ou cadeias (alfa) numéricas quaisquer. Como estamos tratando aqui de erros ortográficos, nossas cadeias serão sempre *palavras*, e os *tokens* serão *letras* ou o *hífen*, usado para separar termos compostos (e.g., guarda-chuva).

Existem várias medidas comumente utilizadas para computar a distância entre cadeias, algumas que consideram a ordem de ocorrência dos caracteres e outras que computam apenas as diferenças entre os caracteres, independentemente da sua ordem de ocorrência. Algumas das medidas mais usadas são (ARAUJO; BENEVIDES, SANSÃO, 2021): distância de Levenshtein (que computa inserção, exclusão e substituição de tokens nas duas sequências), distância de Damerau-Levenshtein (que inclui também a transposição de tokens adjacentes), distância de Hamming (que analisa apenas substituições), subsequência comum mais longa (do inglês, *longest commom sequence* – LCS, que computa o tamanho da subsequência mais longa de caracteres iguais nas duas cadeias) e as distâncias de Jaro e Jaro-Winkler (que medem a similaridade entre duas cadeias considerando também a ordem de ocorrência dos tokens).

Segundo Damerau (1964), as quatro categorias de edição básicas (adição, remoção, substituição e transposição de caracteres adjacentes) abrangem cerca de 80% dos erros ortográficos, embora outros estudos apresentem percentuais diferentes. Por exemplo, Mitton (1987) observou que 69% dos erros envolvem apenas um caractere diferente, enquanto Pollock e Zamora (1984) registraram um

[37] Distância mínima de edição - https://en.wikipedia.org/wiki/Edit_distance

percentual acima de 90%. Essas diferenças podem ser atribuídas às características dos diferentes *corpora* utilizados em cada estudo. Mitton (1987) usou um *corpus* de textos escritos à mão por jovens de 15 anos, enquanto Pollock e Zamora (1984) utilizaram textos científicos e acadêmicos. O nível de formalidade e instrução dos autores dos textos justifica a maior proporção de erros de um único caractere em um grupo em relação ao outro.

Considerando-se a frequência com que essas quatro categorias ocorrem nos textos, conclui-se que a distância mínima de edição é central na criação de algoritmos de correção ortográfica. A implementação dessa técnica parte do pressuposto de que, quando uma determinada cadeia não é encontrada no dicionário em uso, ela contém algum erro ortográfico.

Para selecionar possíveis candidatas corretas e auxiliar o escritor na correção da cadeia malformada, são usadas as quatro operações básicas, que retornam as cadeias mais semelhantes no dicionário em uso (DAMERAU, 1964). Caso existam múltiplas candidatas válidas, é necessário estabelecer critérios para apresentá-las em ordem de preferência, como frequência de ocorrência das candidatas, índice de peculiaridade, distância mínima de edição (diferentes distâncias são possíveis quando usamos diferentes medidas), similaridade sonora, entre outros aspectos, podendo ser utilizada a combinação de dois ou mais desses critérios.

A biblioteca *PyEnchant* (seção 3.2.6.1) utilizada neste trabalho utiliza internamente a distância de Levenshtein, que é a medida tradicionalmente mais utilizada para esta tarefa.

3.2.6 Ferramentas de correção ortográfica

Como já mencionado, os corretores ortográficos automáticos podem operar em duas direções opostas. Uma opção é utilizar regras de derivação morfológica para detectar os afixos de uma palavra dada e removê-los, preservando a raiz da palavra. Em seguida, verifica-se se a raiz (ou radical) está presente no dicionário. Em caso afirmativo, considera-se a palavra original como correta do ponto de vista ortográfico.

A outra opção busca gerar o vocabulário (quase) completo de uma língua pela afixação das raízes das palavras. A seguir, as palavras do texto a ser corrigido são comparadas às entradas desse vocabulário expandido. Quando a palavra é

encontrada com a mesma grafia do vocabulário, será considerada correta. Quando não ocorre um “casamento” perfeito, a palavra de entrada poderá ser substituída automaticamente pela palavra mais semelhante a ela.

Em todos os casos, o corretor ortográfico utiliza uma medida de *distância de edição* entre palavras (seção 3.2.5), a fim de identificar as opções mais plausíveis para corrigir a palavra com erros. Se o corretor é apenas uma ferramenta de auxílio à edição, ele irá apresentar ao usuário as palavras selecionadas ordenadas com base em algum critério previamente definido (que sempre irá incluir o critério de distância de edição). Porém, quando o corretor é totalmente automático, ele terá que escolher a melhor opção entre as candidatas, substituindo automaticamente a palavra com erros. Este é o caso do nosso trabalho de mestrado (Capítulo 4).

Conforme evidenciado na seção 3.2.1, há diversas ferramentas disponíveis para realizar esta tarefa. Neste trabalho, escolhemos a ferramenta PyEnchant por sua eficiência, usabilidade e possibilidade de personalização do dicionário.

3.2.6.1 PyEnchant com provedor Hunspell

Como já mencionado anteriormente, utilizamos neste trabalho a biblioteca *PyEnchant* com o provedor *Hunspell*[38] para verificação e correção ortográfica. O Hunspell, desenvolvido por László Németh (NÉMETH, 2009), é um corretor ortográfico e analisador morfológico recente baseado no Myspell. Ele foi inicialmente criado para a língua húngara, sendo posteriormente adaptado para outras línguas. Dentre suas características mais importantes, destacam-se:

- Suporta todas as características do Myspell, incluindo os dicionários no formato Myspell.
- Análise morfológica
- Suporta codificação UTF-8, além de ISO8859-1.
- A interface de programação está disponível em várias linguagens de programação.
- As sugestões de correção são baseadas na similaridade de *n-gram*, regras do dicionário e informações para substituição fonética.
- Possui suporte para regras de prefixos e sufixos.

[38] Hunspell - https://sourceforge.net/projects/hunspell/

- Biblioteca C++ sob licença tri GPL/LGPL/MPL.

Atualmente, Hunspell é o verificador ortográfico do LibreOffice, OpenOffice.org, Mozilla Firefox e Thunderbird, Google Chrome, e também é usado por pacotes de softwares proprietários, como macOS, InDesign, memoQ, Opera e SDL Trados. (NÉMETH, 2022).

Os léxicos do Hunspell são compostos por dois arquivos: um arquivo .dic que contém uma lista de lemas com etiquetas de afixos, e um arquivo .aff que contém as regras de afixação, que determinam como os afixos são combinados com as formas básicas. As etiquetas de afixos no arquivo de lemas são separados por uma barra (/) após cada entrada. Cada caractere que segue a barra especifica um grupo de regras de afixo que podem ser aplicadas ao lema, desde que as regras definidas no arquivo .aff sejam atendidas.

A seguir, temos alguns exemplos de entradas no arquivo (.dic).

- impensado/DJ
- impensante/BP
- impensável/BIKX
- Imperar/ajkLYÑ
- *irreparável/BIKX*

Como já mencionado, o arquivo de afixos contém principalmente regras de afixação para gerar palavras flexionadas e derivadas a partir dos lemas. Cada linha corresponde a uma regra e cada conjunto de regras é identificado por um caractere. O conjunto é introduzido por uma regra de cabeçalho que fornece informações sobre o tipo da regra, seja prefixo (PFX) ou sufixo (SFX_), o identificador, se as regras do conjunto podem ser combinadas com outras regras (“Y” para sim e “N” para não), e o número de regras no conjunto.

As regras de afixação especificam, em primeiro lugar, o tipo *(SFX)* e o identificador *(X)*, semelhantes às regras de cabeçalho *(SFX X Y 22)*. Em seguida, é indicada uma substring *(ável)* que deve ser excluída do lema *(irreparável)*, no início para regras de prefixo e no final para regras de sufixo. Depois, é indicada uma substring *(abilidade)* que deve ser inserida no mesmo local da subtring excluída *(ável)*. Por fim, é especificada uma Expressão Regular que deve ser correspondida no lema original para que a regra seja aplicada. Abaixo temos um exemplo das primeiras linhas de um conjunto de regras de sufixo chamado X:

- # Substantivos -X- Terminação: idade
- SFX X Y 22
  1. SFX X 0 idade [^e]l
  2. SFX X 0 idade r
  3. SFX X 0 idades [^e]l
  4. SFX X 0 idades r
  5. SFX X ável abilidade ável
  6. SFX X ável abilidades ável
  7. SFX X e idade e
  8. SFX X e idades e
  9. SFX X ível ibilidade ível
  10. SFX X ível ibilidades ível

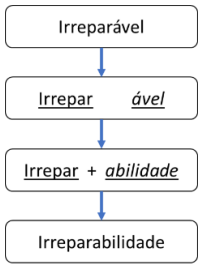
Tabela 1 - Exemplo de aplicação da quinta regra para o sufixo X.

Tipo de afixo	Identificador	Substring removida	Substring inserida	Regra para aplicação
SFX	X	ável	abilidade	ável

Fonte: A autora (2023).

Com base nos exemplos de entradas nos arquivos .dic e .aff acima, a forma flexionada "*irreparabilidade*" pode ser derivada do lema "*irreparável*" e aplicando a quinta regra do sufixo *X*, que exclui o morfema *"ável"*, reduzindo o lema à string *"irrepar"*, e acrescentando ***"abilidade"***. Isso é permitido, pois o lema tem um sinalizador X, e a quinta regra X é aplicável se o padrão *"ável"* corresponder ao sufixo do lema. Este exemplo de aplicação da regra de afixação X à palavra Irreparável é ilustrado na Figura *5*.

Figura 5 - Demonstração da aplicação da regra de afixação X.



Fonte: A autora (2023).

3.3 PÓS-PROCESSAMENTO DE DOCUMENTOS CENTENÁRIOS

Esta seção se dedica a descrever trabalhos relacionados ao nosso tema. Contudo, como já mencionado, não encontramos na literatura trabalhos semelhantes ou equivalentes ao nosso, que realizem um processo de adaptação de dicionários centenários para utilização em corretores ortográficos atuais.

Assim, vamos apresentar brevemente dois trabalhos relacionados ao nosso contexto de correção ortográfica de documentos centenários um na língua portuguesa (SÁ e MAIA, 2020), e outro para Finlandês e Sueco (DROBAC, 2020). Esses trabalhos também utilizam a abordagem lexical, porém o objetivo geral é distinto do nosso.

Por fim, vale frisar que a temática de correção pós-OCR de textos antigos já foi tratada por outros trabalhos que foram estudados durante a revisão bibliográfica realizada neste mestrado (e.g., (HAUSER, 2007) para a língua alemã), e (RICHTER et. al., 2018) (para língua feroesa). Porém, as soluções desenvolvidas estão muito distantes do foco do nosso trabalho.

Inicialmente, destacamos aqui o trabalho de Gildácio Sá e José Maia (2020), que tem como objetivo final possibilitar o acesso ao formato digital do Jornal “O NORDESTE”, publicado pela Arquidiocese de Fortaleza/CE durante o período de 1922 a 1964. Os autores tiveram acesso a um corpus de 36.617 imagens de páginas do referido Jornal, que foi utilizado como entrada para um processo de extração de texto, correção pós-OCR, extração da estrutura de tópicos da coleção de textos e disponibilização para acesso via interface de navegação.

O trabalho de Gildácio Sá e José Maia (2020) se baseou em vários dicionários, além do dicionário de 1913 que utilizamos na nossa solução. Esse trabalho adotou uma abordagem diferente da nossa na construção da solução de correção ortográfica, bem como no objetivo final dos trabalhos. Veremos a seguir detalhes sobre as etapas do pós-processamento realizadas, e ressaltamos que o objetivo final desse trabalho foi, diferentemente do nosso, a visualização/navegação dos documentos a partir de uma hierarquia de tópicos referentes aos temas tratados no jornal.

O processo proposto pelos autores consiste em 4 etapas, que foram executadas a partir do corpus coletado. Inicialmente, as imagens dos documentos históricos foram segmentadas. Após essa segmentação, os textos foram extraídos.

Essas duas etapas foram realizadas utilizando a *ferramenta comercial de OCR AbbyyFine Reader CE* ©. Em seguida o texto foi pós-processado, com dois objetivos principais: correção de erros ortográficos oriundos do OCR, e conversão de palavras arcaicas para uma escrita mais atual.

O pós-processamento do texto se inicia com a extração dos radicais e demais formatações das palavras utilizando técnicas de PLN como tratamento de *stopwords*, e identificação de substantivos e verbos. Foram tratadas apenas essas duas classes gramaticais, pois as demais categorias não agregariam semântica mínima necessária para o objetivo do trabalho.

Após essas etapas, as palavras arcaicas foram convertidas para uma ortografia mais atual (SÁ, 2020). Essa conversão se deu a partir de consultas a dicionários da época dos jornais devidamente estruturados e associados a uma base de dados construída pelo autor. Segundo o autor, para realizar as comparações morfológicas, e em seguida as conversões cabíveis, foi construída uma base de dados de dicionários. A criação dessa base englobou a criação de um banco de dados de verbos, a construção de um arquivo de flexão verbal completa e a construção de um arquivo de sinônimos.

Por fim, descartamos ainda o trabalho de doutorado de Senka Drobac (2020), que se baseou na análise de um corpus de jornais e periódicos históricos publicados na Finlândia, com mais de 11 milhões de páginas de textos históricos digitalizadas e que datam desde o século 18 ao início do século 20. Esse trabalho explorou uma abordagem baseada na melhoria do OCR e no pós-processamento do texto extraído.

Inicialmente, a Biblioteca Nacional da Finlândia (NLF) fez o OCR do corpus de imagens utilizando o *ABBYY FineReader*, um software comercial que fornece modelos de OCR pré-treinados em fontes históricas gerais. A precisão estimada do texto extraído ficou entre 87% e 92% no nível do caractere, o que é bastante baixa mesmo para pesquisas acadêmicas.

Assim, em seu trabalho, Drobac explorou métodos e práticas para treinar modelos de OCR de alta precisão para reconhecer com eficiência todo o corpus de jornais e periódicos históricos da Finlândia. Foram selecionadas do corpus 13.000 linhas de texto finlandesas e 11.000 suecas, das quais metade está impressa em *Blackletter* e metade em fontes *Antiqua*.

Depois de transcrever essas linhas, elas foram utilizadas para treinar e testar modelos de OCR com dois kits de ferramentas de OCR de código aberto, *Ocropy* e *Calamari* (ver seção 2.4, Capítulo 2). Foram realizados experimentos com diferentes configurações de dados de treinamento, juntamente com diferentes configurações e arquiteturas de redes neurais. Além disso, o autor também testou como o mecanismo de votação se comporta com diferentes modelos de OCR.

Também foram explorados diferentes métodos de correção pós-OCR. Um corretor ortográfico baseado em uma máquina de estados finitos foi implementado utilizando máquina de estados finitos e um léxico histórico finlandês. A precisão final obtida foi entre 97,2% e 98,4% no nível do caractere. O método automático de correção pós-OCR empregado, que é relativamente simples, obteve uma melhoria relativa de 0,9 a 12,5%, dependendo da precisão inicial do OCR (DROBAC, 2020).

3.4 CONSIDERAÇÕES FINAIS

Este capítulo teve como foco os corretores ortográficos automáticos para documentos históricos. Inicialmente, aprofundamos os conceitos sobre a abordagem de correção baseada em palavras isoladas com o uso de léxicos (seção 3.2), por ser este o foco do nosso trabalho. A seguir, a seção 3.3 destacou dois trabalhos relacionados que têm como foco a extração de textos a partir de documentos históricos e o seu pós-processamento. O primeiro trabalho focou na atualização do vocabulário antigo, enquanto o segundo trabalho investiu na melhoria do processo de OCR e na correção pós-OCR baseada em léxicos históricos.

O Capítulo 4, a seguir, irá detalhar o trabalho desenvolvido neste mestrado, com foco na adaptação do dicionário histórico de 1913 para utilização com o corretor ortográfico Hunspell. O objetivo final do trabalho era de corrigir os textos extraídos a partir de jornais centenários editados no Brasil do século XIX.

4 UM PROCESSO DE CORREÇÃO E MELHORIA DO TEXTO EXTRAÍDO DE IMAGENS UTILIZANDO TÉCNICAS DE PLN

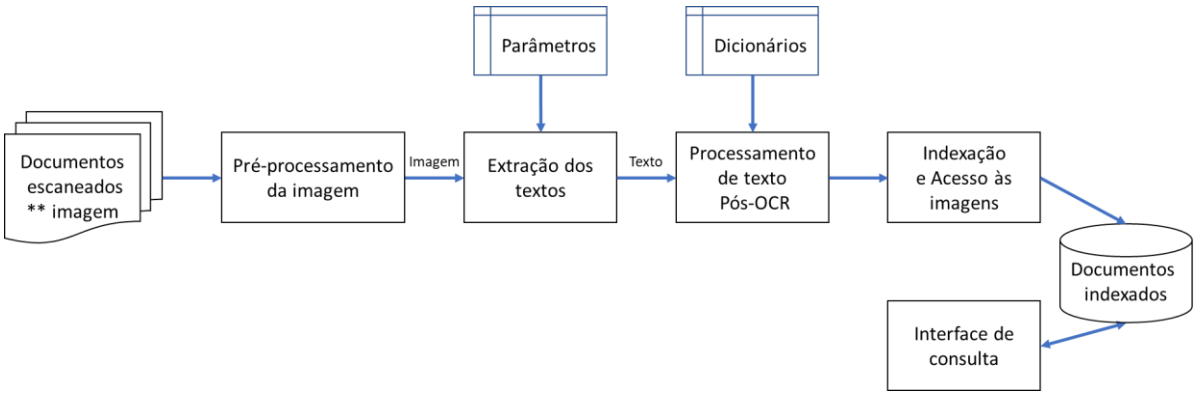
O projeto “Programa de Curadoria do patrimônio memorial de Jornais centenários de Pernambuco” objetiva *resgatar*, *preservar* e *prover livre acesso* a jornais periódicos centenários em língua portuguesa.

Na seção 4.1 será apresentada uma visão geral da solução proposta. A seção 4.2 descreve as técnicas de pré-processamento aplicadas às imagens digitalizadas com o intuito de otimizar a eficiência da ferramenta utilizada na extração de texto via OCR. A seção 4.3 detalha a etapa de extração de texto, que utiliza a ferramenta Tesseract. Em seguida, a seção 4.4 apresenta em detalhes a abordagem desenvolvida neste trabalho para correção e melhoria dos textos extraídos, implementada na etapa de correção pós-OCR. A seção 4.5 mostra a etapa de indexação das imagens a partir dos textos extraídos, bem como a interface de consulta desenvolvida para facilitar o acesso a essas informações. Por fim, a seção 4.6 traz algumas considerações finais sobre o conteúdo deste capítulo.

4.1 SOLUÇÃO PROPOSTA – VISÃO GERAL

O processo proposto para promoção de acessibilidade ao conteúdo de documentos históricos recebe como entrada imagens escaneadas, realiza a correção e a melhoria dos textos extraídos via OCR e, por fim, indexa as imagens de entrada a partir das palavras que ocorrem em cada imagem. Esse processo conta com quatro etapas, como mostra a Figura *6*.

