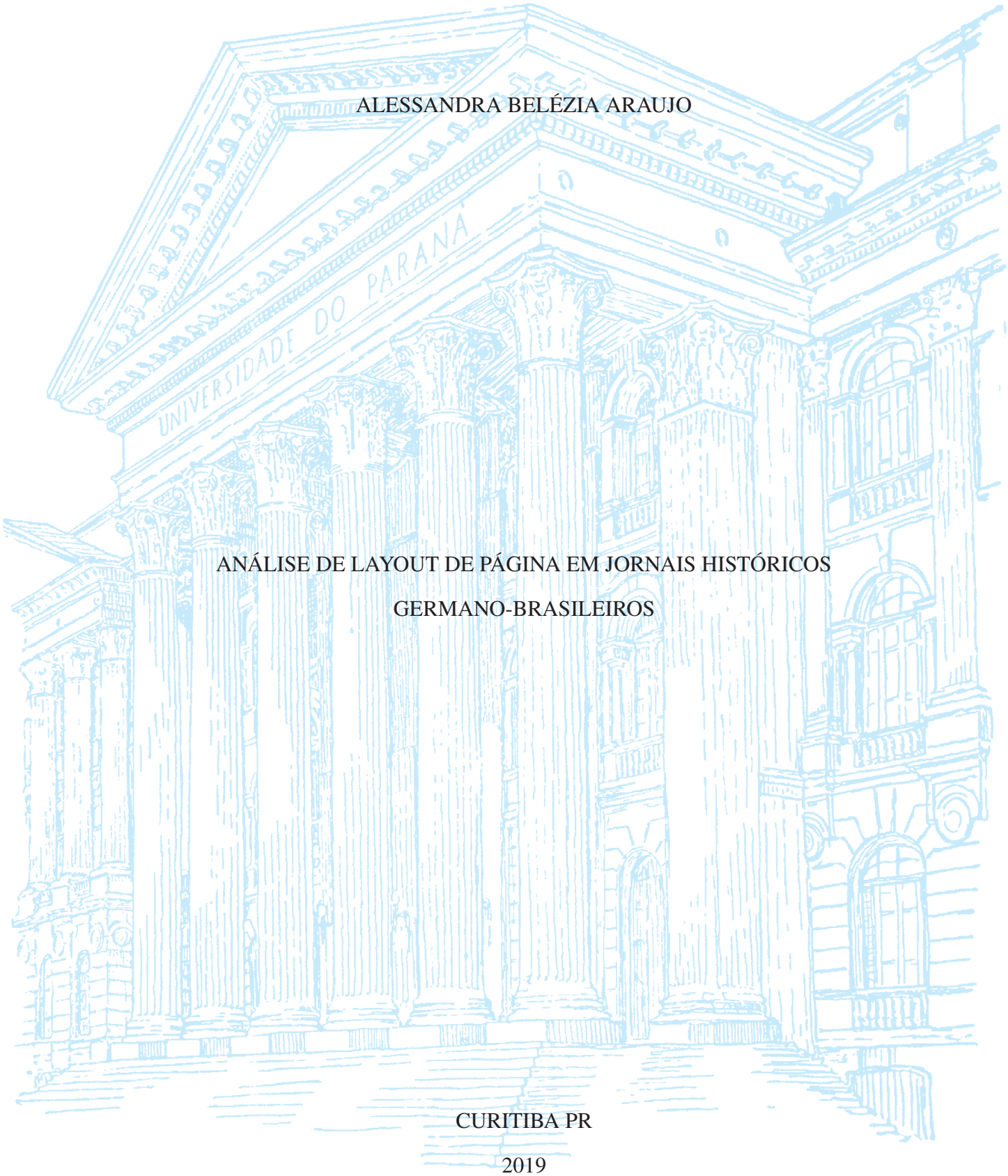
UNIVERSIDADE FEDERAL DO PARANÁ

ALESSANDRA BELÉZIA ARAUJO



ANÁLISE DE LAYOUT DE PÁGINA EM JORNAIS HISTÓRICOS GERMANO-BRASILEIROS

CURITIBA PR

2019

ALESSANDRA BELÉZIA ARAUJO

ANÁLISE DE LAYOUT DE PÁGINA EM JORNAIS HISTÓRICOS GERMANO-BRASILEIROS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Daniel Weingaertner.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

A663 Araújo, Alessandra Belézia

Análise de layout de página em jornais históricos germano-brasileiros [recurso eletrônico] / Alessandra Belézia Araújo, 2019.

Dissertação (mestrado) - Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Daniel Weingaertner.

1. Redes neurais (Computação). 2. Algoritmos de computador. 3. Inteligência artificial. I. Universidade Federal do Paraná. II. Weingaertner, Daniel. III. Título.

CDD 006.3

Bibliotecária: Vilma Machado CRB9/1563

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **ALESSANDRA BELÉZIA ARAUJO** intitulada: **ANÁLISE DE LAYOUT DE PÁGINA EM JORNAIS HISTÓRICOS GERMANO-BRASILEIROS**, sob orientação do Prof. Dr. DANIEL WEINGAERTNER, que após após terem inquirido a aluna e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 30 de Agosto de 2019.

DANIEL WEINGAERTNER
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

BOGDAN TOMOYUKI NASSU
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

LUIZ EDUARDO SOARES DE OLIVEIRA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

A todos que me ajudaram a tornar este trabalho possível.

AGRADECIMENTOS

Agradeço primeiramente à Deus, por ter me dado saúde, força e discernimento para concluir esta pesquisa.

Agradeço à minha família, em especial aos meus pais Maria Cecília S. Belézia e Waldecir de Araujo pela educação, ensinamentos de vida e principalmente pela força e apoio durante o desenvolvimento deste trabalho. À minha irmã Fernanda Belézia Araujo, por seu companheirismo e por sempre torcer pelo meu sucesso.

Ao meu namorado e companheiro Renan Cecchi, sou grata pelo seu amor, apoio e dedicação que tem por nós. Renan, obrigada pelos seus gestos e palavras de incentivo nos momentos em que eu mais precisei. Enfim, você sabe que conseguimos isso juntos!

Agradeço aos professores do DInf e a todos os demais professores que contribuíram para a minha formação. Em especial, agradeço ao meu orientador Prof. Daniel Weingaertner pela confiança depositada em mim desde o início deste trabalho, sou grata por sua disponibilidade em me orientar e por sempre me amparar nos momentos de dúvida.

Agradeço aos colegas do Sistema de Bibliotecas da UFPR pelas trocas de experiência e pelo carinho que têm por mim. Um agradecimento especial à amiga Ligia Eliana Setenareski, primeira pessoa a acreditar que este trabalho seria possível.

A todos os amigos do laboratório de Visão, Robótica e Imagem (VRI) da UFPR, agradeço pelas valiosas conversas e momentos de descontração e incentivo.

Agradeço ao coordenador da iniciativa *Dokumente.br*, Prof. Paulo Astor Soethe, pela confiança no meu trabalho e pela liberação do material para criação da base de dados utilizada nesta pesquisa.

Aos membro da banca, Prof. Bogdan T. Nassu e Prof. Luiz Eduardo S. de Oliveira, agradeço pelas valiosas contribuições.

RESUMO

Projetos de digitalização em massa têm surgido em todo mundo. No Brasil, um dos exemplos é a iniciativa *Dokumente.br* que preocupa-se em disponibilizar acervos brasileiros em língua alemã. Parte de seu acervo é composto por jornais históricos escritos com a fonte gótica Fraktur e precisam ter seus caracteres reconhecidos opticamente. Um bom desempenho nesta tarefa está relacionado ao sucesso da etapa anterior do *workflow* de OCR, a análise de layout. As ferramentas OCR *open source* existentes não conseguem atingir bons resultados de análise de layout neste tipo de material. Com o objetivo de corrigir esta lacuna, propomos duas abordagens para a análise de layout dos jornais da iniciativa *Dokumente.br*: a primeira delas, que chamamos de GBN-MHS, é uma implementação do algoritmo "MHS 2017 System" proposto por Tran et al. (2017). A segunda abordagem é baseada em *deep learning* e a nomeamos de GBN-DL. Para avaliar o desempenho dos nossos métodos criamos o *German-Brazilian Newspaper Dataset (GBN 1.0)* e já preparamos seu *ground truth* para análise de layout e também para OCR. Comparamos os resultados obtidos pelo analisador de layout do software Tesseract no *dataset* proposto e os resultados obtidos pelos métodos GBN-MHS e GBN-DL. Criamos dois cenários de avaliação: um composto por jornais que foram representados no conjunto de treinamento (Cenário 1) e outro com páginas de jornais que não foram representados no conjunto de treinamento (Cenário 2). GBN-MHS e GBN-DL atingiram melhores resultados que Tesseract nos dois cenários avaliados. No Cenário 1, GBN-DL conseguiu 92,81% de acurácia, GBN-MHS obteve 88,12% e Tesseract atingiu apenas 71,83%. No Cenário 2, GBN-DL atingiu 96,96%, GBN-MHS conseguiu 95,16% e Tesseract obteve 88,15% de acurácia. Os bons resultados atingidos pelos métodos propostos demonstram o potencial das nossas abordagens e o experimento também comprova como as ferramentas OCR *open source* existentes não estão totalmente preparadas para trabalhar com documentos históricos.

Palavras-chave: digitalização de jornais. sistemas OCR. análise de layout de página. segmentação de páginas de jornais. OCR. OCR em Fraktur. Tesseract. OCRopy.

ABSTRACT

Mass digitization projects have emerged around the world. In Brazil, one example is the *Dokumente.br* initiative that aims at providing Brazilian collections in the German language. Part of its collection consists of historical newspapers written in the Gothic font Fraktur which need to have their characters recognized optically. A good performance in this task is related to the success of the previous OCR workflow step, the page layout analysis. The available open source OCR tools are not able to achieve good layout analysis results in this type of material. In order to correct this gap, two approaches to the layout analysis of the newspapers from the *Dokumente.br* initiative were proposed in this work: the first of these, which we call GBN-MHS, is an implementation of the "MHS 2017 System" algorithm proposed by Tran et al. (2017). The second proposal is based on deep learning and we call it GBN-DL. To evaluate the performance of the proposed methods we created the German-Brazilian Newspaper Dataset (GBN 1.0) and have already prepared its ground truth for layout analysis and also for OCR. We compared the results obtained by the layout analyzer from software Tesseract in the proposed dataset and the results obtained by the GBN-MHS and GBN-DL methods. We created two evaluation scenarios: one of them consists of newspapers that were represented in the training dataset (Scenario 1) and the other consists of newspaper pages that were not represented in the training dataset (Scenario 2). GBN-MHS and GBN-DL achieved better results than Tesseract in the two scenarios evaluated. In Scenario 1, GBN-DL achieved 92.81% in accuracy, GBN-MHS achieved 88.12% and Tesseract only 71.83%. In Scenario 2, GBN-DL reached 96.96%, GBN-MHS reached 95.16 % and Tesseract achieved 88.15 % in accuracy. The good results achieved by the proposed methods demonstrate the potential of our approaches, and the experiments also evidence that available open source OCR tools are not fully prepared to work with historical documents.

Keywords: digitalization of newspapers. OCR systems. page layout analysis. page segmentation of newspapers. OCR. Fraktur OCR. Tesseract. OCRopy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

1D	Uma Dimensão
2D	Duas Dimensões
AOSM	*Adaptive Over-Split and Merge*
API	*Application Programming Interface*
C3SL	Computação Científica e Software Livre
CC	Componente Conexo
CNN	*Convolutional Neural Network*
CP	Partição de Coluna
CRF	*Conditional Random Fields*
DEPAC	Departamento de Polonês, Alemão e Letras Clássicas
DFKI	*German Research Center for Artificial Intelligence*
DTA	*Deutsches Textarchiv*
FCNN	*Fully Convolutional Neural Network*
FN	*False Negative*
FP	*False Positive*
GBN	*German-Brazilian Newspaper Dataset*
GBN-DL	Método baseado em *deep learning*
GBN-MHS	Método baseado no MHS 2017 System
GCDH	Centro de Göttingen para Humanidades Digitais
HTML	*Hypertext Markup Language*
ICDAR	*International Conference on Document Analysis and Recognition*
IMPACT	*Improving Access to Text*
ISRI	*Information Science Research Institute*
IUPR	*Image Understanding Pattern Recognition*
LSTM	*Long Short-Term Memory*
MHS	*Multilevel Homogeneity Structure*
MLP	*Multi-Layer Perceptron*
OCR	*Optical Character Recognition*
OSD	Somente detecção de script e orientação
PI	Pixels de interesse
PRImA	*Pattern Recognition & Image Analysis*
RDCL	Competição em *Recognition of Documents with Complex Layouts*
Sibi	Sistema de Bibliotecas
SVM	*Support Vector Machine*
TP	*True Positive*

UFPR	Universidade Federal do Paraná
UNLV	Universidade de Nevada, Las Vegas
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Markup Language*

LISTA DE SÍMBOLOS

CC_i	i-ésimo componente do conjunto dos componentes conexos
$B(CC_i)$	*Bounding box* de CC_i
Xl_i, Yl_i	Coordenada superior esquerda do $B(CC_i)$
Xr_i, Yr_i	Coordenada inferior direita do $B(CC_i)$
$\mathcal{A}_i$	Área do CC_i
λ_i	Densidade do CC_i
$\mathcal{A}_i^B$	Área do *bounding box* do CC_i
Inc_i	Número de *bounding box* contidos em $B(CC_i)$
HW_i^{rate}	Razão entre a largura e altura do CC_i
T^{area}	Limiar de $\mathcal{A}_i$ para o filtro heurístico
T^{inside}	Limiar de Inc_i para o filtro heurístico
T^{rate}	Limiar de HW_i^{rate} para o filtro heurístico
$\Re$	Região da página
$a \times b$	Tamanho da região $\Re$
P_H	Projeção horizontal da região $\Re$
p_x	Pixels do eixo x de P_H
Z_H	Histograma suavizado de P_H
s	Parâmetro para a suavização de P_H
G_H	Gradientes da projeção suavizada Z_H
L_H	Pontos de mínimos e máximos locais de G_H
Δ	Conjunto das distâncias entre dois pontos adjacentes de L_H
V	Variância de Δ
T_{Var}	Limiar de V para a classificação de homogeneidade da região
w_i	i-ésima linha branca da região $\Re$
b_i	i-ésima linha preta da região $\Re$
w	Conjunto das w_i
b	Conjunto das b_i
k	Quantidade de *folds* usados na validação cruzada
lr	Valor de *learning rate* usado no treinamento da CNN

SUMÁRIO

1 INTRODUÇÃO

Uma grande quantidade de materiais digitais tem sido produzida por parte de iniciativas de empresas e instituições públicas em todo o mundo. Isso explica a proliferação das bibliotecas digitais e a consequente criação da Federação das Bibliotecas Digitais[1] (*Digital Library Federation*).

Um exemplo de projeto de digitalização em massa é o Google Books[2], anteriormente conhecido como Google Book Search e Google Print. O Google Books é um serviço criado pela Google que permite buscas por textos completos de livros que foram digitalizados a partir do acervo de várias bibliotecas no mundo, tais como a Universidade de Harvard dos Estados Unidos e a Universidade Keio do Japão. Na Europa, há o Projeto IMPACT[3], financiado pela Comissão Europeia e coordenado pela Biblioteca Nacional dos Países Baixos. Seu objetivo é melhorar significativamente o acesso a textos históricos e eliminar as barreiras que impedem a digitalização em massa do patrimônio cultural europeu.

Outro exemplo de projeto de digitalização na Europa é o DTA[4] (em Alemão *Deutsches Textarchiv*). Ele tem digitalizado um grande corpus interdisciplinar de documentos em língua alemã do período entre 1600 e 1900. Atualmente, existem mais 3360 obras disponíveis para pesquisa.

No Brasil, projetos de digitalização em massa também têm surgido como, por exemplo, a iniciativa *Dokumente.br*. A iniciativa *Dokumente.br* é uma parceria entre a UFPR (Universidade Federal do Paraná), a Fundação Fritz Thyssen e o Centro de Göttingen para Humanidades Digitais (GCDH) na Alemanha. Na UFPR, este projeto envolve o Sistema de Bibliotecas (Sibi), o Centro de Computação Científica e Software Livre (C3SL), o Departamento de Polonês, Alemão e Letras Clássicas (DEPAC) e o Programa de Pós Graduação em Letras. Seu objetivo é disponibilizar acervos brasileiros em língua alemã a fim de ampliar sua visibilidade e permitir a pesquisa em seu conteúdo.

Segundo a página web[5] do projeto, esses materiais, que encontram-se em museus, bibliotecas, arquivos e coleções particulares, têm grande relevância histórico-cultural e literária, entretanto seu potencial como objeto ou fonte de pesquisa permanece ainda quase inexplorado.

Isso deve-se, em parte, ao fato de que realizar pesquisa em imagens digitalizadas é um trabalho tedioso e lento, sendo assim, a transcrição do material é necessária. A transcrição manual normalmente é uma tarefa custosa, pode exigir muitas horas de trabalho humano. Também há casos em que uma especialização em algum idioma por parte do humano é necessária. Devido a essas problemáticas, os esforços para automatizar esse processo têm aumentado no decorrer dos anos. O Reconhecimento Óptico de Caracteres (OCR) avançou muito, mas o estado da arte ainda não alcançou resultados equivalentes a humanos em muitas das tarefas do *workflow* de OCR. A identificação correta de regiões de texto em páginas de layout variável, com intercalação de imagens e texto, em papel de baixa qualidade, é uma dessas tarefas. O desempenho desta tarefa, conhecida como análise de layout, é crucial para o bom resultado de OCR do documento.

Jornais históricos, em particular, apresentam características desafiadoras para a análise de layout, pois a maioria dos algoritmos descritos na literatura foram desenvolvidos para

[1] https://www.diglib.org/
[2] https://books.google.com/
[3] http://www.impact-project.eu/home/
[4] http://www.deutschestextarchiv.de
[5] https://dokumente.ufpr.br

reconhecer documentos contemporâneos, tais como revistas e livros, com layouts e fontes mais regulares (Dai-Ton et al., 2016; Tran et al., 2017). Sendo assim, os sistemas OCR de código aberto existentes, tais como o Tesseract (Smith, 2007) e Ocropy (Breuel, 2008), não são capazes de identificar corretamente as regiões de texto de jornais históricos e, por consequência, o resultado final de seu OCR fica prejudicado.

A última RDCL (Competição em *Recognition of Documents with Complex Layouts*) promovida pela ICDAR (*International Conference on Document Analysis and Recognition*) em 2017 também provou como os métodos de segmentação e análise de layout das ferramentas OCR atuais estão ficando ineficientes comparados aos novos métodos que têm surgido (Clausner et al., 2017).

A lacuna apresentada por essas ferramentas demostra a importância desta pesquisa, que tem por objetivo desenvolver técnicas de análise de layout a fim de melhorar o resultado final do OCR dos jornais históricos Germano-Brasileiros gerenciados pela iniciativa *Dokumente.br*. Esses jornais, produzidos pela imprensa brasileira entre 1863 e 1940, retratam o cotidiano das comunidades germânicas do Brasil e eram fontes de informação altamente valorizadas. Circulavam de casa em casa e de mão em mão, sendo lidos em voz alta em uma época em que a maioria das pessoas não sabiam ler. Eles foram escritos usando o idioma Português e o Alemão, sendo que grande parte do texto em Alemão está em Fraktur, uma fonte de estilo gótico utilizada em alguns idiomas da Europa entre o século XVI e meados do século XX.

Para avaliar nossas técnicas de análise de layout criamos, em parceria com o Programa de Pós Graduação em Letras Alemão da UFPR, uma base de dados com parte deste acervo de jornais e a rotulamos para análise de layout e OCR de Fraktur/Português. Atualmente, a *German-Brazilian Newspaper Dataset* (GBN 1.0) é a única base de dados de jornais históricos em Fraktur rotulada para as duas tarefas do *workflow* de OCR.

As técnicas propostas neste trabalho para a análise de layout das páginas do GBN *Dataset* exploram duas vertentes do Reconhecimento de Padrões. Na primeira delas, que chamamos de GBN-MHS, as características para a classificação são definidas de forma heurística baseada em uma análise dos componentes conexos da página. O GBN-MHS é uma implementação do algoritmo vencedor da RDCL 2017, o "MHS 2017 System" (Tran et al., 2017), com algumas modificações propostas por nós. Para a segunda técnica, que nomeamos de GBN-DL, usamos uma abordagem com *deep learning* para a extração automática das características e classificação das regiões da página.

Por fim, comparamos o resultado alcançado pelas técnicas propostas de análise de layout com o resultado obtido com o analisador de layout embutido no Tesseract no *dataset* proposto.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver técnicas de análise de layout de página para os jornais Germano-Brasileiros a fim de melhorar o resultado de seu OCR. A partir disso, almejam-se os seguintes objetivos específicos:

- Compreender o funcionamento das ferramentas OCR *open source* existentes para Fraktur: Tesseract e OCRopy.
- Criar um *dataset* com amostras representativas dos jornais em Fraktur da iniciativa *Dokumente.br* e rotulá-los para análise de layout e OCR.
- Desenvolver um método de análise de layout baseado no "MHS 2017 System" (Tran et al., 2017) devido ao seu bom desempenho na RDCL 2017.

- Desenvolver um método de análise de layout baseado em *deep learning*.
- Comparar os métodos propostos por este trabalho com o método de análise de layout disponibilizado no Tesseract, usando o *dataset* criado.

1.2 CONTRIBUIÇÕES DO TRABALHO

Entre as principais contribuições deste trabalho estão:

- Estudo teórico e levantamento bibliográfico sobre os principais métodos de análise de layout de página existentes na literatura.
- Implementação em código aberto dos dois métodos de análise de layout propostos.
- *Dataset* de jornais em Fraktur rotulados para análise de layout e OCR.
- Comparação de três métodos de análise de layout (GBN-MHS, GBN-DL e Tesseract) no *dataset* proposto.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido conforme os seguintes capítulos: o Capítulo 2 apresenta uma visão geral das tarefas do *workflow* para OCR e os principais fundamentos sobre Redes Neurais Artificiais. No Capítulo 3 são descritas as técnicas de análise de layout de página presentes na literatura e como está o estado-da-arte em OCR de Fraktur. Também detalhamos o funcionamento das duas ferramentas *open source* existentes para o *workflow* de OCR em Fraktur: Tesseract e OCRopy. O *dataset* criado e as duas abordagens propostas para análise de layout são apresentadas no Capítulo 4. O Capítulo 5 contém a descrição dos experimentos, os parâmetros utilizados e as métricas propostas para as avaliações. Os resultados obtidos nos experimentos são mostrados e discutidos no Capítulo 6. Por fim, no Capítulo 7 apresentamos nossas conclusões e projeções para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo apresentamos uma visão geral das tarefas do *workflow* de OCR de documentos históricos (Seção 2.1) e introduzimos o conceito de Redes Neurais Artificiais (Seção 2.2), abordagem esta presente em uma de nossas propostas.

2.1 VISÃO GERAL DE OCR

Para que um documento no formato de imagem tenha seus caracteres reconhecidos e seu conteúdo torne-se pesquisável, são necessários vários processamentos. O conjunto desses processamentos (Figura 2.1) compõem um sistema OCR (Eikvil, 1993). Inicialmente é feita a aquisição da imagem. Em seguida a imagem é pré-processada. As regiões de texto e não-texto são identificadas e classificadas. Por fim, os caracteres das regiões de texto são reconhecidos opticamente e são gerados arquivos de saída com essas informações. Descrevemos cada um desses processos detalhadamente a seguir.

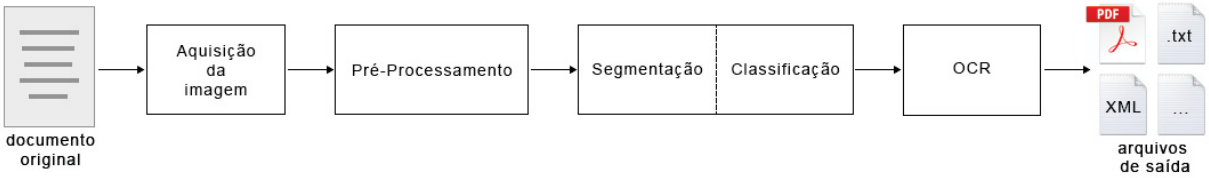


Figura 2.1: Processos básicos de um sistema OCR: aquisição da imagem, pré-processamento, segmentação e classificação das regiões, OCR e geração de arquivos de saída.
Fonte: Adaptado de Eskenazi et al. (2017).

Como mostra a Figura 2.1, o primeiro estágio de um sistema OCR é a aquisição da imagem. Ou seja, é a transformação do dado analógico que se deseja reconhecer em um dado de formato digital. Essa transformação é realizada por meio de um *scanner* óptico ou outro dispositivo de entrada digital, como por exemplo uma câmera, que captura opticamente os dados e transforma-os em uma imagem digital.

Após a aquisição da imagem é comum que ela passe por algum processo de correção e binarização. Esses processos objetivam melhorar a eficiência das tarefas posteriores. Apesar de não ser um processo obrigatório, a maioria dos sistemas OCR implementam algum tipo de pré-processamento na imagem. Também há sistemas que trabalham com imagens coloridas ou em tons de cinza (ou seja, não a binarizam) e não realizam correções.

