Segmentação de Documentos Jurídicos usando Supervisão Fraca

Marlon Daltro Tosta

SERVIÇO DE PÓS-GRADUAÇÃO DA FACOM-UFMS

Data de Depósito:

Assinatura:_____

Marlon Daltro Tosta

Orientador: *Prof. Dr. Eraldo Luís Rezende Fernandes*

Dissertação apresentada à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul como parte dos requisitos necessários à obtenção do título de Mestre em Ciência da Computação.

UFMS - Campo Grande
Março/2023

Abstract

Millions of cases are currently being processed in the Brazilian judicial system. The court decisions, known as *acórdãos*, are collective decisions made by Brazilian courts and are of high relevance in ensuring a unified understanding among judges and different courts. Therefore, developing and implementing effective technological solutions to assist judges judges, appellate and other professionals involved in the judicial process to cope with the growing volume of court cases in Brazil. These solutions should be able to speed up decision making and reduce workload ensuring the efficiency of the judicial system and the satisfaction of citizens who depend on it. The judgments of Brazilian courts are publicly available, However, as these documents are not in a structured format their automatic processing is hampered. However, the lack of structured format in which these documents are available makes their automatic processing challenging. This work collected over 960,000 PDF-format *acórdãos* documents from five Brazilian courts and used available tools to extract textual content and layout characteristics from 624,161 of them. An automatic annotation method was used to segment the documents into five mandatory segments of *acórdãos*. A total of 500 documents were manually annotated and they were used as validation and test sets for machine learning models trained on weakly annotated data. Classic and deep learning-based machine learning models were evaluated, with deep learning models outperforming traditional algorithms. Additionally, models that used both textual content and layout information achieved even better results. Models trained and tested on the same court tend to perform comparably or even better than automatic annotation methods, while performance for models trained on one court and tested on another depends on the correlation between the courts. Models trained on judgments from four courts and validated on a fifth achieved even better performance, with an average F1 above 90% in the best models. General segmentation models showed a trend of improving performance as the variety of layouts in the training data increased, suggesting that expan-

ding the variety of courts in the training data can lead to satisfactory practical performance. In this work, several resources that can be used in future work have been made available. All collected documents in PDF format, as well as the corresponding TSV and JSON files with automatic annotations, are freely available. The automatic segmentation scripts are also available, as are the scripts used for model training and evaluation. Finally, the manually reviewed annotations of 500 documents (100 from each court) are also available.

Resumo

Milhões de processos estão em tramitação no sistema judiciário brasileiro. Os acórdãos são decisões colegiadas de tribunais brasileiros e, por conta disto, são fundamentais na uniformização do entendimento entre os magistrados e entre diferentes tribunais.

Portanto, desenvolver e implementar soluções tecnológicas eficazes para auxiliar juízes, desembargadores e outros profissionais envolvidos no processo judicial a lidar com o crescente volume de processos judiciais no Brasil. Essas soluções devem ser capazes de acelerar a tomada de decisões e reduzir a carga de trabalho, garantindo a eficiência do sistema judiciário e a satisfação dos cidadãos que dependem dele. Os acórdãos dos tribunais brasileiros estão disponíveis publicamente, entretanto, como estes documentos não estão em formato estruturado, o processamento automático deles é dificultado. Um dos primeiros passos para o processamento de acórdãos é a segmentação destes documentos que são compostos por diversas seções com diferentes informações sobre a decisão. Neste trabalho, foram coletados mais de 960 mil documentos contendo o inteiro teor de acórdãos de cinco tribunais de diferentes esferas do judiciário brasileiro. Estes documentos estão em formato PDF e foram coletados nos portais de busca dos tribunais. Após a coleta, o conteúdo textual e características de layout de 624.161 acórdãos foram extraídos utilizando diferentes ferramentas. Adicionalmente, um método de segmentação automática foi desenvolvido para cada tribunal e todos os documentos foram segmentados em cinco seções obrigatórias em acórdãos de tribunais brasileiros. A segmentação de 100 documentos de cada tribunal (500 no total) foi manualmente revisada para servirem como validação e teste de modelos de Aprendizado de Máquina (AM). Modelos de AM clássicos e baseados em aprendizado profundo foram avaliados utilizando estes datasets. Todos os modelos foram treinados usando os dados anotados automaticamente (supervisão fraca). Diversos experimentos foram realizados para avaliar diferentes aspectos do problema de segmentação de acórdãos. Particularmente, um as-