Figura 6 - Etapas de processamento da solução proposta.



Fonte: A autora (2023).

4.2 PRÉ-PROCESSAMENTO DA IMAGEM (CORPUS DIGITALIZADO)

Conforme discutido no Capítulo 2, ao lidar com imagens digitalizadas contendo texto, é essencial preparar o conteúdo para uma extração precisa e confiável. Como já mencionado, as imagens tratadas neste trabalho são oriundas de jornais centenários digitalizados pela equipe do LIBER, responsável por essa tarefa no projeto geral. Atualmente, estamos lidando com páginas dos jornais “Diário da Manhã”, de 1930, e do “Diário de Pernambuco”, de 1840 e 1861. Devido ao desgaste dos documentos de origem, as imagens obtidas apresentam falhas, manchas e até buracos devidos à ação de fungos e traças. Assim, foi necessário pré-processar essas imagens, a fim de melhorar os resultados do OCR realizado para extração do texto.

A ferramenta utilizada para a extração de texto via OCR, *Tesseract* (ver seção 2.4), já inclui diversas técnicas de processamento de imagem através da biblioteca *Leptonica* antes de realizar a extração de texto. Embora essas técnicas sejam eficientes na melhoria da precisão da extração de texto em uma ampla variedade de imagens, inevitavelmente existem casos que demandam o emprego de técnicas e ajustes personalizados, a serem utilizados anteriormente ao Tesseract, para a ferramenta alcançar uma precisão satisfatória.

Assim, neste trabalho foi definida uma etapa de pré-processamento externo de imagens, com o objetivo de investigar e destacar a importância dessa atividade na precisão final da extração de texto. Esta etapa se baseia em um conjunto de técnicas disponíveis via *OpenCV,* uma biblioteca popular e poderosa para processamento de imagens e visão computacional em Python. Com suporte a várias plataformas e linguagens de programação, o *OpenCV* fornece uma ampla gama de funcionalidades para lidar com tarefas relacionadas ao processamento e à análise de imagens (BRADSKI, 2008).

O *OpenCV* oferece uma variedade de recursos para melhorar a qualidade da imagem e facilitar o reconhecimento de caracteres pelo OCR. A seguir, são apresentadas as operações de pré-processamento de imagem que foram aplicadas neste trabalho, visando aprimorar a qualidade da imagem previamente ao uso do Tesseract, para alcançar maior precisão e legibilidade do texto extraído.

Ajuste do Gamma: Na imagem é uma técnica comumente aplicada para melhorar a legibilidade do texto antes da extração dos caracteres. Essa técnica envolve a manipulação da curva de resposta tonal da imagem, realçando ou atenuando os níveis de brilho. Por meio do ajuste do gamma, é possível aprimorar o contraste e destacar as informações relevantes do texto, especialmente quando a imagem apresenta iluminação desigual ou baixo contraste. Isso contribui para melhorar a capacidade do OCR em reconhecer os caracteres com mais precisão, proporcionando resultados mais confiáveis na etapa de reconhecimento.

O ajuste adequado do gamma depende das características específicas da imagem, sendo necessário equilibrar a melhoria do contraste sem comprometer a qualidade geral da imagem. A função usada para realizar o ajuste do gamma é apresentada a seguir.

```
def adjust_gamma(image, gamma=1.2):
        invGamma = 1.0 / gamma
        table = np.array([((i / 255.0) ** invGamma) * 255
                        for i in np.arange(0, 256)]).astype("uint8")
        return cv2.LUT(image, table)
gama_corrected = adjust_gamma(img)
```

Método Divisão de Ganho: Consiste no contraste entre o texto e o plano de fundo desempenhando um papel crucial na legibilidade dos caracteres. Ao construir um modelo do plano de fundo, é possível entender a distribuição geral de intensidade da imagem. Com base nesse modelo, cada pixel de entrada é multiplicado por um fator de ganho apropriado. O resultado desse processo é uma imagem em que o contraste entre o texto e o plano de fundo é aprimorado de forma consistente em toda a imagem. Isso ajuda a destacar os caracteres e torná-los mais distintos, o que, por sua vez, melhora a precisão do OCR.

```
imageMedian = cv2.medianBlur(img_original, filterSize)
kernelSize = 15
maxKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernelSize,
kernelSize))
localMax = cv2.morphologyEx(imageMedian, cv2.MORPH_CLOSE, maxKernel,
None, None, 1, cv2.BORDER_REFLECT101)
gainDivision = np.where(localMax == 0, 0, (imageMedian/localMax))
gainDivision = np.clip((255 * gainDivision), 0, 255)
gainDivision = gainDivision.astype("uint8")
```

Converter a imagem para escala de cinzas: Até o passo anterior, a imagem que está sendo processada possui 3 canais de cores no modelo RGB (*Red-Green-Blue*), referente às cores primárias vermelho, verde e azul. Nesta etapa é realizada uma conversão da imagem para escala de cinza com apenas um canal para a aplicação das demais técnicas de processamento.

```
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Ajuste de contraste: Nesta etapa é realizado um aumento de contraste entre as letras e o plano de fundo, de modo a destacar mais os caracteres. Isso é feito através da normalização dos valores de intensidade de cada pixel de acordo com os limites máximos e mínimo desejados.

```
grayscaleImage = np.uint8(cv2.normalize(gray_image, gray_image, 0, 255, cv2.NORM_MINMAX))
```

Binarização: Essa técnica é utilizada para converter uma imagem em tons de cinza em uma imagem binária, na qual cada pixel é classificado como preto ou branco. Embora alguns outros métodos de binarização, como Otsu adaptativo e Sauvola, tenham sido incluídos pelo *Tesseract* recentemente, seu método de binarização padrão é o Otsu, que se baseia na análise do histograma de intensidades da imagem para encontrar um valor de limite ótimo que separa os pixels em duas classes, normalmente preto e branco.

Após extensiva experimentação com outros métodos de binarização, como limiarização padrão, adaptativa, e Otsu, escolhemos utilizar aqui o método Sauvola por apresentar melhor desempenho com as imagens testadas. Essa técnica é particularmente eficaz para documentos ou imagens com condições de iluminação não uniforme ou variações na intensidade do fundo.

O algoritmo de binarização Sauvola calcula um limiar local para cada pixel com base nas estatísticas locais de sua vizinhança. Ele leva em consideração a intensidade média e o desvio padrão dos pixels dentro de uma janela definida centrada no pixel de interesse. O tamanho da janela geralmente é um parâmetro que

pode ser ajustado com base nas características da imagem. O limiar ***T*** é calculado para cada pixel na imagem usando a seguinte fórmula:

$$T = m(x,y) * \left[1 + k * \left(\frac{s(x,y)}{R} - 1\right)\right]$$

Onde,

T(x, y) é o valor do limiar para o pixel nas coordenadas (x, y).

m(x, y) é a média local das intensidades dos pixels dentro da janela centrada em (x, y).

s(x, y) é o desvio padrão local das intensidades dos pixels dentro da janela centrada em (x, y).

k é um parâmetro que controla a sensibilidade às variações locais.

R é o desvio padrão máximo dentro da imagem ou uma faixa específica.

Uma vez que o limiar local é calculado, a intensidade de cada pixel é comparada com o limiar. Se a intensidade for maior ou igual ao limiar, o pixel é definido como branco; caso contrário, é definido como preto (SAUVOLA; PIETIKAINEN, 2000). Esse processo é aplicado a todos os pixels da imagem, resultando em uma representação binária.

Neste trabalho utilizamos a função Sauvola implementada na biblioteca *skimage*[39], conforme apresentado a seguir.

```
from skimage.filters import (threshold_niblack, threshold_sauvola)
window_size = 15
thresh_sauvola = threshold_sauvola(grayscaleImage, window_size=window_size,
k=0.08, r= 28)
binary_sauvola = grayscaleImage > thresh_sauvola
binary_sauvola = skimage.util.img_as_ubyte(binary_sauvola)
```

Remoção de ruído: Ruído é uma variação aleatória de brilho ou cor em uma imagem, que pode tornar o texto da imagem mais difícil de ser extraído. Certos tipos

[39] Skimage - https://scikit-image.org/docs/stable/api/skimage.html

de ruído podem não ser removidos pelo *Tesseract* na etapa interna de binarização, o que pode fazer com que as taxas de precisão caiam.

A biblioteca *OpenCV* disponibiliza diversas técnicas para lidar com o ruído das imagens. Neste trabalho, optou-se por implementar um filtro de área no qual apenas os componentes conectados com área maior ou igual a um determinado valor *minArea* são preservados, enquanto os demais são removidos. Isso permite uma segmentação mais refinada dos objetos de interesse na imagem binária.

```
componentsNumber, labeledImage, componentStats, componentCentroids = \
cv2.connectedComponentsWithStats(binaryImage, connectivity=4)
minArea = 10
remainingComponentLabels = [i for i in range (1, componentsNumber) if
componentStats[i][4] >= minArea]
filteredImage = np.where(np.isin(labeledImage, remainingComponentLabels) ==
True, 255, 0).astype("uint8")
```

Erosão e Dilatação: Caracteres em negrito ou finos podem afetar o reconhecimento de detalhes e reduzir sua precisão. A dilatação é utilizada para expandir as regiões de interesse e preencher pequenas lacunas entre os caracteres, o que ajuda a melhorar a conectividade e a integridade dos caracteres. Por outro lado, a erosão é usada para reduzir o tamanho dos objetos e eliminar ruídos, tornando os caracteres mais distintos e separados uns dos outros. Ao combinar a dilatação e a erosão de maneira adequada, é possível realçar os caracteres e melhorar a segmentação e o reconhecimento de texto durante o processo de OCR.

```
def thin_font(image):
        kernel = np.ones((2, 2), np.uint8)
        image = cv2.erode(image, kernel, iterations=1)
return (image)

def thick_font(image):
        kernel = np.ones((2, 2), np.uint8)
        image = cv2.dilate(image, kernel, iterations=1)
return (image)

eroded_image = thin_font(filteredImage)
dilated_image = thick_font(eroded_image)
```

Fechamento de elementos: Frequentemente, fragmentos dos caracteres nas imagens de documentos históricos podem desaparecer com a ação do tempo, ou mesmo em decorrência de técnicas de pré-processamento de imagem utilizadas. Para repor essas lacunas nos caracteres e melhorar a qualidade do texto extraído podem ser aplicadas operações morfológicas que reconectam as adjacências nas lacunas e restauram o caractere deteriorado.

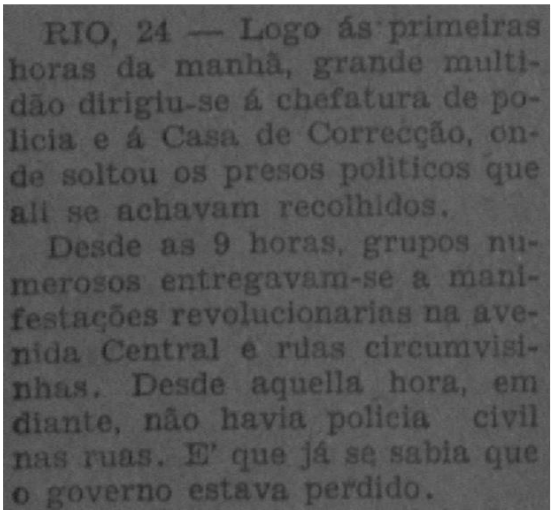
```
kernelSize = 3
opIterations = 1
maxKernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernelSize, kernelSize))
closingImage = cv2.morphologyEx(eroded_image, cv2.MORPH_CLOSE, maxKernel, None, None, opIterations, cv2.BORDER_REFLECT101)
closingImage = 255 - closingImage
```

4.2.1 Considerações finais

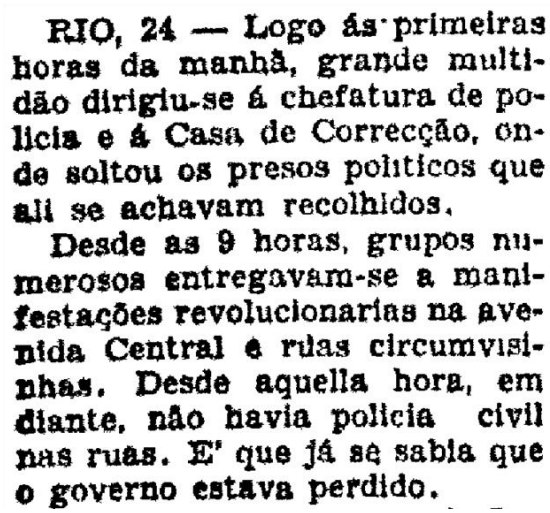
As técnicas de pré-processamento de imagens descritas aqui foram aplicadas às imagens dos jornais previamente à extração do texto via OCR *Tesseract*. A Imagem *1* apresenta um exemplo do impacto do pré-processamento da imagem implementado aqui, aplicado a um trecho de uma das páginas do jornal “Diário da Manhã”.

Imagem 1 - Exemplo do resultado obtido após a realização do pré-processamento de imagens. (a) imagem original e (b) imagem pré-processada.

a)



b)



Fonte: A autora (2023).

É importante destacar que observamos uma variação considerável quanto à melhoria que o emprego dessas técnicas trouxe ao texto extraído de cada página. Como é possível notar, o elevado número de técnicas de pré-processamento e parâmetros de ajuste torna a tarefa de definir um único conjunto ótimo de parâmetros para todos os jornais uma tarefa impraticável, uma vez que existem alterações do estado nas diferentes regiões de cada página nos jornais históricos.

Embora não tenha sido o foco principal deste trabalho, a investigação do uso das técnicas de pré-processamento de imagem comprovou seu potencial na melhoria do texto extraído, conforme será apresentado no Capítulo 5. Além disso, as dificuldades encontradas na generalização das técnicas de pré-processamento fomentaram a indicação de uma possível abordagem para tratar o problema, apresentada na seção 6.2 sobre Trabalhos Futuros.

4.3 EXTRAÇÃO DOS TEXTOS VIA TESSERACT

A extração de texto a partir das imagens digitalizadas foi realizada utilizando-se o *PyTesseract*, uma biblioteca em Python que disponibiliza a tecnologia OCR para reconhecer e extrair texto a partir de imagens. Ele é baseado no *Tesseract*, que é um mecanismo de OCR de código aberto desenvolvido pelo Google, como já mencionado na seção 2.4. Essa ferramenta suporta vários tipos de entradas incluindo PNG, JPEG, TIFF, possui também várias funções e parâmetros que podem ser usados para obter resultados diferentes, como por exemplo: PDF pesquisável, alto xml, hOCR, string, dentre outras. Para nosso trabalho, TIFF é a extensão das imagens utilizadas e escolhemos a função que permite a conversão de imagem para *string*. A assinatura completa da função é a seguinte:

```
custom_oem_psm_config = r'--oem 1 --psm 1'
text = pytesseract.image_to_string(image, lang='por', config='', nice=0,
output_type=Output.STRING)
```

Essa função recebe a *imagem* cujo texto será extraído. Caso a imagem passada seja colorida ou não seja binarizada, o Tesseract irá realizar a binarização internamente. Caso a imagem já seja binarizada, o Tesseract interpreta que o usuário já realizou o pré-processamento previamente e não realiza essa etapa novamente.

Além disso, o Tesseract OCR oferece suporte ao *idioma* português, permitindo aos usuários selecionar esse idioma específico para o reconhecimento de texto. Ao escolher o idioma português, o Tesseract adapta seu modelo interno para reconhecer as características e as particularidades da língua portuguesa, incluindo padrões de caracteres, regras gramaticais e convenções de escrita. Isso resulta em uma maior precisão na extração de texto de qualquer tipo de conteúdo em língua portuguesa.

Durante o processo de segmentação, o *módulo linguístico* escolhe a melhor sequência de palavras disponível em várias categorias, incluindo: Palavra mais frequente, Palavra do dicionário mais frequente, Palavra numérica mais frequente, Palavra em MAIÚSCULAS mais frequente, dentre outras. A decisão final para uma determinada segmentação é simplesmente a palavra com a menor avaliação de distância total, em que cada uma das categorias mencionadas é multiplicada por uma constante diferente (SMITH, 2007).

Outro argumento fundamental nessa função é o *config***,** que fornece opções de configuração adicionais para o Tesseract OCR personalizar o processo de reconhecimento de caracteres. Neste trabalho, o arquivo de configuração foi definido como:

```
custom_oem_psm_config = r'--oem 1 --psm 1’
```

onde,

- **mecanismo de OCR** (*--oem*) = 1. A opção “--oem 1” no Tesseract OCR refere-se à configuração do modo de reconhecimento de caracteres do mecanismo OCR. OEM é a sigla para "*OCR Engine Mode*". O valor '1' atribuído a essa opção indica o modo de OCR baseado em "Redes neurais LSTM apenas". Nesse modo, o Tesseract utiliza redes neurais de memória de longo e curto prazo para realizar o reconhecimento de caracteres. Esse modelo é geralmente mais preciso e eficiente do que as demais opções de configuração, permitindo uma melhor interpretação de diferentes tipos de fontes e melhorias na detecção de palavras.
- **método de segmentação da página** (*--psm) = 1.* A configuração '--psm 1' no Tesseract OCR refere-se ao modo de segmentação de página do mecanismo OCR. PSM é a sigla para "*Page Segmentation*

Mode". O valor '1' atribuído a essa opção indica o modo de segmentação automática, no qual o Tesseract tentará identificar automaticamente e segmentar blocos de texto em uma página.

Por fim, *nice* e *output_type* são parâmetros opcionais. O padrão é *Output.STRING*, que retorna o texto extraído como uma *string*, que atende o tipo de saída escolhida nesse projeto.

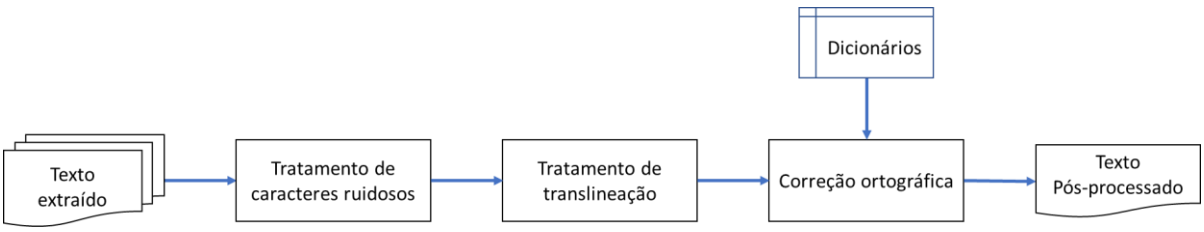
Vale ressaltar que os demais modos de configuração foram testados para as imagens das páginas dos jornais tratadas, e os modos escolhidos apresentaram melhor desempenho na extração do texto.

4.4 PROCESSAMENTO DE TEXTO PÓS-OCR

Embora o Tesseract seja uma ferramenta poderosa de OCR, inevitavelmente o texto extraído está sujeito a imprecisões, sobretudo em se tratando de documentos históricos. Neste trabalho, optamos por implementar uma etapa de correção do texto extraído para reduzir as imprecisões do OCR, melhorando a qualidade do texto final. Esta é a principal contribuição deste trabalho de mestrado.

A Figura *7* traz uma visão geral da solução desenvolvida, mostrando os módulos de processamento implementados. A pós correção do texto desenvolvida tratou inicialmente da identificação e eliminação de caracteres ruidosos presentes no texto extraído. Em seguida, há uma etapa para resolver casos de translineação de palavras, recompondo as palavras que estavam divididas. Note que essas etapas são necessárias para viabilizar a aplicação do corretor ortográfico. Finalmente, temos a aplicação do corretor ortográfico *PyEnchant* utilizando um dicionário personalizado construído neste trabalho.

Figura 7 - Etapas de processamento do texto Pós-OCR.



Fonte: A autora (2023).

A seguir, a seções 4.4.1 e 4.4.2 apresentam etapas de limpeza e normalização do texto. A seção 4.4.3 trata da correção ortográfica do texto usando o *PyEnchant*. Por fim, a secção 4.4.4 detalha a preparação do dicionário de 1913 para uso pelo corretor ortográfico.

4.4.1 Tratamento de caracteres ruidosos

A tokenização é geralmente o primeiro passo na análise de texto, objetivando dividir um texto em unidades menores chamadas *tokens* (ver seção 2.3.2.2). Um token pode ser uma palavra, uma frase, um caractere ou qualquer outra unidade significativa de texto, levando em consideração pontuações e espaços em branco.

Embora existam várias bibliotecas populares em Python para realizar a tokenização de texto, neste trabalho optamos por utilizar uma funcionalidade dessa linguagem chamada de “compreensão de listas” para realizar essa tarefa. Com esta funcionalidade é possível, em apenas uma linha de código, criar uma nova lista ao aplicar uma expressão a cada elemento de uma lista existente, filtrar elementos com base em uma condição ou até mesmo combinar múltiplas listas. O trecho do código utilizado para realiza a tokenização de palavras é apresentado a seguir.