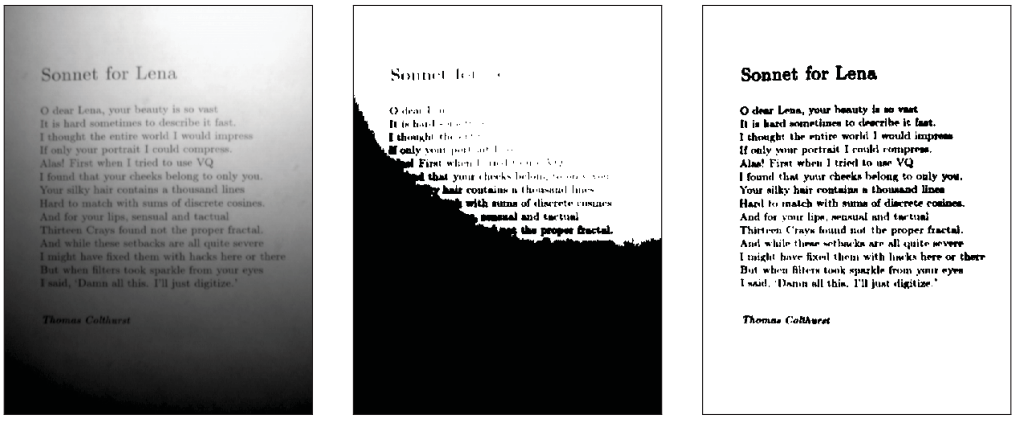
A binarização ou limiarização consiste em converter uma imagem de vários níveis de cinza para uma imagem com apenas dois níveis. Para isso normalmente são usadas as cores preta e branca. Sua função é separar regiões de interesse e regiões de não interesse através da escolha de um ponto de corte. No caso do reconhecimento de textos, o objetivo é segmentar os pixels de caracteres e elementos gráficos dos pixels do fundo da imagem.

A escolha de um limiar global é o método mais simples de binarização. Baseando-se na imagem toda, é definido um único ponto de corte para a segmentação. O pixel com valor abaixo do nível de corte é convertido para uma cor (por exemplo: preto) e o pixel com valor acima do nível de corte é convertido para outra cor (por exemplo: branco).

Em muitas situações, entretanto, apenas um limiar não é suficiente para segmentar corretamente todas as regiões de interesse da imagem. Nestes casos usam-se técnicas de

limiarização variáveis e multiníveis que se baseiam em janelas deslizantes na imagem. Elas são chamadas de técnicas locais e adaptativas (Gonzalez e Woods, 2006).

A Figura 2.2 mostra um exemplo onde o uso de um limiar adaptativo é mais adequado devido à imagem apresentar regiões com diferença de iluminação.



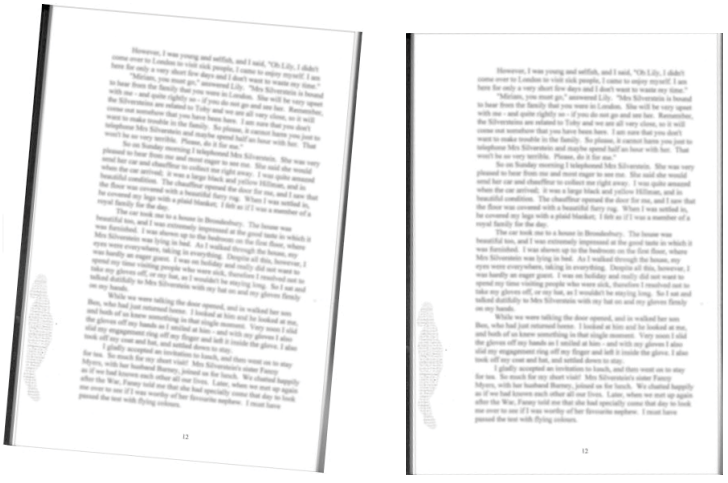
(a) (b) (c)

Figura 2.2: (a) Imagem original, (b) binarização com limiar global e (c) binarização com limiar adaptativo.
Fonte: Adaptado de Robert Fisher e Wolfart (2003).

Em seguida podem ser realizadas correções na imagem. As mais comuns implementadas pelos sistemas OCR são as de inclinação, as de ondulação da página e as de remoção de ruído. Elas são necessárias para corrigir falhas do processo de aquisição da imagem.

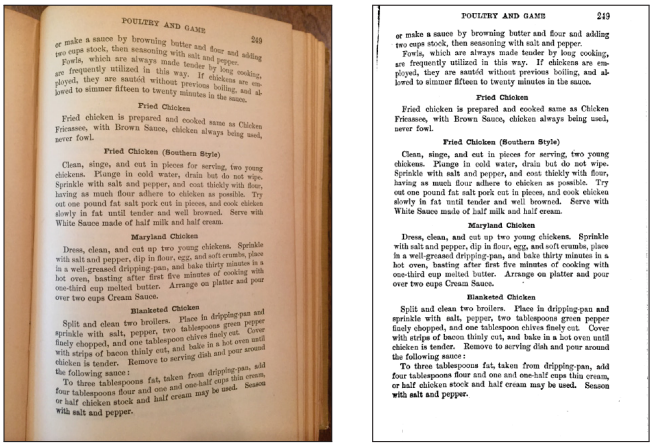
Na correção da inclinação da página, o ângulo de inclinação é detectado e com base nele é feita a rotação da página (Figura 2.3). Já a ondulação normalmente é ocasionada devido à encadernação do material ser do tipo lombada. A Figura 2.4 mostra um exemplo de página com ondulação e sua correção.

Em documentos históricos é comum a presença de ruídos. Ruídos são variações indesejadas de cor e/ou luminância descendentes do processo de aquisição ou transmissão da imagem. Vários fatores influenciam no surgimento de ruído: condições do meio ambiente, mal desempenho nos sensores no momento da aquisição, falhas de transmissão, etc. Em sistemas OCR



(a) (b)

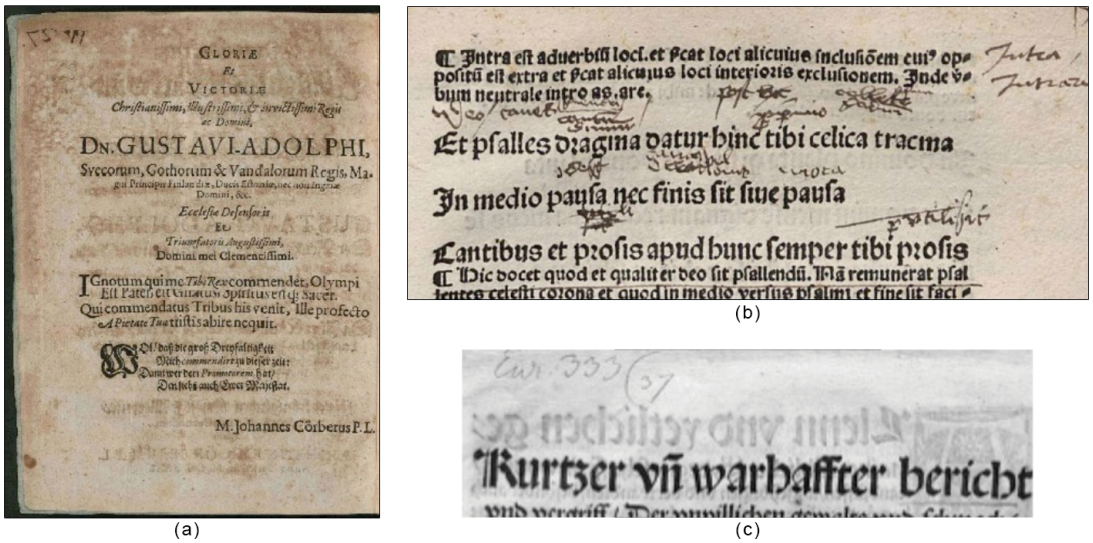
Figura 2.3: Página com inclinação (a) e sua correção (b).
Fonte: Autoria própria.



(a) (b)

Figura 2.4: Página com ondulação (a) e sua correção (b).
Fonte: Adaptado de Zucker (2016).

é considerado como ruído tudo que prejudique o processo de segmentação e consequentemente o reconhecimento dos caracteres. A Figura 2.5 mostra alguns tipos de ruído que são comuns em documentos históricos: bordas criadas no processo de aquisição da imagem (Figura 2.5(a)), manchas de envelhecimento do papel (Figura 2.5(a)), anotações manuscritas (Figura 2.5(b)) e exibição de caracteres da página de trás devido à transparência do papel (Figura 2.5(c)).



(a) (b) (c)

Figura 2.5: Alguns tipos de ruído: bordas da página e manchas de envelhecimento do papel (a), anotações manuscritas (b) e exibição de caracteres da página de trás devido à transparência do papel (c).
Fonte: Adaptado de Neudecker (2010).

Depois da imagem ter sido adquirida e pré-processada, ela precisa ser segmentada em regiões de texto e não-texto. Esse processo é conhecido como análise de layout. Existem dois tipos de análise para as regiões: a análise de layout física e a análise de layout lógica (Eskenazi et al., 2017).

A análise de layout física, também conhecida como análise geométrica, classifica as regiões de acordo com a natureza de seu conteúdo, ou seja, em regiões de texto, figura, gráfico, separadores de coluna, etc. A análise de layout lógica leva em consideração a semântica da

área, ou seja, a função que ela desempenha no documento, tal como título, legenda, corpo do documento, nota de rodapé, etc.

A análise de layout física é um processo obrigatório em sistemas OCR, pois estes precisam identificar quais das áreas rotuladas possuem texto a ser reconhecido. Já a análise de layout lógica é um processo que traz um maior detalhamento do documento, sendo assim, é um processo opcional no *workflow* de OCR.

A Figura 2.6 mostra o resultado esperado da análise de layout física em dois exemplos do conjunto de dados do projeto IMPACT[1]. Este conjunto de dados contém páginas digitalizadas de livros e jornais históricos das principais bibliotecas da Europa (Papadopoulos et al., 2013).

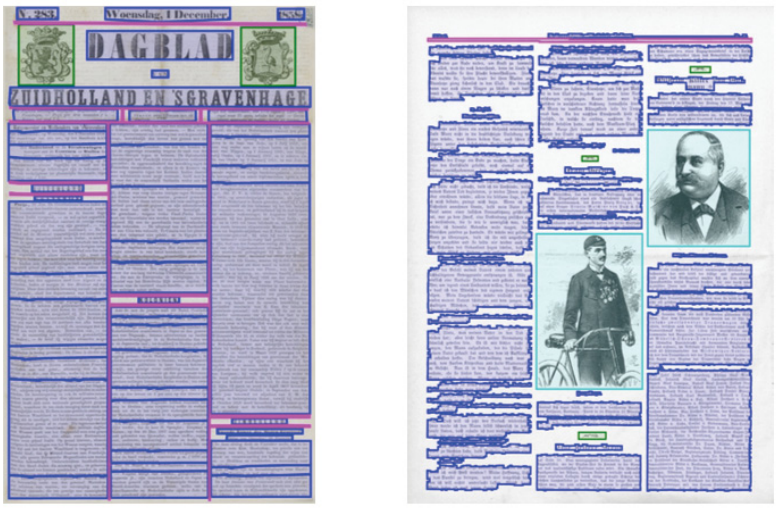


Figura 2.6: Contornos esperados da análise de layout física de duas amostras do conjunto de dados do projeto IMPACT. As regiões de texto estão representadas em **azul**, regiões de imagem em **ciano**, elementos gráficos em **verde** e separadores de coluna em **magenta**.

Fonte: Antonacopoulos et al. (2013).

Com a análise de layout concluída, as regiões de texto encontradas são dadas como entrada a um algoritmo responsável por reconhecer os caracteres. Com base nas características de cada caractere, o algoritmo decide que letra, número ou sinal de pontuação aquele segmento de imagem representa.

Os métodos reconhecedores de caractere podem ser divididos em dois grandes grupos: os que usam uma abordagem com segmentação (*segmentation-based*) e os que usam uma abordagem sem segmentação (*segmentation-free*).

Os métodos com segmentação segmentam as linhas de texto em caracteres ou candidatos a caractere e em seguida aplicam um classificador de forma individualizada em cada caractere (ou candidato a caractere). O resultado deste processo geralmente é apresentado em uma estrutura de reconhecimento em malha, que contém alternativas de segmentação e reconhecimento.

Na abordagem sem segmentação, tendo como base um conjunto de imagens de linhas de texto e sua devida transcrição, o classificador aprende as características de cada caractere e como segmentá-los. Esse tipo de classificador é baseado em Redes Neurais Artificiais (assunto explorado na Seção 2.2).

Com os caracteres já reconhecidos é necessário organizar esta e outras informações do documento em um arquivo de saída. A saída de um sistema OCR é sempre um arquivo que contém texto. Este arquivo pode ser de diversos formatos, como por exemplo um PDF pesquisável que, além de conter o texto reconhecido, contém também os elementos gráficos do

[1] http://www.impact-project.eu/

documento (figuras, separadores de colunas, tabelas, etc.). A saída também pode ser somente o texto puro, contido em um arquivo com extensão ".txt" por exemplo.

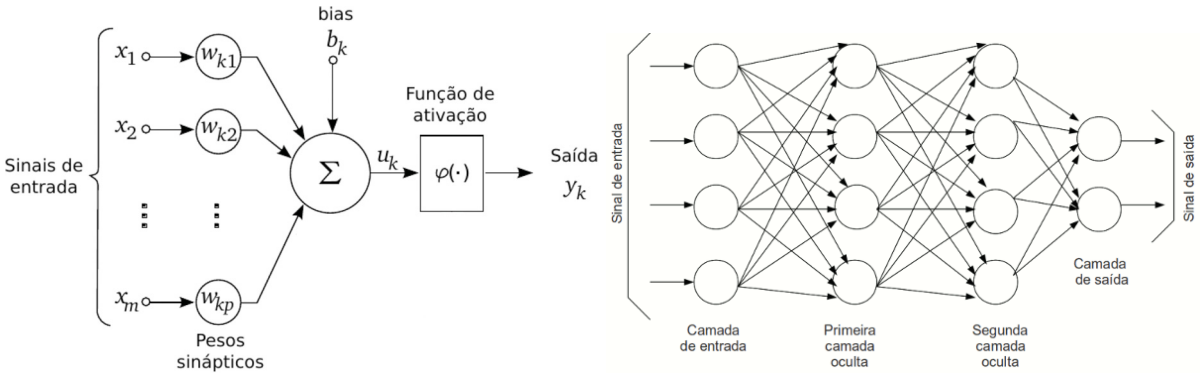
Há ainda outro tipo de saída bastante utilizado pelos sistemas OCR, a saída no formato hOCR (Breuel, 2007). O hOCR é um padrão aberto criado especificamente para representar os dados resultantes do processo de OCR. Ele usa linguagem de marcação como XML (*Extensible Markup Language*), HTML (*Hypertext Markup Language*) e XHTML (*Extensible Hypertext Markup Language*) para codificar informações referentes ao texto, layout e outras características do documento. Já nas bases de dados rotuladas para análise de layout, o padrão de representação comumente usado é o XML PAGE (Pletschacher e Antonacopoulos, 2010) que é muito semelhante ao hOCR.

2.2 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNAs) foram inicialmente propostas por Pitts e McCulloch (1947). Tratam-se de modelos computacionais inspirados no comportamento do cérebro humano que são capazes de realizar processos de aprendizado. Uma RNA é composta por unidades básicas de processamento (neurônios artificiais) interconectadas. Cada neurônio artificial é constituído por seus pesos, *bias* e uma função de ativação que determina se o neurônio está ativo ou não para um determinado conjunto de características dado como entrada.

Um neurônio artificial isolado funciona como um classificador linear e é conhecido como "Perceptron" (Rosenblatt, 1958). A Figura 2.7(a) mostra seu funcionamento. Cada sinal de entrada x_j na sinapse j conectado a um neurônio k é multiplicado por seu peso sináptico w_{kj} e o resultado final é somando ao *bias*. Esse valor (u_k) é passado para a função de ativação que determina a saída do classificador.

Sendo assim, neurônios artificiais interconectados em um fluxo unidirecional (*feed-forward*) constituem uma RNA ou "Perceptron Multi-Camadas" (*Multi-Layer Perceptron* - MLP). O aprendizado da rede ocorre da seguinte maneira: na tentativa de representar um comportamento desejado, são atribuídos valores aos pesos de entrada. Os parâmetros (pesos e *bias*) dos neurônios das camadas intermediárias da rede são atualizados em função do erro calculado na camada de saída. Quando a saída não é a correta, o erro calculado é retropropagado até a camada de entrada. Esse processo é conhecido como *backpropagation*. A Figura 2.7(b) mostra a estrutura de uma RNA.



(a) Neurônio Artificial (Perceptron) (b) Rede Neural Artificial (RNA)

Figura 2.7: Um Neurônio Artificial (Perceptron) em (a) e um exemplo de arquitetura de RNA com duas camadas intermediarias (também conhecidas como camadas ocultas) em (b).
Fonte: Adaptado de HAYKIN (2001).

Arquiteturas de RNAs formadas por diversas camadas intermediárias treináveis (camadas densas) e capazes de extraírem características particulares do problema, são utilizadas nas técnicas de Aprendizagem Profunda (*deep learning*) (LeCun et al., 1998).

Diferente dos métodos tradicionais, onde os descritores de características do problema são pré-definidos por um humano, no aprendizado profundo a representação das características é extraída automaticamente pela máquina. O dado é repassado entre as camadas de uma RNA, sendo que nas camadas mais profundas ele tende a ter um nível de abstração mais alto, ou seja, possui uma representação mais próxima da máquina (LeCun et al., 2015).

Técnicas de aprendizagem profunda vêm sendo aplicadas na resolução de diversos problemas tais como na identificação biométrica (Severo et al., 2018; Zanlorensi et al., 2018; Lucio et al., 2018), segmentação de imagens médicas (Alves et al., 2018), reconhecimento óptico de caracteres (Laroca et al., 2019) e reconhecimento de dígitos manuscritos (Hochuli et al., 2018).

O aprendizado profundo colocou os problemas relacionados à área de Reconhecimento de Padrões em um outro patamar, pois agora é possível criar modelos capazes de aprender funções complexas que amplificam a relevância de características representativas e suprimem as características menos relevantes. Um dos exemplos de arquitetura de aprendizado profundo é a Rede Neural Convolucional. Ela foi usada em um dos nossos métodos de segmentação de layout propostos e por isso será abordada a seguir.

2.2.1 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs) são redes projetadas para realizarem a auto-aprendizagem de filtros e consequentemente reconhecerem padrões em dados dispersos na forma de matrizes, como por exemplo imagens e espectrogramas de áudio (LeCun et al., 2010, 2015).

Nas CNNs as características dos dados são extraídas pelas diversas camadas da rede que podem ser de três tipos: convolucionais, de ativação (por exemplo *Rectified Linear Unit* - ReLU) e de agregação (*pooling*). Além dessas camadas, também há as camadas totalmente conectadas (*fully connected*) que são posicionadas ao final da rede e são responsáveis por classificar os mapas de características (*feature maps*) aprendidos nas camadas anteriores. A Figura 4.10 mostra um exemplo desta arquitetura.

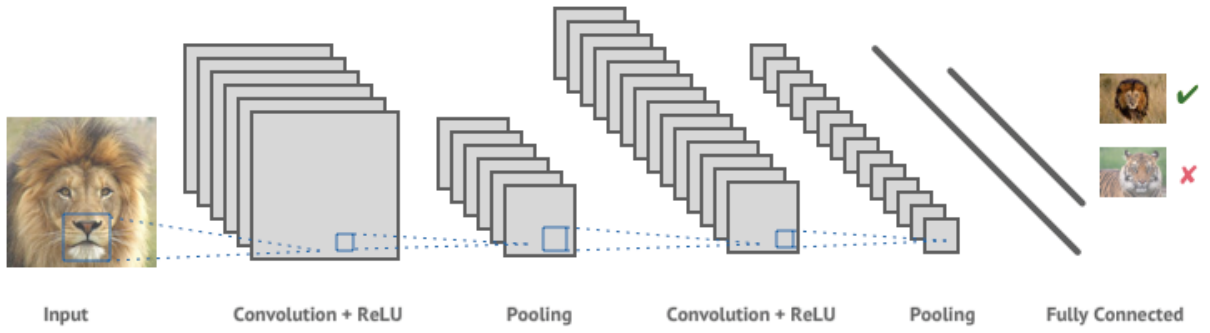


Figura 2.8: Um exemplo de arquitetura CNN.
Fonte: https://shafeentejani.github.io/2016-12-20/convolutional-neural-nets/.

As camadas convolucionais aplicam filtros (*kernels*) nos pixels da imagem através de janelas deslizantes. Novas imagens (*feature maps*) um pouco menores são então geradas com as características mais enfatizadas detectadas pelos filtros. Sendo assim os filtros, que são matrizes com pesos (ou valores), podem ser considerados como descritores de características. A Figura

2.9 mostra um exemplo da operação de convolução com um filtro (*kernel*) de dimensão 2×2 que se desloca em uma matriz de entrada (*input*) a cada passo 1 (*stride* = 1). Os elementos da janela da matriz são multiplicados pelos elementos do *kernel* e em seguida são somados. O resultado (*output*) é uma matriz de tamanho 3×2.

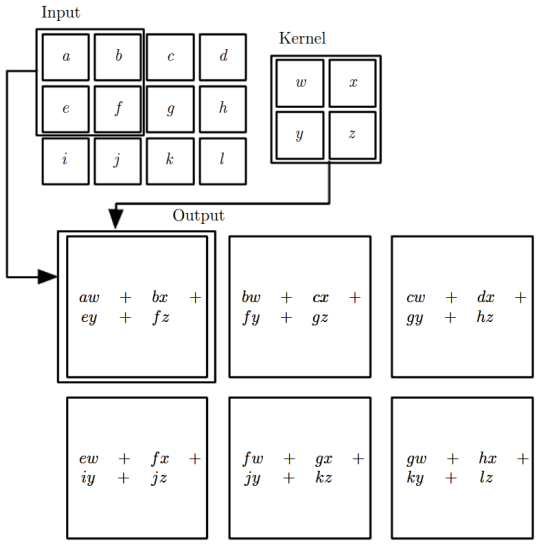
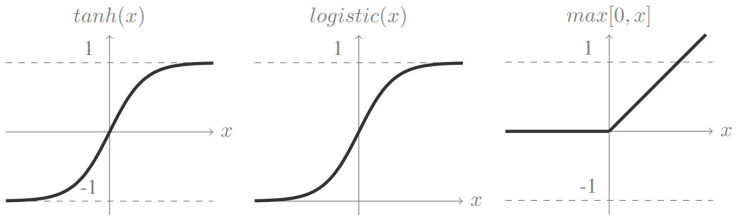


Figura 2.9: Um exemplo de operação de convolução. Um filtro (*kernel*) de dimensões 2×2 é aplicado a uma matriz de entrada (*input*) 3×4. Os elementos da janela da matriz são multiplicados pelos elementos do *kernel* e em seguida são somados. O *kernel* é deslocado a cada passo 1 (*stride* = 1) e a operação é realizada novamente, resultando em uma matriz de saída (*output*) do tamanho 3×2.
Fonte: Goodfellow et al. (2016).

A saída da camada de convolução é passada para uma função de ativação. A função de ativação é um componente matemático incluído na rede a fim de permitir a solução de problemas mais complexos, problemas que não podem ser resolvidos de forma linear (LeCun et al., 2015). A função de ativação transforma a saída linear da camada de convolução em uma saída não linear. É ela quem decide se o neurônio será ativado ou não, ou seja, se ele é relevante para informação fornecida ou não. Existem diversos tipos funções de ativação, a maioria delas seguem a regra de normalizar a saída do neurônio para o intervalo de 0 a 1 ou -1 a 1. Figura 2.10 mostra exemplos de funções de ativação.



(a) Tangente Hiperbólica (b) Logística (c) ReLU

Figura 2.10: Algumas funções de ativação: Tangente Hiperbólica (a), Logística (b) e ReLU (c).
Fonte: Ponti et al. (2017).

Após a camada de convolução é comum a presença de uma camada de agregação (*pooling*). As camadas de agregação adicionam robustez ao modelo, pois contribuem para o aprendizado de características invariantes à translação (Ponti et al., 2017). Nesta etapa os mapas

de características são redimensionados de forma a reduzir a dimensão do problema. O processo consiste em deslizar uma janela na imagem, sem que haja sobreposição das regiões, utilizando uma passo (*stride*) de tamanho maior ou igual ao tamanho da janela.

Os tipos de agregação podem ser Max, Média, Mediana, entre outros. A regra mais utilizada é a do valor máximo (*Max Pooling*), na qual o maior valor da região é extraído. A Figura 2.11 mostra um exemplo dessa regra onde é usada uma janela 2×2 com um *stride* 2, reduzindo assim a dimensão da imagem pela metade.

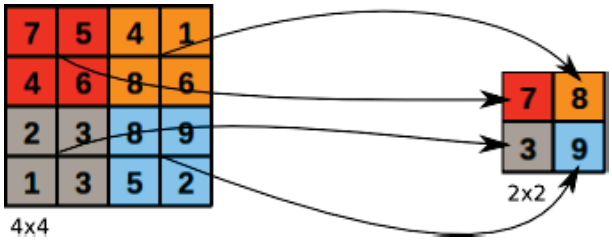


Figura 2.11: *Max Pooling* em uma matriz 4×4 usando um *stride* 2 e uma janela de 2×2.
Fonte: Hochuli (2019).

3 REVISÃO DA LITERATURA

Este Capítulo apresenta as principais técnicas de análise de layout de página presentes na literatura (Seção 3.1) e o estado-da-arte em OCR em Fraktur (Seção 3.2). Também são detalhados os dois sistemas *open source* existentes para OCR em Fraktur: Tesseract (Subseção 3.3.1) e OCRopy (Subseção 3.3.2).