```
words = [line.rstrip().split() for line in Lines]
```

Onde *Lines* é uma lista de *strings* na qual cada elemento contém uma linha do texto extraído do OCR. Para cada elemento *line*, é chamado o método *.rstrip()*, que remove os espaços em branco à direita da *string*. Isso é feito para eliminar possíveis espaços ou quebras de linha extras que possam existir no final de cada linha. Em seguida, o método *.split()* é chamado para tratar a *string* resultante do passo anterior. Esse método divide a *string* em Tokens com base nos espaços em branco encontrados.

Em seguida, foram utilizadas expressões regulares para limpeza de caracteres ruidosos no texto extraído. As expressões regulares são sequências de caracteres especiais que permitem realizar correspondências e a manipulações avançadas em *strings*. Elas são usadas para pesquisar, validar, extrair e manipular padrões complexos de texto. Em Python, as expressões regulares são implementadas por meio do módulo *re*. O código apresentado a seguir remove todos

os caracteres especiais do texto, exceto vírgulas, pontos, hifens e letras com acento.

```
def remove_caracteres_especiais(texto):
    regex = r"[^a-zA-Z0-9,. \-áÁâÂãÃàÀéÉêÊíÍóÓôÔõÕúÚçÇ]"
    texto_sem_especiais = re.sub(regex, "", texto)
    return texto_sem_especiais
```

Desta forma, o texto dividido por linhas é organizado em uma lista, onde cada elemento da lista é outra lista correspondente a uma linha do texto original tokenizada e sem os caracteres especiais. Essa estrutura é então passada para o próximo processo desta etapa.

4.4.2 Tratamento de translineação de palavras

A *translineação* de palavras refere-se à quebra de palavras longas em mais de uma linha, geralmente por motivos de formatação ou restrições de espaço em um documento. Essa quebra deve obedecer às regras de divisão silábica da língua em questão. Embora seja comumente encontrada em documentos impressos antigos, a translineação pode representar um desafio para uma análise correta e coerente do texto. Isso ocorre porque a translineação leva à fragmentação das palavras, prejudicando a compreensão do contexto e a aplicação de corretores ortográficos. Assim, foi necessário implementar um código para realizar a união das partes da palavra translineada, a fim de restaurar a palavra em sua forma original.

Resumidamente, o código implementado divide o texto extraído em uma lista de linhas e testa a linha que termina com o caractere ‘-’. Caso o teste seja positivo, o primeiro token da próxima linha é adicionada ao último token da linha anterior sem o caractere, sendo removido da linha seguinte. A seguir, é apresentado um exemplo do resultado obtido com a translineação.

Texto extraído:	**Texto processado:**
As forças armadas, permanen- tes ou improvisadas, têm sido	As forças armadas, permanentes ou improvisadas, têm sido

4.4.3 Correção ortográfica via PyEnchant Hunspell

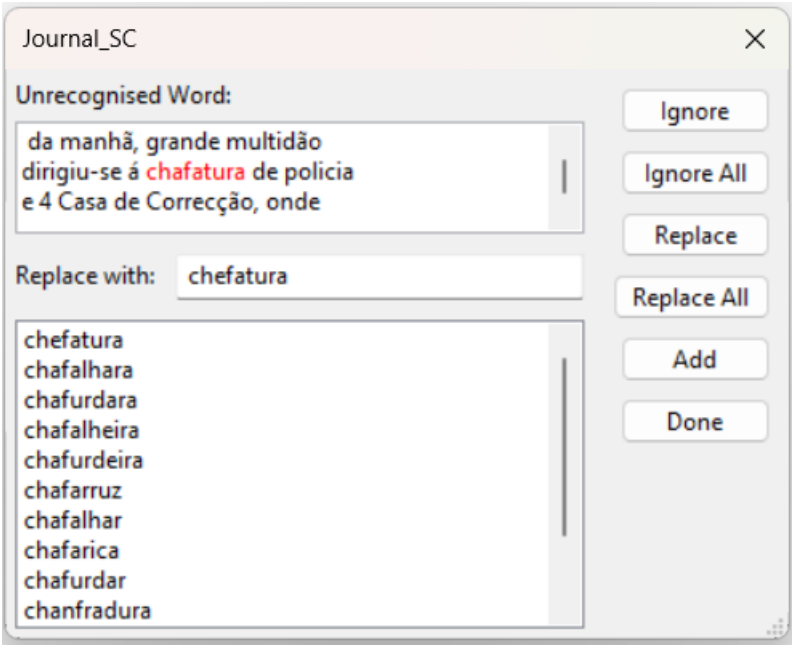
Após a tokenização e o tratamento de translineação do texto extraído, as palavras obtidas são submetidas a um corretor ortográfico. O objetivo é eliminar erros de OCR referentes a ausência ou trocas de letras. A correção do texto é realizada com o auxílio da biblioteca *PyEnchant* com o provedor *Hunspell*, escolhida devido à sua eficiência e facilidade de uso.

Essa biblioteca Python oferece suporte a vários idiomas e permite que sejam utilizados dicionários customizados. Contudo, como já mencionado, a ferramenta só oferece um dicionário do Português atual, que não é adequado para tratar textos históricos. Assim, foi necessário adaptar para os padrões da ferramenta um dicionário também centenário, de 1913 (ver seção 4.4.4). Esse foi o dicionário utilizado na correção dos textos centenários.

O código desenvolvido neste trabalho disponibiliza dois modos de correção ortográfica: automático e semiautomático. No *modo automático de correção*, utiliza-se a biblioteca *PyEnchant* para identificar e corrigir automaticamente os erros ortográficos do texto fornecido. Isso é feito comparando as palavras do texto com o dicionário definido na ferramenta. Se uma palavra não for encontrada no dicionário nem em suas possíveis flexões, o algoritmo de correção trocará a palavra pela primeira sugestão do *PyEnchant* com base na similaridade ortográfica.

No *modo semiautomático de correção*, a biblioteca *PyEnchant* é utilizada juntamente com uma interface personalizada para verificar a ortografia das palavras em um texto fornecido. Cada palavra do texto é verificada, em busca de erros ortográficos. Quando uma palavra é identificada como incorreta, a interface realça a palavra no texto e oferece uma lista de sugestões de correção com base no mecanismo de correção definido pelo provedor ortográfico selecionado, no nosso caso, o *Hunspell*. Se nenhuma das palavras sugeridas for a opção correta para a palavra em questão, o usuário tem a opção de adicionar a palavra correta ao dicionário ou simplesmente ignorar o erro, mantendo o texto sem alterações. A Imagem 2 apresenta um exemplo de utilização da interface desenvolvida para correção semiautomática de textos.

Imagem 2 - Interface desenvolvida para correção semiautomática de textos.



Fonte: A autora (2023).

O modo automático de correção é incomparavelmente mais rápido e prático que o semiautomático, especialmente quando consideramos a dimensão do corpus dos jornais históricos trabalhados. Todavia, os resultados da correção automática ainda poderão conter erros de correção, decorrentes de alguns fatores, como: palavras extraídas erradas do OCR que não existem no léxico do dicionário ou casos em que a escolha feita pela ferramenta da primeira palavra da lista de sugestões de correção não corresponde à palavra original. A escolha do modo de correção a ser utilizado depende dos requisitos e recursos da aplicação. Sendo o modo semiautomático mais confiável, porém mais custoso e o modo automático mais rápido e econômico, porém mais susceptível a erros.

4.4.4 Adaptação do dicionário de Português de 1913 no formato *Hunspell*

Esta seção descreve a adaptação de um dicionário de Português de 1913 para o padrão do *PyEnchant* + *Hunspell*, que utiliza regras de afixação para flexionar a forma base (lema) das palavras, ampliando assim a cobertura do corretor ortográfico. Essa funcionalidade melhora a precisão e a qualidade dos textos históricos corrigidos.

4.4.4.1 Dicionário de 1913

O “Novo Diccionário da Língua Portuguesa” foi originalmente publicado em 1899 e passou por várias reedições, até a 25ª e última edição em 1996. Essa obra é de grande importância na história da lexicografia da língua portuguesa. Foi compilada pelo filólogo, gramático e professor António Pereira Cândido de Figueiredo, sendo amplamente reconhecido como um marco significativo no estudo da língua portuguesa (FIGUEIREDO, 1913). O autor faleceu em 1925, e a edição de 1913 entrou em domínio público no ano 2000. Posteriormente, foi digitalizada e convertida em texto por uma equipe de voluntários do projeto “*Distributed Proofreaders*”[40] entre 2007 e 2010. A obra foi disponibilizada ao público no site do “Projeto Gutenberg”[41] em 8 de março de 2010 (WIKIPEDIA, 2018).

No Projeto Gutemberg, além da versão no formato .pdf também foi publicada a versão digital em formato de texto .txt do dicionário de português de 1913. Esse arquivo foi utilizado na construção do dicionário a ser utilizado pelo corretor ortográfico. Contudo, foi necessário extrair do arquivo .txt todas as informações desnecessárias para uso corretor ortográfico, isto é, as definições de cada palavra, textos introdutórios e todas as demais informações contidas no dicionário, restando apenas o vocabulário. Resumidamente, processo de extração do léxico consistiu em remover os caracteres “\n” do arquivo .txt, em seguida dividir cada linha a partir do delimitador “,”. Na versão de 1913, este léxico contém 125.244 palavras em sua forma base (lema). Porém, vale ressaltar que existem diversas ocorrências de palavras flexionadas, como por exemplo “último, última”. Na próxima seção descreveremos a metodologia utilizada para tratar estes casos de termos redundantes considerando as flexões.

4.4.4.2 Uso do PyEnchant Hunspell

Testes iniciais com PyEnchant usando esse léxico de 1913 revelaram duas falhas distintas na indicação de erros de ortografia: palavras flexionadas e palavras (lemas) ausentes do léxico. Ao observar os erros indicados, verificou-se que muitas

[40] Distributed Proofreaders - https://www.pgdp.net/c/
[41] Projeto Gutemberg - https://www.gutenberg.org/ebooks/31552

das palavras dos jornais identificadas como erradas eram, na verdade, flexões de outras palavras contidas no dicionário. Esse resultado já era esperado, uma vez que esse dicionário contém apenas a forma base das palavras. Além disso, algumas palavras corretas, e em sua forma base, estavam de fato ausentes desse léxico. Isso também era esperado, uma vez que os dicionários não pretendem cobrir o vocabulário completo da língua, apenas as formas mais frequentes.

O problema da ausência da forma base foi contornado oferecendo-se a opção de inclusão de palavras via interface (como já explicado na seção 4.4.3 acima). Já o caso de palavras flexionadas foi tratado através da construção de um dicionário no formato *Hunspell*, cujas entradas são compostas a partir do léxico de 1913 expandido através de regras de afixação. Essa abordagem, que é uma alternativa à inclusão no dicionário de palavras já flexionadas, assegura um funcionamento mais eficiente dos recursos do analisador e corretor em relação ao tempo de processamento, uso de memória, qualidade das sugestões do dicionário em caso de erro, e detecção efetiva de um erro.

Como apresentado na seção 3.2.6.1 do Capítulo 3, o *PyEnchant Hunspell* utiliza um dicionário no formato “lema + tags de afixos”. Cada entrada do dicionário contém um lema (forma base da palavra) e as *tags* (símbolos específicos) que indicam as regras de afixação que podem ser usadas para formar palavras flexionadas a partir daquele lema. As entradas do dicionário são armazenadas em um arquivo com extensão “.dic”, enquanto as regras de afixação são armazenadas em um arquivo “.aff”.

Quando executamos o corretor ortográfico, ele consulta o arquivo “.dic” para identificar a entrada correspondente à palavra sendo validada, e depois aplica as regras de afixação referentes aos símbolos associados a esse lema, a fim de verificar se a palavra em questão é uma flexão do lema encontrado.

4.4.4.3 Descrição do processo de adaptação do dicionário de 1913

De modo simples, poderíamos pensar que a adaptação do léxico de 1913 para o padrão *Hunspell* poderia ser feita de forma manual, já que consiste principalmente na inclusão de *tags* que indicam as regras de afixação de cada lema existente. Contudo, considerando a quantidade de vocábulos (mais de 128 mil) e a variedade das regras de afixação, fica claro que essa tarefa teria que ser

automatizada. Assim, foram implementados cinco processos para incluir automaticamente *tags* de afixação no léxico de 1913.

Inicialmente, as palavras constantes do léxico de 1913, nomeado aqui de “pt_1913”, foram armazenadas no arquivo “.dic”, que chamaremos de “pt_1913.dic”. A seguir, foi construído o arquivo “.aff”, que contém as regras de afixação a serem utilizadas pelo PyEnchant. Esse arquivo, chamado de “pt_1913.aff”, foi construído espelhando o arquivo “pt-BR.aff” do Português brasileiro atual, que contém mais de 27 mil linhas de regras de afixos. O arquivo pt-BR.aff foi obtido a partir do “Dicionário de Português do Brasil para Hunspell” (“pt-BR”) desenvolvido e publicado pelo projeto VERO - Verificador Ortográfico Livre[42] - em sua versão 3.2. Esse é um projeto desenvolvido em Software Livre utilizado no software LibreOffice[43], criado em janeiro de 2006, com objetivo de identificar e corrigir erros de ortografia (MOURA, 2021).

Após criados os dois arquivos, foi necessário um laborioso processo de atualização das 125.244 entradas do dicionário pt_1913.dic, de modo a inserir as *tags* de afixação de cada palavra. A *Figura 8* mostra que esse processo de adaptação contou com 5 passos, sendo necessário escrever um código diferente para cada um deles. Os passos realizados para a atualização dos termos base são descritos a seguir.

Figura 8 - Etapas de adaptação do dicionário 1913

Fonte: A autora (2023).

[42] VERO - https://pt-br.libreoffice.org/projetos/vero
[43] Dicionários LibreOffice - https://pt.libreoffice.org/descarregar/dicionarios/

1) **Comparação direta**: Essa etapa inicial verificou quais palavras do dicionário pt_1913.dic estavam contidas no dicionário pt-BR.dic sem nenhuma alteração. Foram identificadas 61.995 palavras nesta etapa, e foram adicionadas as respectivas *tags* de afixos. Restaram 62.313 palavras do dicionário pt_1913.dic sem afixos.
2) **Equivalência entre palavras**: A seguir, algumas palavras do pt_1913.dic ainda sem *tags* de afixação foram "espelhadas" em sua grafia atual, a fim de permitir a comparação direta com o dicionário pt-BR.dic. Algumas letras foram substituídas, e letras repetidas foram eliminadas. É importante informar que esse espelhamento não modificou as palavras originais. Ele foi realizado apenas em tempo de execução do código de etiquetagem de afixos deste passo. As ações realizadas foram:
    - Substituição de **ph** por **f**
    - Substituição de **th** por **t**
    - Substituição de **y** por **i**
    - Eliminação de letras repetidas

    Em seguida, foi realizada nova comparação direta das palavras modificadas com o dicionário pt-BR.dic (passo 1). Foram identificadas mais 10.615 palavras, e as respectivas *tags* de afixos foram adicionadas. Restaram ainda 51.698 palavras do dicionário pt_1913.dic sem afixos.
3) **Uso do PyEnchant com pt_1913 para eliminação de termos redundantes**: Nesta etapa, a versão parcial do dicionário pt_1913.dic construída a partir dos passos anteriores, cujas palavras já contêm as *tags* de afixos, foi utilizada como dicionário no corretor ortográfico PyEnchant. Em seguida, cada uma das 51.698 palavras que ainda restavam sem afixos foi submetida ao corretor ortográfico, a fim eliminar dessa lista as palavras que fossem identificadas como corretas pelo PyEnchant -- pois isso evidencia que essa palavra já é contemplada a partir da expansão (flexão) de alguma palavra do dicionário parcial pt_1913.dic sendo utilizado. Neste passo, foram eliminadas 22.722 palavras da lista de palavras que ainda restavam sem afixo. Após essa etapa, restaram 28.976 palavras do dicionário pt_1913.dic sem afixo.
    - Exemplo:

- **último/D** (palavra do dicionário parcial pt_1913.dic com afixos)
- **última** (palavra identificada como expansão de “último/D” e eliminada das palavras restantes sem afixo)

4) **Uso do PyEnchant com pt-BR para obtenção de afixos:** Nesta etapa, o dicionário pt-BR.dic foi utilizado como o dicionário no PyEnchant. Em seguida, cada uma das 28.976 palavras do pt_1913.dic que ainda restavam sem afixos foi submetida ao corretor ortográfico a fim de identificar as que estavam contempladas a partir da expansão (flexão) de alguma palavra do dicionário pt_BR.dic definido. Aqui, cada palavra restante sem afixo que foi identificada como correta pelo corretor ortográfico foi substituída pela entrada correspondente do dicionário pt_BR.dic (forma base e suas respectivas *tags* de afixo). Foram identificadas e substituídas 6.306 palavras neste passo. Restaram 22.670 palavras do dicionário pt_1913.dic sem afixo.
   - Exemplo:
     - **abadia** (termo do grupo de palavras restantes do pt_1913.dic sem afixos)
     - **abadiar/ajkLMY** (termo presente no dicionário pt-BR que gera o termo “abadia” após flexão e foi inserido no pt_1913.dic)

   É importante notar que esse passo tem o potencial de inserir entradas duplicadas no dicionário, uma vez que o pt_1913.dic contém algumas palavras flexionadas, oriundas da mesma forma base pela aplicação de afixos. Assim, foi necessário verificar novamente a existência de redundâncias no pt_1913.dic.

5) **Uso do PyEnchant para eliminação de termos redundantes:** Uma vez que o léxico do pt_1913.dic com os afixos cresceu significativamente, os passos (3) e (4) foram repetidos, para eliminar possíveis redundâncias, ou seja, palavras que seriam obtidas a partir da expansão de alguma palavra contida no dicionário parcial pt_1913.dic ou do pt-BR.dic. Foram eliminadas mais 4.353 palavras que ainda restavam sem afixos. Após essa etapa restaram 18.317 palavras do dicionário pt_1913.dic sem afixo.

Algumas das 18.317 restantes que escaparam aos processos implementados foram examinadas, e identificamos situações muito complexas para serem automaticamente resolvidas no momento, devido ao pouco tempo que resta para conclusão deste trabalho de mestrado. Vejam alguns exemplos a seguir:

- Termos compostos acentuados:
  - patrão-mór -> patrão-mor
  - pedra-íman -> pedra-ímã
- Termos que perderam o hífen:
  - outro-tanto, há-de-haver

Devido à falta de tempo para prosseguir manualmente com a inclusão de afixos nas palavras restantes, optamos por finalizar a construção do dicionário após o passo (5).