3.1 ANÁLISE DE LAYOUT

Diversos métodos de análise de layout de página têm sido propostos nos últimos anos. Eles podem ser divididos em três categorias: métodos *top-down*, métodos *bottom-up* e métodos híbridos.

A abordagem *top-down* é a mais antiga (Baird et al., 1990; Nagy et al., 1992). Ela analisa a estrutura global do documento e em seguida corta-o recursivamente nas direções vertical e horizontal com base nos espaçamentos em branco. O algoritmo parte do pressuposto que esses espaçamentos são os limites de colunas e parágrafos e, em seguida, segmenta os blocos de texto em linhas e em caracteres.

Apesar dos métodos *top-down* serem simples e rápidos (porque não são iterativos), apresentam uma desvantagem: só conseguem bons resultados de segmentação em documentos de layouts retangulares (também chamados de layouts Manhattan), ou seja, são ineficientes para revistas e jornais, já que estes materiais possuem tipos diversos de layouts. Além dos citados anteriormente, Breuel (2002) e Sun (2005) também são exemplos do uso da abordagem *top-down*.

Uma outra abordagem de segmentação é a *bottom-up* (Wahl et al., 1982; O'Gorman, 1993; Chowdhury et al., 2007). Baseando-se na análise de componentes conexos, o algoritmo agrupa os componentes em caracteres, depois em palavras, em seguida em linhas e finalmente em parágrafos ou regiões.

Métodos *bottom-up* são aplicáveis a vários tipos de layout porque não requerem hipóteses sobre a estrutura global do documento, porém, por realizarem segmentação e agrupamento de maneira iterativa, têm um alto custo computacional. Além disso, a sensibilidade da medida usada como parâmetro para o agrupamento dos componentes iniciais pode levar a erros de segmentação em páginas que possuem fontes variadas e/ou de tamanhos diferentes. Como cada estilo de documento exige a escolha de parâmetros diferentes (tamanho da fonte, características de ruído, distribuição do tamanho do componente conexo, etc.), a segmentação fica mais propensa a erros. Exemplos recentes de implementações *bottom-up* foram propostos por Ferilli et al. (2010) e Pan et al. (2010).

Na tentativa de superar as fraquezas das abordagens *top-down* e *bottom-up*, Okamoto e Takahashi propuseram em 1993 um método híbrido que combinava características das duas abordagens. Nos métodos híbridos a abordagem *bottom-up* é usada para localizar delimitadores, como por exemplo espaçamentos retangulares em branco e limites de regiões. Esses delimitadores são usados para inferir um layout *top-down* no documento. Depois disso, com base nos dois layouts, o algoritmo detecta as regiões.

Devido a essa combinação, algoritmos híbridos (Smith, 2009; Dai-Ton et al., 2016; Tran et al., 2017) podem superar erros de segmentação excessiva (*over-segmentation*) causados por métodos *bottom-up* e alcançarem melhores resultados em documentos de layouts não retangulares comparado os métodos *top-down*.

Entretanto, corrigir erros de *over-segmentation* não é uma tarefa trivial pois é difícil definir com precisão os limites das áreas de texto. Eles podem estar muito próximos uns dos outros, causando erros de *under-segmentation*; ou os componentes conexos podem estar com um espaçamento muito grande, causando erros de *over-segmentation* (Figura 3.1).

Figura 3.1: Erros de *under-segmentation* e *over-segmentation*.
Fonte: Dai-Ton et al. (2016).

Em 2016, Dai-Ton et al. criaram um algoritmo usando uma abordagem de corte e fusão adaptativas que reduziu bastante a ocorrência desses tipos de erros de segmentação. Juntamente com o método AOSM (*Adaptive Over-Split and Merge*) de Dai-Ton et al., outros dois algoritmos híbridos de segmentação e análise de layout vêm se destacando na área: o CVML de Sangyu Han, Soyeon Kim e Hyung Il Koo da Universidade de Ajou na Coreia (Antonacopoulos et al., 2015); e o MHS 2017 System (Sistema para análise de layout de documento que usa uma *Multilevel Homogeneity Structure*) de Tran et al. (2017).

AOSM, CVML e MHS 2017 System atingiram altas taxas de acerto nos três cenários de avaliação propostos pela RDCL2017 (Clausner et al., 2017) – competição de *Recognition of Documents with Complex Layouts* – da ICDAR2017 – *International Conference on Document Analysis and Recognition* – (Figura 3.2).

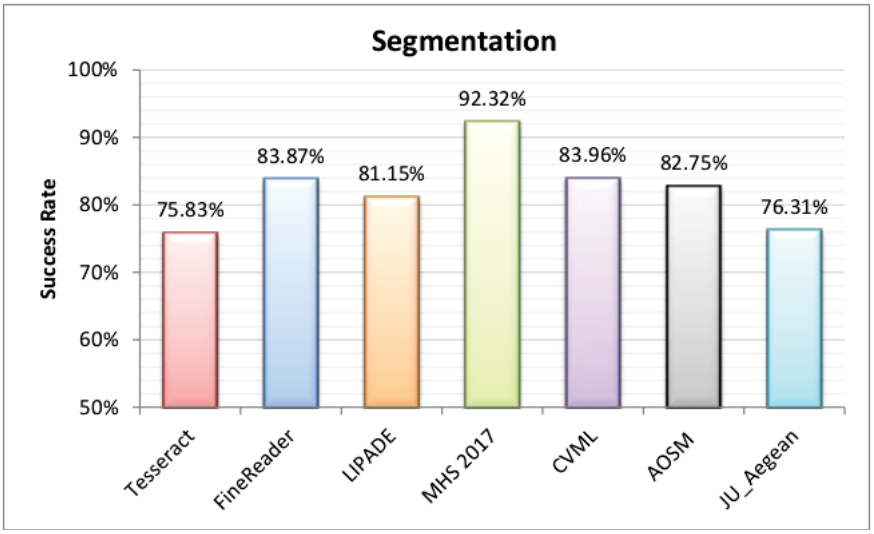
O MHS 2017 System superou até mesmo o principal software proprietário da área, o ABBYY FineReader Engine 11[1], e o sistema *open source* mais popular de análise de layout e OCR – Tesseract 3.04.

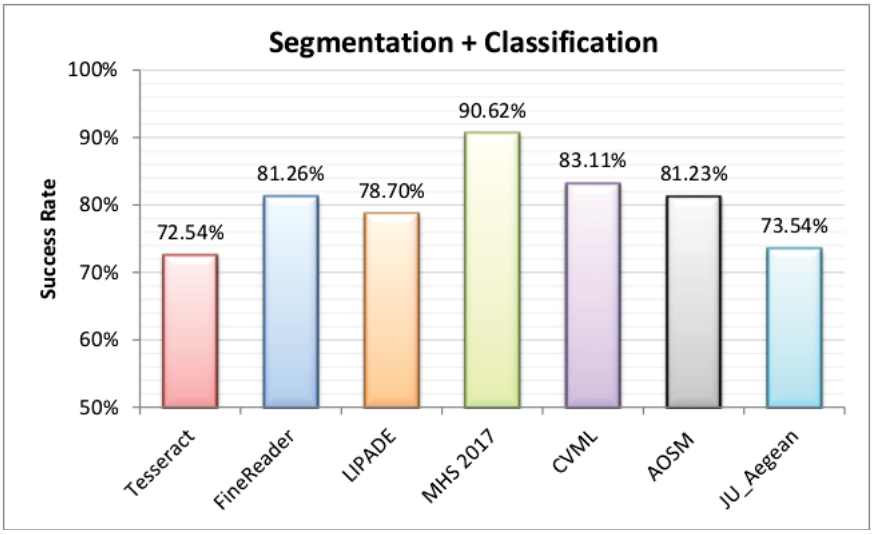
Embora na última RDCL não tenham ocorrido submissões de métodos baseados em aprendizagem de máquina, esse tipo de abordagem também vêm sendo estudada. Bukhari et al. (2010) e Chen et al. (2013) propuseram métodos que são baseados parcialmente nesta abordagem.

Bukhari et al. (2010) aplicou um classificador MLP (*Multi-Layer Perceptron*) com características de forma e contexto para classificar regiões de texto e não-texto. O método PAL de Chen et al. (2013) usa um classificador SVM (*Support Vector Machine*) com características extraídas do esqueleto, largura do traçado e cor do texto.

Mais recentemente autores têm proposto segmentadores de layout baseados em redes neurais artificiais, do tipo CNN (*Convolutional Neural Network*) e FCNN (*Fully Convolutional*

[1]https://www.abbyy.com/pt-br/ocr-sdk/





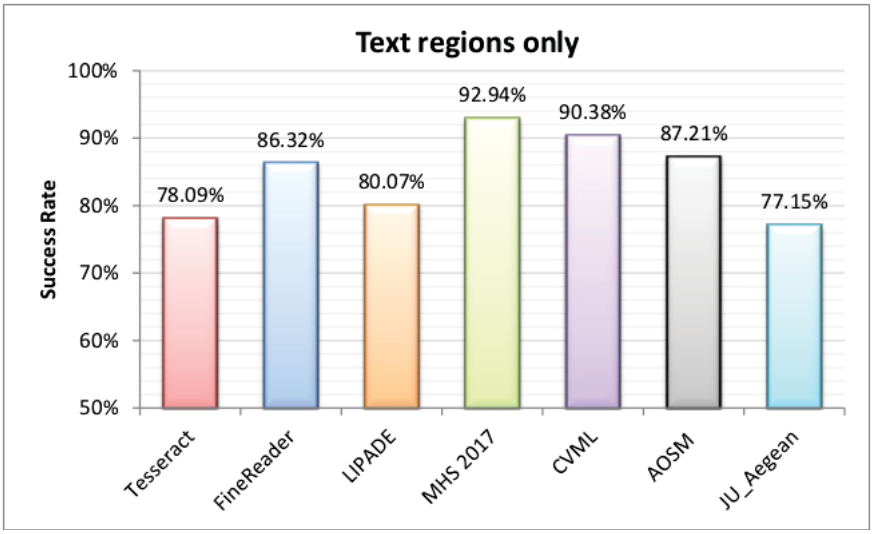


Figura 3.2: Resultado dos métodos submetidos na RDCL2017 nos três perfis de avaliação propostos pela competição.
Fonte: Clausner et al. (2017).

Neural Network). Com base em um conjunto de dados rotulados, essas redes aprendem automaticamente as características dos *pixels* das páginas. Elas têm mostrado um resultado superior em relação aos métodos que realizam a extração de características heuristicamente.

Chen et al. (2017) propuseram uma CNN de três camadas com apenas uma de convolução, que é capaz de inferir rótulos de superpixel (Farabet et al., 2013) em imagens de documentos históricos. A arquitetura mostrou resultados superiores em relação a métodos baseados em SVM (Chen et al., 2016a) e CRF – *Conditional Random Fields* – (Chen et al., 2016b).

Já Xu et al. (2017) e Wick e Puppe (2018) implementaram arquiteturas de FCNN que aprendem diretamente a partir dos *pixels* brutos, ou seja, não são necessárias etapas de pré-processamento como por exemplo a geração de superpixels.

A Tabela 3.1 lista os trabalhos mencionados anteriormente e a abordagem de segmentação utilizada em cada um.

Tabela 3.1: Trabalhos relacionados à análise de layout

Artigo	**Técnica de segmentação**
Breuel (2002)	*Top-down*
Sun (2005)	*Top-down*
Chowdhury et al. (2007)	*Bottom-up*
Ferilli et al. (2010)	*Bottom-up*
Pan et al. (2010)	*Bottom-up*
Smith (2009)	Híbrida
Dai-Ton et al. (2016)	Híbrida
Tran et al. (2017)	Híbrida
Bukhari et al. (2010)	classificador MLP
Chen et al. (2013)	classificador SVM
Chen et al. (2016a)	classificador SVM
Chen et al. (2016b)	CRF
Chen et al. (2017)	CNN
Xu et al. (2017)	FCNN
Wick e Puppe (2018)	FCNN

3.2 OCR EM FRAKTUR

Documentos históricos possuem características que os diferem dos documentos atuais, como por exemplo, o emprego de fontes que já caíram em desuso. Apesar da maioria dos sistemas OCR terem se especializado no reconhecimento óptico de caracteres de fontes atuais, a literatura mostra que os modelos treinados para as fontes consideradas do estilo Gótico, como a Fraktur, também têm atingido bons resultados.

Em 2011 o Arquivo Público de Zurique, na Suiça, digitalizou e realizou o reconhecimento óptico dos caracteres das resoluções de 1887 a 1902 do Governo Cantonal de Zurique escritas em Fraktur. O software comercial ABBYY Recognition Server 3.0[2] conseguiu reconhecer corretamente 99.09% dos caracteres das resoluções (Furrer e Volk, 2011). A Tabela 3.2 apresenta as seis confusões mais cometidas pelo software neste conjunto de dados e a Figura 3.3 mostra exemplos de como alguns caracteres em Fraktur são realmente difíceis de serem distinguidos.

No ambiente *open source* as ferramentas existentes para OCR em Fraktur são o Tesseract e OCRopy (anteriormente denominado OCRopus). Elas possuem modelos pré-treinados para

[2]https://www.abbyy.com

Tabela 3.2: As seis confusões mais cometidas pelo ABBYY Recognition Server 3.0 nas resuluções do Governo Cantonal de Zurique. Fonte: Adaptado de Furrer e Volk (2011).

Ranking	Frequência	{correto}–{reconhecido}
1	49	{u}–{n}
2	38	{n}–{u}
3	14	{a}–{ä}
4	14	{}–{.}
5	13	{d}–{b}
6	11	{s}–{f}

Figura 3.3: Exemplos de caracteres em Fraktur difíceis de serem distinguidos (da esquerda para a direita): *s* (na forma de "*long s*"), *f*, *u*, *n*, *u* ou *n*, *B*, *V*, *R* e *N*.
Fonte: Furrer e Volk (2011).

o reconhecimento de caracteres em Fraktur e também disponibilizam implementações para o treinamento em conjuntos de dados específicos.

Breuel et al. (2013) treinou novos modelos no Tesseract e OCRopy e comparou o desempenho com o ABBYY Recognition Server. Para o treinamento e avaliação foram utilizados dois conjuntos de dados compostos por linhas em Fraktur: (1) linhas do livro *Wanderungen durch die Mark Brandenburg* do escritor alemão Theodor Fontane, digitalizadas em alta resolução e com poucos ruídos; e (2) linhas da enciclopédia *Ersch-Gruber*, digitalizadas em baixa resolução e com ruídos presentes. No conjunto de dados de maior resolução (*Fontane*), os modelos treinados nas ferramentas *open source* se mostraram superiores em relação ao modelo da ferramenta proprietária. Entretanto, no conjunto de dados com ruídos e em baixa resolução (*Ersch-Gruber*) a ferramenta ABBYY obteve a menor taxa de erro, 0.43%. Os resultados estão apresentados na Tabela 3.3.

Tabela 3.3: Taxa de erro de caractere dos sistemas OCR avaliados em Breuel et al. (2013). Fonte: Adaptado de Breuel et al. (2013).

Ferramenta OCR	*Fontane*	*Ersch-Gruber*
OCRopus-LSTM	**0.15**	1.37
Tesseract	0.90	1.47
ABBYY	1.23	**0.43**

Por fim, pode-se concluir que o reconhecimento óptico de caracteres em Fraktur é um problema solucionado de maneira satisfatória pelas ferramentas *open source* existentes. Quando treinados de forma adequada, os modelos conseguem atingir altas taxas de acurácia. Sendo assim, este trabalho não aprofundará esta tarefa do *workflow* de OCR.

3.3 SISTEMAS *OPEN SOURCE* EXISTENTES PARA OCR EM FRAKTUR

As duas subseções seguintes apresentam a evolução, detalhes do funcionamento e características das duas ferramentas *open source* existentes para o OCR em Fraktur: Tesseract (Subseção 3.3.1) e OCRopy (Subseção 3.3.2).

3.3.1 Tesseract

O Tesseract (Smith, 2007) é considerado um sistema OCR *open source* completo: faz a binarização, segmentação e o reconhecimento óptico dos caracteres da imagem.

Iniciou como um projeto de Doutorado (Smith, 1987) desenvolvido nos laboratórios da HP em Bristol, Reino Unido, entre os anos de 1984 e 1994. Neste período houve uma associação entre o HP Lab e a divisão de scanner da HP do Colorado que propiciou ao Tesseract uma vantagem significativa em precisão em relação aos softwares de OCR existentes na época. Apesar deste ganho, na época Tesseract não se tornou um produto.

O próximo estágio de desenvolvimento foi uma investigação do OCR para compressão de documentos e para a melhoria da eficiência da taxa de rejeição.

Em 1995, Tesseract provou seu valor ao atingir bons resultados no Teste Anual de Acurácia de OCR (S.V. Rice, 1995) promovido pelo *Information Science Research Institute* (ISRI) da Universidade de Nevada, Las Vegas (UNLV). Dez anos depois seu código[3] se tornou *open source* e desde 2006 é mantido pelo Google.

Tesseract é uma ferramenta de linha de comando disponibilizada sob a licença *Apache 2.0*[4] para as plataformas Linux, macOS e Windows. Atualmente está na versão 4.1.0. Possui uma API (*Application Programming Interface*) de desenvolvimento na linguagem C/C++ para a versão 3, *wrappers* e ferramentas não-oficiais que auxiliam no treinamento.

As tarefas de OCR no Tesseract (pré-processamento, análise de layout e reconhecimento dos caracteres) são executadas todas juntas através de um único comando:

```
>> tesseract imagename outputbase [-l lang]
[--oem ocrenginemode] [--psm pagesegmode] [configfiles...]
```

Esses processos serão detalhados a seguir.

3.3.1.1 Pré-Processamento

Tesseract faz a análise de layout e, consequentemente, o reconhecimento dos caracteres somente em imagens binarizadas. Documentos coloridos dados como entrada ao sistema serão binarizados pelo próprio Tesseract. O método de limiarização utilizado na binarização é o de Otsu (1979), implementado na biblioteca de processamento de imagens Leptonica[5]. O algoritmo usa um limiar global para a separação dos pixels de fundo e de primeiro plano. Um limiar global geralmente não é adequado para imagens que possuem variação de contraste e isso torna o processo de binarização do Tesseract bastante limitado (Figura 3.4).

Além disso, não há no Tesseract um tratamento para correção da inclinação da página e remoção de ruídos. Apesar de seu analisador de layout e OCR serem bastante dependentes da qualidade desses processos, eles precisam ser tratados com alternativas externas ao Tesseract.

3.3.1.2 Análise de Layout

A análise de layout do Tesseract é baseada nas tabulações do documento (*tab-stops*). Uma *tab-stop* é uma posição na horizontal que limita o início e o fim de uma linha. Os outros elementos não retangulares, como por exemplo as imagens, também são limitados por *tab-stops*.

Tesseract tem um método de análise de layout híbrido, ele combina as abordagens *bottom-up* e *top-down* para segmentar o documento em regiões de texto e não-texto (Figura

[3]https://github.com/tesseract-ocr/

[4]http://www.apache.org/licenses/LICENSE-2.0

[5]http://www.leptonica.com/

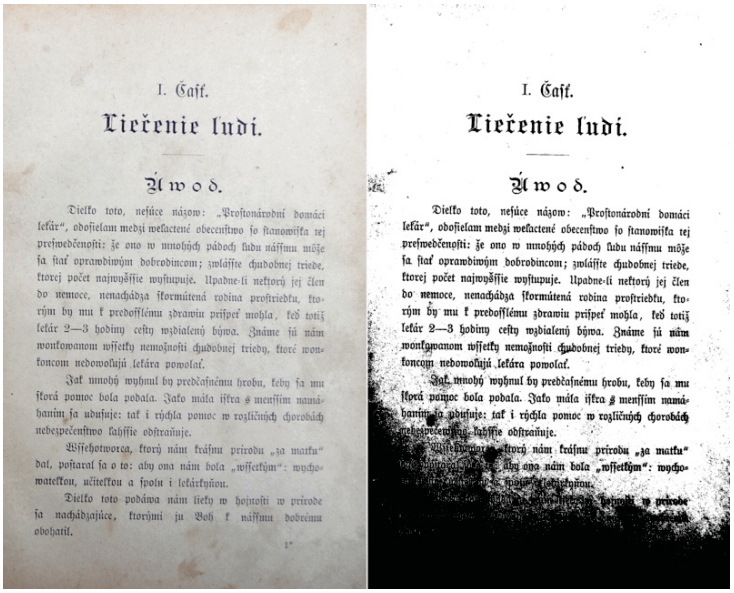


Figura 3.4: Exemplo de imagem com variação de contraste binarizada pelo Tesseract.
Fonte: https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality

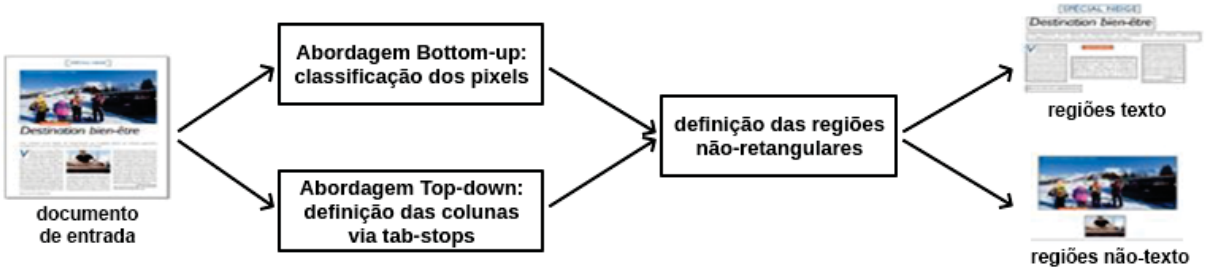


Figura 3.5: A abordagem híbrida do analisador de layout do Tesseract.
Fonte: Adaptado de Soua et al. (2017).

3.5). A abordagem *bottom-up* é usada para rotular os *pixels* em componentes conexos (*CCs*) e localizar as *tab-stops*, enquanto que a análise *top-down* encontra as colunas do layout.

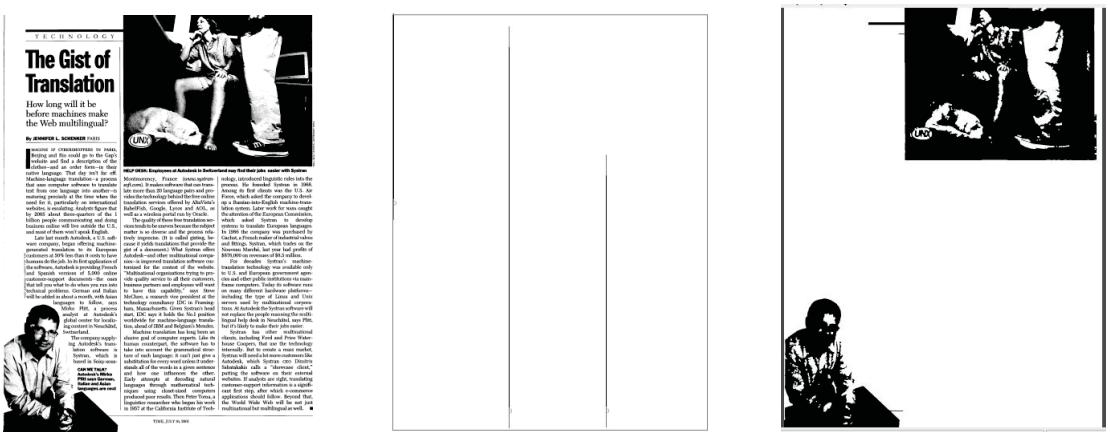
O método começa detectando os separadores de texto e as regiões de imagem através de operações matemáticas e morfológicas da biblioteca Leptonica (Figura 3.6). Esses elementos são retirados e no restante da imagem é feita uma análise de componentes conexos.

Conforme a altura e largura, os *CCs* são classificados em pequenos, médios e grandes. Os componentes pequenos são identificados como ruído ou diacríticos e descartados por um filtro. Um componente grande é classificado como texto se possuir vizinhos à esquerda ou à direta de largura similar de traçado (isso indica que provavelmente ele seja um título). Os elementos grandes de texto e os médios permanecem na imagem (Figura 3.7(a)).

Dentre os elementos de texto restantes são localizados candidatos a *tab-stops*. Uma busca radial é feita em cada *CC* verificando se o mesmo possui vizinhos alinhados e conforme o resultado encontrado o *CC* é classificado como candidato a *tab-stop* direito, esquerdo ou não candidato (Figura 3.7(b)).

Em seguida os candidatos a *tab-stops* são agrupados em linhas. *CCs* adjacentes alinhados verticalmente e que estão entre dois *tab-stops* também são conectados por uma linha. As conexões entre as linhas permitem a identificação e remoção dos falsos positivos.

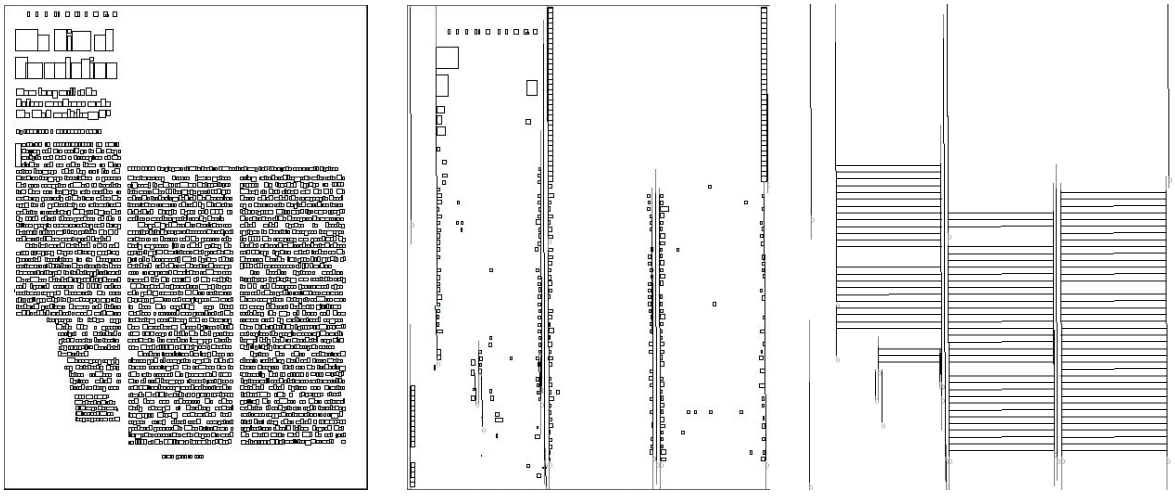
Uma análise da esquerda para a direita e do topo para baixo é feita na página e *CCs* similares são agrupados em partições de colunas (*CPs*) (Figura 3.8(a)) de um modo que



(a) (b) (c)

Figura 3.6: Imagem de entrada (a) e os dois processos inciais da análise de layout do Tesseract: detecção das linhas verticais (b) e elementos de imagem (c).

Fonte: Smith (2009).



(a) (b) (c)

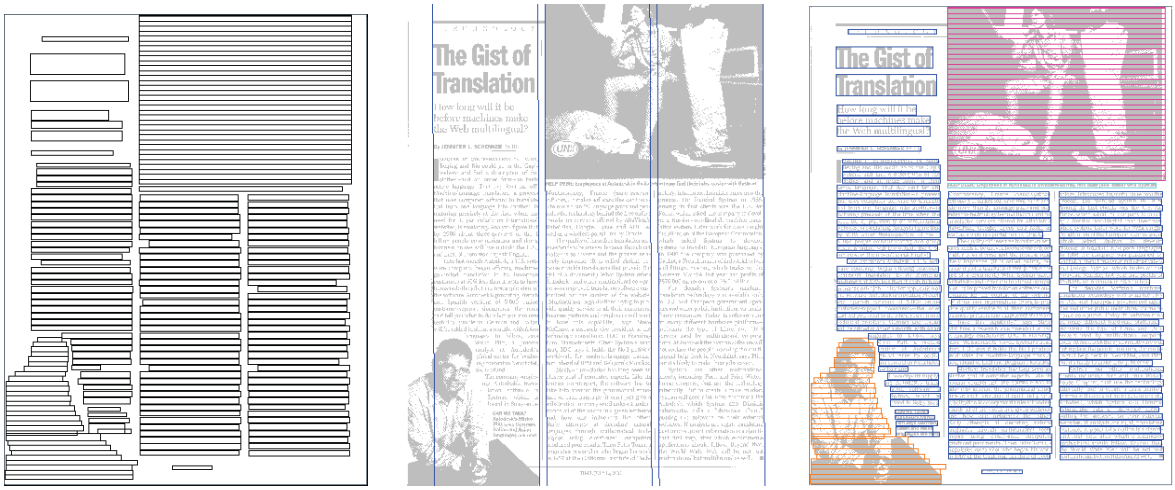
Figura 3.7: *CCs* de texto (a); *CCs* candidatos a *tab-stop* (b); linhas *tab-stops* e linhas de texto (c).
Fonte: Smith (2009).

nenhuma *CP* ultrapasse as linhas *tab-stops*. Cada coleção de *CPs* encontrada horizontalmente é armazenada em um conjunto de partições de coluna (*CPset*). Cada *CPset* é um possível divisor da página em colunas na posição vertical.

Encontrar o layout das colunas (Figura 3.8(b)) é encontrar, dentre os *CPsets* gerados, aqueles que melhor descrevem a página. Para isso, os *CPsets* são combinados e comparados através de critérios definidos pelo algoritmo.

Depois que as colunas são encontradas, cada *CP* é rotulada com um tipo de acordo com a quantidade de colunas que ela cobre. *CPs* inseridas em uma única coluna recebem o tipo "*flowing*", partições localizadas em mais de uma coluna mas que não tocam suas bordas são marcadas como "*pull-out*" e partições que abrangem mais de uma coluna são "*heading*".

A Figura 3.8(c) mostra o resultado da classificação: em **azul** as *CPs* de texto "*flowing*", em **ciano** textos "*heading*", em **magenta** uma imagem "*heading*" e em **laranja** uma imagem "*pull-out*".



(a) (b) (c)

Figura 3.8: (a) *CPs*, (b) colunas e (c) tipos de *CPs*.
Fonte: Smith (2009).

As *CPs* finais são usadas para determinar a ordem de leitura da página que é baseada em algumas regras como por exemplo: blocos "*flowing*" são ordenados conforme suas posições y dentro da coluna, *CPs* que estiverem abaixo de um texto "*heading*" são ordenadas depois dele, e etc.

Por fim, as *CPs* são agrupadas em regiões representadas por polígonos ortogonais (Figura 3.9). Suas bordas são ajustadas para resultarem em um número menor de vértices. As *CPs* devem estar contidas em seu polígono de região e não são permitidas sobreposições entre as regiões. A Figura 3.9 mostra os resultados obtidos em exemplos do conjunto de dados da ICDAR2007.

O analisador de layout, assim como todos os processos do Tesseract, foi projetado inicialmente para receber como entrada documentos em Inglês. Tempos depois, numa tentativa de torná-lo independente de idioma, Smith et al. propuseram algumas alterações no sistema (Smith et al., 2009) e uma delas é em relação ao analisador de layout.

Como pode-se notar, o algoritmo inicial (Smith, 2009) não prevê a ocorrência de textos na vertical. Esse tipo de texto é bastante comum em idiomas não latinos, como o Chinês, Japonês e Coreano, onde muitas vezes aparecem junto com textos na horizontal. Textos na vertical também ocorrem em páginas de revistas e jornais.

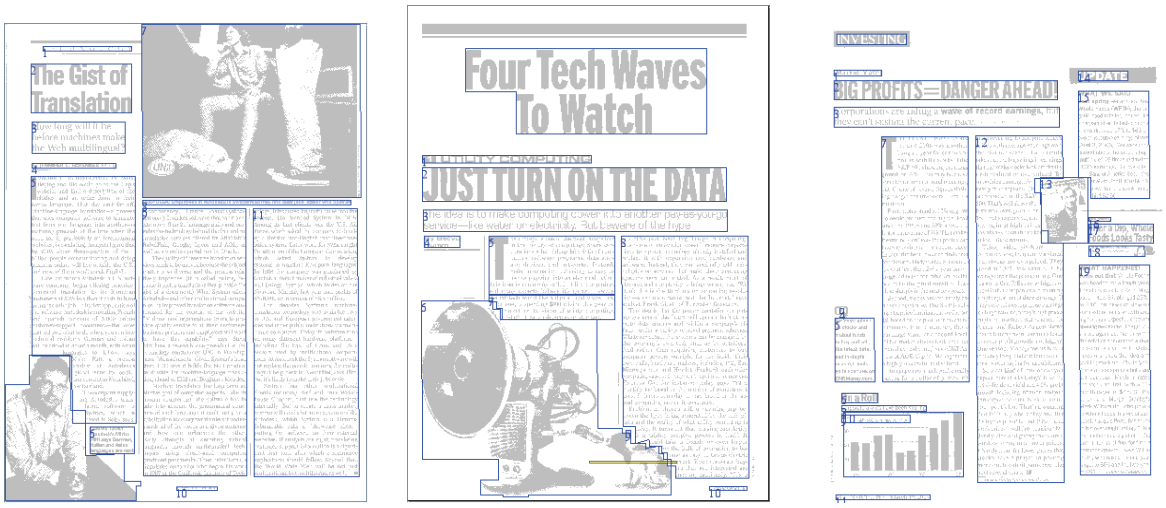


Figura 3.9: Resultado da análise de layout do Tesseract em alguns exemplos do conjunto de dados da ICDAR2007.
Fonte: Smith (2009).

Para corrigir esta questão, inicialmente o algoritmo verifica o alinhamento dos *CCs*. Se a maioria dos *CCs* estiverem alinhados verticalmente, estes serão rotacionados internamente pelo Tesseract em 90 graus no sentido anti-horário para que então seja feita a detecção do layout de colunas. Os textos com alinhamento horizontal permanecem na página (Figura 3.10).

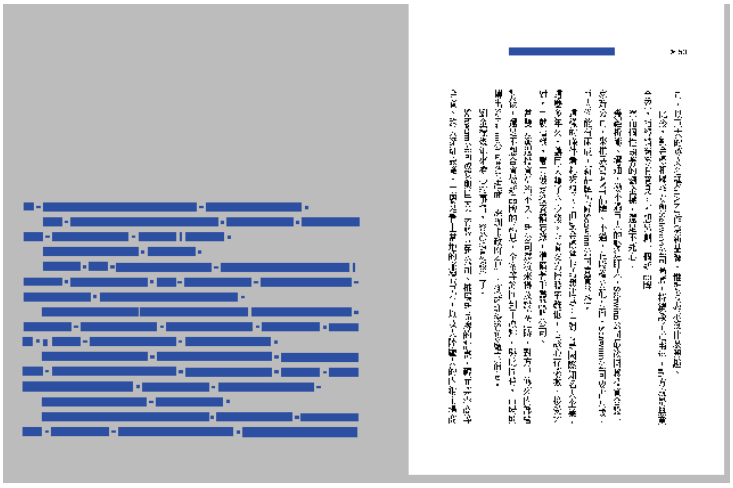


Figura 3.10: Rotação no sentido anti-horário de blocos originalmente verticais em um exemplo de página do idioma Chinês.

Fonte: Smith et al. (2009).

Depois desta verificação de alinhamento e rotação, o algoritmo de análise de layout fica apto para seguir os demais passos normalmente.

Com o passar dos anos, opções de segmentação de página baseadas neste algoritmo foram sendo adicionadas ao Tesseract e atualmente, na versão 4.1.0, estas são as disponíveis:

1. Somente detecção de script e orientação (OSD).
2. Segmentação automática de página com OSD.
3. Segmentação automática de página, mas sem OSD, ou OCR. (não implementado)

4. Segmentação completamente automática de página, mas sem OSD. (padrão)
5. Assume uma única coluna de texto de tamanhos variáveis.
6. Assume um único bloco uniforme de texto alinhado verticalmente.
7. Assume um único bloco uniforme de texto.
8. Trata a imagem como uma única linha de texto.
9. Trata a imagem como uma única palavra.
10. Trata a imagem como uma única palavra em um círculo.
11. Trata a imagem como um único carácter.
12. Texto disperso. Encontra o máximo de texto sem nenhuma ordem específica.
13. Texto disperso com OSD.
14. Linha bruta. Trata a imagem como uma única linha de texto, passando *hacks* que são específicos do Tesseract.

3.3.1.3 Reconhecimento do texto

Com as regiões de texto definidas o próximo passo é reconhecer seus caracteres. Para isso uma nova análise de componentes conexos é feita na região e em seguida são encontrados os *blobs*.

Um *blob* é uma unidade classificável. Ele é composto por um ou mais CCs sobrepostos na horizontal juntamente com seus contornos internos. Normalmente um *blob* corresponde a um caractere.

Depois de definir quais contornos formam os *blobs*, um algoritmo de detecção de linha proposto por Smith (1995) usa essas informações e as linhas de texto são encontradas.

As linhas de texto são então segmentadas em caracteres. Até a versão 3, lançada em 16 de Fevereiro de 2017, Tesseract usava somente uma abordagem de OCR baseada na segmentação dos caracteres (Figura 3.11). Nesta abordagem as linhas são analisadas quanto ao seu espaçamento entre as letras. Linhas com texto de espaçamento fixo (*fixed pitch*) são imediatamente quebradas em caracteres e textos de espaçamento proporcional são quebrados em palavras usando espaços definidos e espaços *fuzzy*.

O processo de reconhecimento é realizado na palavra como um todo. Inicialmente, seus *blobs* são classificados e a palavra é apresentada ao Reconhecedor de Palavras (Figura 3.12). Se o resultado do reconhecimento for considerado insatisfatório, os caracteres de baixa acurácia serão cortados pelo Divisor de Caracteres (*Character Chopper*).

O *Character Chopper* corta os caracteres em determinados pontos buscando melhorar a taxa de acurácia do reconhecimento da palavra. A Figura 3.13 mostra um exemplo de atuação do *Character Chopper*, no qual as setas representam os possíveis pontos de corte. Qualquer ponto de corte que piorar a taxa de acurácia da palavra será desfeito mas não completamente descartado, pois poderá ser reutilizado pelo Associador de Caracteres (*Character Associator*).

Depois das possibilidades de corte se esgotarem e, caso o classificador ainda apresente uma taxa de reconhecimento baixa para a palavra, seus fragmentos de carácter serão então combinados e inseridos num espaço de busca A* (busca do melhor-primeiro ou *best-first*).

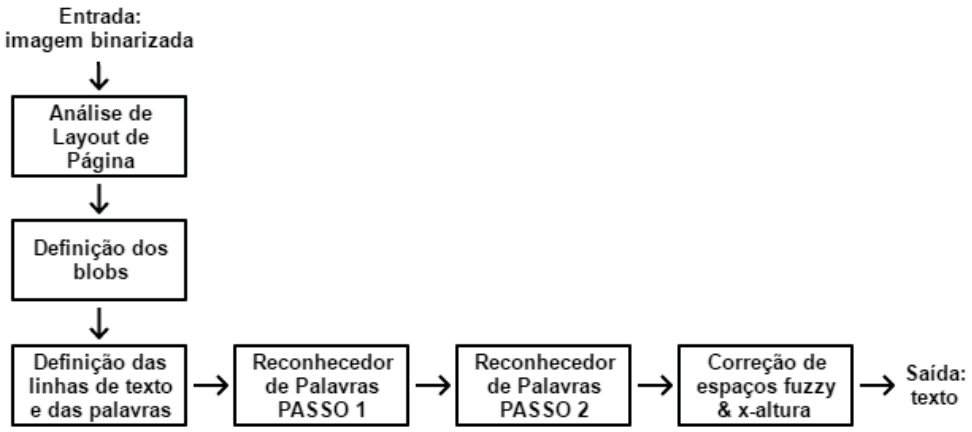


Figura 3.11: Diagrama de blocos do Tesseract 3 (abordagem baseada em segmentação).
Fonte: Adaptado de Smith et al. (2009).

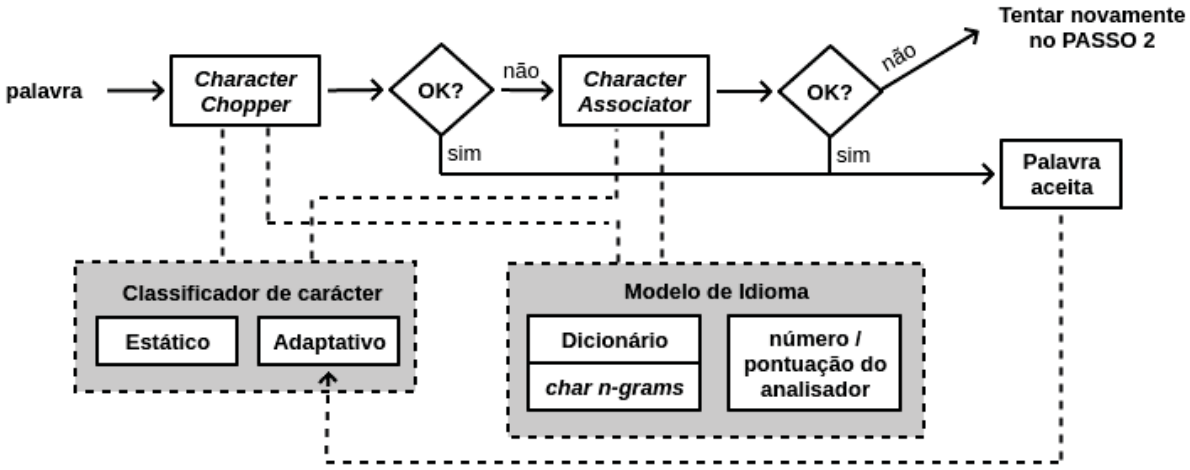


Figura 3.12: Reconhecedor de Palavras do Tesseract 3.
Fonte: Adaptado de Smith (2013).

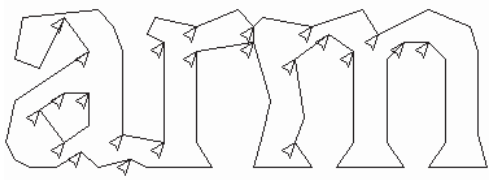


Figura 3.13: Candidatos a pontos de corte.
Fonte: Smith (2007).

Em cada passo da busca, os resultados de classificação dos caracteres são combinados com o modelo de idioma. Este modelo é composto por um dicionário de palavras (definidas pelo sistema e/ou pelo usuário) e por estruturas que auxiliam o classificador.

A cadeia de caracteres que possuir a melhor classificação é a que será considerada como resultado da palavra. Cada palavra aceita é passada como dado de treinamento para o classificador adaptativo.

As palavras do documento são processadas duas vezes. Na primeira vez, as que tiveram um bom reconhecimento são passadas como dado de treinamento para o classificador adaptativo. Assim que o classificador tiver amostras suficientes, ele já pode fornecer resultados mesmo na primeira passagem.

Na segunda vez, as palavras que não atingiram um bom percentual de reconhecimento na primeira vez são processadas e reconhecidas novamente, pois agora o classificador adaptativo possui mais informações.

Na versão 4 do Tesseract, lançada em 29 de Outubro de 2018, foi incluída uma nova opção para OCR: a abordagem sem segmentação (*segmentation-free*) das redes neurais LSTM 2D (*Long Short-Term Memory* de duas dimensões). Ela pode ser utilizada sozinha (Figura 3.14) ou combinada com o resultado do classificador legado (versão 3). Atualmente as opções para OCR no Tesseract são:

0. Somente modo legado.
1. Somente modo redes neurais LSTM.
2. Modo legado + LSTM.
3. Padrão, baseada no que estiver disponível.

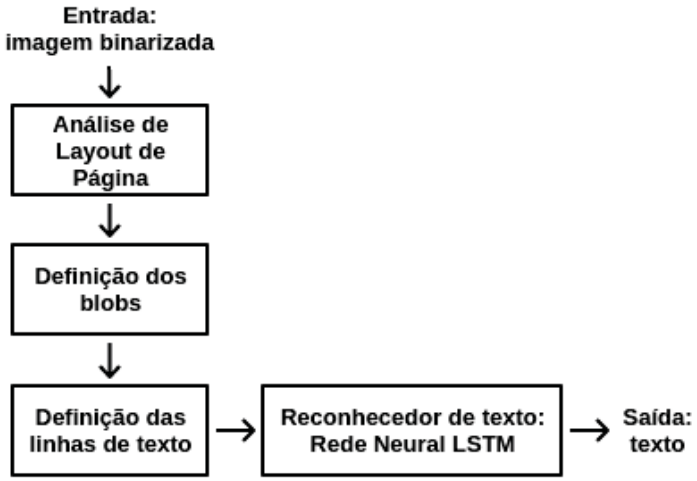


Figura 3.14: Diagrama de blocos do Tesseract 4, opção 1 (abordagem sem segmentação).
Fonte: Autoria própria.

No repositório da ferramenta estão disponíveis modelos treinados em diversos idiomas para a versão 3 e 4.

Atualmente o Tesseract suporta os seguintes arquivos de saída: texto puro, hOCR (Breuel, 2007), PDF, TSV e XML ALTO[6] (experimental).

[6]https://www.loc.gov/standards/alto/

3.3.2 OCRopus / OCRopy

O OCRopus é uma coleção de scripts para análise de documentos e OCR. Não é considerado um sistema OCR, pois seus processos foram desenvolvidos em módulos e podem ser executados separadamente.

Ele surgiu como um projeto de pesquisa patrocinado pelo Google no IUPR *Research Group* (Grupo de Pesquisa em *Image Understanding Pattern Recognition*) sob a liderança do Prof. Thomas Breuel do DFKI (Centro Alemão de Pesquisa em Inteligência Artificial) na Alemanha.

O projeto de pesquisa foi proposto com o objetivo de avançar o estado da arte em reconhecimento óptico de caracteres e em tecnologias relacionadas, buscando um sistema OCR de alta qualidade para conversões de documentos, bibliotecas digitais, usuários com deficiência visual, análise de documentos históricos e uso geral em desktops.

O código do OCRopus se tornou *open source* em 9 de Abril de 2007 e foi disponibilizado sob a licença *Apache 2.0*. A implementação inicial continha o Tesseract 1.x como seu reconhecedor de caracteres e alguns algoritmos de análise de layout.

A primeira versão empacotada do OCRopus (v 0.1.0) foi escrita na linguagem C++ e lançada em 22 de Outubro de 2007 somente para a plataforma Linux. Nesta versão o Tesseract 1.x foi subsituído pelo Tesseract 2.x, funcionalidades de pré-processamento de imagem e um classificador baseado em MLP (*Multi-Layer Perceptron*) foram adicionados.

Do ano de 2007 até início de 2010, OCRopus passou por muitas mudanças (versões 0.x). Novas funcionalidades de pré-processamento foram incluídas, como por exemplo a binarização pelos métodos de Otsu (1979) e, Sauvola e Pietikäinen (2000). Novos algoritmos de análise de layout baseados no diagrama de Voronoi (Kise et al., 1998) e *X-y Cut* (Nagy et al., 1992) foram adicionados. Um reconhecedor de linha foi incluído, ferramentas para treinamento dos dados, suporte a saída hOCR, compatibilidade com Mac OS X e Windows, e etc. Durante este período seu código foi hospedado em *https://github.com/michaelyin/ocropus-git*.

Em Março de 2010 o OCRopus foi totalmente refatorado e transformado em um conjunto de módulos em Python. Foi renomeado para OCRopy e passou a ser hospedado em *https://github.com/tmbdev/ocropy*.

Nas versões 0.7 a 0.8 (de Abril/2013 a Novembro/2014), OCRopy disponibilizava dois tipos de classificadores para o reconhecimento dos caracteres. Um deles, implementado inicialmente na versão 0.6, usava um modelo de Classificadores Locais Logísticos (Yousefi e Breuel, 2012). A outra opção era uma implementação das redes neurais LSTM 1D.

A partir da versão 1.0, de 2 Novembro de 2014, o classificador da v0.6 foi descontinuado e a abordagem LSTM se tornou o reconhecedor padrão de caracteres do OCRopy.

No OCRopy, as tarefas para OCR estão muito bem distribuídas em seus scripts. O *ocropus-nlbin.py* faz o pré-processamento da página, o *ocropus-gpageseg.py* é o responsável pela análise de layout, o *ocropus-rpred.py* reconhece os caracteres das linhas e o *ocropus-hocr.py* gera a saída em formato hOCR. Essas implementações serão detalhadas a seguir.

Além desses scripts existem outros, como o *ocropus-rtrain.py* que é usado para treinar os modelos LSTM em cada idioma e o *ocropus-gtedit.py* que gera uma página HTML que facilita a tarefa de transcrever as linhas. A Figura 3.15 mostra como todos os processo do OCRopy se relacionam.

3.3.2.1 Pré-processamento

O método de binarização implementado no OCRopy é o proposto por Afzal et al. (2013). Ele usa estatísticas dos *pixels* de primeiro e segundo plano, filtros de percentil e um limiar

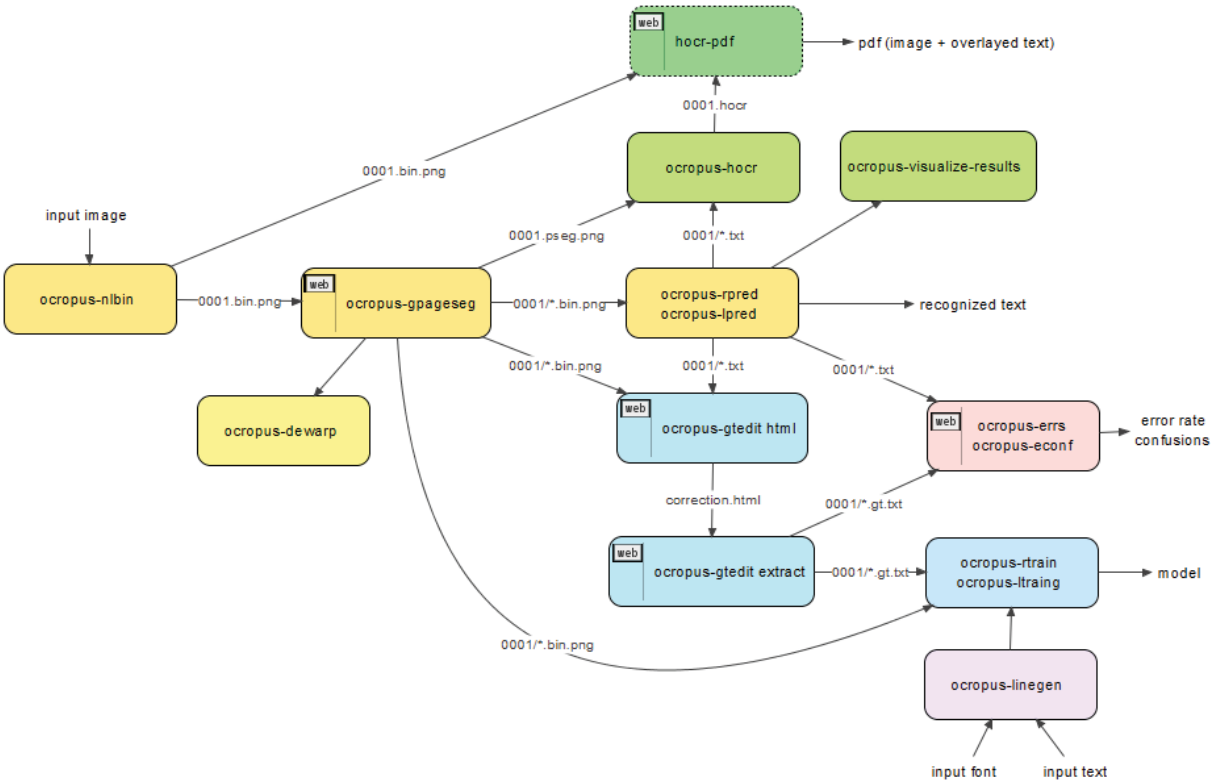


Figura 3.15: Workflow do OCRopy
Fonte: https://rawgit.com/tmbdev/ocropy/master/doc/workflow.html

adaptativo para binarizar a imagem. Segundo o autor, o método proposto gera bons resultados de binarização em imagens com foco e também nas desfocadas.