Observem que os termos restantes não podem ser eliminados do pt_1913.dic, pois eram corretos naquela época. Assim, eles foram mantidos no dicionário, porém sem regras de afixação. Em síntese, das 125.244 palavras iniciais do pt_1913.dic sem afixos, foram obtidas 81.900 palavras com regras de afixo definidas pelos passos descritos, mais as 18.317 palavras sem afixos restantes, totalizando 100.217 palavras no pt_1913.dic final. Todavia, ao contabilizar todas as possíveis flexões das palavras com *tags* de afixação, a versão final do dicionário pt_1913.dic construído abrange um total de 210.793 palavras. Assim, o total de palavras final do dicionário pt_1913.dic é aproximadamente 69% maior do que inicialmente. Espera-se que o expressivo incremento no léxico proporcione importante melhoria na eficiência da correção dos textos históricos obtidos do OCR.

A seção 6.2 do último capítulo deste documento apresenta sugestões para trabalhos futuros que pretendem cobrir as lacunas deixadas pelo nosso trabalho atual.

4.5 INDEXAÇÃO E ACESSO ÀS IMAGENS COM PALAVRAS-CHAVE

Esta seção se dedica a apresentar a etapa final do processo proposto, que se refere à indexação e ao acesso às imagens digitalizadas. Como essa etapa envolve técnicas da área de Recuperação de Informação (RI), iniciamos com uma breve apresentação dos conceitos básicos dessa área relativos à construção e uso de

sistema de RI tradicionais (seção 4.5.1). A seguir, a seção 4.5.2 descreve alguns detalhes do sistema de RI implementado neste trabalho.

4.5.1 Recuperação de Informação (RI)

A *Recuperação de Informação* (RI) tem por objetivo principal *facilitar o acesso* a documentos (itens de informação) relevantes à *necessidade de informação* atual do usuário (BAEZA-YATES; RIBEIRO-NETO, 2011). Assim como o PLN, a RI também pode ser considerada como uma área interdisciplinar, pois utiliza métodos e técnicas de várias áreas, tais como Ciência da Computação, Estatística, Matemática e Ciência da Informação (MANNING; RAGHAVAN; SCHÜTZE, 2008). Esta seção tem por objetivo trazer uma visão geral dessa área de pesquisa e desenvolvimento, além de citar algumas ferramentas disponíveis para auxiliar na construção desses sistemas.

Uma definição mais detalhada da área seria: A Recuperação de Informação é uma área de pesquisa e desenvolvimento que investiga métodos e técnicas para a representação, a organização, o armazenamento, a busca e a recuperação de itens de informação (BAEZA-YATES; RIBEIRO-NETO, 2011). Considerando um *corpus* de documentos (itens de informação) e uma consulta do usuário (geralmente representada por palavras-chave), queremos receber uma lista ordenada de documentos que são relevantes para a consulta.

Os sistemas de RI são comumente conhecidos como “Mecanismos de busca” ou “Engenhos de busca” (do inglês, *Search Engines*). Esses sistemas podem ser de uso livre, como os engenhos de busca na Web (e.g., Google), ou de uso restrito, como os sistemas privativos de empresas. Veremos a seguir um pouco sobre os sistemas de RI, destacando os temas de interesse do nosso trabalho de mestrado.

4.5.1.1 Arquitetura geral e Fases dos sistemas de RI

Uma definição mais atual e abrangente de engenhos de busca seria: Um sistema de RI pode ser visto como a parte do sistema de informação responsável pelo armazenamento ordenado de documentos (itens de informação) em um BD e sua posterior recuperação para responder à consulta do usuário (BAEZA-YATES e RIBEIRO-NETO, 2011). Essa definição corresponde ao cenário do nosso trabalho,

uma vez que o sistema de RI implementado não é uma aplicação *stand-alone*, e sim trata-se de mais um módulo de um processo maior.

Esses sistemas podem ser compostos por diversos módulos de processamento, tendo duas fases principais (BAEZA-YATES e RIBEIRO-NETO, 2011). Indexação dos documentos - Criação da Base de índices e Consulta à Base de índices de documentos. Essas fases serão descritas a seguir. Ressaltamos que o foco dessa apresentação está no processamento de documentos textuais.

4.5.1.1.1 Etapas da Fase 1 – Criação da Base de índices (Indexação dos documentos)

A Fase 1 se inicia com a **Aquisição (seleção)** dos documentos que serão indexados. Essa tarefa pode ser realizada de modo manual, semiautomático ou automático. No primeiro caso, o responsável pelo sistema seleciona manualmente os documentos de interesse. Os documentos também podem ser enviados ao responsável por outras pessoas que têm relação com a base. Citamos como exemplo a Biblioteca Digital da UFPE (BDTD/UFPE), que é parte do ATTENA[44], o repositório digital da UFPE. A BDTD armazena e disponibiliza as dissertações e teses de doutorado produzidas na UFPE, e esses arquivos são enviados para a Biblioteca pelos alunos depois de serem aprovados pela banca examinadora. Neste trabalho, a seleção dos documentos é manual, pois os documentos são obtidos a partir do grupo de pesquisa do laboratório LIBER.

Outros sistemas de RI utilizam *crawlers*, que são softwares capazes de navegar na Web ou em outras redes de computadores, e guardar em um computador local cópias dos documentos de interesse. Esses *crawlers* são usados para aquisição semiautomática ou automática de documentos. Esta é a opção adotada pelos engenhos de busca na Web, devido ao imenso volume de documentos a serem indexados, e também por se tratar da indexação de uma base em atualização contínua.

A seguir, vem a etapa de **Preparação (pré-processamento)**, cujo objetivo é criar uma representação estruturada dos documentos que seja adequada para

[44] ATTENA - https://repositorio.ufpe.br/static/jsp/apresentacao.jsp

processamentos futuros, visando sempre facilitar o acesso futuro aos documentos. A maioria dos sistemas adota representações internas na forma de listas ou vetores de palavras relevantes. Essa etapa consiste em um conjunto de *transformações* realizadas no texto do documento: Tokenização, *POS-tagging*, Remoção de *stopwords*, Lematização ou *Stemming*, representação na forma de *n-grams*, entre outras possibilidades. A escolha das operações que serão realizadas deve ser feita pelo desenvolvedor do sistema de RI, pois elas não são mandatórias (exceto a tokenização). A maioria dessas operações já foi apresentada na seção 2.3.2 do Capítulo 2. Assim, apenas a remoção de *stopwords* será descrita aqui.

A remoção de *stopwords* tem por objetivo é eliminar da representação final dos documentos as palavras sem relevância para o contexto do sistema atual. Geralmente, essas palavras são artigos, preposições, pronomes, e outras palavras sem significado semântico de valor para a RI. A eliminação das palavras pode ser baseada em uma lista previamente construída (*stoplist*) ou pode se basear em um *POS-tagger* (para eliminar palavras de determinadas classes gramaticais). Esta etapa devolve uma representação estruturada dos documentos de entrada (geralmente uma lista/vetor de palavras/termos), sendo que cada documento terá sua lista correspondente. A eliminação das *stopwords* ajuda a diminuir o tamanho dessas listas, mantendo as palavras mais relevantes para o contexto do sistema. Isso também irá afetar positivamente a precisão do sistema na Fase 2, de consulta.

A Fase 1 termina com a **Criação da base de índices**, que realiza a indexação de cada documento a partir das palavras/termos que constam da sua lista correspondente. Também é possível associar um “peso” (valor numérico) que representa a relevância de cada termo para representar cada documento sendo indexado. No nosso caso, os documentos a serem indexados são os textos extraídos e as imagens correspondentes. Porém, apenas as imagens serão retornadas através das consultas, na Fase 2.

4.5.1.1.2 Etapas da Fase 2 – Consulta à Base de índices

A Fase 2 se inicia com a **Construção da consulta (*query*)** pelo usuário, que irá usar a interface do sistema para digitar as palavras-chave que vão guiar a busca. Alguns sistemas também utilizam tesauros locais para estender as consultas automaticamente, a fim de recuperar mais documentos relevantes. Essa opção foi

considerada no nosso trabalho, como um caminho para fazer a atualização vocabular dos textos extraídos. Essa ideia será apresentada no Capítulo 6, como possibilidade de trabalho futuro.

Com base na consulta, o sistema realiza a **Busca e Recuperação** dos documentos que “casam” (match) com a consulta atual. Assim, quando o usuário digita uma palavra, o sistema de RI devolve a lista de documentos associados a aquela palavra. Se a consulta tiver mais de uma palavra, as listas de documentos recuperados são combinadas e apenas uma lista final é passada para a etapa de ordenação, a seguir.

A etapa de **Ordenação dos resultados** tem por objetivo ranquear a lista final de acordo com a relevância de cada documento com relação à consulta do usuário. Para ser possível ordenar essa lista, é necessário guardar o peso de cada token para representar cada documento. Os documentos que tiverem maiores pesos associados às palavras da consulta serão ordenados no topo da lista. Por fim, temos a **Apresentação dos resultados** através da interface do sistema. Nessa fase, o usuário poderá ter acesso direto aos documentos que ele selecionar através da interface.

4.5.1.2 Ferramentas de RI

Assim como a área de PLN conta com várias ferramentas de auxílio ao processamento de texto (seção 2.3.2.3), a área de RI também dispõe de ferramentas e bibliotecas para auxiliar na criação de sistemas de indexação e busca de documentos através de palavras-chave (também chamados de “engenhos de busca” – do inglês “*search engines”*). Vale ressaltar que existem softwares para indexação de vários tipos de dados, não apenas texto. Atualmente, podemos contar com ferramentas de recuperação de imagens a partir de características como cor e formas, indexação de áudio e de vídeos também, porém estão fora do nosso escopo.

Existem diversas opções de ferramentas e bibliotecas para construção de engenhos de busca, algumas pagas e outras para livre uso, sendo algumas também

open-source[45]. Dentre elas, destacamos as ferramentas/bibliotecas oferecidas pela *Apache Software Foundation*[46], que são as mais populares nessa área (*Lucene* e *Solr*).

De acordo com o site da fundação, o projeto *Apache Lucene*[47], desenvolve software *open-source* para criação de engenhos de busca através da biblioteca *Lucene core*. *Lucene core* é uma biblioteca de livre uso *open-source* que foi originalmente escrita em Java. *Apache* oferece também a biblioteca *PyLucene*, para fazer a comunicação com o *Lucene*.

O *Apache Solr*[48] é uma plataforma para desenvolvimento de engenhos de busca que nasceu como um subprojeto da *Apache Lucene*, tendo se destacado por seu excelente desempenho, flexibilidade e facilidade de uso. De acordo com o site do produto, o *Solr* oferece busca e navegação em muitos dos maiores sites na Internet, tendo uma comunidade de desenvolvimento ativa e com atualizações regulares. O *Solr* permite armazenar, indexar e recuperar grandes volumes de dados de maneira rápida e precisa, sendo escalável, tolerante a falhas, entre outras capacidades. O *Solr* trabalha com documentos em formato JSON, XML, CVS, entre outros.

Por fim, citamos aqui o *Elasticsearch, "*um mecanismo de busca e análise de dados distribuído, gratuito e aberto para todos os tipos de dados, incluindo textuais, numéricos, geoespaciais, estruturados e não estruturados"[49]. Ele é um componente do Elastic Stack, que oferece ferramentas gratuitas para armazenamento, análise e visualização de dados. Esse engenho de busca foi construído com base no *Apache Lucene*, porém não é um projeto da *Apache Foundation.* Esse software é da empresa *Elastic*[50]. O *Elasticsearch* oferece suporte às linguagens Java, JavaScript (Node.js), PHP, Python, entre outras.

Como já mencionado, optamos aqui pelo uso da plataforma *Apache Solr*, acessada através do *PySolr.* Essa biblioteca em Python oferece uma interface de fácil utilização para interagir com o *Solr*, atuando como uma camada de abstração sobre o mecanismo de indexação e busca. O *PySolr* disponibiliza uma API Python

---

[45] Engenhos de busca - https://en.wikipedia.org/wiki/List_of_search_engine_software
[46] Apache Software Foundation - https://www.apache.org/
[47] Apache Lucene - https://lucene.apache.org/
[48] Apache Solr - https://solr.apache.org/
[49] Elastic - https://www.elastic.co/pt/what-is/elasticsearch
[50] Elastic - https://www.elastic.co/pt/

que simplifica o envio de consultas ao *Solr* e o processamento dos resultados retornados diretamente no ambiente Python.

A comunicação entre o *PySolr* e o *Solr* é realizada por meio de requisições HTTP, com o *PySolr* enviando as consultas e recebendo os resultados para manipulação no ambiente Python. Essa interface consiste em uma coleção de métodos e classes que permitem interagir com o *Solr* de maneira simples e conveniente, abstraindo os detalhes da comunicação HTTP com o servidor. Através do *PySolr* é possível executar operações de indexação, pesquisa e recuperação de documentos de forma eficiente. Essa biblioteca simplifica o processo de interagir com o *Solr* e aproveitar todo o potencial dessa ferramenta de busca em projetos *Python*.

4.5.2 Etapa de Indexação das imagens via PySolr

Esta seção descreve a etapa final do processo geral, que consiste na indexação das imagens digitalizadas a fim de possibilitar consultas via palavras-chave. É importante frisar que cada imagem tratada apresenta apenas uma página de jornal, e não sua edição completa. Assim, as palavras extraídas de cada imagem via OCR correspondem apenas aos termos que ocorrem naquela página. Isso possibilitou a criação de uma correspondência mais precisa entre palavras e imagens, facilitando a indexação da imagem a parir dos termos que de fato ocorrem naquela página.

O resultado final foi a criação de uma base de imagens indexadas através de palavras e termos compostos, que podem ser recuperadas através de uma interface simples baseada em palavras-chave (ver Imagem *29*). Esse engenho de busca local provê acesso simplificado ao conteúdo dos jornais centenários digitalizados, cumprindo assim com mais um dos objetivos iniciais deste trabalho de mestrado.

Como já mencionado, utilizamos aqui a biblioteca *PySolr* para interagir com o *Solr* por meio do Python. Essa escolha nos permitiu indexar os dados de forma eficiente e realizar consultas de maneira direta, tornando o processo mais ágil e eficaz para acessar os conteúdos desses documentos históricos.

Antes de iniciar o processo de indexação, é necessário configurar a plataforma *Solr* de acordo com os requisitos do sistema que será construído. Essas escolhas são registradas no arquivo de configuração do núcleo do *Solr*, chamado de

“*CORE*”. As configurações do *CORE* permitem um controle detalhado sobre diversos aspectos da indexação e busca, como tarefas de análise de texto a serem executadas (e.g., eliminação de stopwords, stemming etc.), relevância dos resultados (e.g., binária ou baseada em pesos), entre outros. Essa configuração é essencial para otimizar os resultados e atender às necessidades específicas do sistema de indexação e busca.

O arquivo *managedschema.xml* é usado para configurar o *CORE* para executar um pré-processamento dos dados indexados e das consultas enviadas ao sistema. No entanto, em nosso caso, optamos por não utilizar o pré-processamento disponível.

Em vez disso, aproveitamos um recurso que nos permite definir campos com metadados para a indexação das imagens. Assim, garantimos que cada ID atribuído seja único, evitando duplicidade dos documentos indexados. Essa abordagem é essencial para garantir que os resultados das buscas sejam precisos e livres de ambiguidades. Ao personalizar a configuração do *Solr* de acordo com as características dos dados e das necessidades do sistema, alcançamos um desempenho mais eficiente e uma experiência de busca mais aprimorada.

No projeto geral, as imagens digitalizadas das páginas dos jornais são identificadas individualmente pela sigla do nome do jornal, sua data de publicação e a página constante da imagem. Por exemplo, a página 4 do Diário da manhã do dia 25 de outubro de 1930 será nomeado como “DM_25_10_1930P4”. Esse padrão foi mantido nos arquivos armazenadas no diretório local acessado pelo *Solr*. Isso possibilitou a criação de uma correspondência entre palavras extraídas e imagens, bem como a indexação das imagens através dessas palavras. Assim, tornou-se possível identificar e recuperar as imagens a partir das consultas através de palavras-chave.

Como já mencionado, o *Solr* aceita dados em vários formatos, como XML, JSON, CSV, entre outros. Neste trabalho, utilizamos o formato JSON *(JavaScript Object Notation*) para representar os dados a serem indexados. Esse formato é definido utilizando a estrutura de dados “dicionário” em Python. Assim, cada dicionário representa uma unidade básica de informação a ser indexada (i.e., uma imagem). No *Solr*, essa unidade básica de informação é denominada de ***documento***. A seguir é apresentado um exemplo de um arquivo no formato JSON com os documentos a serem indexados.