Nesta etapa de pré-processamento, OCRopy também tenta corrigir, se necessário, a inclinação da página. O documento é rotacionado em 32 ângulos diferentes e aquele que gerar a maior variância entre as somatórias de *pixels* no eixo y é o angulo escolhido. Isso funciona muito bem pois quando o documento está perfeitamente alinhado, as lacunas entre as linhas de texto tornam o valor da variância alto.

A Figura 3.16 mostra o resultado do pré-processamento realizado pelo script *ocropus-nlbin.py* do OCRopy.

3.3.2.2 Segmentação em linhas

O OCRopy não segmenta o documento em regiões de parágrafo, imagem ou separador. Ao invés disso, ele apenas extrai as linhas de texto e cria uma ordenação entre elas.

Inicialmente, é feita uma análise dos componentes conexos. OCRopy tenta então estimar a "escala" do texto. Este valor é considerado como a "altura-x" da sua fonte e é calculado através da mediana das dimensões dos componentes conexos. De acordo com a escala encontrada, o algoritmo remove os componentes muito grandes e muito pequenos que claramente não podem ser letras.

Em seguida é aplicado o filtro da derivada de y com um *kernel* Gaussiano para detectar as bordas superiores e inferiores dos componentes. Essas bordas são então desfocadas horizontalmente e se agrupam em duas regiões. A Figura 3.17 mostra o resultado desta operação, onde as áreas verdes representam os topos de linha, as áreas vermelhas são as partes inferiores de linha e os retângulos em azul correspondem aos componentes conexos. Os *pixels* entre as áreas verde e vermelha são considerados uma linha de texto. É possível observar neste

(a) (b)

Figura 3.16: Imagem original (a) e pré-processamento realizado com o OCRopy (b).
Fonte: Autoria própria.

exemplo que o OCRopy classificou erroneamente algumas partes da gravura do documento como componentes de letra e tentou agrupa-las em linhas. Isso ocorre devido ao processo de binarização ter segmentado a gravura em pequenas partes de tamanho semelhante às letras do texto.

O script *ocropus-gpageseg.py* é o responsável pela segmentação e detecção das linhas mencionadas acima. Para melhorar seu desempenho vários parâmetros podem ser adicionados, dentre eles podemos citar:

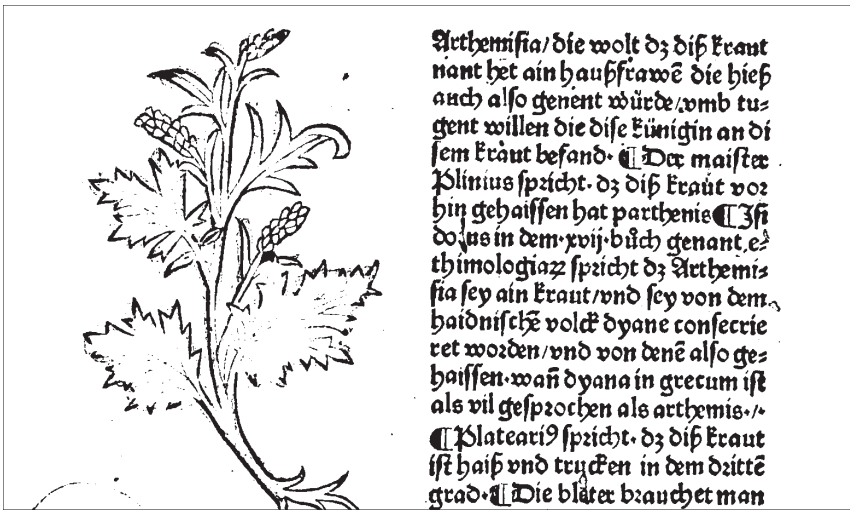
- *maxcolseps*: quantidade máxima de separadores de coluna em branco na imagem. Por exemplo, uma imagem com duas colunas de texto possui um separador em branco entre elas. Já imagens com somente uma coluna de texto não possuem separadores. Se este parâmetro não for ajustado pelo usuário, o número máximo de separadores de coluna em branco considerado pelo OCRopy será 3.

- *maxseps*: quantidade máxima de separadores pretos de coluna. Por padrão, separadores pretos não são verificados pelo algoritmo, mas isso pode ser alterado com a inclusão deste parâmetro.

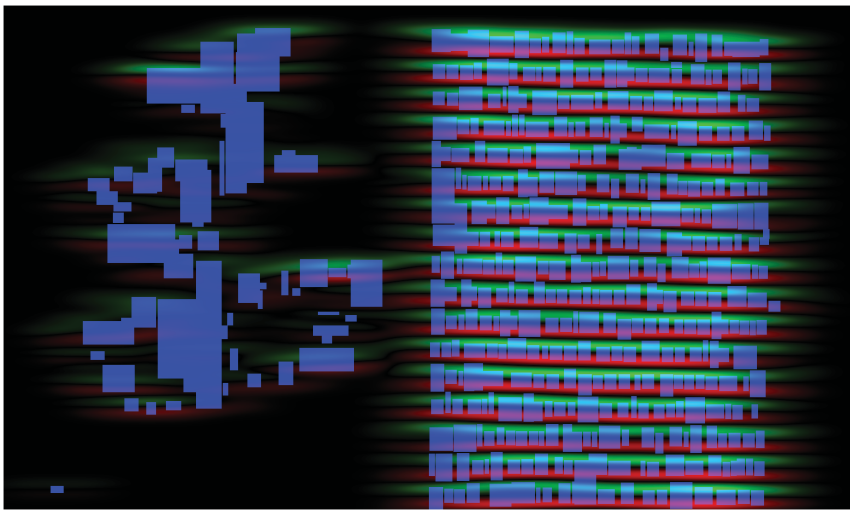
- *scale*: escala da fonte do texto. Quando ajustado, este parâmetro faz com que a estimação automática da escala do texto realizada pelo algoritmo seja descartada.

Com as linhas já segmentadas, OCRopy tenta então estimar uma ordem de leitura entre elas. Para cada par de linhas são determinadas restrições em relação ao seu arranjo geométrico. Segundo Breuel (2003) são usados apenas dois critérios para esta ordenação:

1. Um segmento de linha A vem antes de um segmento de linha B se houver uma sobreposição entre os intervalos de suas coordenadas x; e se o segmento de linha A estiver acima do segmento de linha B na página.



(a)



(b)

Figura 3.17: (a) Pré-processamento com o *ocropus-nlbin.py* e (b) detecção dos componentes de linha.
Fonte: Autoria própria.

2. Um segmento de linha A vem antes de um segmento de linha B se A estiver inteiramente à esquerda de B e se não existir um segmento de linha C cujas coordenadas y estejam entre A e B e as coordenadas x se sobreponham às coordenadas de A e B.

O primeiro critério garante a ordenação dos segmentos de linha pertencentes a uma mesma coluna. Já o segundo critério faz a ordenação das colunas da esquerda para a direita, desde que elas não tenham um cabeçalho em comum. A Figura 3.18 mostra a aplicação desses critérios: no exemplo 1 a linha A vem antes da linha B devido ao critério 1; no exemplo 2 e 3 a linha A está inteiramente à esquerda da linha B e não há nenhuma outra linha entre elas; no exemplo 4 a linha A vem antes da linha B devido ao critério 2, já que entre eles há uma linha C que os separa.

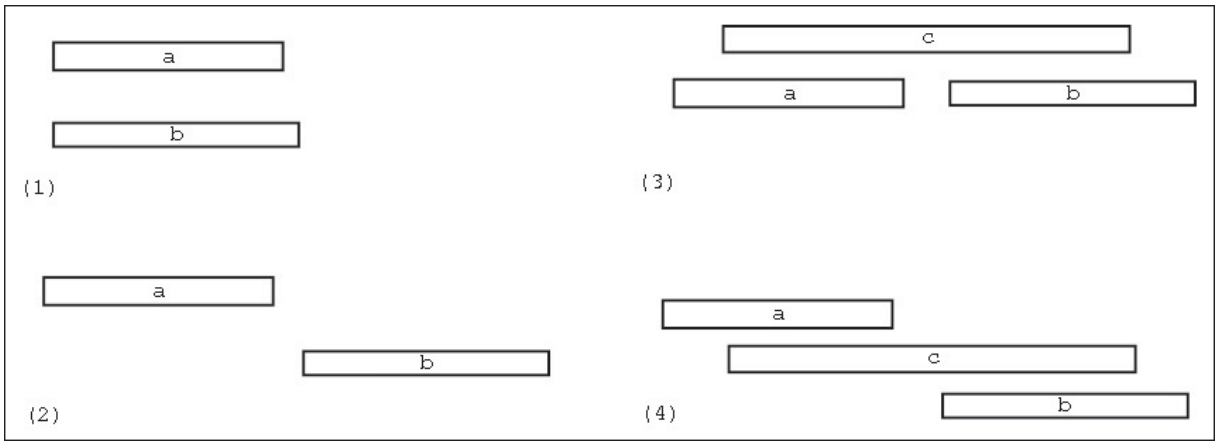


Figura 3.18: Possíveis arranjos geométricos para ordenação dos segmentos de linha no ORCopy.
Fonte: Breuel (2003).

Depois da ordenação finalizada, uma imagem de cada linha do texto é salva em uma estrutura de pastas padronizada pelo OCRopy. A partir deste processo o usuário pode escolher entre reconhecer o texto contido nelas (script *ocropus-rpred*) ou criar uma página HTML para transcrever as linhas e gerar destas uma base de treinamento (script *ocropus-gtedit*).

3.3.2.3 Reconhecimento do texto

Para o reconhecimento do texto das linhas, OCRopy tem uma implementação das redes LSTM bidirecionais de uma dimensão. A rede é alimentada pelas colunas de *pixels* da linha e sua saída são pontuações para cada possível letra. A Figura 3.19 mostra um exemplo deste processo de reconhecimento, na qual a imagem superior é uma possível entrada para a rede e a inferior é a saída produzida. As pequenas linhas coloridas na imagem inferior representam o quão forte uma resposta é: a cor vermelha representa a resposta mais forte e o azul a resposta mais fraca. Neste exemplo a saída sugerida pela rede é: “Auguat S, 1934.” sendo que o correto seria “August 5, 1934.”. Perceba como os segmentos de linha dos caracteres classificados incorretamente pela rede expressam baixa confiança (tendem para a cor azul).

Após todas as linhas serem reconhecidas, elas podem ser agrupadas por página em um arquivo de extensão “.txt” ou convertidas em um arquivo de formato hOCR (Breuel, 2007) com o script *ocropus-hocr.py*.

Figura 3.19: Entrada e saída de uma linha na rede LSTM do OCRopy.
Fonte: https://www.danvk.org/2015/01/09/extracting-text-from-an-image-using-ocropus.html

4 ANÁLISE DE LAYOUT DE PÁGINA EM JORNAIS GERMANO-BRASILEIROS

Como pode ser visto no Capítulo 3, o reconhecimento óptico de caracteres em Fraktur é uma das tarefas do *workflow* de OCR que apresenta bons resultados na literatura. Entretanto, em Sistemas OCR projetados para realizar todo o *pipeline* de OCR, bons resultados assim só são alcançados quando as regiões de texto e não-texto foram bem definidas pela etapa de segmentação e análise de layout.

Na Seção 3.3 apresentamos as abordagens de segmentação e análise de layout das duas ferramentas OCR *open source* existentes para Fraktur: Tesseract e OCRopy. Testes iniciais mostraram que esta tarefa do *workflow* de OCR tem muito que melhorar nas duas ferramentas. Sendo assim, propomos aqui duas novas implementações para segmentação e análise de layout. A primeira delas, descrita na Seção 4.2, é baseada no conceito de extração de regiões homogêneas proposto por Tran et al. (2017). Na segunda abordagem, descrita na Seção 4.3, usamos a arquitetura de CNN proposta por Chen et al. (2017) para a classificação dos *pixels* das páginas e um algoritmo de agrupamento baseado nos conceitos de Maia et al. (2018) para a extração das regiões. Para a avaliação dos resultados criamos o *German-Brazilian Newspaper (GBN) Dataset*. Até o momento, o GBN é o único *dataset* de jornais rotulados tanto para a análise de layout quanto para OCR em Fraktur presente na literatura. Ele será detalhado na Seção 4.1.

4.1 GBN 1.0 DATASET

O *German-Brazilian Newspaper (GBN) Dataset*[1] é composto por 152 imagens de páginas coloridas de oito jornais diferentes procedentes da coleção *dbp digital* da iniciativa *Dokumente.br*. As páginas foram escolhidas de maneira a representar todo o corpus de jornais da coleção. A Figura 4.1 mostra exemplos desses jornais e seu respectivo *ground truth* de classificação a nível de *pixel*.

"*Der Gemeindebote*" possui um layout simples com o texto dividindo em duas colunas e escrito em uma única fonte, com variação pequena no tamanho da fonte. É um jornal sem componentes de imagem.

"*Der Jugendfreund*" passou por três mudanças de layout consideráveis durante a sua existência. As duas primeiras foram mudanças bem simples, já na terceira o título se tornou mais rebuscado artisticamente. Possui poucos elementos gráficos e nenhuma imagem.

"*Der Pioner*" tem um dos maiores corpus, totalizando 1.295 páginas. Apresenta um layout complexo com diversos estilos e tamanhos de fonte. Tem elementos gráficos bastante diversos entre si.

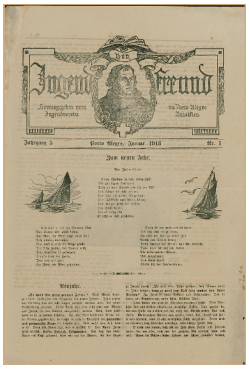
"*Der Landwirt*" é o jornal com a maior quantidade de elementos de imagem. A maioria de suas páginas possuem pelo menos uma imagem ou elemento gráfico ocupando metade da página. Seu título foi escrito em letra cursiva.

O "*Evangelisch-Lutherisches Kirchenblatt*" possui um layout com duas colunas e sem separadores entre elas. Apresenta imagens e elementos gráficos. Assim como o "*Der Jugendfreund*", também possui um título bastante rebuscado.

O maior corpus é do "*Kolonie Zeitung*". Seu layout é bem irregular, com grande variabilidade no número de colunas e na intercalação entre o texto e os elementos gráficos no decorrer da página.

[1] O GBN 1.0 Dataset está parcialmente disponível em `https://web.inf.ufpr.br/vri/databases/gbn/`

(a) *Der Gemeindebote*

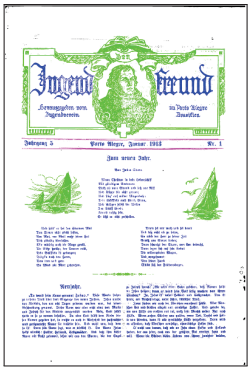


(b) *Der Jugendfreund*

(c) *Der Pioner*

(d) *Der Landwirt*

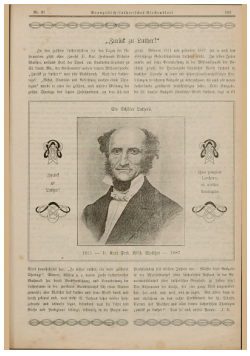
(e)



(f)

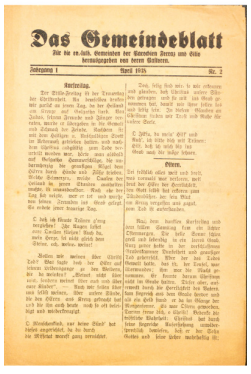
(g)

(h)

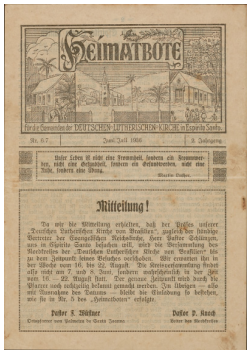


(i) *Ev.-Luth. Kirch.*

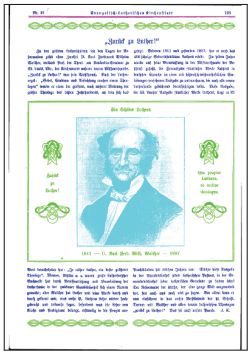
(j) *Kolonie Zeitung*



(k) *Gemeindeblatt*

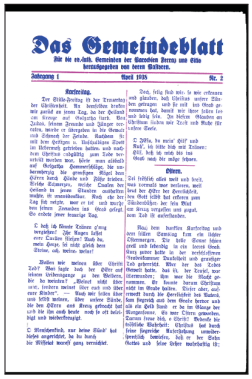


(l) *Heimatbote*

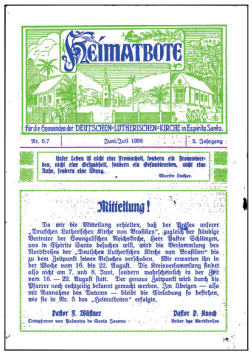


(m)

(n)



(o)



(p)

Figura 4.1: Exemplos de páginas dos jornais do *GBN 1.0 Dataset* (a–d, i–l) com seu *ground truth* de classificação a nível de *pixel* (e–h, m–p) onde a classe **text** é representada em **azul**, **image** em **ciano**, **graphic** em **verde**, **separator** em **magenta**, a cor preta é usada para os *pixels* que não são de interesse e a cor branca para os *pixels* de fundo.

Páginas de dois jornais: "*Gemeindeblatt*", com apenas duas edições de quatro páginas em um layout simples, com duas colunas; e "*Heimatbote*", com 14 páginas escritas em papel levemente transparente, não estão disponíveis no conjunto de treinamento. O objetivo é verificar como os classificadores se comportam em relação a títulos de jornais desconhecidos pelo treinamento. As Tabelas 4.1 e 4.2 fornecem informações detalhadas sobre a composição e a distribuição de cada título de jornal do *GBN 1.0 Dataset*.

Das 152 páginas do GBN 1.0, 102 compõem o conjunto de treinamento e 50 compõem o conjunto de teste (Tabela 4.2). As páginas foram digitalizadas em 600 dpi e estão disponíveis no formato PNG. O *ground truth* das páginas é composto por arquivos no formato XML PAGE (Pletschacher e Antonacopoulos, 2010) e arquivos no formato de texto puro. Os arquivos XML PAGE foram produzidos com o software Aletheia (Clausner et al., 2011a) e contêm a representação geométrica, a classificação de cada região (**text**, **image**, **graphic** ou **separator**) e a transcrição das regiões de texto.

Nos arquivos de texto são fornecidas as coordenadas de cada *pixel* de primeiro plano juntamente com sua classe correspondente (Tabela 4.3), ou seja, cada linha do arquivo de texto contém a coordenada de um *pixel de interesse* e sua respectiva classe.

Tabela 4.1: Composição do dataset GBN 1.0.

Título do jornal	**Data de publicação**	**Fonte**	**Idioma**
"*Der Gemeindebote*"	1935-1938	Fraktur	Alemão
"*Der Jugendfreund*"	1911-1917	Fraktur	Alemão
"*Der Pioner*"	1888-1891	Fraktur / Latin	Português / Alemão
"*Der Landwirt*"	1934-1940	Fraktur	Alemão
"*Ev.-Luth. Kirchenblatt*"	1916-1919	Fraktur	Alemão
"*Kolonie Zeitung*"	1863-1889	Fraktur / Latin	Português / Alemão
"*Gemeindeblatt*"	1938	Fraktur	Alemão
"*Heimatbote*"	1936	Fraktur	Alemão

Tabela 4.2: Distribuição das imagens do dataset GBN 1.0.

Título do jornal	**Tamanho da imagem (em *pixels*)**	**Páginas de Treinamento**	**Páginas de Teste**
"*Der Gemeindebote*"	3850 x 5480	19	05
"*Der Jugendfreund*"	de 2650 x 3950 a 5320 x 8050	15	03
"*Der Pioner*"	7100 x 10590	17	05
"*Der Landwirt*"	4250 x 6020	17	05
"*Ev.-Luth. Kirchenblatt*"	2590 x 3690	17	05
"*Kolonie Zeitung*"	5470 x 7010, 6700 x 8400, e 7050 x 9300	17	05
"*Gemeindeblatt*"	3850 x 5870	-	08
"*Heimatbote*"	3850 x 5480	-	14

Tabela 4.3: Rótulos das classes nos arquivos de *ground truth* de formato texto.

Rótulo da classe	Descrição
0	**text**: texto em qualquer tamanho de fonte.
1	**image**: fotos.
2	**graphic**: brasões de família e outros elementos gráficos (com ou sem texto em seu interior).
3	**separator**: separadores de coluna e linhas.

4.2 MÉTODO BASEADO NO "MHS 2017 SYSTEM" (GBN-MHS)

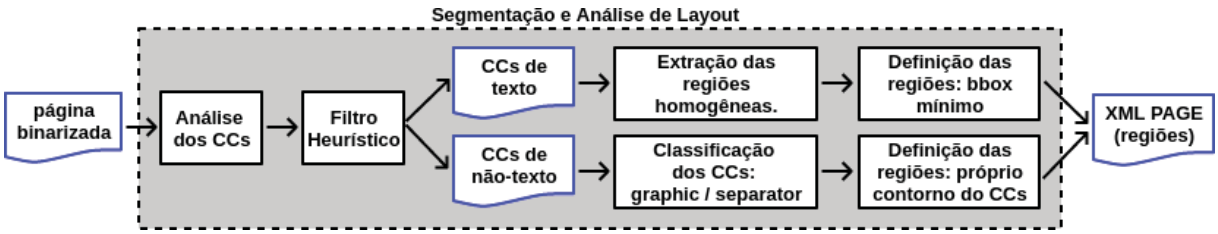


Figura 4.2: Fluxograma da análise de layout baseada no MHS 2017 System.
Fonte: Autoria própria.

Criamos o método GBN-MHS com conceitos propostos no "MHS 2017 System" de Tran et al. (2017). A Figura 4.2 mostra o fluxograma da nossa abordagem. A entrada do algoritmo é uma página de jornal já binarizada. Em seguida, uma análise dos componentes conexos é realizada com o objetivo de extrair características geométricas que serão utilizadas por um filtro heurístico. O filtro heurístico classifica os componentes conexos em texto e não-texto. Os elementos de texto são então agrupados em regiões homogêneas e os não-texto são classificados em **graphic** e **separator**. Por fim, os contornos das regiões resultantes são refinados e armazenados em um arquivo no formato XML PAGE (Pletschacher e Antonacopoulos, 2010) juntamente com a sua classificação.

Sendo CC_i o i-ésimo componente do conjunto dos componentes conexos CCs, e $B(CC_i)$ o *bounding box* de CC_i onde (Xl_i, Yl_i) corresponde a coordenada superior esquerda e (Xr_i, Yr_i) corresponde a coordenada inferior direita; Tran et al. (2017) propõem a extração das seguintes características:

(i) $\mathcal{A}_i$: a quantidade de *pixels* do CC_i, ou em outras palavras, a área do CC_i.

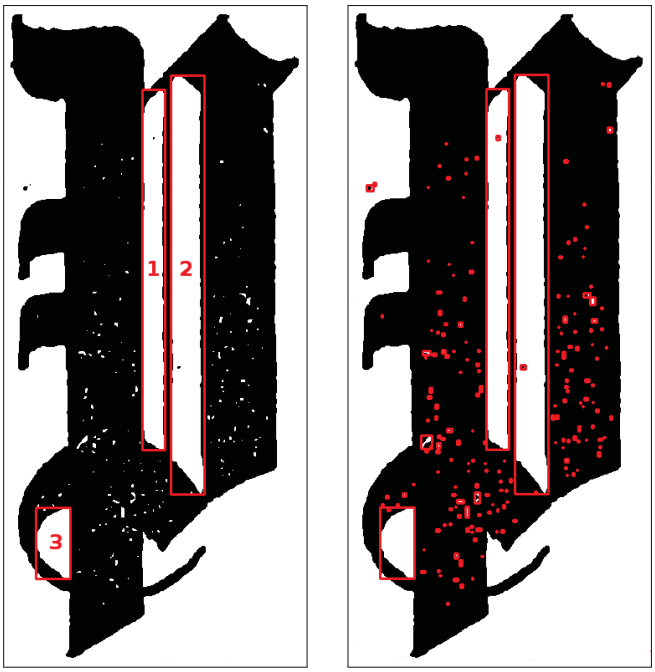
(ii) λ_i: a densidade do CC_i.
$\lambda_i = \mathcal{A}_i / \mathcal{A}_i^B$; $0 < \lambda_i \leq 1$. Sendo que $\mathcal{A}_i^B$ corresponde a área do *bounding box* do CC_i.