```
newspapers = [
  {
    "id": "1",
    "title": "DM_25_10_1930P4",
    "content": "O conteúdo do arquivo de texto extraído da imagem.",
    "img_path": "C:\\Caminho\\Para\\O\\Diretorio\\imagem.tif"
  },
  {...}
  ]
```

No array ***newspapers,*** cada item é um objeto JSON contendo as chaves ***id, title, content e img_path***, e os valores associados a cada chave são preenchidos com informações específicas de cada documento. Em nosso contexto, cada página do jornal digitalizada é um documento indexado.

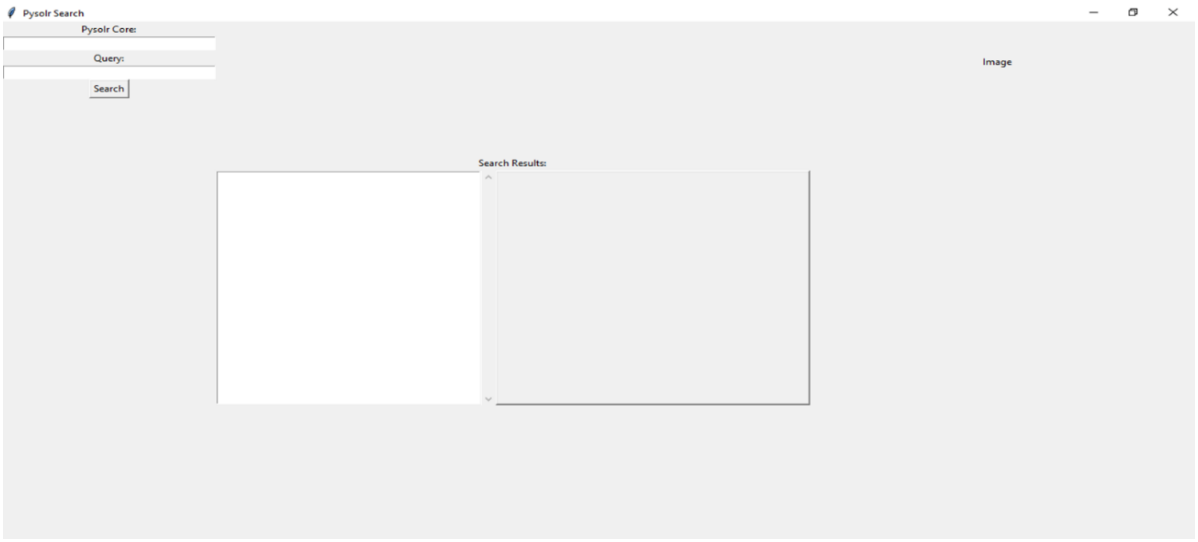
O campo ***id*** é utilizado como um identificador exclusivo para cada documento, garantindo que não haja duplicidade. O campo ***title*** contém as iniciais do jornal, a data de publicação e a página associada ao documento. O campo ***content*** armazena o conteúdo do arquivo de texto extraído da imagem da página do jornal, enquanto o campo ***img_path*** armazena o caminho do diretório onde a respectiva imagem está armazenada. Essa abordagem nos permite organizar e indexar os dados de forma estruturada, possibilitando buscas eficientes e precisas no *Solr*.

A fim de possibilitar o fácil acesos aos documentos indexados, foi implementada aqui uma interface para realização de buscas eficientes a partir de palavras-chave dos documentos indexados via *PySolr*. A interface foi desenvolvida utilizando a biblioteca de interface gráfica de usuário (GUI) Python chamada *Tkinter*[51].

A Imagem 3 apresenta a interface desenvolvida para a fase de busca dos dados indexados a partir de palavras-chave. Conforme pode ser observado, a interface atual é funcional, porém ainda relativamente simples, e a busca é realizada em uma base de imagens armazenada localmente.

[51] Tkinter - https://docs.python.org/3/library/tkinter.html

Imagem 3 - Interface desenvolvida para a fase de busca.



Fonte: A autora (2023).

No âmbito do projeto geral executado pelo grupo do laboratório LIBER existe um grande número jornais históricos disponíveis, muitos já digitalizados. Contudo, por falta de tempo, nosso trabalho utilizou apenas algumas páginas de jornal para testar o sistema de indexação e busca desenvolvidos. Além disso, a estrutura de indexação e busca desenvolvida no nosso trabalho não seria capaz de comportar esse número tão elevado de documentos.

Assim, como trabalho futuro, está indicado o uso de uma estrutura de armazenamento mais robusta, como um banco de dados. Isso iria viabilizar o acesso e a disponibilização desses documentos ao público em geral, por meio da criação de uma página web com uma interface de consulta abrangendo todo acervo dos jornais digitalizados.

4.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o processo proposto para correção e melhoria do texto extraído de imagens, concluindo com a etapa de indexação das imagens a partir do texto corrigido.

O capítulo 5, a seguir, irá descrever dois estudos de caso realizado a fim de testar o processo implementado, observando os resultados obtidos em cada etapa do processo geral.

5 ESTUDOS DE CASO

Este Capítulo apresenta exemplos de aplicação do protótipo desenvolvido em imagens digitalizadas de dois jornais centenários diferentes. A seção 5.1 apresenta os resultados obtidos com páginas do jornal “Diário da Manhã”, e na seção 5.2 veremos um exemplo com o jornal “Diário de Pernambuco”. Na seção 5.3 comentaremos sobre os resultados obtidos com o processo de correção pós-OCR e da etapa de indexação das imagens.

Devido ao tamanho considerável das páginas dos jornais digitalizados, apresentaremos apenas um trecho de cada página nos estudos de caso. Isso permitirá uma comparação mais adequada entre as imagens resultantes da aplicação de cada função de pré-processamento da imagem, bem como do texto extraído a partir da imagem original e da imagem pré-processada. Por fim, veremos também a correção final do texto extraído.

5.1 JORNAL DIÁRIO DA MANHÃ

Este primeiro estudo de caso foi realizado com base em uma página do jornal Diário da Manhã publicado no dia 25 de outubro de 1930. Essa edição é relativamente recente quando comparada ao jornal Diário de Pernambuco utilizado no segundo estudo de caso, que data de 1840. Talvez por ser mais recente, a página do jornal digitalizada encontra-se em bom estado de conservação, provendo uma imagem digitalizada mais nítida.

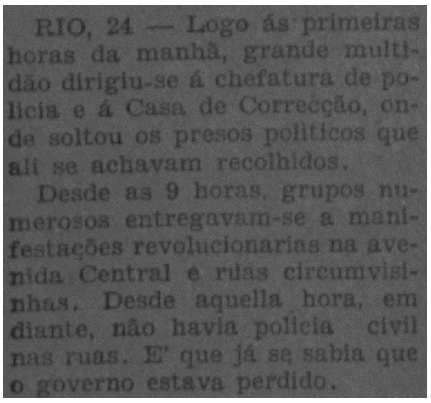
Contudo, quando comparada aos jornais mais antigos, essa página de jornal apresenta uma diagramação mais difícil de ser interpretada pelo OCR, uma vez que traz linhas verticais e horizontais dividindo as notícias, bem como imagens de desenhos e fotografias misturadas ao texto. Essas características acabam por dificultar a segmentação, provocando assim mais erros na extração do texto via OCR.

5.1.1 Pré-Processamento da imagem

A Imagem 4 apresenta a imagem do trecho do jornal que será utilizada neste estudo. As imagens subsequentes ilustram a aplicação de cada função

implementada na etapa de pré-processamento da imagem (ver seção 4.2.), a fim de exemplificar o efeito de cada passo desse pré-processamento.

Imagem 4 - Imagem original de um trecho de página do jornal Diário da Manhã.

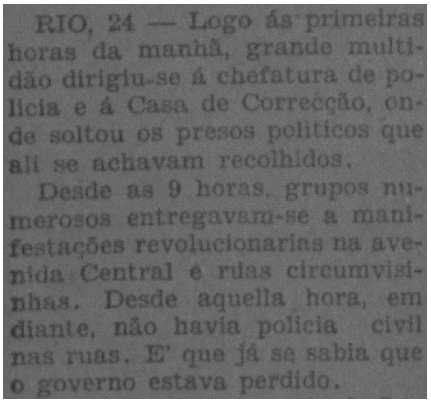

RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Ajuste de Gamma: A Imagem 5 apresenta a imagem com o Gamma ajustado. Utilizamos um fator de ajuste do Gamma de 1.2 nesta página de jornal específica. Apesar da diferença com relação à imagem original ser sutil, notamos uma melhoria na visibilidade da imagem, e consequentemente, na qualidade do texto extraído a partir dessa imagem.

Imagem 5 - Ajuste do Gamma da imagem.


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

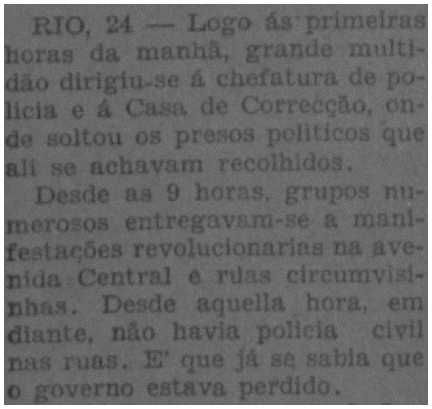
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Conversão para escalas de cinza: A Imagem 6 mostra a imagem obtida a partir da conversão da imagem colorida para outra em escalas de cinza. Neste caso, não notamos diferença entre esta imagem e a imagem da etapa anterior porque o jornal não era colorido. Ainda assim, esta é uma etapa necessária, pois os processos

subsequentes recebem uma imagem em escalas de cinza, ou seja, com apenas um canal de cor.

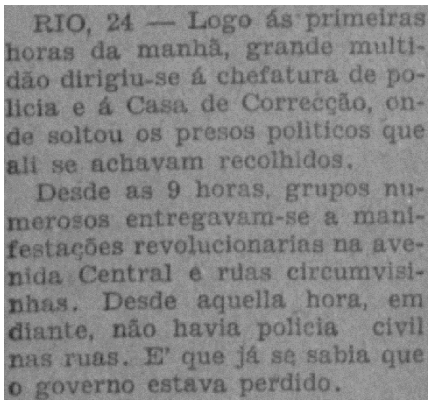
Imagem 6 - Conversão para escalas de cinza.


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Ajuste de Contraste: A Imagem *7* mostra a imagem normalizada de forma a aprimorar o contraste entre o plano de fundo e as letras, auxiliando na extração do texto. Isso é feito expandindo a diferença entre o branco e o preto na imagem para seus limites máximo e mínimo, respectivamente.

Imagem 7 - Normalização da imagem.

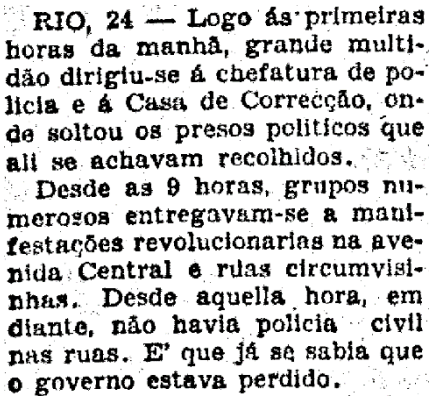

RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Binarização: A Imagem 8 apresenta a imagem binarizada a partir do método de *binarização gaussiano adaptativo*. O método de binarização *cv2.adaptiveThreshold()* é uma função da biblioteca *OpenCV* em Python utilizada para converter uma imagem em escala de cinza em uma imagem binária. Ele utiliza uma abordagem adaptativa para calcular o valor de limiar para cada pixel com base em sua vizinhança local. O limiar adaptativo é calculado como a média ponderada dos valores dos pixels na vizinhança local. Se o valor do pixel central for maior que esse

limiar, o pixel é definido como branco; caso contrário, é definido como preto, resultando em uma imagem binária segmentada.

Imagem 8 - Binarização da imagem.



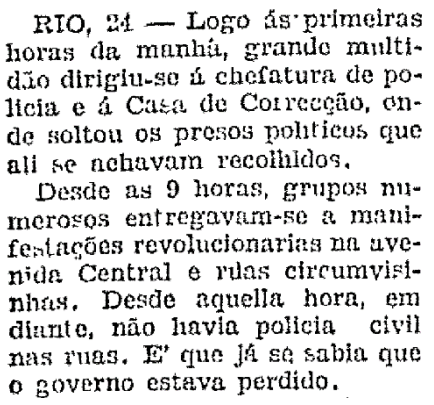
RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Filtro de área: Um filtro de área foi aplicado à imagem binarizada de forma a remover o ruído presente no passo anterior. Foi definida uma área mínima de 15x15 pixels para a aplicação da filtragem. O resultado é apresentado na Imagem 9 a seguir.

Imagem 9 - Filtro de área.



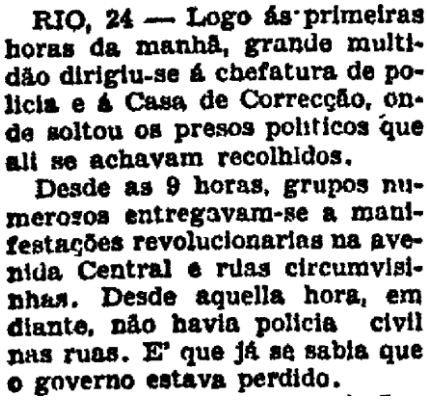
RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Dilatação: A seguir foi realizada uma operação morfológica de dilatação para aumentar a espessura dos traços de cada caractere, conforme apresentado na Imagem 10.

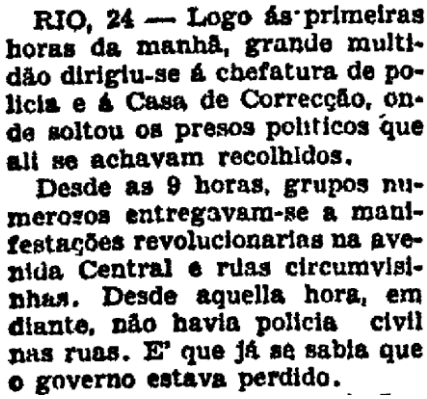
Imagem 10 - Operação morfológica de dilatação.


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Fechamento de elementos: A seguir foi realizada uma operação morfológica para reconectar algumas regiões nos caracteres que foram segmentadas nos passos anteriores, obtendo assim caracteres mais contínuos e com menos lacunas internas. O resultado desta operação está apresentado na Imagem 11.

Imagem 11 - Operação morfológica de fechamento de elementos.


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

É importante destacar que algumas das técnicas implementadas no nosso trabalho não foram aplicadas neste exemplo específico de página de jornal, pois verificamos que o uso delas não contribuiu com o desempenho da extração do texto. As funções de pré-processamento de imagem implementadas no trabalho que optamos por não utilizar foram o método de divisão de ganho, erosão e remoção de ruído via suavização.

5.1.2 Optical Character Recognition (OCR)

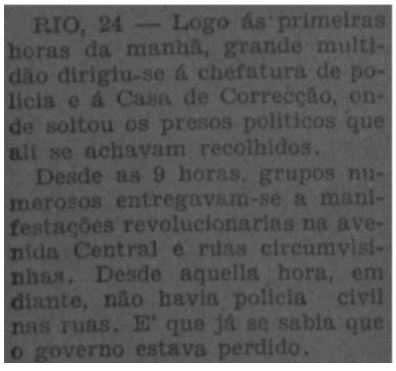
Conforme apresentado na seção 4.3, os textos contidos na imagem foram extraídos utilizando a biblioteca *Tesseract OCR*. Para melhorar a precisão e o controle da extração, foi aplicada uma configuração personalizada com os parâmetros --oem 1 e --psm 1.

O parâmetro --oem 1 seleciona o motor OCR baseado em LSTM, que geralmente produz resultados mais precisos em comparação com o modo *Legacy*. Já o parâmetro --psm 1 especifica o modo de segmentação automática da imagem com orientação de texto e script, de modo que o *Tesseract* tentará detectar automaticamente o layout das páginas e tentará identificar áreas de texto.

As Imagem 12(a) e 13(b) apresentam o mesmo trecho selecionado do jornal, que foram utilizadas para comparar o efeito do pré-processamento externo da imagem na extração do texto via OCR: (a) imagem original e (b) imagem pré-processada. Relembramos que o Tesseract OCR também realiza passos de pré-processamento interno das imagens, porém não temos acesso à imagem resultante desse processo interno. Assim, a Imagem abaixo apresenta apenas as imagens que são fornecidas como entrada para o Tesseract OCR.

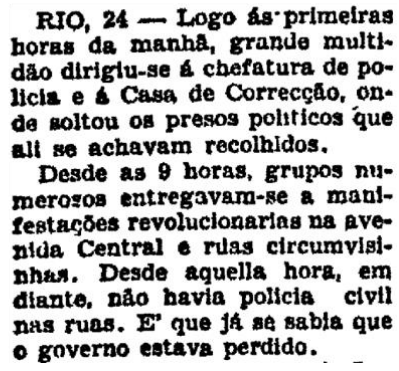
Imagem 12 - Imagens utilizadas para realização da extração do texto (a) Imagem original e (b) Imagem pré-processada.

a)


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

b)


RIO, 24 — Logo ás primeiras horas da manhã, grande multidão dirigiu-se á chefatura de policia e á Casa de Correcção, onde soltou os presos politicos que ali se achavam recolhidos.

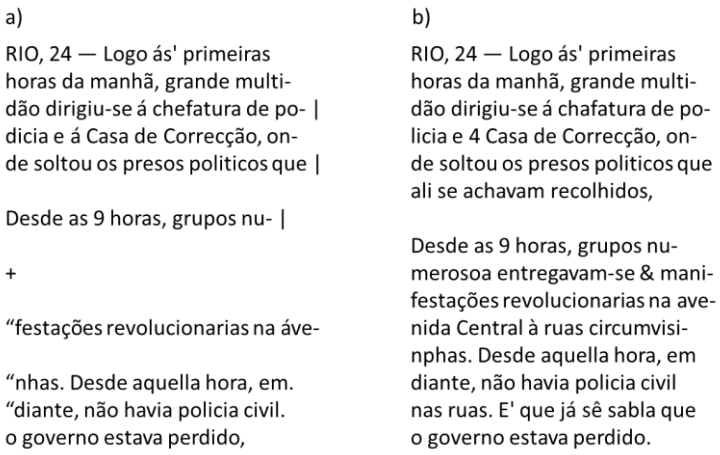
Desde as 9 horas, grupos numerosos entregavam-se a manifestações revolucionarias na avenida Central e ruas circumvisinhas. Desde aquella hora, em diante, não havia policia civil nas ruas. E' que já se sabia que o governo estava perdido.

Fonte: A autora (2023).

Os textos extraídos de cada imagem são apresentados na Imagem 13. Verifica-se que o texto obtido da imagem original possui diversos caracteres ruidosos, linhas em branco, palavras com erros e até mesmo uma linha que não foi reconhecida (“nas ruas. É que já se sabia que”). Já a extração a partir da imagem

pré-processada resultou em um texto com menos ruídos e mais bem segmentado. Embora o texto extraído contenha palavras com erros do OCR, claramente o resultado foi melhor que o obtido a partir da imagem original.

Imagem 13 - Texto extraído a partir de (a) Imagem original e (b) Imagem pré-processada.



a)

RIO, 24 — Logo ás' primeiras
horas da manhã, grande multi-
dão dirigiu-se á chefatura de po- |
dicia e á Casa de Correcção, on-
de soltou os presos politicos que |

Desde as 9 horas, grupos nu- |

+

“festações revolucionarias na áve-

“nhas. Desde aquella hora, em.
“diante, não havia policia civil.
o governo estava perdido,

b)

RIO, 24 — Logo ás' primeiras
horas da manhã, grande multi-
dão dirigiu-se á chafatura de po-
licia e 4 Casa de Correcção, on-
de soltou os presos politicos que
ali se achavam recolhidos,

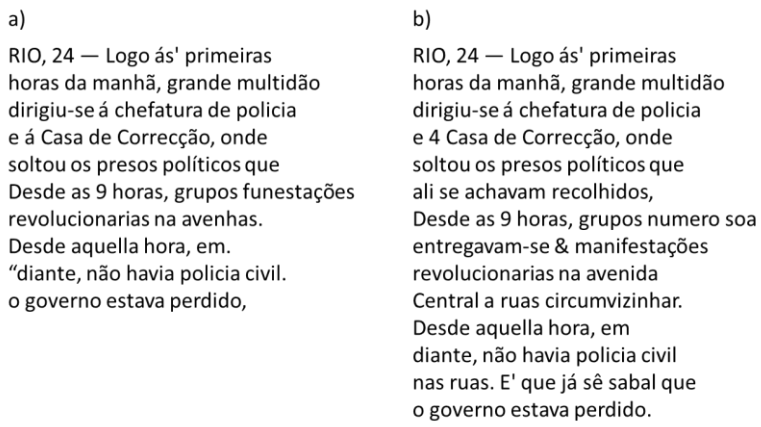
Desde as 9 horas, grupos nu-
merosoa entregavam-se & mani-
festações revolucionarias na ave-
nida Central à ruas circumvisi-
nphas. Desde aquella hora, em
diante, não havia policia civil
nas ruas. E' que já sê sabla que
o governo estava perdido.

Fonte: A autora (2023).

5.1.3 Correção Pós-OCR

Os textos extraídos foram submetidos à etapa de correção pós-OCR descrita na seção 4.4. A Imagem 14 apresenta o resultado da aplicação das funções de remoção de caracteres ruidosos, tratamento da translineação e correção ortográfica automática com uso do dicionário de 1913 desenvolvido.

Imagem 14 - Resultado da correção Pós-OCR aplicada ao texto obtido a partir da (a) imagem original (b) imagem pré-processada.



a)

RIO, 24 — Logo ás' primeiras
horas da manhã, grande multidão
dirigiu-se á chefatura de policia
e á Casa de Correcção, onde
soltou os presos políticos que
Desde as 9 horas, grupos funestações
revolucionarias na avenhas.
Desde aquella hora, em.
“diante, não havia policia civil.
o governo estava perdido,

b)

RIO, 24 — Logo ás' primeiras
horas da manhã, grande multidão
dirigiu-se á chefatura de policia
e 4 Casa de Correcção, onde
soltou os presos políticos que
ali se achavam recolhidos,
Desde as 9 horas, grupos numero soa
entregavam-se & manifestações
revolucionarias na avenida
Central a ruas circumvizinhar.
Desde aquella hora, em
diante, não havia policia civil
nas ruas. E' que já sê sabal que
o governo estava perdido.

Fonte: A autora (2023).

Neste exemplo, é possível observar a melhoria no texto extraído a partir da correção realizada. Porém, também houve casos de palavras identificadas como incorretas que foram corrigidas para outras que não correspondem ao texto original, como "numerosoa", que foi corrigida para “numero soa” ao invés de “numerosos”. Neste caso, o corretor ortográfico considerou que o erro identificado na palavra foi devido à falta de espaço entre duas palavras do dicionário (“numero” e “soa”). A seção 5.3.2 apresenta uma análise mais detalhada do texto corrigido.

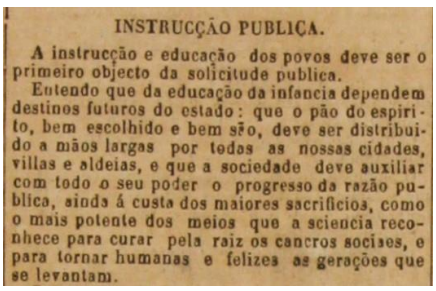
5.2 JORNAL DIÁRIO DE PERNAMBUCO

Nesta seção apresentaremos um exemplo semelhante ao anterior, porém utilizando um trecho de página do jornal Diário de Pernambuco do dia 02 de abril de 1861. Essa edição é cerca de 70 anos mais antiga que a do jornal Diário da Manhã apresentada na seção 5.1. Podemos observar aqui considerável diferença no estado de conservação desse jornal, o que impactou na escolha dos parâmetros e funções utilizadas no pré-processamento da imagem, bem como no resultado obtido.

5.2.1 Pré-Processamento da imagem

O trecho do jornal que será utilizado para ilustrar a ação de cada técnica de pré-processamento executada é apresentado na Imagem 15.

Imagem 15 - Imagem original de trecho da página do jornal diário de Pernambuco.



INSTRUCÇÃO PUBLICA.

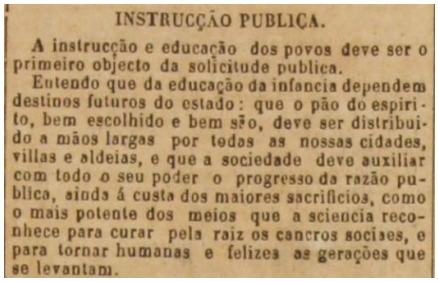
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem são, deve ser distribuido a mãos largas por todos as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Ajuste de Gamma: A Imagem 16 apresenta a imagem com o resultado da aplicação desta função. Utilizamos um fator de ajuste do Gamma de 1.16. nesta página. Embora a diferença com relação à imagem original seja sutil, ainda assim observamos uma melhoria na visibilidade da imagem, e consequentemente, na qualidade do texto extraído a partir dessa imagem.

Imagem 16 - Ajuste do Gamma da imagem.


INSTRUCÇÃO PUBLICA.

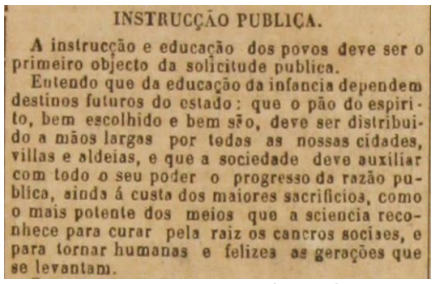
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem sêo, deve ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Ajuste de Contraste: A Imagem *17* mostra a imagem normalizada de forma a aprimorar o contraste entre o plano de fundo e as letras.

Imagem 17 - Normalização da imagem.


INSTRUCÇÃO PUBLICA.

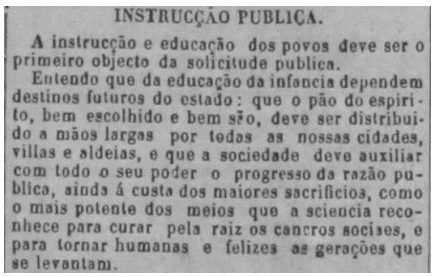
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem sêo, deve ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Conversão para escalas de cinza: Em seguida, a Imagem 18 mostra conversão da imagem de uma imagem colorida para outra em escalas de cinza.

Imagem 18 - Conversão para escalas de cinza.


INSTRUCÇÃO PUBLICA.

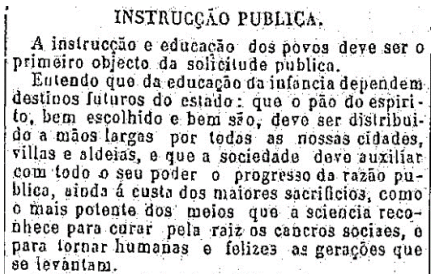
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem sêo, deve ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Binarização: a Imagem 19 apresenta a imagem binarizada a partir do método *Sauvola* com os parâmetros *window_size = 3* e *k=0.03*. Essa função transforma a imagem em escalas de cinza para uma com apenas dois níveis, preto e branco, através da aplicação da fórmula apresentada na seção 4.2.5 em cada pixel da imagem. Como é possível observar, com os parâmetros usados foi possível obter uma boa definição dos caracteres. Contudo, também observamos a introdução de ruído na imagem resultante, que precisou ser tratado nos passos subsequentes.

Imagem 19 - Binarização da imagem.


INSTRUCÇÃO PUBLICA.

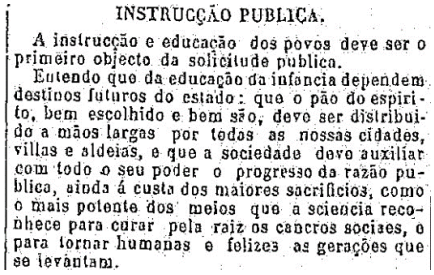
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem são, devo ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Abertura de elementos: A seguir Imagem 20, foi realizada uma operação morfológica para desconectar algumas regiões que apareciam conectadas de forma indesejada nos caracteres em consequência dos passos anteriores, obtendo-se assim maior definição nos caracteres.

Imagem 20 - Operação morfológica de abertura de elementos.


INSTRUCÇÃO PUBLICA.

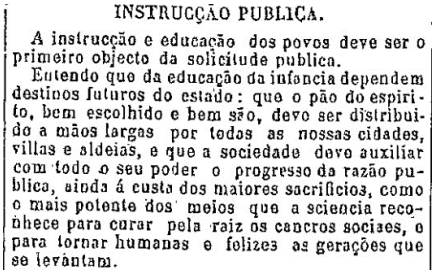
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem são, devo ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Filtro de área: Um filtro de área foi aplicado na imagem obtida no passo anterior, de forma a remover o ruído ainda presente na imagem. Foi definida uma área mínima de 3x3 pixels para a aplicação do filtro. O resultado é apresentado na Imagem 21.

Imagem 21 - Aplicação de filtro de área na imagem.


INSTRUCÇÃO PUBLICA.

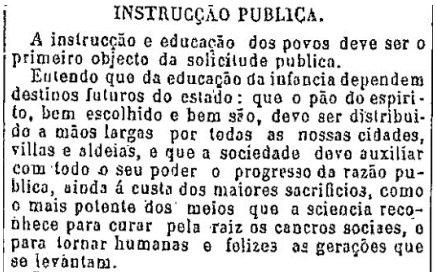
A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem são, devo ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levantam.

Fonte: A autora (2023).

Fechamento de elementos: Aqui também foi realizada uma operação morfológica para reconectar algumas regiões que foram segmentadas nos passos anteriores, obtendo-se assim caracteres mais contínuos e com menos lacunas internas. O resultado desta operação está apresentado na Imagem 22.

Imagem 22 - Operação morfológica de fechamento de elementos.



INSTRUCÇÃO PUBLICA.

A instrucção e educação dos povos deve ser o primeiro objecto da solicitude publica.

Entendo que da educação da infancia dependem destinos futuros do estado: que o pão do espirito, bem escolhido e bem são, devo ser distribuido a mãos largas por todas as nossas cidades, villas e aldeias, e que a sociedade deve auxiliar com todo o seu poder o progresso da razão publica, ainda á custa dos maiores sacrificios, como o mais potente dos meios que a sciencia reconhece para curar pela raiz os cancros sociaes, e para tornar humanas e felizes as gerações que se levàntam.

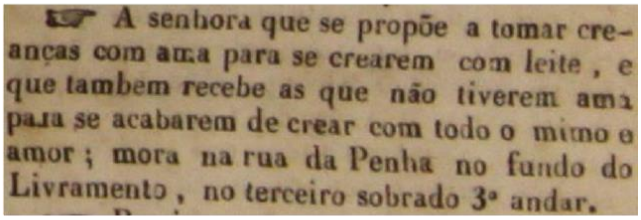
Fonte: A autora (2023).

5.2.2 Optical Character Recognition (OCR)

Neste estudo de caso, um trecho de página do jornal Diário de Pernambuco do dia 08 de julho de 1840, também foi aplicada uma configuração personalizada para o OCR através dos parâmetros --oem 1 e --psm 1. As Imagem *23*(a) e 24(b) apresentam o mesmo trecho selecionado do jornal, e foram utilizadas para comparar o efeito do pré-processamento externo da imagem na extração do texto via OCR: (a) imagem original e (b) imagem pré-processada.

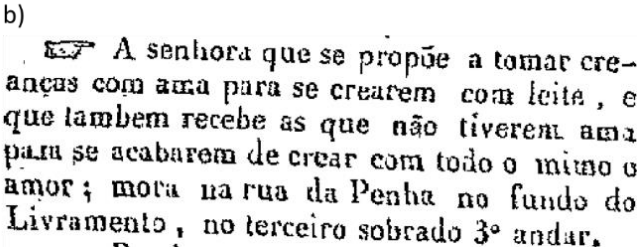
Imagem 23 - Imagens utilizadas para realização da extração do texto (a) Imagem original e (b) Imagem pré-processada.

a)



☞ A senhora que se propõe a tomar creanças com ama para se crearem com leite, e que tambem recebe as que não tiverem ama para se acabarem de crear com todo o mimo e amor; mora na rua da Penha no fundo do Livramento, no terceiro sobrado 3º andar.

b)



☞ A senhora que se propõe a tomar creanças com ama para se crearem com leite, e que tambem recebe as que não tiverem ama para se acabarem de crear com todo o mimo e amor; mora na rua da Penha no fundo do Livramento, no terceiro sobrado 3º andar.

Fonte: A autora (2023).

Na Imagem 24, apresentamos os textos extraídos a partir das imagens original e pré-processada.

Imagem 24 - Texto extraído a partir de: (a) Imagem original e (b) Imagem pré-processada.

a)