(iii) Inc_i: o número de *bounding boxes* $B(CC_j)$ contidos em $B(CC_i)$, sendo que $j \neq i$.

(iv) HW_i^{rate}: razão entre a largura e altura do CC_i.
$HW_i^{rate} = \min(W_i, H_i)/\max(W_i, H_i)$; $0 < HW_i^{\text{rate}} \leq 1$. Sendo que W_i e H_i correspondem respectivamente a largura e altura do *bounding box* do CC_i.

A densidade do CC_i – característica (ii) – não foi calculada na nossa implementação pois ela não é utilizada no filtro heurístico proposto por Tran et al. (2017), é utilizada somente em processo sucessores do "MHS 2017 System".

A característica (iii) funciona muito bem para caracteres não degradados. Entretanto, no *GBN Dataset* temos muitos caracteres com ruídos de degradação. Esses ruídos interferem na contagem dos *bounding boxes* internos do CC_i. Como observamos no exemplo da Figura 4.3, o retorno esperado da contagem de *bounding boxes* internos seria 3, mas devido às degradações o resultado é muito maior (Figura 4.3(b)). Para corrigir essa característica, só consideramos os $B(CC_j)$ cuja a área seja maior que 5% da área total do $B(CC_i)$.



(a) Resultado esperado da característica (iii): $B(CC_j) = 3$.

(b) Resultado obtido: $B(CC_j) = 181$.

Figura 4.3: Ajuste na definição da característica (iii) devido à presença de ruídos nos caracteres.
Fonte: Autoria própria.

Nesta etapa também removemos todos os CC_i cujo contorno toca um dos cantos da página (coordenadas [0,0], [0,largura], [0,altura] ou [largura,altura]), pois são considerados ruídos de borda.

Após a extração das características é aplicado um filtro heurístico. O filtro heurístico tem o objetivo de eliminar componentes que claramente não podem ser do tipo texto. Pelas definições de Tran et al. (2017), um CC_i é considerado um componente não-texto se satisfizer a pelo menos uma das seguintes condições:

(i) $\mathcal{A}_i < T^{\text{area}}$, $T^{\text{area}} = 6$ pixels: os componentes com áreas muito pequenas serão removidos.

(ii) $Inc_i > T^{\text{inside}}$, $T^{\text{inside}} = 4$: se um $B(CC_i)$ contém em seu interior mais de 4 componentes, ele será classificado como um elemento não-texto.

(iii) $HW_i^{\text{rate}} < T^{\text{rate}}$, $T^{\text{rate}} = 0.06$: a largura de um componente texto não pode ser 16.66 vezes maior que sua altura.

Esses limiares foram definidos com base nos conjuntos de dados utilizados por Tran et al. (2017). Para o *GBN Dataset* foi necessário o ajuste desses parâmetros.

Cada componente não-texto detectado é removido da imagem. Em seguida, pelo algoritmo original, são aplicadas na imagem as classificações *Multi-Level* (Tran et al., 2016)

e *Multi-Layer* (Tran et al., 2017). As duas classificações trabalham com um conceito de homogeneidade de regiões de texto e usam um filtro recursivo para extrair o restante dos componentes não-texto da imagem. Nós optamos por extrair as regiões de texto da mesma maneira que Tran et al. (2017), ou seja, calculando as regiões homogêneas, entretanto, após esse processo não utilizamos o filtro recursivo.

Uma região $\Re$ de tamanho $a \times b$, será considerada uma região homogênea se possuir uma determinada estrutura de homogeneidade. Caso contrário ela será considerada uma região heterogênea e precisará ser segmentada em regiões menores homogêneas $\Re_i, i = \overline{1,n}$.

$$\Re = \Re_1 \cup \Re_2 \cup \cdots \cup \Re_n \tag{4.1}$$

A homogeneidade de uma região de texto é verificada da seguinte forma:

1. O histograma da projeção horizontal P_H da região $\Re$ é extraído (Figura 4.4):

$$P_H = \left\{ p_x \mid p_x = \sum_{y=1}^{b} \Re(x,y), 1 \leq x \leq a \right\} \tag{4.2}$$

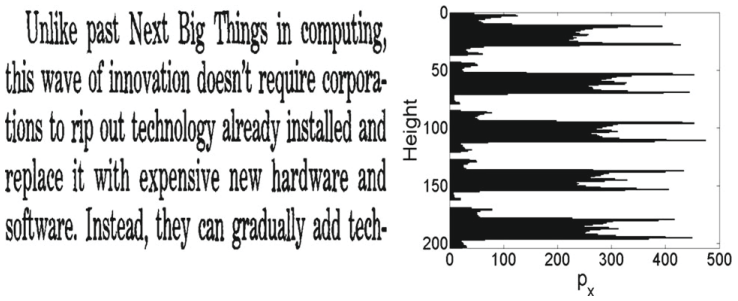


Figura 4.4: Documento original e sua projeção horizontal.
Fonte: Tran et al. (2016).

2. O histograma é suavizado (Z_H) através da Equação 4.3, onde s é o tamanho do *kernel* (Figura 4.5(a)):

$$Z_H = \left\{ z_x \mid z_x = \left\lfloor \frac{1}{2s} \sum_{-s}^{s} p_x \right\rfloor, 1 \leq x \leq a, p_x \in P_H \right\} \tag{4.3}$$

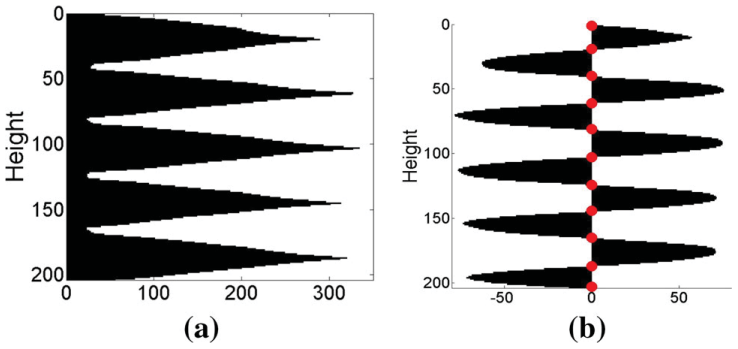


Figura 4.5: (a) Projeção suavizada Z_H e (b) derivada da projeção suavizada G_H.
Fonte: Tran et al. (2016).

3. É calculada a primeira derivada G_H (gradiente) da projeção suavizada Z_H (Figura 4.5(b)):

$$G_H = \left\{ g_x \mid g_x = \frac{\partial z_x}{\partial y}, 1 \leq x \leq a, z_x \in Z_H \right\} \tag{4.4}$$

4. São encontrados os extremos locais L_H (máximos e mínimos locais) de G_H, ou seja, onde sua derivada é igual a 0 (pontos vermelhos da Figura 4.5(b)):

$$L_H = \{t \mid (g_t < 0 \wedge g_{t+1} \geq 0) \vee (g_t > 0 \wedge g_{t+1} \leq 0), g_t \in G_H\} \tag{4.5}$$

5. Sendo Δ o conjunto das distâncias entre dois extremos locais adjacentes:

$$\Delta = \{d_i \mid d_i = t_{i+1} - t_i, t_i \in L_H\} \tag{4.6}$$

   A variância V de Δ é calculada. Este valor indica a distribuição da região considerada, onde μ é a média e n é o número de elementos de Δ:

$$V = \frac{\sum (d_i - \mu)^2}{n}, d_i \in \Delta \tag{4.7}$$

Se a variância for pequena, $V \leq T_{\text{Var}}$ (Tran et al. (2017) sugerem $T_{\text{Var}} = 1.2$) a região é homogênea como na Figura 4.6(a), caso contrário ela é heterogênea (Figura 4.6(b)) e precisará ser segmentada.

Figura 4.6: Exemplo de região homogênea (a) e heterogênea (b).
Fonte: (Tran et al., 2016).

A segmentação proposta por Tran et al. (2017) é feita da mesma maneira que no algoritmo *X-y Cut* de Nagy et al. (1992). Ela é baseada na altura das linhas de texto da região (*black lines*) e em seus espaçamentos entre linhas (*white lines*).

Sendo w_i, b_i a i-ésima linha branca e preta respectivamente; w e b o conjunto de w_i e b_i, a região é segmentada se:

1. Uma de suas linhas brancas possuir altura maior do que a altura das demais linhas da região (Figura 4.7(a)):

$$(w_i > \text{média}(w)) \wedge (w_i = \max(w)) \tag{4.8}$$

2. Ou se uma de suas linhas pretas possuir altura maior do que a altura das demais linhas da região (Figura 4.7(b)):

$$(b_i > \text{média}(b)) \wedge (b_i = \max(b)) \tag{4.9}$$

Figura 4.7: Exemplo de *white division* em w_3 (a) e *black division* em w_1 e w_2 (b).
Fonte: Tran et al. (2016).

As regiões resultantes deste processo são re-examinadas em relação à sua homogeneidade e re-segmentadas caso necessário. São realizados esses mesmos passos verticalmente para estimar e segmentar, se necessário, as regiões na direção vertical. No algoritmo de Tran et al. (2017), uma região é considerada homogênea se possuir homogeneidade em ambas as direções.

Na implementação desta etapa do algoritmo, uma dificuldade que tivemos foi em relação ao parâmetro s da Equação 4.3. Ele é um parâmetro essencial para a correta extração das regiões homogêneas, entretanto Tran et al. (2017) não explicam como ele deve ser definido. Concluímos que seu valor deve ser suficiente para suavizar os picos e vales da projeção horizontal das linhas de texto. No entanto, na projeção vertical não chegamos a uma conclusão sobre como se deve comportar o parâmetro s, decidimos então considerar se uma é região homogênea somente com base na variância dos pontos de mínimo e máximo da projeção horizontal.

Outra situação que não fica clara no algoritmo de Tran et al. (2017) é a ordem de segmentação das regiões heterogêneas caso estas possuam uma linha branca e também uma linha preta que satisfaçam as condições das Equações 4.8 e 4.9. Decidimos implementar da seguinte maneira: consideramos como *white lines* as linhas com texto e *black lines* os espaçamentos entre as linhas de texto. Em seguida calculamos a altura da maior *black line* na projeção vertical e na projeção horizontal da região. Uma *black division* é feita no sentido (horizontal ou vertical) que tiver a *black line* maior. Sendo assim, nossa implementação sempre segmentará uma região pelos seus maiores espaçamentos entre linhas (*black lines*).

Caso as *black lines* da direção horizontal e vertical tenham o mesmo tamanho, são verificadas então as *white lines*. Da mesma maneira, o sentido que possuir a maior das *white lines* é a sentido em que ocorrerá a *white division*.

Depois de todas as regiões de texto se tornarem homogêneas, Tran et al. (2017) aplicam um filtro recursivo para extrair seus CCs não-texto restantes, que não foi implementado neste trabalho.

Para finalizar classificamos todos os CCs não-texto encontrados pelo filtro heurístico em CCs de **graphic** ou **separator**. Não incluímos a classe **image** porque não conseguimos encontrar características representativas que a diferenciasse da classe **graphic**. Como no *GBN Dataset* existem mais componentes **graphic** do que **image**, optamos por classificar todos como **graphic**.

Identificamos os CC_i da classe **separator** da mesma maneira que Tran et al. (2017): se seu HW_i^{rate} for menor que um determinado limiar ele é considerado da classe **separator**, caso contrário ele é automaticamente classificado como um CC_i **graphic**.

A saída do método é um arquivo no formato XML PAGE com as coordenadas das regiões e sua classificação. Consideramos como coordenadas das regiões não-texto as próprias coordenadas do seu CC_i, ou seja, nesta nossa implementação cada região não-texto contém somente um CC_i. As coordenadas das regiões de texto são as mesmas quatro coordenadas

encontradas pelas *black division* e *white division*. Para a geração dos arquivos XML PAGE utilizamos uma biblioteca[2] em C++ disponibilizada pelo PRImA Lab.

Regiões sobrepostas não são permitidas no formato XML PAGE, somente regiões aninhadas (regiões totalmente contidas em outras). Entretanto não tivemos tempo hábil para tratar esses casos.

4.3 MÉTODO BASEADO EM *DEEP LEARNING* (GBN-DL)

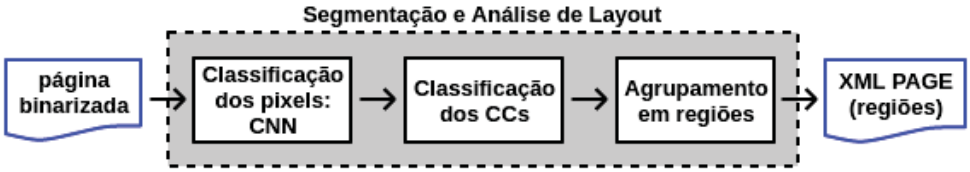
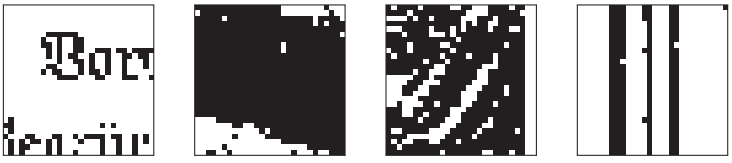


Figura 4.8: Fluxograma da abordagem para análise de layout baseada em *deep learning*.
Fonte: Autoria própria.

A Figura 4.8 mostra o fluxograma da nossa implementação para análise de layout baseada em *deep learning*. O algoritmo recebe como entrada uma página de jornal binarizada. Seus *pixels* são então classificados por uma CNN. Em seguida os componentes conexos são classificados de acordo com a classe de maior ocorrência entre seus *pixels*. Um agrupamento dos componentes conexos em regiões baseado nos conceitos de Maia et al. (2018) é realizado. As regiões resultantes e sua classificação são agrupadas em um arquivo de saída no formato XML PAGE.

A classificação dos *pixels* é realizada por uma arquitetura de CNN proposta por Chen et al. (2017). A entrada da rede consiste em pedaços (*patches*) de imagem das páginas dos jornais do conjunto de treinamento. Os *patches* devem medir 28×28 *pixels* e receber como rótulo a classe do seu *pixel* central. Exemplos de *patches* podem ser observados na Figura 4.9.



(a) **text** (b) **image** (c) **graphic** (d) **separator**

Figura 4.9: Exemplos de *patches* de entrada para a CNN e sua classe correspondente.
Fonte: Autoria própria.

A Figura 4.10 ilustra a arquitetura da rede (Chen et al., 2017). Sua estrutura pode ser resumida como $28 \times 28 \times 1 - 26 \times 26 \times 4 - 100 - 4$. A CNN contém apenas uma camada de convolução que, com um *kernel* de 3×3, gera 4 filtros (*feature maps*). Não há camadas de agregações (*pooling*) propostas nesta arquitetura, então logo após a camada de convolução já há uma camada totalmente conectada (*fully connected*) com 100 neurônios. A última camada é uma *softmax* com regressão logística que, com base nas probabilidades encontradas por classe, classifica o *patch*. A função de ativação usada nos neurônios da camada convolucional e na camada totalmente conectada é a ReLU (Ponti et al., 2017).

Após a classificação dos *pixels* realizada pela CNN, identificamos então todos os componentes conexos (CC) da página e atribuímos a cada CC a classe de maior ocorrência

[2]https://www.primaresearch.org/tools/PAGELibraries

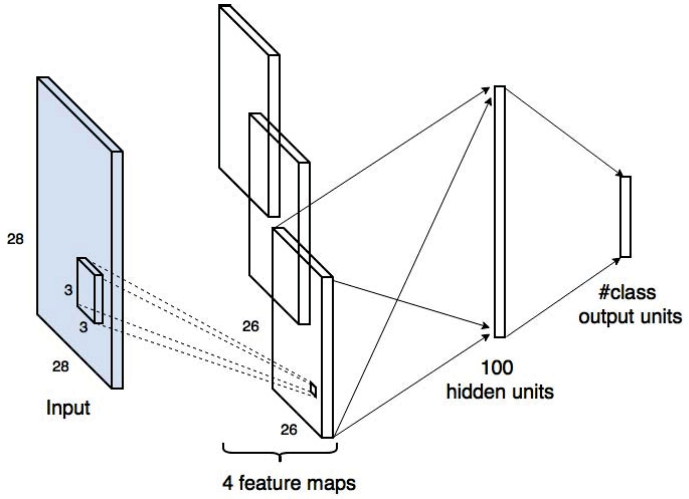


Figura 4.10: Arquitetura da CNN usada para a classificação dos *pixels*/*patches*.
Fonte: Chen et al. (2017)

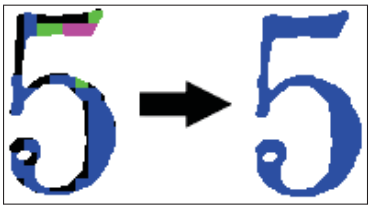


Figura 4.11: Exemplo da classificação final de um CC no tipo **text**.
Fonte: Autoria própria.

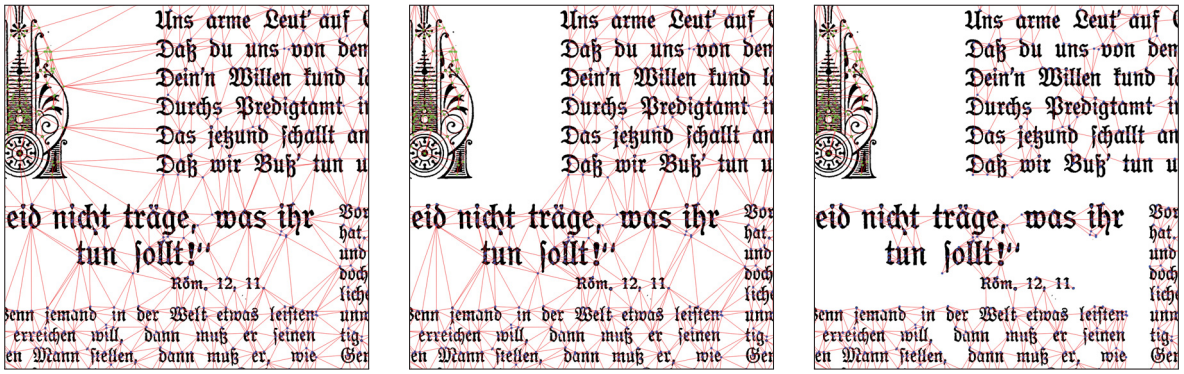
entre seus *pixels*. A Figura 4.11 mostra um exemplo deste processo no qual a classe de maior ocorrência é a **text**, representada pela cor **azul**.

Após a classificação dos CCs, eles são então agrupados em regiões. Uma região deve conter apenas CCs do mesmo tipo e o ideal é que textos em colunas diferentes pertençam a regiões diferentes.

Para o agrupamento dos CCs usamos uma abordagem semelhante à de Maia et al. (2018). Neste trabalho os autores propõem uma representação baseada em grafo para o problema de agrupamento dos CCs. Eles são considerados como nós de um grafo planar. O grafo é criado da seguinte maneira: com base nos nós, o diagrama de Voronoi é calculado e cada par de regiões adjacentes criadas pelo diagrama tem seus nós conectados por uma aresta. As arestas que conectam CCs que devem pertencer a regiões diferentes são removidas por uma CNN que leva em consideração características geométricas dos CCs e também características extraídas pela própria rede.

Na nossa implementação, o problema do agrupamento dos CCs também é representado na forma de um grafo planar (Figura 4.12(a)). Entretanto, ao invés de usarmos características geométricas ou um classificador com *deep learning*, propomos as seguintes regras heurísticas para a remoção das arestas indesejáveis (possíveis resultados de cada regra podem ser acompanhados pelas Figuras 4.12 e 4.13):

1. Arestas que ligam CCs de classes diferentes são removidas (Figura 4.12(b)).
2. Arestas que ligam CCs da classe **text** e que são maiores que um determinado limiar são removidas (Figura 4.12(c)).
3. Arestas que cruzam CCs da classe **separator** são removidas (Figura 4.13(b)).



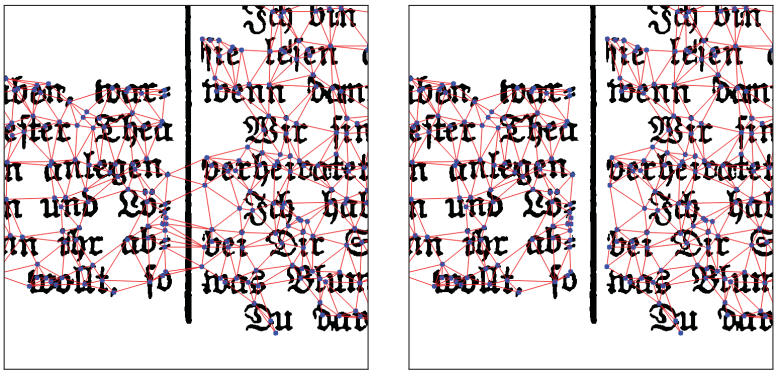
(a) Grafo planar completo.

(b) Resultado após processamento da regra 1.

(c) Resultado após processamento da regra 2.

Figura 4.12: Remoção das arestas indesejáveis no processo de agrupamento. Os pontos coloridos representam a classe do CC, neste exemplo temos CCs da classe **graphic** (em **verde**) e CCs da classe **text** (em **azul**).
Fonte: Autoria própria.

Como os jornais possuem tipos de layouts bem diversificados, a escolha do limiar da regra 2 pode não funcionar bem para todos os tipos de jornais, ou seja, CCs de colunas diferentes podem ainda permanecer conectados (Figura 4.13(a)). A regra 3 tenta corrigir parte desses casos (Figura 4.13(b)). É claro que, caso os processamentos das regras 1 e 2 não consigam remover todas as arestas indesejáveis e não exista um separador entre as regiões, elas ainda permanecerão conectadas.



(a) Resultado após processamento das regras 1 e 2.

(b) Resultado após processamento das regras 1, 2 e 3.

Figura 4.13: Divisão de uma região de texto em duas devido à presença de um separador de colunas entre elas.
Fonte: Autoria própria.

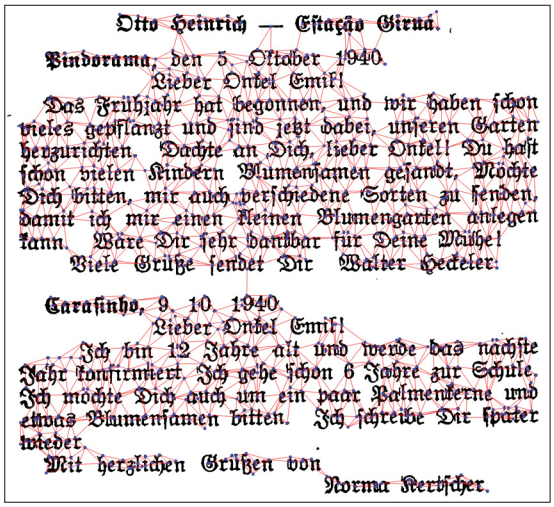
As arestas da classe **text** que restaram são plotadas em uma imagem que servirá como uma máscara (Figura 4.14(b)) para a detecção das regiões. Os contornos externos da imagem de máscara são encontrados. No exemplo mostrado na Figura 4.14, serão encontrados dois contornos na máscara, ou seja, os CCs serão agrupados em duas regiões. Para encontrar os limites de cada região concatenamos todos os contornos dos CCs pertencentes a uma mesma região e geramos um polígono *convex hull* (Figura 4.14(c)).

Não implementamos um limiar para a remoção das arestas que ligam dois ou mais elementos gráficos ou imagens. Concluímos que este processo é desnecessário pois a maioria dos jornais possui um layout onde esses elementos já estão afastados uns dos outros. Com essas arestas também geramos uma imagem de máscara, detectamos seus contornos externos,

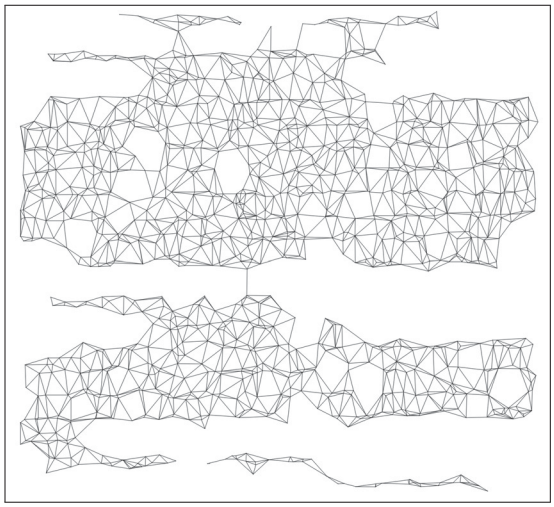
concatenamos os contornos dos CCs de cada região e criamos seu contorno final com um polígono *convex hull*. Nas arestas da classe **separator** essa tarefa não é realizada porque consideramos como uma região o próprio contorno do CC.

Regiões sobrepostas não são permitidas no formato XML PAGE, somente regiões aninhadas (regiões totalmente contidas em outras). Devido a isso, tratamos todas as sobreposições parciais entre regiões. Regiões da mesma classe e que possuem alguma sobreposição são mescladas (Figura 4.14(d)). Em sobreposições de regiões de classes diferentes, é verificada a classe dos CCs da área sobreposta e a região rotulada com a mesma classe dos CCs incorpora a área.