```
Os fundos portuguezes os 5 por 100 ticam a de Menezes junior, rua do Colegio , ena| ES" A senhora que se propõe a tomar cre-
351/82 5/8, os 3 por roca 34 1/8. Os botica de joão Moreira Marques , rua do Ca jancas com ama para se crearem com leite Ss
Hespanhoes a 28 /8, t bugá ; ena Boa-Vista , no Atterro » loja de que tambem recebe as que não tiverem ama
À imprensa franceza se occupa ácerca do Mauoel de Souza Rapozo pasa se acabarem de crear com todo o mimo e
tractado com esta unção , e sobre as bases que —"- amor ; mora na rua da Penha no fundo do
acima lhe aporito fazem varias reflexões. À LOTERIA DO THEATRO, Livramento , no terceiro sobrado 3º andar.
```

b)

```
. EF A senhora que se propõe a tomar cre-
anças com ama para se crearem com leite e
que tambem recebe as que não tiverem ama
pasa se acabarem de cerear com todo o mito &
JaAmor; mora narua da Penha no fundo do

Livramento , no terceiro sobrado 3º andar,
```

Fonte: A autora (2023).

Verifica-se que o texto extraído da imagem original não foi segmentado de forma bem-sucedida, resultando em uma informação confusa e de difícil interpretação. Já a imagem pré-processada foi segmentada de forma correta, embora o texto extraído ainda contenha palavras com erros do OCR. Desta forma, verificamos o potencial que o pré-processamento externo ao OCR possui no resultado final do texto extraído.

5.2.3 Correção Pós-OCR

Similarmente ao exemplo da seção 5.1.3, o texto extraído foi submetido à etapa de correção pós-OCR e o resultado é apresentado na Imagem 25.

Imagem 25 - Correção pós-OCR aplicada à imagem escaneada original. a) Texto extraído e b) Texto corrigido.

a)

```
ES A senhora que se propõe a tomar crê35182
58, os 3 por roca 34 18. Os botica de joão Moreira Marques , rua do Ca canjas com ama para se crearem com leite Ás
Hespanhol a 28 8, t bogá ena Boa-Vista , no Aterro loja de que também recebe as que não tiverem ama
A imprensa francesa se occupa ácer ca do Mameluco de Souza Raposo paca se acabarem de crear com todo o mimo e
tratado com esta unção , e sobre as bases que - amor mora na rua da Penha no fundo do
acima lhe a porito fazem varias reflexões. A LOTERIA DO THEATRO, Livramento , no terceiro sobrado 3 andar.
```

b)

```
EFÓ A senhora que se propõe a tomar crenças
com ama para se crearem com leite e
que também recebe as que não tiverem ama
paca se acabarem de cere ar com todo o meto
Amor mora na rua da Penha no fundo do
Livramento , no terceiro sobrado 3a andar,
```

Fonte: A autora (2023).

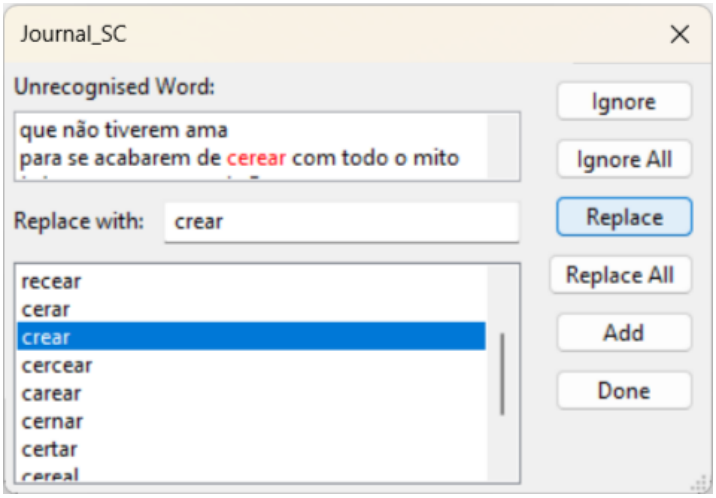
Neste exemplo, é possível observar a correção da translineação e a remoção de caracteres ruidosos. No entanto, a correção ortográfica automática não se mostrou particularmente eficiente neste caso, resultando em algumas palavras

corretas sendo erroneamente modificadas, como no caso de "creanças", que não constava do léxico do dicionário de 1913, usado como base para desenvolver o dicionário utilizado pelo PyEnchant.

Além disso, em outros casos, palavras reconhecidas como incorretas foram corrigidas para outras que não correspondem ao texto original, como a palavra "mito", que foi corrigida para “meto” ao invés de “mimo”. Isto ocorreu porque, dentre as duas palavras sugeridas pelo corretor, “meto” obteve menor valor de distância de edição em relação a “mito” com base na medida de Levenshtein.

Nos casos citados acima, o uso do corretor ortográfico semiautomático desenvolvido neste trabalho solucionaria os problemas detectados a partir da fácil seleção das melhores alternativas para as palavras identificadas com erro ortográfico, conforme apresentado na Imagem 26. Também é possível na própria interface adicionar palavras ao dicionário construído apenas digitando o termo correto para palavra errada e clicando no botão adicionar.

Imagem 26 - Uso da interface para correção semiautomática.



Fonte: A autora (2023).

5.3 RESULTADOS ALCANÇADOS

A metodologia desenvolvida para a extração do texto dos jornais históricos foi comprovadamente benéfica, conforme evidenciado nos exemplos apresentados nas seções 5.1 e 5.2. Foi constatado que tanto as etapas de pré-processamento quanto a correção pós-OCR desempenham papéis importantes na melhoria da qualidade final do texto extraído.

Esta seção apresenta algumas considerações sobre os resultados alcançados com a execução das etapas do processo geral proposto no capítulo 4. Inicialmente vamos comentar sobre os resultados da etapa de pré-processamento das imagens, seguindo com a correção do texto pós-OCR, destacando onde acertamos e onde erramos. A seguir, faremos uma breve discussão sobre o uso do engenho de busca via PySolr já apresentado no capítulo anterior.

Vale esclarecer que utilizamos uma pequena quantidade de imagens para realizar os testes com o processo implementado – sete páginas de Diário da Manhã e três páginas do Diário de Pernambuco. Isso se deveu ao limite de tempo para concluir esta dissertação, e à dificuldade em analisar visualmente os resultados obtidos, que é uma tarefa que demanda muito tempo e atenção. Testes mais robustos estão propostos como trabalhos futuros, no Capítulo 6, de conclusão.

5.3.1 Resultados do Pré-processamento das imagens

As seções anteriores mostraram a influência benéfica da melhoria da imagem na extração de texto. Assim, não vamos ressaltar novamente essa contribuição. Vamos apenas comentar sobre o que pode ser aprimorado.

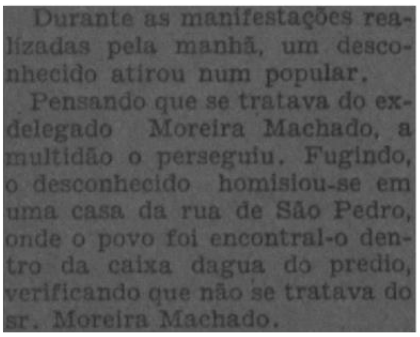
Em alguns casos, as funções de pré-processamento aplicadas à imagem resultaram em um texto extraído via OCR de qualidade inferior ao texto extraído a partir da imagem original. As Imagem 27 lustram um desses casos, que ocorreu com um trecho do jornal Diário da Manhã.

Ao comparamos os textos nas imagens (c) e (d), observamos que o OCR errou na extração das palavras “realizadas” e “homisiou-se”, além de ter inserido caracteres especiais como ruído (e.g., “-“, “!” e “|”). Contudo, nos testes realizados, foi possível observar que apenas em casos isolados o pré-processamento não resultou em melhoria do texto extraído das imagens utilizadas. No geral, essa etapa de fato ajudou na correta extração dos textos.

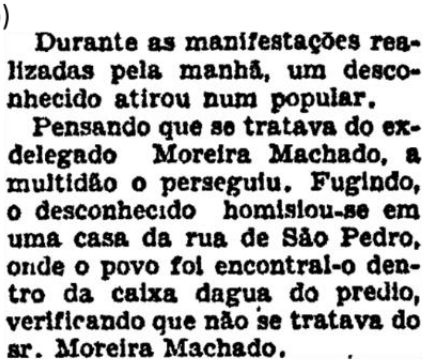
Como trabalho futuro, recomendamos aprimorar essa etapa com o uso de algoritmos mais robustos, como os baseados em redes neurais. Além disso, é de grande importância realizar a segmentação prévia da imagem em notícias individuais. Dessa forma, será possível realizar pré-processamentos individualizados em cada trecho da página (segmento que corresponde a uma notícia), buscando aperfeiçoar ainda mais a qualidade da extração do texto.

Imagem 27 - Exemplo de um trecho onde o pré-processamento implementado não melhorou o texto extraído. (a) imagem original; (b) imagem pré-processada; (c) texto extraído da imagem original; (d) texto extraído da imagem pré-processada.

a)


Durante as manifestações realizadas pela manhã, um desconhecido atirou num popular.
Pensando que se tratava do exdelegado Moreira Machado, a multidão o perseguiu. Fugindo, o desconhecido homisiou-se em uma casa da rua de São Pedro, onde o povo foi encontral-o dentro da caixa dagua do predio, verificando que não se tratava do sr. Moreira Machado.

b)


Durante as manifestações realizadas pela manhã, um desconhecido atirou num popular.
Pensando que se tratava do exdelegado Moreira Machado, a multidão o perseguiu. Fugindo, o desconhecido homisiou-se em uma casa da rua de São Pedro, onde o povo foi encontral-o dentro da caixa dagua do predio, verificando que não se tratava do sr. Moreira Machado.

c)

Durante as manifestações realizadas pela manhã, um. desconhecido atirou num popular, "Pensando que se tratava do exdelegado — Moreira Machado, a multidão o perseguiu, Fugindo, o desconhecido homisiou-se em uma casa da rua de São Pedro, onde o povo foi encontral-o dentro da caixa dagua do predio, | verificando que não se tratava do | sr. Moreira Machado,

d)

Durante as manifestações reglizadas pela manhã, um desconhecido atirou num popular, Pensando que se tratava do exdelegado — Moreira Machado, a multidão o perseguiu, Fugindo, ! o desconhecido homislou-se em uma casa da rua de São Pedro, onde o povo foi encontral-o dentro da caixa dagua do predio, | verificando que não se tratava do | sr. Moreira Machado,

Fonte: A autora (2023).

5.3.2 Resultados da Correção pós-OCR

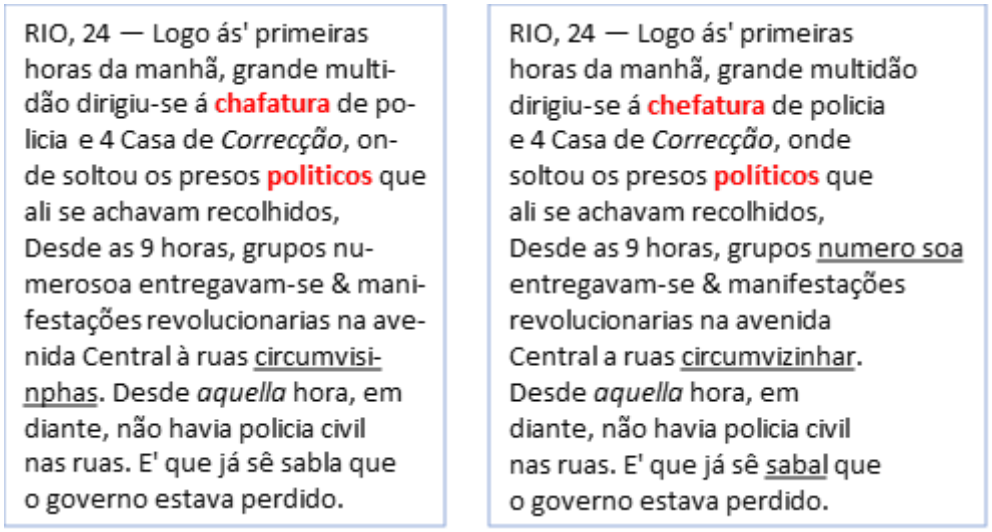
Vamos aqui apresentar a análise da correção pós-OCR de um trecho de texto extraído de uma imagem do jornal Diário da Manhã. Na Imagem 28, os textos (a) e (b) reproduzidos abaixo foram copiados da seção 5.1.3. O texto (a) é resultado da extração do texto via OCR, e o texto (b) é resultado da correção pós-OCR.

Todas as correções de translineação foram bem-sucedidas. As palavras que estavam separadas entre linhas foram todas restauradas corretamente, assim não vamos comentar sobre essa tarefa. Porém, ressaltamos que esse passo é essencial para o bom funcionamento do corretor ortográfico.

A seguir, vamos ressaltar aqui três situações distintas:

- Negrito vermelho – destaca as correções ortográficas bem-sucedidas;
- Sublinhado – destaca modificações errôneas feitas pelo corretor ortográfico;
- Itálico – indica palavras que necessitam de atualização vocabular.

Imagem 28 - Análise da correção pós-OCR (a) resultado da extração do texto via OCR, e (b) resultado da correção pós-OCR.



Fonte: A autora (2023).

Contabilizamos como modificações positivas as sete correções de translineação e a correção das palavras “chefatura” e “políticos”.

Como saldo negativo, destacamos três palavras que foram substituídas erroneamente. A palavra "numerosoa" (adjetivo) foi trocada por “numero soa” (substantivo e verbo). A palavra “circumvisinphas” foi trocada por “circunvizinhar” (verbo), cuja ortografia está correta, porém não corresponde à palavra que seria correta no texto, i.e., “circunvizinhas” (adjetivo). Por fim, a palavra “sabla” foi trocada por “sabal” (planta), em vez de “sabia” (verbo). Já a palavra “policia” permaneceu errada, sem acento agudo no primeiro “i”. Note que aqui o correto seria “polícia” (substantivo), e não “policia” (verbo).

Nos quatro casos destacamos acima, a solução vislumbrada seria a realização de uma análise mais aprofundada do texto, com uso de um *POS-tagger* aliado a regras gramaticais da língua portuguesa. Essas regras seriam capazes de indicar a classe gramatical da palavra sendo corrigida, evitando assim algumas substituições errôneas. Essa tarefa também será indicada como trabalho futuro.

Por fim, vale mencionar as palavras que necessitam de atualização vocabular, porém apenas para fins de facilitar a busca das imagens a partir de palavras-chave. Essa tarefa também será indicada como trabalho futuro.

5.3.3 Resultados da Indexação via PySolr

O engenho de busca implementado via PySolr foi utilizado para indexar as 7 imagens de páginas de jornais já mencionadas na seção anterior. Essa baixa quantidade de documentos inviabilizou a realização de testes baseados nas medidas tradicionais de sistemas de RI, que são Precisão, Cobertura e *F-measure* (Baeza-Yates e Ribeiro-Neto, 2011). Esses testes ficam indicados como trabalho futuro, com o aumento do tamanho do *corpus* de imagens.

Contudo, foi possível realizar testes de usabilidade da interface desenvolvida para realização de consultas. Observando o funcionamento do sistema, podemos afirmar que os resultados foram bastante satisfatórios.

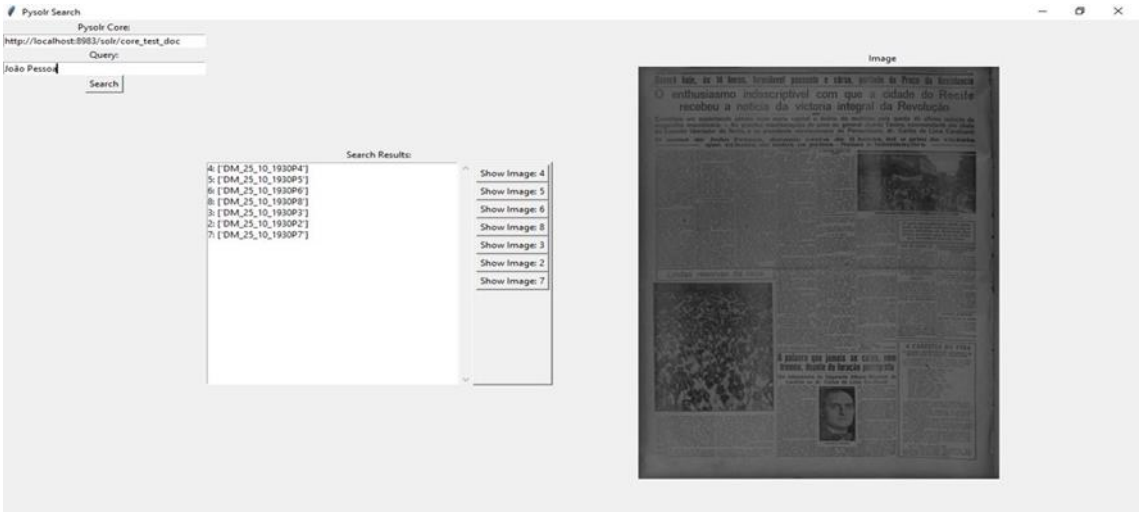
A seguir, vamos descrever os passos para realizar a indexação das imagens e as consultas posteriores através do engenho de busca implementado. Relembramos aqui que utilizamos o formato JSON *(JavaScript Object Notation*) para representar os dados a serem indexados (texto extraído e endereço da imagem correspondente).

Inicialmente, as imagens a serem indexadas foram armazenadas em um diretório local na máquina utilizada para desenvolver e executar o engenho de busca. A seguir, os arquivos JSON foram criados, utilizando o formato apresentado na seção 4.5.2. O caminho do diretório das imagens e os textos extraídos das respectivas imagens foram organizados em um *array* de dicionários em Python, conforme definido na seção 4.5.2. Em seguida foi realizada a indexação desses documentos no Solr por meio do PySolr.

A Imagem 29 apresenta um exemplo de utilização da interface desenvolvida para etapa de busca via PySolr a partir dos dados indexados. A interface disponibiliza o campo ***Pysolar Core*** para o usuário especificar a URL definida na criação da conexão com o servidor Solr, seguido do nome atribuído ao CORE (arquivo de configurações).

O campo ***Query*** é uma janela para o usuário digitar as palavras-chave para realização das consultas de interesse. Ao clicar no botão ***Search***, é realizada a busca e recuperação dos documentos relacionados às palavras da consulta. A apresentação dos resultados é feita no campo ***Search results*** e seus respectivos botões de visualização. Por fim, a imagem de cada página recuperada contendo o termo buscado pode ser visualizada clicando no botão ***Show Image***.

Imagem 29 - Interface desenvolvida para etapa de busca via PySolr.



Fonte: A autora (2023).

5.4 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo apresentar mais detalhes do processo implementado neste trabalho, iniciando com uma breve análise dos resultados obtidos com o pré-processamento das imagens de jornais. Essa análise, ainda que superficial, mostrou os ganhos obtidos com essa etapa de melhoria das imagens de entrada, confirmando sua importância no contexto do nosso trabalho. Em seguida, discutimos brevemente os resultados da etapa de OCR evidenciando a importância que esta etapa também possui na melhoria da qualidade do texto final, e por fim falamos sobre o engenho de busca e sua utilização.

O Capítulo 6, a seguir, finaliza este documento apresentando nossas principais contribuições e as várias indicações de trabalhos futuros.

6 CONCLUSÃO

Este trabalho de mestrado teve como objetivo ampliado contribuir com um projeto de interesse da sociedade, o “Programa de Curadoria do patrimônio memorial de Jornais centenários de Pernambuco”, cujo objetivo é resgatar, preservar e prover livre acesso a jornais periódicos centenários em língua portuguesa. Nosso objetivo específico foi viabilizar o acesso a esse acervo tão relevante, já em formato digital, através de consultas simples baseadas em palavras-chave.

Com base nos resultados obtidos, consideramos que os objetivos iniciais deste trabalho foram alcançados, tendo sido materializados no dicionário de 1913 adaptado ao *PyEnchant*, e na base de imagens indexadas automaticamente através de termos que ocorrem em cada imagem. Devido a restrições de tempo, não foi possível implementar todas as melhorias inicialmente planejadas, estando indicadas como trabalhos futuros.

Este capítulo finaliza o documento de dissertação destacando algumas das dificuldades encontradas na construção da solução oferecida, as principais contribuições e, por fim, algumas sugestões de trabalhos futuros. Diferentemente da organização tradicional deste capítulo em dissertações de mestrado, vamos seguir aqui o fluxo de execução do processo apresentado na seção 4. 1 (Visão Geral da solução proposta), uma vez que cada etapa de processamento envolve uma área de pesquisa distinta, com problemas e soluções próprias.

Iniciamos com o pré-processamento da imagem, seguido da extração do texto via OCR. A próxima seção trata da correção do texto pós-OCR (nossa principal contribuição), finalizando com a criação e disponibilização da base de imagens indexadas. Com isso pretendemos dar uma visão de cada processo implementado, com suas dificuldades individuais, soluções adotadas e sugestões de melhorias futuras.

6.1 PRÉ-PROCESSAMENTO DAS IMAGENS

6.1.1 Dificuldades encontradas

Como vimos nos exemplos mostrados neste texto, as imagens digitalizadas a partir dos jornais centenários em papel apresentam problemas diversos. Porém, independentemente do problema identificado, todos afetam a precisão da extração do texto via OCR (próxima seção).

Problemas como dobras, rasgos e buracos causados por traças e fungos acarretam na extração incompleta das palavras. Já os hifens que separam sílabas, bem como os traços verticais e horizontais que separam notícias podem ser confundidos com caracteres especiais. Além disso, sem uma segmentação prévia da imagem, o OCR poderá apenas unir o texto de cada linha da página, sem respeitar as bordas verticais que separam notícias. Desenhos com textos internos também podem causar dificuldades, pois o algoritmo de OCR poderá misturar o texto da notícia ao texto da figura.

Inicialmente, nosso projeto não considerou a ideia de realizar pré-processamento externo de imagens, uma vez que a ferramenta Tesseract, adotada para realizar o OCR, já conta com filtros internos de imagens. Contudo, testes com a ferramenta mostraram que esses filtros internos não eram suficientes para melhorar satisfatoriamente as nossas imagens – os erros do OCR ainda eram muitos. Assim surgiu a necessidade de realizarmos um pré-processamento externo das imagens. Nesse contexto, nossa *dificuldade* foi de encontrar uma solução adequada para melhorar a imagem antes de enviá-la para a etapa de extração do texto via OCR. Porém, como os problemas são variados, um tratamento único de imagem dificilmente conseguiria minorar ou eliminar todos esses problemas.

6.1.2 Contribuição da solução implementada

Buscando identificar a melhor opção disponível para limpeza/restauro das imagens, realizamos inúmeros testes com ferramentas e bibliotecas variadas. Esses testes consumiram muito tempo de trabalho, trazendo descobertas interessantes. Foi possível deduzir, por exemplo, que não é possível obter o mesmo resultado em todas as imagens aplicando as mesmas operações de tratamento de imagens, pois

o número extensivo de operações torna inviável o ajuste manual de todos os seus parâmetros dado a alta variabilidade no estado geral das imagens e do grande volume do acervo.

Ao final dos testes, optou-se por utilizar a biblioteca Python OpenCV na implementação dessa etapa. Nosso objetivo foi de melhorar qualidade geral da imagem, a fim de auxiliar o OCR a alcançar maior precisão e legibilidade do texto extraído.

Nossa contribuição com relação ao processamento da imagem está relatada nos capítulos 4 e 5. A seção 4.2 descreve as operações realizadas e exemplifica os resultados alcançados, destacando a visível melhora na extração correta dos textos após essa etapa. Por fim, o Capítulo 5 também apresenta exemplos de melhorias das imagens tratadas. Porém, nosso trabalho não se dedicou a resolver todos os problemas apontados na seção 6.1.1 acima, por serem muito complexos e por não ser este o objetivo principal da nossa pesquisa de mestrado.

6.1.3 Sugestões de trabalhos futuros

Uma sugestão de trabalho futuro aqui é a *segmentação prévia das imagens* em notícias individuas, bem como a separação de figuras, fotografias e outras imagens contidas nas páginas dos jornais. O objetivo é dividir a página do jornal em segmentos menores onde, consequentemente, haverá menor variação no estado da imagem. Desta forma, se tornará mais simples a otimização da qualidade da imagem via etapa de pré-processamento em cada segmento de notícia. Acreditamos que essa operação irá oferecer um grande salto de qualidade na extração dos textos, bem como na correção pós-OCR.

Como verificamos um número considerável de casos em que os textos das notícias são extraídos misturados (por linha), nem sempre é possível, por exemplo, *restaurar o texto original* através do OCR. Esse era mais um objetivo do projeto geral de preservação desse acervo, porém ainda não conseguimos uma solução viável para alcançá-lo. Também não seria muito útil realizar uma análise morfossintática dos textos, a fim de melhorar a correção ortográfica, uma vez que as frases não estrariam completas.