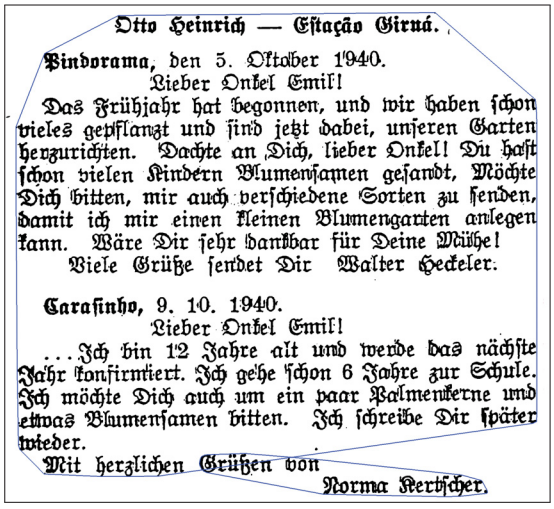
Com as coordenadas geométricas e a classe de cada região definidas, é gerado finalmente o arquivo no formato XML PAGE de resultado. Para isso utilizamos a biblioteca[3] implementada em C++ disponibilizada pelo PRImA Lab.



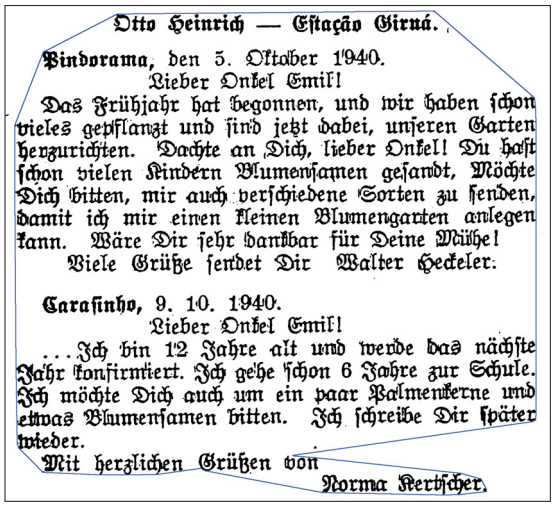
(a) Um exemplo de grafo planar final.



(b) Máscara criada para a detecção das regiões.



(c) Dois polígonos *convex hull* (em **azul**) criados a partir dos CCs pertencentes a cada contorno externo da máscara.



(d) Resultado final (mesclagem das regiões devido a área sobreposta ser do mesmo tipo de ambas.)

Figura 4.14: Etapas da concepção das regiões.
Fonte: Autoria própria.

[3]https://www.primaresearch.org/tools/PAGELibraries

5 MATERIAIS E MÉTODOS

Este Capítulo descreve os experimentos realizados e os procedimentos para a avaliação dos resultados. São apresentados detalhes de execução como, por exemplo, os parâmetros utilizados, quantidade de imagens e procedimentos de validação.

Foram realizados dois experimentos usando o *GBN Dataset*. O primeiro deles, apresentado na Seção 5.2, tem o objetivo de avaliar o desempenho da CNN responsável pela classificação inicial dos *pixels* do método GBN-DL. O outro, descrito na Seção 5.3, contém os procedimentos para uma comparação e avaliação dos métodos propostos neste trabalho para análise de layout (GBN-MHS e GBN-DL) e o analisador de layout embutido no Tesseract.

5.1 ESPECIFICAÇÃO DE *HARDWARE* E *SOFTWARE*

Todos os experimentos foram realizados em um servidor com sistema operacional Debian GNU/Linux 9, processador Intel®Xeon®CPU E5-2620 2.00GHz, 32GB RAM, GPU Nvidia TITAN Xp, CUDA versão 9.0. As implementações foram feitas na linguagem Python 2.7.13. Usamos o Framework Keras 2.2.4 (com Tensor Flow 1.12.0) e as bibliotecas Matplotlib 2.2.3, Numpy 1.15.4, OpenCV 3.4.5 e Sklearn 0.20.2.

5.2 CLASSIFICAÇÃO DOS *PIXELS* PELA CNN DO GBN-DL

Como apresentado na Seção 4.3, no método GBN-DL utilizamos uma arquitetura de CNN para treinar e classificar os *pixels* das páginas dos jornais. Esta Seção descreve os parâmetros e fornece informações sobre o treinamento da rede, bem como as métricas utilizadas para sua avaliação.

Para criar os *patches* de treinamento, inicialmente fizemos um processo de normalização nas imagens, já que cada título de jornal possui uma resolução diferente. As páginas foram reduzidas de forma que duas linhas do corpo de seu texto pudessem ser contidas em *patches* de 28×28 *pixels*. O fator de redução para cada jornal está listado na Tabela 5.1.

Tabela 5.1: Fator de redução usado em cada jornal para que duas linhas do corpo de seu texto pudessem ser contidas em *patches* de 28×28 *pixels*.

Título do jornal	Fator de redução
"*Der Gemeindebote*"	7×
"*Der Jugendfreund*"	6×
"*Der Pioner*"	7×
"*Der Landwirt*"	7×
"*Ev.-Luth. Kirchenblatt*"	4×
"*Kolonie Zeitung*"	7×
"*Gemeindeblatt*"	6×
"*Heimatbote*"	6×

Após o redimensionamento das 102 páginas do conjunto de treinamento, passamos em cada página uma janela deslizante de tamanho 28×28 a cada 3 *pixels* de largura e altura e produzimos os *patches*. Definimos como rótulo do *patch* o mesmo rótulo de seu *pixel* central. Este processo resultou em 1.137.196 *patches* da classe **text**, 116.823 *patches* da classe **image**,

89.126 *patches* da classe **graphic** e 54.973 *patches* da classe **separator**. Para o modelo treinado não tendenciar para alguma determinada classe, realizamos o balanceamento entre as classes, ou seja, utilizamos apenas 54.973 dos *patches* de cada classe.

Para o treinamento, utilizamos a técnica de validação cruzada *k-folds*. O conjunto de 54.973 *patches* de cada classe (totalizando 219.892 *patches*) foi dividido em 5 (k) subconjuntos (*folds*) mutualmente exclusivos, sendo assim, cada *fold* ficou com 10.994 exemplos de cada classe. Dos 5 *folds* resultantes, utilizamos 4 deles para o treinamento e 1 para a validação. Na busca por melhores resultados executamos cada treinamento por 100 e 200 épocas, com um *learning rate* de 10^{-4} e 10^{-5}. Utilizamos a quantidade de 64 *patches* nos *batches* de entrada. O modelo que atingiu a melhor acurácia dentre as combinações de parâmetros propostas foi utilizado para predizer os *pixels* das páginas do conjunto de teste.

As 50 páginas do conjunto de teste foram redimensionadas da mesma maneira que suas páginas de treinamento. Todos os *pixels* das páginas de teste foram transformados em *patches* de 28×28 e testados. Os *pixels* que não pertencem a nenhuma das 4 classes (**text**, **image**, **graphic** ou **separator**) foram rotulados como *pixels* de ruído e também receberam uma predição da rede. A Tabela 5.2 mostra a quantidade de *patches* de teste por classe em cada título de jornal.

Tabela 5.2: Quantidade de *patches* de teste por classe em cada título de jornal.

Título do jornal	text	image	graphic	separator	ruído
"*Der Gemeindebote*"	233.661	0	7.810	5.201	4.181
"*Der Jugendfreund*"	211.508	0	16.963	5.591	66
"*Der Pioner*"	997.856	0	155.054	68.780	6.693
"*Der Landwirt*"	212.971	148.495	112.745	21.688	682
"*Ev.-Luth. Kirchenblatt*"	339.032	41.292	35.857	12.943	4.435
"*Kolonie Zeitung*"	893.070	0	38.315	33.481	45.354
"*Gemeindeblatt*"	466.781	0	0	23.703	16.695
"*Heimatbote*"	1.137.924	0	58.472	16.331	6.758
Total	4.492.803	189.787	425.216	187.718	84.864

A avaliação deste experimento foi baseada no número de acertos e erros das predições da rede. As seguintes métricas foram calculadas por classe:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

onde TP (*True Positive*) corresponde ao número de *pixels* da classe classificados corretamente, FP (*False Positive*) é a quantidade de *pixels* classificados incorretamente como sendo da classe e FN (*False Negative*) são os *pixels* da classe que foram classificados incorretamente.

5.3 ANÁLISE DE LAYOUT DOS MÉTODOS GBN-MHS, GBN-DL E TESSERACT

Comparamos os resultados da análise de layout dos métodos propostos (GBN-MHS e GBN-DL) com o resultado da análise de layout do Tesseract. As 50 páginas do conjunto de teste

do *GBN Dataset* foram processadas nos 3 métodos. Os parâmetros utilizados na execução de cada método foram definidos com base no conjunto de treinamento e estão descritos nas Seções 5.3.1, 5.3.2 e 5.3.3. Por fim, a Seção 5.3.4 apresenta o protocolo de avaliação do experimento.

5.3.1 Método GBN-MHS

- $T^{\text{area}} = 0.00005$; os CCs com uma área 0.005% menor que a área total da página são removidos pelo Filtro Heurístico.
- $T^{\text{inside}} = 5$; CCs com mais de 5 componentes conexos internos são removidos pelo Filtro Heurístico.
- $T^{\text{rate}} = 0.06$; CCs com largura maior que 16.66 vezes sua altura são removidos pelo Filtro Heurísticos.
- $T_{\text{Var}} \leq 50$; uma região será considerada homogênea se sua variância horizontal for menor que 50.
- $s = 2\times$ a mediana das alturas dos CCs da sua região. Parâmetro utilizado na suavização do histograma da projeção horizontal.
- Um CC_i não-texto é classificado como **separator** se possuir $HW_i^{\text{rate}} \leq 0.1$, caso contrário, ele é da classe **graphic**.

5.3.2 Método GBN-DL

No início do método GBN-DL classificamos os *patches* das páginas através de uma CNN. O modelo CNN treinado que utilizamos foi escolhido no experimento anterior e seus resultados estão descritos na Seção 6.1.

No agrupamento em regiões usamos um limiar para a remoção das arestas de texto que possivelmente conectam regiões de texto diferentes. Elas são removidas se forem maiores que duas vezes a mediana das arestas de texto.

5.3.3 Tesseract

Apesar do formato XML PAGE ser o arquivo de representação mais comum em bases de dados para análise de layout, este formato de saída não está implementado no Tesseract. Sendo assim, optamos pelo uso da ferramenta *Tesseract OCR to PAGE*[1] do PRImA Lab que processa a imagem de entrada com os algoritmos do Tesseract e já retorna a saída no formato XML PAGE. Executamos a seguinte linha de comando para as páginas do conjunto de teste:

```
>> TesseractToPAGE.exe -inp-img "01.png" -out-xml "01.xml" -rec-mode layout
```

O parâmetro `-rec-mode` refere-se ao nível de reconhecimento desejado, neste caso ajustado para retornar apenas as informações referentes ao layout da página.

[1] https://www.primaresearch.org/tools/TesseractOCRToPAGE

5.3.4 Protocolo de Avaliação para a Análise de Layout

Neste experimento de análise de layout avaliamos a nível de *pixel* as saídas XML PAGE produzidas pelos métodos. Para isso geramos um arquivo de texto contendo as coordenadas dos *pixels* de primeiro plano que foram englobados por alguma região resultante e atribuímos para cada *pixel* a mesma classe da sua região. Comparamos os *pixels* deste arquivo de texto com os *pixels* do arquivo de texto do *ground truth*.

Dois cenários de avaliação foram criados. O primeiro deles é composto por 28 páginas de jornais que estavam representados no conjunto de treinamento, são eles: "*Der Gemeindebote*", "*Der Jugendfreund*", "*Der Pioner*", "*Der Landwirt*", "*Ev.-Luth. Kirchenblatt*" e "*Kolonie Zeitung*". O segundo cenário foi criado para avaliar a capacidade de generalização dos métodos. Ele é composto por páginas de jornais que não foram representados no conjunto de treinamento, são eles: "*Gemeindeblatt*" (8 páginas) e "*Heimatbote*" (14 páginas). Nesses jornais não há regiões da classe **image**.

Em cada cenário, os métodos foram avaliados por métricas globais e individualizadas por classe. Nas métricas globais consideramos somente as predições dos *pixels* de interesse, ou seja, predições de *pixels* que não estavam listados no arquivo de texto de *ground truth* foram descartadas e seus erros de classificação não foram penalizados. Definimos as métricas globais como:

$$Precision = \frac{TP}{TP + FP} \tag{5.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.5}$$

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.6}$$

$$\text{Acurácia} = \frac{TP}{PI} \tag{5.7}$$

onde TP (*True Positive*) corresponde ao número de *pixels* classificados corretamente, FP (*False Positives*) refere-se ao número de *pixels* classificados incorretamente; FN (*False Negatives*) corresponde ao total de *pixels* de interesse que não foram classificados pelo método, ou seja, *pixels* que não estão contidos em nenhuma região do XML PAGE resultante mas deveriam estar; e PI é o número total de *pixels* de interesse.

Ajustamos as métricas *Precision*, *Recall* e *F-measure* para a avaliação por classes, considerando como: TP o número de *pixels* classificados corretamente em cada classe; FP os *pixels* que foram rotulados incorretamente como pertencentes à classe; e FN a soma dos *pixels* que são da classe mas foram classificados como sendo de outra classe, mais a quantidade de *pixels* de interesse da classe que não foram classificados pelo método. Para cada cenário também apresentamos a Matriz Confusão de cada método.

6 RESULTADOS E DISCUSSÃO

Neste Capítulo são apresentados os resultados obtidos conforme a metodologia experimental discutida anteriormente. A Seção 6.1 apresenta e discute o desempenho da classificação dos *pixels* realizada pela CNN do método GBN-DL e a Seção 6.2 fornece os resultados e considerações sobre a análise de layout dos métodos GBN-MHS, GBN-DL e Tesseract no *GBN Dataset*.

6.1 CLASSIFICAÇÃO DOS *PIXELS* PELA CNN DO GBN-DL

Esta Seção apresenta os resultados obtidos pela CNN do método GBN-DL nos conjuntos de treinamento e teste. A Tabela 6.1 mostra as acurácias atingidas no conjunto de treinamento pelos modelos, usando as combinações de parâmetros propostas.

Tabela 6.1: Acurácias atingidas pela CNN do método GBN-DL nas páginas de treinamento do GBN *Dataset* usando as combinações de parâmetros: k = [1–5], $lr = [10^{-4}, 10^{-5}]$ e épocas = [100,200].

	k	100 épocas	200 épocas
$lr = 10^{-4}$	1	91,79	92,31
	2	91,16	90,06
	3	86,79	**92,52**
	4	91,53	92,51
	5	88,48	92,02
$lr = 10^{-5}$	1	84,50	60,07
	2	81,92	86,76
	3	84,17	87,86
	4	85,92	87,97
	5	86,60	87,08

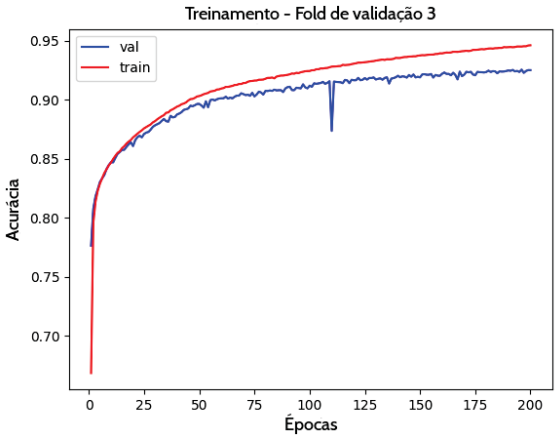


Figura 6.1: Curvas de acurácia do melhor modelo encontrado no treinamento: $k = 3$, $lr = 10^{-4}$ e épocas = 200. Fonte: Autoria própria.

Como podemos observar pela Tabela 6.1, o modelo treinado no *fold* de validação número 3 por 200 épocas e com um *learning rate* de 10^{-4} foi o que obteve a acurácia mais alta entre

os modelos: 92,52%. O gráfico da Figura 6.1 mostra as curvas de acurácia do treinamento e validação deste modelo durante as 200 épocas. Devido ao melhor desempenho, este foi o modelo utilizado na etapa de classificação dos *pixels* das páginas de teste no método GBN-DL. O resultado nesses *pixels* do conjunto de teste encontram-se listados na Tabela 6.2, separados por classe.

Tabela 6.2: Resultados atingidos pela CNN do método GBN-DL nas páginas de teste do GBN *Dataset*.

Classe	**Qnt. de *patches***	***Precision***	***Recall***	***F1-measure***
Text	4.492.803	99,26	93,78	96,28
Image	189.787	93,91	91,30	92,61
Graphic	425.216	59,83	77,76	66,75
Separator	187.718	82,42	95,72	87,77

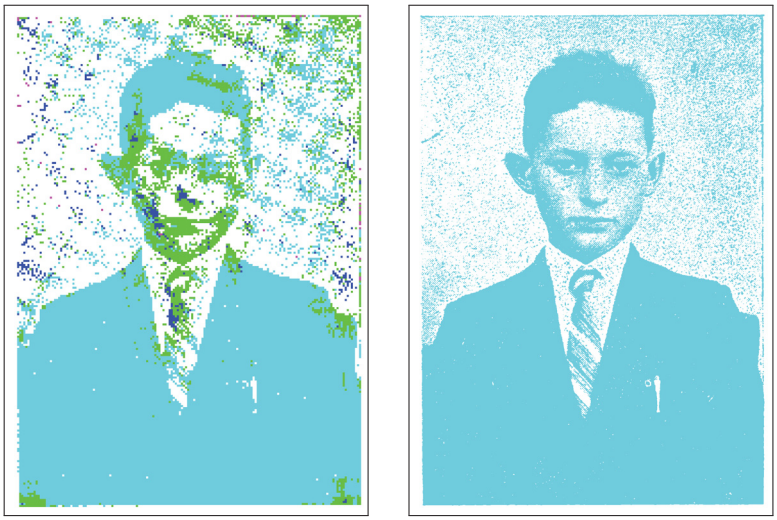
A classe **text** foi a classe na qual a rede conseguiu reconhecer corretamente a maioria dos *pixels*, atingiu a *precision* mais alta em relação às outras classes, 99,26%. A maioria dos erros da classe **text** foram em *pixels* contidos em caracteres de fontes grandes, como nos títulos e subtítulos do jornal. Como podemos observar no exemplo da Figura 6.2(a), a rede atribuiu a classe **graphic** (representada em **verde**) a muitos desses *pixels* de texto (representados pela cor **azul**). Concluímos que isso ocorreu devido a esses caracteres serem maiores que os caracteres do corpo do texto e então não puderam ser totalmente contidos nos *patches* de 28×28. Como no treinamento havia poucos exemplos com essa característica, a rede não conseguiu aprender suficientemente bem a identificar *pixels* de caracteres segmentados pelos *patches*.

(a) Resultado da classificação de *pixels* da CNN. (b) *Ground truth*.

Figura 6.2: Resultado da classificação de *pixels* feita pela CNN em parte da página de um jornal "*Der Pionier*" (a) e seu *ground truth* (b). *Pixels* da classe **text** são representados em **azul**, *pixels* **image** em **ciano**, *pixels* **graphic** em **verde**, e *pixels* **separator** em **magenta**. Em (a) constatamos que muitos *pixels* de caracteres de fonte grande foram classificados incorretamente como sendo da classe **graphic**.
Fonte: Autoria própria.

A classe **graphic** obteve as piores taxas de avaliação. Uma *precision* de apenas 59,83% indica uma grande quantidade de falso positivos na classe. Isso deve-se ao fato de que muito dos *pixels* da classe **image** foram classificados como sendo da classe **graphic**. A Figura 6.3 mostra parte de uma página do jornal "*Evangelisch-Lutherisches Kirchenblatt*" onde houve essa confusão.

Outra classe com grande ocorrência de falsos positivos foi a **separator**. Atingiu 82,42% em *precision* e 95,72% em *recall*. O bom resultado em *recall* mostra que a rede conseguiu identificar corretamente os *pixels* de separadores, já a taxa de *precision* não tão alta ocorre porque muito dos *pixels* de outras classes foram confundidos como sendo da classe **separator**. A Figura



(a) Resultado da classificação de *pixels* da CNN.

(b) *Ground truth*.

Figura 6.3: Resultado da classificação de *pixels* feita pela CNN em parte de um página do jornal "*Evangelisch-Lutherisches Kirchenblatt*" (a) e seu *ground truth* (b). *Pixels* da classe **text** são representados em **azul**, *pixels* **image** em **ciano**, *pixels* **graphic** em **verde**, e *pixels* **separator** em **magenta**. Em (a) é possível notar uma grande quantidade de FP da classe **graphic**.

Fonte: Autoria própria.

(a) Resultado da classificação de *pixels* da CNN.

(b) *Ground truth*.

Figura 6.4: Resultado da classificação de *pixels* feita pela CNN em parte da página de um jornal "*Der Gemeindebote*" (a) e seu *ground truth* (b). *Pixels* da classe **text** são representados em **azul**, *pixels* **image** em **ciano**, *pixels* **graphic** em **verde**, e *pixels* **separator** em **magenta**. Em (a) notamos a ocorrência de FPs da classe **separator** em algumas extremidades de caracteres e em elementos gráficos com traçado fino.

Fonte: Autoria própria.

6.4 ilustra onde ocorreram esses erros: em algumas extremidades de caracteres e em elementos gráficos com traçado fino.

Concluímos que a classificação dos *pixels* pela nossa arquitetura de CNN foi satisfatória. Ela consegue distinguir elementos texto de elementos não-texto, tarefa esta fundamental para

um bom resultado de OCR. Com o objetivo de melhorar o resultado, experimentos futuros podem ser realizados principalmente explorando a arquitetura da rede e seus parâmetros, como a adição de mais camadas de convolução e alterações no *learning rate*. Um treinamento com mais informações de contexto dos *pixels*, como no trabalho de Julca-Aguilar et al. (2017), também é um experimento interessante a se testar.

6.2 ANÁLISE DE LAYOUT DOS MÉTODOS GBN-MHS, GBN-DL E TESSERACT

Esta Seção relata os resultados obtidos na análise de layout das páginas de teste do GBN *Dataset* pelos métodos propostos neste trabalho (GBN-MHS e GBN-DL) e pelo analisador de layout do Tesseract.

Podemos observar pela Tabela 6.3 que o GBN-DL foi o método que atingiu as melhores taxas na avaliação global tanto no Cenário 1 com uma acurácia de 92,81% por exemplo, quanto no Cenário 2 com 96,96%. Em seguida temos o GBN-MHS com uma acurácia de 88,12% no Cenário 1 e 95,65% no Cenário 2 e o pior desempenho foi do analisador de layout do Tesseract com 71,83% de acurácia no Cenário 1 e 88,15% no Cenário 2.

Tabela 6.3: Avaliação global da classificação dos *pixels* nas análises de layout dos métodos GBN-MHS, GBN-DL e Tesseract nos dois cenários propostos.

	Método	***Precision***	***Recall***	***F1-measure***	**Acurácia**
Cenário 1	GBN-MHS	91,12	96,40	93,68	88,12
	GBN-DL	**93,36**	**99,36**	**96,27**	**92,81**
	Tesseract	76,04	92,84	83,61	71,83
Cenário 2	GBN-MHS	96,18	99,42	97,77	95,65
	GBN-DL	**97,03**	**99,92**	**98,45**	**96,96**
	Tesseract	92,22	95,23	93,70	88,15

A Tabela 6.4 mostra os resultados por classe. As métricas da classe **image** estão zeradas do GBN-MHS porque não implementamos esta classificação no método. Na Matriz Confusão do GBN-MHS (Tabela 6.5) podemos notar que muito dos *pixels* da classe **graphic** foram classificados como **text**, isso explica a baixa taxa de *recall* nos dois Cenários: 54,06% no Cenário 1 e 5,13% no Cenário 2.

O GBN-DL atingiu as melhores taxas em todas as métricas por classe nos dois Cenários propostos, ficou abaixo apenas na *precision* do Tesseract na classe **text** no Cenário 2, com 99,48%. As piores taxas do GBN-DL foram na classe **graphic**, com a *precision* de 64,93% no Cenário 1 e 58,47% no Cenário 2. Isso mostra a grande ocorrência de falsos positivos. Pela Matriz Confusão do método (Tabela 6.6(a)) observamos que esses falsos positivos são principalmente *pixels* de regiões de texto (7.138.926 *pixels*).

O Tesseract não classifica as regiões em **graphic**, somente em **image**, **text** e **separator**. Por esse motivo as métricas da classe **graphic** estão zeradas. Como a grande maioria dos *pixels* de **graphic** foram classificados como *pixels* de **image**, a *precision* da classe **image** foi de apenas 21,12% no Cenário 1.

No GBN-MHS e no GBN-DL fizemos uma etapa de identificação e remoção das bordas das páginas. Esse processo não é realizado pelo Tesseract. Como podemos observar pela sua Matriz Confusão (Tabela 6.7), Tesseract classificou esses componentes em regiões de imagem e separador. *Pixels* da classe **separator** também foram classificados como texto, o que contribuiu para um *recall* de apenas 40,89% no Cenário 1 e 61,11% no Cenário 2.

Tabela 6.4: Avaliação por classe da classificação dos *pixels* nas análises de layout dos métodos GBN-MHS, GBN-DL e Tesseract nos dois cenários propostos.

	Método	Classe	*Precision*	*Recall*	*F1-measure*
Cenário 1	GBN-MHS	Text	92,33	98,36	95,25
		Image	0	0	0
		Graphic	81,08	54,06	64,87
		Separator	83,04	81,83	82,43
	GBN-DL	Text	98,42	94,02	96,17
		Image	99,19	91,31	95,08
		Graphic	64,93	87,96	74,71
		Separator	89,91	83,65	86,67
	Tesseract	Text	96,41	81,06	88,07
		Image	21,12	99,77	34,86
		Graphic	0	0	0
		Separator	52,28	40,89	45,89
Cenário 2	GBN-MHS	Text	96,45	99,45	97,93
		Image	-	-	-
		Graphic	40,93	5,13	9,13
		Separator	94,95	76,11	84,49
	GBN-DL	Text	99,48	97,34	98,40
		Image	-	-	-
		Graphic	58,47	85,60	69,48
		Separator	86,74	98,31	92,16
	Tesseract	Text	98,92	92,05	95,36
		Image	-	-	-
		Graphic	0	0	0
		Separator	45,69	61,11	52,29

Tabela 6.5: Matriz Confusão do Método GBN-MHS no Cenário 1 (a) e no Cenário 2 (b).

(a) Método GBN-MHS no Cenário 1

		Predição				
		text	**image**	**graphic**	**separator**	**ruído**
Classe	**text**	125.543.812	0	1.805.171	8.276	273.278
	image	3.843.209	0	274.573	8.415	3.813.497
	graphic	5.700.099	0	8.955.720	1.115.270	792.746
	separator	878.155	0	9.499	5.544.072	343.150

(b) Método GBN-MHS no Cenário 2

		Predição				
		text	**image**	**graphic**	**separator**	**ruído**
Classe	**text**	57.441.123	0	155.300	5.807	155.306
	image	-	-	-	-	-
	graphic	1.947.527	0	108.395	5.3054	0
	separator	161.311	0	1.097	1.107.263	185.003

Tabela 6.6: Matriz Confusão do Método GBN-DL no Cenário 1 (a) e no Cenário 2 (b).

(a) Método GBN-DL no Cenário 1

Classe \ Predição	text	image	graphic	separator	ruído
text	119.998.291	44.624	7.138.926	240.286	208.410
image	20.357	7.249.767	559.246	108.778	1.546
graphic	1.187.546	14.474	14.569.581	286.347	505.887
separator	716.978	0	168.074	5.667.633	222.191

(b) Método GBN-DL no Cenário 2

Classe \ Predição	text	image	graphic	separator	ruído
text	56.224.377	19.920	1.275.044	194.416	43.779
image	-	-	-	-	-
graphic	273.340	5.786	1.805.420	24.153	277
separator	17.531	0	6.844	1.430.171	128

Tabela 6.7: Matriz Confusão do Tesseract no Cenário 1 (a) e no Cenário 2 (b).

(a) Tesseract no Cenário 1

Classe \ Predição	text	image	graphic	separator	ruído
text	103.462.866	16.582.855	0	1.983.249	5.601.567
image	0	7.921.685	0	17.867	142
graphic	2.416.548	12.124.648	0	526.909	1.495.730
separator	1.428.731	872.254	0	2.770.639	1.703.252

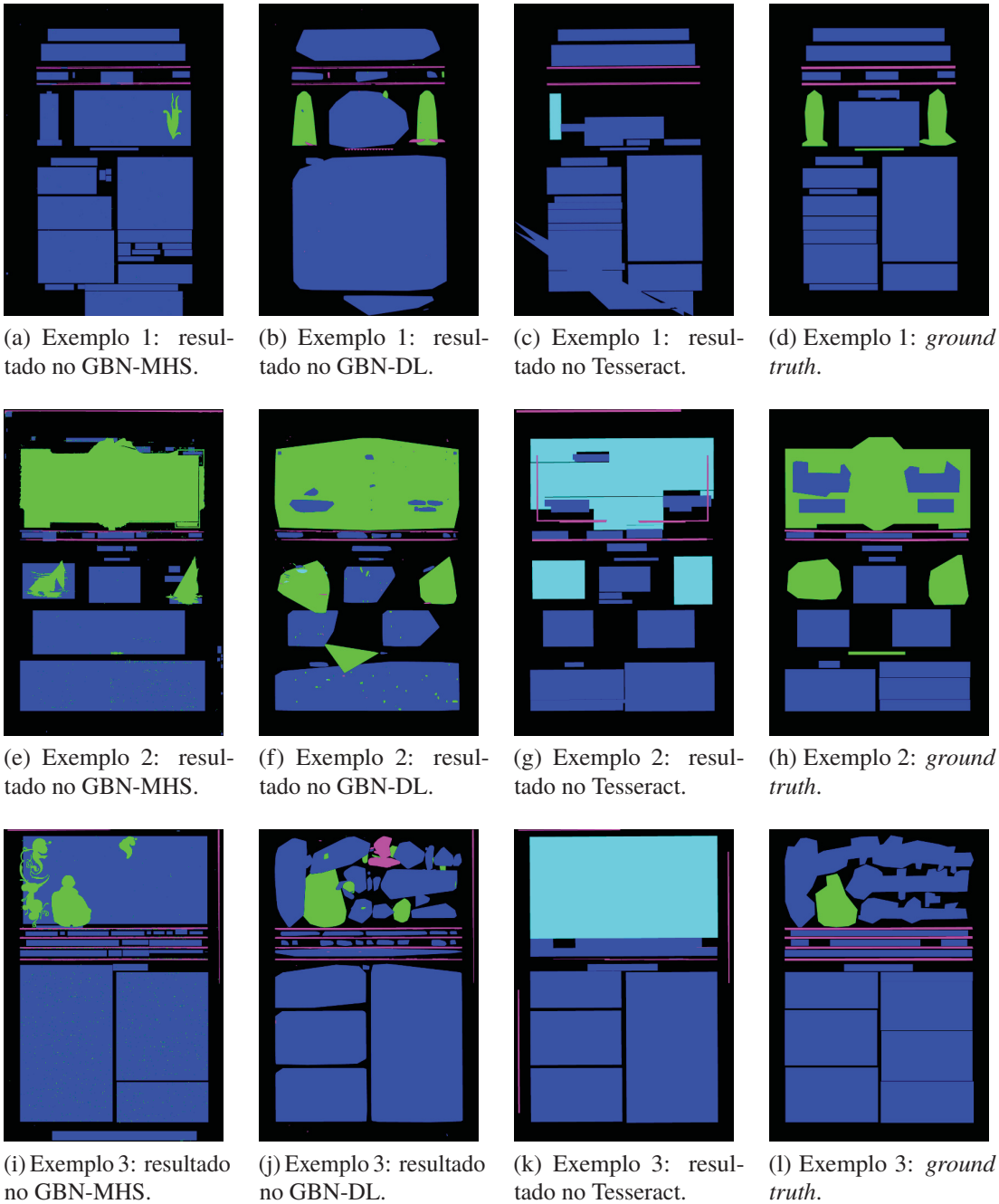
(b) Tesseract no Cenário 2

Classe \ Predição	text	image	graphic	separator	ruído
text	53.169.027	1.386.092	0	932.592	2.269.825
image	-	-	-	-	-
graphic	458.087	1.494.828	0	123.900	32.161
separator	122.029	41.554	0	889.005	402.086

Os três métodos avaliados conseguiram diferenciar os *pixels* da classe **text** das demais classes e isso é fundamental para que um sistema OCR consiga atingir altas taxas de acerto no reconhecimento dos caracteres. Porém é necessário que as regiões de texto estejam segmentadas de uma forma que não prejudique a ordem de leitura do texto.

Apesar dos métodos propostos neste trabalho realizarem a segmentação de regiões, nós optamos por avaliá-los a nível de *pixel*. Observe com em nossas métricas não há uma penalização para o método caso ele agrupe regiões de texto pertencentes a colunas diferentes (*under-segmentation*) ou faça uma segmentação excessiva (*over-segmentation*) em alguma região de texto que prejudique a ordem de leitura. Para a avaliação deste cenário seriam necessárias métricas como as propostas por Clausner et al. (2011b). Entretanto, mesmo sem poder mensurá-las mostramos na Figura 6.5 algumas situações onde o agrupamento dos CCs / segmentação de regiões não funcionou da forma esperada. O GBN-MHS super segmentou algumas regiões de texto no Exemplo 1 (Figura 6.5(a)) e agrupou regiões de texto de colunas diferentes no Exemplo 2 (Figura 6.5(e)). As *over-segmentations* do GBN-MHS ocorreram nas situações em que o parâmetro s não foi suficiente para suavizar todos os pontos de mínimos e máximos locais do histograma da projeção horizontal. Isso tornou a variância do conjunto Δ alta e então o algoritmo segmentou a região. Já os erros de *under-segmentation* provavelmente tenham ocorrido devido à decisão de usar somente a projeção horizontal para definir se uma região é homogênea ou não.

No GBN-DL também é necessário um melhor estudo sobre os parâmetros. O limiar usado para desconectar regiões de texto que possivelmente pertencem a colunas diferentes não funcionou bem para todos os tipos de jornais. Observe nos Exemplos 1 (Figura 6.5(b)) e 2 (Figura 6.5(f)) como há regiões de texto que permaneceram conectadas. Já no Exemplo 3 (Figura 6.5(j)) o método conseguiu atingir um bom resultado com o mesmo parâmetro utilizado nos Exemplos 1 e 2.



(a) Exemplo 1: resultado no GBN-MHS. (b) Exemplo 1: resultado no GBN-DL. (c) Exemplo 1: resultado no Tesseract. (d) Exemplo 1: *ground truth*.

(e) Exemplo 2: resultado no GBN-MHS. (f) Exemplo 2: resultado no GBN-DL. (g) Exemplo 2: resultado no Tesseract. (h) Exemplo 2: *ground truth*.

(i) Exemplo 3: resultado no GBN-MHS. (j) Exemplo 3: resultado no GBN-DL. (k) Exemplo 3: resultado no Tesseract. (l) Exemplo 3: *ground truth*.

Figura 6.5: Resultado da análise de layout do GBN-DL, GBN-MHS e Tesseract em três páginas de jornais do *GBN Dataset* e seus respectivos *ground truth*. As regiões da classe **text** estão representadas em **azul**, regiões **image** em **ciano**, regiões **graphic** em **verde** e regiões **separator** em **magenta**.

7 CONCLUSÃO E TRABALHOS FUTUROS

O número de instituições e empresas interessadas em digitalizar e disponibilizar seu acervo em papel tem crescido cada vez mais. Devido a isto, projetos de digitalização em massa de documentos têm surgido ao redor do mundo. No Brasil, temos a iniciativa *Dokumente.br* que surgiu com o objetivo de disponibilizar acervos brasileiros produzidos em língua alemã. Parte deste acervo é composto por jornais históricos.

Jornais históricos possuem características como ruídos de degradação do papel e layout complexo que desafiam as ferramentas OCR *open source* existentes, já que estas foram desenvolvidas para a digitalização de documentos contemporâneos.

A maior lacuna nestas ferramentas é na tarefa de análise de layout. A análise de layout é responsável por segmentar e classificar as regiões da página em regiões de texto e não-texto, sendo assim, se a ferramenta não apresentar bons resultados nesta tarefa, o desempenho do OCR nas regiões de texto fica comprometido.

Motivados por essa problemática, propomos neste trabalho duas abordagens para a análise de layout das páginas dos jornais da iniciativa *Dokumente.br*. A primeira delas, que nomeamos de GBN-MHS, é baseada nos conceitos do "MHS 2017 System" proposto por Tran et al. (2017). A outra abordagem, que chamamos de GBN-DL, usa *deep learning* para a classificação do pixels e conceitos de Maia et al. (2018) para o agrupamento dos componentes conexos em regiões.

Para avaliar nossos métodos criamos o *German-Brazilian Newspaper Dataset* (GBN 1.0), composto por 152 páginas representativas de 8 jornais do acervo da iniciativa *Dokumente.br*. Realizamos dois experimentos com o *dataset*. No primeiro deles (Seção 5.2) avaliamos o resultado da CNN do método GBN-DL que foi treinada para classificar os pixels em **text**, **image**, **graphic** e **separator**. O modelo teve uma boa precisão nas classes **text** (99,26%), **image** (93,91%) e **separator** (82,42%). Entretanto na classe **graphic** conseguiu apenas uma *precision* de 59,83%. Isso mostra que a nossa CNN precisa de ajustes principalmente relacionados ao reconhecimento da classe **graphic**.

No segundo experimento (Seção 5.3), comparamos os resultados da análise de layout dos métodos propostos neste trabalho com o resultado do analisador de layout do Tesseract. Os jornais de teste do *GBN Dataset* foram divididos em dois Cenários para a avaliação: o Cenário 1 foi composto por jornais que foram representados no conjunto de treinamento e o Cenário 2 foi composto por páginas de jornais que não foram representadas no conjunto de treinamento. Tanto GBN-MHS, quanto GBN-DL conseguiram resultados melhores que o Tesseract em todas as métricas dos dois Cenários pospostos. No Cenário 1, GBN-DL atingiu 92,81% de acurácia, GBN-MHS teve 88,12% e Tesseract conseguiu apenas 71,83%. Tesseract foi um pouco melhor no Cenaŕio 2 comparado ao seu desempenho no Cenário 1, atingiu 88,15% de acurácia; GBN-MHS conseguiu 95,16% e GBN-DL obteve 96,96% de acerto.

Concluímos assim que os objetivos pretendidos com este trabalho foram alcançados. As ferramentas *open source* para OCR em Fraktur foram compreendidas. Os bons resultados atingidos nos experimentos por nossos métodos propostos, GBN-MHS e GBN-DL, comprovam o potencial das nossas abordagens e mostram como essas ferramentas OCR *open source* existentes não estão preparadas para trabalhar com documentos históricos.

7.1 TRABALHOS FUTUROS

Durante o desenvolvimento deste trabalho e após a análise dos resultados, identificamos os seguintes pontos que necessitam de melhorias e que podem ser objetos de pesquisas futuras:

- No GBN-MHS é necessário definir características que diferenciem componentes da classe **image** e **graphic**.
- O treinamento e arquitetura da CNN do método GBN-DL precisa ser melhor explorado. São necessárias mais informações de contexto para os pixels.
- Como o Tesseract não classifica regiões em **graphic** é necessário algum processo de normalização.
- O agrupamento dos componentes conexos em regiões precisa ser melhor avaliado, pois nossas métricas não penalizam regiões sobrepostas e erros de *under-segmentation* e *over-segmentation*.
- Os métodos propostos neste trabalho não definem a ordem de leitura das regiões de texto e essa é uma etapa fundamental para a compreensão da informações da página.
- Os métodos para análise de layout propostos neste trabalho podem ser adicionados a um *workflow* de OCR e seu resultado final (taxa de OCR) pode ser avaliado e comparado com o resultado final das ferramentas OCR *open sources* existentes: Tesseract e OCRopy.

REFERÊNCIAS

Afzal, M. Z., Krämer, M., Bukhari, S. S., Yousefi, M. R., Shafait, F. e Breuel, T. M. (2013). Robust binarization of stereo and monocular document images using percentile filter. Em *International Workshop on Camera-Based Document Analysis and Recognition*, páginas 139–149. Springer.

Alves, J. H., Neto, P. M. M. e Oliveira, L. F. (2018). Extracting lungs from ct images using fully convolutional networks. Em *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–8. IEEE.

Antonacopoulos, A., Clausner, C., Papadopoulos, C. e Pletschacher, S. (2013). ICDAR 2013 Competition on historical newspaper layout analysis (HNLA 2013). Em *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, páginas 1454–1458. IEEE.

Antonacopoulos, A., Clausner, C., Papadopoulos, C. e Pletschacher, S. (2015). ICDAR2015 Competition on recognition of documents with complex layouts - RDCL2015. Em *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, páginas 1151–1155. IEEE.

Baird, H. S., Jones, S. E. e Fortune, S. J. (1990). Image segmentation by shape-directed covers. Em *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, páginas 820–825. IEEE.

Breuel, T. M. (2002). Two geometric algorithms for layout analysis. Em *International Workshop on Document Analysis Systems*, páginas 188–199. Springer.

Breuel, T. M. (2003). High performance document layout analysis. Em *Proceedings of the Symposium on Document Image Understanding Technology*, páginas 209–218.

Breuel, T. M. (2007). The hOCR microformat for OCR workflow and results. Em *ICDAR 2007. Ninth International Conference on Document Analysis and Recognition*, volume 2, páginas 1063–1067. IEEE.

Breuel, T. M. (2008). The ocropus open source ocr system. Em *Document Recognition and Retrieval XV*, volume 6815, página 68150F. International Society for Optics and Photonics.

Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A. e Shafait, F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks. Em *2013 12th International Conference on Document Analysis and Recognition*, páginas 683–687. IEEE.

Bukhari, S. S., Azawi, A., Ali, M. I., Shafait, F. e Breuel, T. M. (2010). Document image segmentation using discriminative learning over connected components. Em *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, páginas 183–190. ACM.

Chen, K., Liu, C.-L., Seuret, M., Liwicki, M., Hennebert, J. e Ingold, R. (2016a). Page segmentation for historical document images based on superpixel classification with unsupervised feature learning. Em *Document Analysis Systems (DAS), 2016 12th IAPR Workshop on*, páginas 299–304. IEEE.

Chen, K., Seuret, M., Hennebert, J. e Ingold, R. (2017). Convolutional neural networks for page segmentation of historical document images. Em *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, páginas 965–970. IEEE.

Chen, K., Seuret, M., Liwicki, M., Hennebert, J., Liu, C.-L. e Ingold, R. (2016b). Page segmentation for historical handwritten document images using conditional random fields. Em *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, páginas 90–95. IEEE.

Chen, K., Yin, F. e Liu, C.-L. (2013). Hybrid page segmentation with efficient whitespace rectangles extraction and grouping. Em *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, páginas 958–962. IEEE.

Chowdhury, S., Mandal, S., Das, A. e Chanda, B. (2007). Segmentation of text and graphics from document images. Em *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, páginas 619–623. IEEE.

Clausner, C., Antonacopoulos, A. e Pletschacher, S. (2017). ICDAR2017 Competition on recognition of documents with complex layouts - RDCL2017. Em *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, páginas 1404–1410. IEEE.

Clausner, C., Pletschacher, S. e Antonacopoulos, A. (2011a). Aletheia-an advanced document layout and text ground-truthing system for production environments. Em *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, páginas 48–52. IEEE.

Clausner, C., Pletschacher, S. e Antonacopoulos, A. (2011b). Scenario driven in-depth performance evaluation of document layout analysis methods. Em *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, páginas 1404–1408. IEEE.

Dai-Ton, H., Duc-Dung, N. e Duc-Hieu, L. (2016). An adaptive over-split and merge algorithm for page segmentation. *Pattern Recognition Letters*, 80:137–143.

Eikvil, L. (1993). Optical character recognition. *citeseer. ist. psu. edu/142042. html*.

Eskenazi, S., Gomez-Krämer, P. e Ogier, J.-M. (2017). A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognition*, 64:1–14.

Farabet, C., Couprie, C., Najman, L. e LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

Ferilli, S., Basile, T. e Esposito, F. (2010). A histogram-based technique for automatic threshold assessment in a run length smoothing-based algorithm. Em *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, páginas 349–356. ACM.

Furrer, L. e Volk, M. (2011). Reducing OCR errors in Gothic-script documents. Em *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, páginas 97–103.

Gonzalez, R. C. e Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

HAYKIN, S. (2001). *Redes Neurais - 2ed.* Bookman.

Hochuli, A. G. (2019). *Abordagens livres de segmentação para reconhecimento automático de cadeias numéricas manuscritas utilizando aprendizado profundo*. Tese de doutorado, Programa de Pós-Graduação em Informática - Universidade Federal do Paraná, Curitiba - PR.

Hochuli, A. G., Oliveira, L. S., de Souza Britto, A. e Sabourin, R. (2018). Segmentation-free approaches for handwritten numeral string recognition. Em *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–8. IEEE.

Julca-Aguilar, F. D., Maia, A. L. e Hirata, N. S. (2017). Text/non-text classification of connected components in document images. Em *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, páginas 450–455. IEEE.

Kise, K., Sato, A. e Iwata, M. (1998). Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382.

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R. e Menotti, D. (2019). Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging*, 28:1–14.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Kavukcuoglu, K. e Farabet, C. (2010). Convolutional networks and applications in vision. Em *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, páginas 253–256. IEEE.

Lucio, D. R., Laroca, R., Severo, E., Britto Jr., A. S. e Menotti, D. (2018). Fully convolutional networks and generative adversarial networks applied to sclera segmentation. Em *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, páginas 1–7.

Maia, A. L. L. M., Julca-Aguilar, F. D. e Hirata, N. S. T. (2018). A machine learning approach for graph-based page segmentation. Em *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, páginas 424–431. IEEE.

Nagy, G., Seth, S. e Viswanathan, M. (1992). A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22.

Neudecker, C. (2010). OCR challenges in historic documents and the contribution of IMPACT. https://www.slideshare.net/impactproject/ocr-challenges-in-historic-documents-and-the-contribution-of-impact. Acessado em 07/05/2018.

O'Gorman, L. (1993). The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173.

Okamoto, M. e Takahashi, M. (1993). A hybrid page segmentation method. Em *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, páginas 743–746. IEEE.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

Pan, Y., Zhao, Q. e Kamata, S. (2010). Document layout analysis and reading order determination for a reading robot. Em *TENCON 2010-2010 IEEE Region 10 Conference*, páginas 1607–1612. IEEE.

Papadopoulos, C., Pletschacher, S., Clausner, C. e Antonacopoulos, A. (2013). The IMPACT dataset of historical document images. Em *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, páginas 123–130. ACM.

Pitts, W. e McCulloch, W. S. (1947). How we know universals the perception of auditory and visual forms. *The Bulletin of mathematical biophysics*, 9(3):127–147.

Pletschacher, S. e Antonacopoulos, A. (2010). The PAGE (page analysis and ground-truth elements) format framework. Em *Pattern Recognition (ICPR), 2010 20th International Conference on*, páginas 257–260. IEEE.

Ponti, M. A., Ribeiro, L. S. F., Nazare, T. S., Bui, T. e Collomosse, J. (2017). Everything you wanted to know about deep learning for computer vision but were afraid to ask. Em *2017 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*, páginas 17–41. IEEE.

Robert Fisher, Simon Perkins, A. W. e Wolfart, E. (2003). Hypertext Image Processing Reference. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm`. Acessado em 02/04/2018.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sauvola, J. e Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern recognition*, 33(2):225–236.

Severo, E., Laroca, R., Bezerra, C. S., Zanlorensi, L. A., Weingaertner, D., Moreira, G. e Menotti, D. (2018). A benchmark for iris location and a deep learning detector evaluation. Em *International Joint Conference on Neural Networks (IJCNN)*, páginas 1–7.

Smith, R. (1995). A simple and efficient skew detection algorithm via text row accumulation. Em *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, páginas 1145–1148. IEEE.

Smith, R. (2007). An overview of the Tesseract OCR engine. Em *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, páginas 629–633. IEEE.

Smith, R., Antonova, D. e Lee, D.-S. (2009). Adapting the tesseract open source ocr engine for multilingual ocr. Em *Proceedings of the International Workshop on Multilingual OCR*, página 1. ACM.

Smith, R. W. (1987). *The Extraction and Recognition of Text from Multimedia Document Images*. Tese de doutorado, University of Bristol.

Smith, R. W. (2009). Hybrid page layout analysis via tab-stop detection. Em *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, páginas 241–245. IEEE.

Smith, R. W. (2013). History of the tesseract ocr engine: what worked and what didn't. Em *Document Recognition and Retrieval XX*, volume 8658, página 865802. International Society for Optics and Photonics.

Soua, M., Benchekroun, A., Kachouri, R. e Akil, M. (2017). Real-time text extraction based on the page layout analysis system. Em *Real-Time Image and Video Processing 2017*, volume 10223, página 1022305. International Society for Optics and Photonics.

Sun, H.-M. (2005). Page segmentation for Manhattan and non-Manhattan layout documents via selective CRLA. Em *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, páginas 116–120. IEEE.

S.V. Rice, F.R. Jenkins, T. N. (1995). The fourth annual test of ocr accuracy. Relatório Técnico Technical Report 95-03, Information Science Research Institute, University of Nevada, Las Vegas.

Tran, T. A., Na, I. S. e Kim, S. H. (2016). Page segmentation using minimum homogeneity algorithm and adaptive mathematical morphology. *International Journal on Document Analysis and Recognition (IJDAR)*, 19(3):191–209.

Tran, T. A., Oh, K., Na, I.-S., Lee, G.-S., Yang, H.-J. e Kim, S.-H. (2017). A robust system for document layout analysis using multilevel homogeneity structure. *Expert Systems With Applications*, 85:99–113.

Wahl, F. M., Wong, K. Y. e Casey, R. G. (1982). Block segmentation and text extraction in mixed text/image documents. *Computer graphics and image processing*, 20(4):375–390.

Wick, C. e Puppe, F. (2018). Fully convolutional neural networks for page segmentation of historical document images. Em *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, páginas 287–292. IEEE.

Xu, Y., He, W., Yin, F. e Liu, C.-L. (2017). Page segmentation for historical handwritten documents using fully convolutional networks. Em *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, páginas 541–546. IEEE.

Yousefi, M. R. e Breuel, T. M. (2012). Local logistic classifiers for large scale learning.

Zanlorensi, L. A., Luz, E., Laroca, R., Britto Jr., A. S., Oliveira, L. S. e Menotti, D. (2018). The impact of preprocessing on deep representations for iris recognition on unconstrained environments. Em *Conference on Graphics, Patterns and Images (SIBGRAPI)*, páginas 289–296.

Zucker, M. (2016). Page dewarping. `https://mzucker.github.io/2016/08/15/page-dewarping.html`. Acessado em 03/05/2018.