Uma outra abordagem que acreditamos ter um potencial de melhoria ainda mais significativo para os problemas enfrentados seria desenvolver uma metodologia

de pré-processamento de imagem adaptativa e robusta, possivelmente com uso de técnicas de aprendizagem de máquina, de forma a lidar de forma mais eficiente com a alta variabilidade no estado da imagem nas páginas do jornal.

6.2 EXTRAÇÃO DO TEXTO VIA OCR

6.2.1 Dificuldades encontradas

Como sabemos, existem diversas ferramentas de OCR disponíveis, muitas delas com excelente desempenho quando tratam de imagens com boa resolução e com textos atuais. Contudo, nosso contexto é um pouco diferente. Todos os problemas citados na seção 6.1.1 impactaram a extração do texto de forma negativa, como já foi discutido.

Além desses problemas relacionados diretamente às imagens, tivemos que lidar também com o vocabulário utilizado nos jornais centenários. Lembramos aqui que algumas palavras tinham grafia diferente da atual (por exemplo, *pharmácia* - *farmácia*), dificultando a correção automática do texto com base nos dicionários modernos contidos nas ferramentas de OCR e de PLN.

6.2.2 Contribuição da solução implementada

Essa etapa de extração do texto também envolveu demorados testes com variadas ferramentas e bibliotecas. Ao final, adotamos a biblioteca em Python *PyTesseract*, por ter apresentado o melhor resultado para nossas imagens tão antigas e com falhas devidas às avarias no papel original dos jornais.

Contudo, os problemas relacionados às diferenças no vocabulário persistiram. Logo ficou claro que melhorias gerais e segmentação adequada da imagem não resolveriam esse problema. Realizamos então um estudo bibliográficos com foco em corretores ortográficos, bem como sobre trabalhos relacionados, que tratam de textos centenários (Capítulo 3). Porém, não conseguimos encontrar uma solução trivial, pronta para ser aplicada no nosso contexto. A partir desses estudos, decidimos implementar outra etapa de processamento de texto, a fim de dirimir os erros relacionados às variações vocabulares.

Assim, nossa contribuição com relação à etapa de extração de textos centenários via OCR consiste nos estudos da bibliografia relacionada ao nosso problema, bem como nos vários testes realizados para selecionar a ferramenta que foi utilizada aqui. Esses estudos tiveram importância central na condução dos passos posteriores do nosso trabalho, materializados na etapa de correção pós-OCR.

6.2.3 Sugestões de trabalhos futuros

As sugestões de trabalhos nesse tema envolvem testar novas configurações do *PyTesseract*, com variações dos parâmetros existentes. Quando for possível avançar na etapa de pré-processamento das imagens, com segmentação das notícias, será necessário verificar novamente quais são as melhores configurações do *PyTesseract* nesse novo contexto. Podemos ainda testar outras ferramentas de OCR com as imagens segmentadas.

6.3 PROCESSAMENTO DE TEXTO PÓS-OCR

6.3.1 Dificuldades encontradas

Como sabemos, esta etapa se dedicou a melhorar o texto extraído via OCR, executando processos diversos, como: eliminação de caracteres ruidosos, recomposição das palavras em casos de translineação (etapa necessária para a aplicação do corretor ortográfico), e finalmente, aplicação do corretor ortográfico *PyEnchant* utilizando um dicionário personalizado construído neste trabalho.

Cada um desses processos foi implementado individualmente, apresentando suas dificuldades particulares. Por exemplo, a eliminação de caracteres ruidosos poderia apagar um hífen usado na separação de sílabas da translineação. Assim, esses dois processos foram cuidadosamente implementados, para não causarem efeitos negativos. Além disso, como o OCR apresentou erros de extração variados, alguns caracteres que faziam parte de palavras foram eliminados por engano, prejudicando assim a recomposição da palavra.

Claramente, a maior dificuldade enfrentada aqui está relacionada ao uso do corretor ortográfico *PyEnchant*, que só dispõe de um dicionário atual da língua

portuguesa. Isso nos levou a buscar um dicionário adequado para tratar textos centenários. Iniciamos a busca por um dicionário datado de 1832, que é contemporâneo às edições iniciais do Diário de Pernambuco. Contudo, esse dicionário só foi encontrado em formato de imagem, e uma tentativa de extração dos seus termos via OCR não foi bem-sucedida, pois caímos em um problema semelhante ao que queríamos resolver (extração de texto a partir de documentos centenários).

Depois de muitas buscas por uma solução factível, decidimos utilizar o dicionário de 1913, que dispõe de uma versão digital em formato de texto. As dificuldades enfrentadas aqui foram duas: recuperar as entradas do dicionário e armazená-las em uma lista; e adequar as entradas do dicionário ao padrão do PyEnchant.

6.3.2 Contribuição da solução implementada

A solução implementada nesta etapa é nossa maior contribuição. O dicionário de 1913 foi recuperado e adaptado para o padrão PyEnchant Hunspell. Essa adaptação automática contou com vários passos, que estão detalhadamente descritos na seção 4.4. O dicionário adaptado conta agora com regras de afixação para flexionar a forma base (lema) das palavras, ampliando assim a cobertura do corretor ortográfico. Essa funcionalidade melhora a precisão e a qualidade dos textos históricos corrigidos.

Como resultado, conseguimos um expressivo incremento no dicionário original de 1913. A partir das 125.244 palavras iniciais do dicionário sem afixos, foram obtidas 81.900 palavras com regras de afixo, mais as 18.317 palavras sem afixos restantes, totalizando 100.217 palavras. Ainda, ao contabilizar todas as possíveis flexões das palavras com tags de afixação, a versão final do dicionário construído abrange um total de 210.793 palavras. Assim, o total de palavras do dicionário final é aproximadamente 69% maior do que inicialmente, ampliando as possibilidades de correção dos textos históricos obtidos do OCR.

Outra contribuição dessa etapa se refere à possibilidade de reuso do código implementado. Como utilizamos uma abordagem modular de desenvolvimento, o código referente a cada passo implementado pode ser reusado na adaptação para o PyEnchant de outros dicionários em formato de lista de palavras. Aqui vale ressaltar

que a equipe do LIBER está desenvolvendo uma versão digital do dicionário de 1832, que futuramente poderá ser submetida aos passos de adaptação desenvolvidos neste trabalho.

Por fim, destacamos que testes preliminares mostraram que a correção dos textos extraídos com o uso do dicionário desenvolvido foi bem-sucedida. Porém, ainda restam situações de erros e lacunas neste trabalho.

6.3.3 Sugestões de trabalhos futuros

Como indicado na seção 5.1.3, a solução implementada ainda precisa de melhorias. Por total falta de tempo, não foi possível tratar todos os termos compostos automaticamente, principalmente aqueles que tiveram grande modificação na ortografia ou os que já não são mais usados como termos compostos (e.g., pedra-íman -> pedra-ímã, há-de-haver). Esses passos de adaptação do dicionário serão tratados em trabalhos futuros.

Outras lacunas a serem resolvidas se referem a substituições errôneas de palavras e a casos de ausência de correção de acentuação (e.g., “polícia” (substantivo) em lugar de “policia” (verbo)). Esses casos poderiam ser tratados através de uma análise mais aprofundada do texto, com uso de um POS-tagger aliado a regras gramaticais da língua portuguesa. Essas regras seriam capazes de indicar a classe gramatical da palavra sendo corrigida, evitando assim algumas substituições errôneas. Essa tarefa está aqui indicada como trabalho futuro.

Por fim, ressaltamos a necessidade de se criar um mecanismo para atualização vocabular. Algumas vezes, os usuários não conhecem a grafia original da palavra que desejam buscar na base de imagens indexadas. Assim, seria importante associar as palavras do dicionário de 1913 com grafia antiga à sua correspondente na grafia atual. O objetivo dessa atualização é facilitar a busca das imagens a partir de palavras-chave.

6.4 INDEXAÇÃO

6.4.1 Dificuldades encontradas

Por fim, os textos corrigidos foram utilizados para indexar as imagens correspondentes, criando assim um repositório para livre acesso através de consultas via palavras-chave (seção 4.5). Não enfrentamos grande dificuldades com esta etapa, uma vez que a ferramenta *PySolr* já era do nosso conhecimento, sendo bastante flexível e fácil de configurar. Além disso, nosso trabalho se resumiu a indexar algumas poucas páginas de jornal, apenas como “prova de conceito”. Queríamos mostrar que existe uma solução viável para o problema “reduzido”.

Contudo, entendemos que a estrutura de indexação e busca desenvolvida no nosso trabalho não seria capaz de comportar o número tão elevado de documentos já digitalizados pela equipe do LIBER. A criação de uma nova estrutura de indexação e busca fica indicada como trabalho futuro. Ainda por falta de tempo, não foi possível implementar todas as ideias inicialmente consideradas para indexação. Vamos comentar sobre isso na seção 6.4.3.

6.4.2 Contribuição da solução implementada

A grande contribuição desta etapa foi de mostrar a viabilidade de prover acesso simplificado às imagens indexadas a partir de palavras-chave, sendo esse um dos objetivos principais deste trabalho. A interface desenvolvida é muito simples de ser usada, não oferecendo dificuldades aos usuários menos acostumados a lidar com sistemas informatizados. Trata-se de uma prova de conceito, como já mencionado.

6.4.3 Sugestões de trabalhos futuros

Como trabalhos futuros, relembramos inicialmente a importância da criação de um mecanismo para modernização vocabular, para facilitar as buscas via palavras-chave. Uma possibilidade simples seria o uso de um tesauro associado ao *PySolr*. Porém, não sabemos se a ferramenta comportaria um tesauro com tantas

entradas. Outra opção seria buscar uma solução via *PyEnchant*. Porém essas ideias ainda não foram maturadas, estando assim abertas a sugestões.

Outra sugestão, já mencionada acima, é a criação de uma estrutura de indexação e busca de imagens mais robusta, como um banco de dados. Isso iria viabilizar o acesso e a disponibilização ao público em geral do grande volume de documentos já digitalizados, por meio da criação de uma página web com uma interface de consulta abrangendo todo acervo dos jornais digitalizados.

REFERÊNCIAS

ALLEN, James. **Natural Language Understanding**. 2. ed. Pearson, 1994.

ARAUJO, L. C. de; BENEVIDES, A. de L.; SANSÃO, J. P. H. **Desenvolvimento de um corretor ortográfico**. Texto Livre, Belo Horizonte-MG, v. 14, n. 1, p. e26469, 2021. DOI: 10.35699/1983-3652.2021.26469.

ARAÚJO, Luciana. **Morfemas**. Mundo educação. Disponível em: < https://mundoeducacao.uol.com.br/gramatica/morfemas.htm >. Acesso em: 05 jul. 2023.

ATKINSON, Kevin. **Aspell**. [S.l.: s.n.], 1998. Disponível em: <http://aspell.net/>. Acesso em: 8 jul. 2020.

BAEZA Y. R.; RIBEIRO-NETO, B. ***Modern Information Retrieval*** **- The Concepts and Technology behind Search**. 2nd Edition, Pearson, 2011.

BARROS, Flávia A.; ROBIN, Jacques. **Processamento de Linguagem Natural**. Revista Eletrônica de Iniciação Científica, v. 1, p. 1-61, 2001.

BLEI, D., NG, A. e JORDAN, M. **Latent Dirichlet Allocation**. The Journal of Machine Learning Research, 3, 601-608, 2001.

BRADSKI, G., **The OpenCV Library**, Dr. Dobb's Journal of Software Tools, 2000.

CHURCH, Kenneth W.; GALE, William A. **Probability scoring for spelling correction**. Statistics and Computing, Springer Science e Business Media LLC, v. 1, n. 2, p. 93–103, dez. 1991. DOI: 10.1007/bf01889984.

DAMERAU, Fred J. **A technique for computer detection and correction of spelling errors. Communications of the ACM**, Association for Computing Machinery (ACM), v. 7, n. 3, p. 171–176, mar. 1964.

DE ALMEIDA, José J. D. **Dicionários dinâmicos multi-fonte**. Tese de doutoramento, Universidade do Minho, 2003.

DROBAC, SENKA, **OCR and post-correction of historical newspapers and journals**. Tese (Doutorado em Artes) - Universidade de Helsinki, Finlândia, 2020.

ENGLMEIER, T., FINK, F., & SCHULZ, K. U. **AI-PoCoTo: Combining automated and interactive OCR postcorrection.** In Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage, 2019.

EVERSHED, J., & FITCH, K. **Correcting noisy OCR: Context beats confusion**. In Proceedings of the 1st International Conference on Digital Access to Textual Cultural Heritage. ACM. 2014.

FIGUEIREDO, C. **Novo diccionário da língua portuguesa**, Vol. 2, AM Teixeira, 1913.

FUGGETTA, Alfonso. **Open source software—an evaluation**. Journal of Systems and software, v. 66, n. 1, p. 77-90, 2003.

GARRIDO-ALENDA, A. and M. L. FORCADA, **Comparing nondeterministic and quasideterministic finite-state transducers built from morphological dictionaries. In Procesamiento del Lenguaje Natural**, (XVIII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, pages 291-292, 2007.

GASPAR, Lúcia. **Diario de Pernambuco**. Pesquisa Escolar Online, Fundação Joaquim Nabuco, Recife. Disponível em: <http://basilio.fundaj.gov.br/pesquisaescolar/>. Acesso em: 07/07/2023.

GONZALEZ, Rafael, **Digital image processing**, 4. ed., New York: Pearson, 2018.

GORIN, Ralph E.; WILLISSON, Pace; KUENNING, Geoff. **Ispell**. [S.l.: s.n.], 1971. Disponível em: <https://www.cs.hmc.edu/%7B%5C~%7Dgeoff/ispell.html>. Acesso em: 8 jul. 2020.

HAUSER, A. W. **OCR Postcorrection of Historical Texts**, Dissertação (Mestrado) - Faculdade de Linguística Computacional - Universidade de Munique, Munique, 2007.

HIRSCHBERG J., MANNING C. D. **Advances in natural language processing**. Science. 349. 2015.

JURAFSKY, D. e MARTIN, J. H. **Speech and Language Processing**, 2. ed., Pearson, Prentice Hall, 2008.

KARTHIKEYAN, Srinidhi et al. **An OCR Post-Correction Approach Using Deep Learning for Processing Medical Reports**. IEEE Transactions on Circuits and Systems for Video Technology, v. PP, p. 1-1, 2021.

KELLY, R. **Pyenchant - Python bindings for the Enchant spellchecking system**, 2021. Disponível em: <https://pypi.org/project/pyenchant/>.

KERNIGHAN, Mark D.; CHURCH, Kenneth W.; GALE, William A. **A spelling correction program based on a noisy channel model**. In: PROCEEDINGS of the 13th conference on Computational linguistics. [S.l.]: Association for Computational Linguistics, 1990. DOI: 10.3115/997939.997975. Disponível em: <https://doi.org/10.3115/997939.997975>.

KNUTH, Donald Ervin. **The Art of Computer Programming.** [S.l.]: Addison-Wesley, 1973. v. 3.

MANNING C. D., RAGHAVAN P. E SCHÜTZE H. **Introduction to Information Retrieval,** Cambridge University Press. 2008.

MEDEIROS, José Carlos. **Análise morfológica e correção ortográfica do Português**. Tese de mestrado, Instituto Superior Técnico, Universidade Técnica de Lisboa, 1995.

MITTAL, R., GARG, A. **Text extraction using OCR: A systematic review,** In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 357-362), 2020.

MITTON, Roger. **Spelling checkers, spelling correctors and the misspellings of poor spellers. Information Processing & Management**, Elsevier BV, v. 23, n. 5, p. 495–505, jan. 1987.

MOURA R. **Vero, the brazilian portuguese spell checker**. Disponível em: <https://pt-br.libreoffice.org/projetos/vero> Acessado em: 01/08/2021.
MURSARI, L.R., & WIBOWO, A. **The Effectiveness of Image Preprocessing on Digital Handwritten Scripts Recognition with The Implementation of OCR Tesseract,** Computer Engineering and Applications Journal, 2021

NÉMETH, László, **Hunspell** 1.7.2, 2022. Disponível em: <http://hunspell.github.io/)>

NGUYEN, Thi. T. H., et. al. 2021. **Survey of Post-OCR Processing Approaches**, ACM Comput. Surv. 54, 6, Article 124, 2022.

ODELL, Margaret King. **The profit in records management Systems**, New York, v. 20, 1956.
KUČERA, Henry; FRANCIS, Winthrop Nelson. **Computational analysis of present-day American English**. [S.l.]: Brown University Press, 1967.

PHILIPS, Lawrence. **Hanging on the metaphone**. Computer Language, v. 7, n. 12, p. 39–43, 1990.

PHILIPS, Lawrence. **The Double Metaphone Search Algorithm**. C/C++ Users J., CMP Media, Inc., USA, v. 18, n. 6, p. 38–43, jun. 2000. ISSN 1075-2838.

POLLOCK, Joseph J.; ZAMORA, Antonio. **Automatic spelling correction in scientific and scholarly text**. Communications of the ACM, Association for Computing Machinery (ACM), v. 27, n. 4, p. 358–368, abr. 1984. DOI: 10.1145/358027.358048

RICHTER, C. et al. **Low-resource post processing of noisy OCR output for historical corpus digitisation**. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation. 2018.*

RUSSELL, Robert C. Robert C. Russell. **Index**. 1918. US 1261167A, Depósito: 25 out. 1917. Concessão: 02 abr. 1918. Disponível em: <https://patents.google.com/patent/US1261167A/en>.

RUSSELL, Robert C. Robert C. Russell. **Index**. 1922. US 1435663A, Depósito: 28 nov. 1921. Concessão: 14 nov. 1922. Disponível em: <https://patents.google.com/patent/US1435663A/en>

SÁ, Gildácio; MAIA, José, **Processamento e Navegação por Tópicos em Imagens de Páginas de Jornais Históricos**, Computer on The Beach 2020, v. 11 n. 1, p 432-439, 2020.

SAUVOLA, J; PIETIKAINEN, M., **Adaptive document image binarization,** Pattern Recognition 33(2), pp. 225-236, 2000.

SHANNON, Claude E. **A mathematical theory of communication**. Bell Syst. Tech. J., v. 27, n. 3, p. 379–423, 1948.

SMITH, R. **An Overview of the Tesseract** OCR Engine, Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society, 2007.

VILELA, M. R. S., **Geração de dicionários para correcção ortográfica do português**, Dissertação (Mestrado em Informática) - Universidade do Minho, Braga, Portugal, 2009.

WIKIPEDIA, **NOVO DICIONÁRIO DA LÍNGUA PORTUGUESA**. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Novo_Dicion%C3%A1rio_da_L%C3%ADngua_Portuguesa&oldid=51736863>. Acesso em: 02/07/2023.