pecto analisado foi a capacidade de generalização dos modelos para tribunais não vistos no treinamento. Outro aspecto considerado nos experimentos foi o impacto de características de layout dos documentos em alguns modelos de AM. Mais especificamente, foram analisados modelos baseados em aprendizado profundo, propostos recentemente na literatura, que consideram como entrada tanto o conteúdo textual quanto algumas características de layout. Modelos de segmentação com incorporação de layout alcançam, em alguns casos, desempenho superior aos métodos desenvolvidos especificamente para um tribunal. Modelos que são treinados e testados no mesmo tribunal também tendem a apresentar um desempenho comparável ou até superior aos métodos de anotação automática. No entanto, quando se trata de modelos treinados em um tribunal e testados em outro, o desempenho depende da correlação encontrada entre eles. Em alguns casos, a perda de desempenho é substancial. Neste trabalho, são disponibilizados diversos recursos que podem ser usados em trabalhos futuros. Todos os documentos coletados em formato PDF, assim como os correspondentes arquivos TSV e JSON com as anotações automáticas, estão disponíveis livremente. Também estão disponíveis os scripts de segmentação automática, assim como os scripts usados para treinamento e avaliação de modelos. Por fim, também estão disponíveis as anotações revisadas manualmente de 500 documentos (100 de cada tribunal).

Palavras-Chave Aprendizado de Máquina, Processamento de Linguagem Natural, Segmentação de Documento Jurídico, Direito Digital

Sumário

Sumário vi
Lista de Figuras viii
Lista de Tabelas x
Lista de Abreviaturas x

1 Introdução **3**
1.1 Layout e Estrutura de um Documento 5
1.2 Variabilidade de Layouts nos Acórdãos Brasileiros 5
1.3 Ferramentas e Métodos Existentes 6
1.4 Segmentação de Documentos Jurídicos usando Supervisão Fraca 8
1.5 Objetivos 8
1.6 Organização do Texto 10

2 Métodos e Ferramentas **12**
2.1 Coleta de Documentos 12
2.1.1 Tratamento e Manipulação do HTML de Resposta 13
2.2 Extração de Texto de Arquivos PDF 14
2.2.1 Características do Formato PDF 14
2.2.2 Análise e Extração de Objetos no Arquivo PDF 16
2.3 Armazenamento de Dados 18
2.4 Redes Neurais Artificiais 18
2.4.1 Treinamento de uma Rede Neural Artificial 22
2.4.2 Modelos de Representação de Linguagem 24
2.4.3 RNNs e Suas Aplicações em Dados Sequenciais 24
2.4.4 Embedding de Palavras 26
2.4.5 Mecanismos de Atenção 26
2.4.6 Transformers 28
2.4.7 BERT 29
2.5 Abordagem Híbrida: Texto e Layout 31
2.5.1 LayoutLMv3 32

3 Datasets de Segmentação **34**
3.1 Tribunais Considerados 34
3.2 Coleta de PDFs de Acórdãos 47
3.3 Extração de Texto e Layout a partir de um PDF 53
3.4 Anotação Automática 56
3.5 Anotação Manual 58
3.6 Estatísticas dos Datasets 58

4 Trabalhos Relacionados **62**
4.1 Segmentação de Documentos Jurídicos 62
4.2 Modelos Profundos de Linguagem com Incorporação de Layout 2D . . . 63
4.3 Modelos Profundos de Linguagem com Incorporação de Imagem 64
4.4 Modelos profundos de Linguagem com Incorporação de Layout 2D e Imagem 65

5 Experimentos **67**
5.1 Métricas de Avaliação 68
5.2 Modelos de Referência 69
5.3 Experimentos Intra-Tribunal 71
5.4 Experimentos Inter-Tribunal 73
5.4.1 Experimentos um-contra-um 73
5.4.2 Experimentos todos-contra-um 75

6 Conclusão **77**

Referências **84**

Lista de Figuras

1.1 Exemplo de um inteiro teor de um acórdão cujos segmentos de interesse estão marcados com diferentes cores. Os segmentos em destaque são: partes (verde), relatório (amarelo), voto (lilás), ementa (laranja) e acórdão (azul). Partes do conteúdo foram removidas para melhorar a exibição. . 4

2.1 A estrutura de contêineres do *pdfplumber* com os tipos de objetos aninhados 17
2.2 A figura mostra um problema comum na identificação da ordem de leitura de um PDF pelo *pdfplumber*, quando há uma confusão entre tabelas e colunas de leitura 18
2.3 Representação de um neurônio artificial com suas operações 19
2.4 Função de degrau unitário. 20
2.5 Funções de transferência não lineares sigmoid e suas derivadas. 21
2.6 Funções de unidade linear retificada (ReLU) e suas funções derivadas. . 21
2.7 Uma rede neural artificial com uma camada oculta 22
2.8 Rede neural artificial recorrente. 23
2.9 Arquitetura de predição de uma RNN profunda. Baseado em Graves (2013) 25
2.10 Demonstração de um grupo de palavras e sua representação de recursos 26
2.11 Mecanismos de atenção 28
2.12 Arquitetura do modelo *Transformer* (Vaswani et al., 2017) 30
2.13 Os *embeddings* de entrada são a soma dos *embeddings* de *token*, de segmento e de sequência 31
2.14 A arquitetura do LayoutLMv3 e suas tarefas de treino (Huang et al., 2022). 33

3.1 Sniffer de rede do protocolo HTTP nas ferramentas do desenvolvedor . . 47
3.2 Estrutura de um arquivo HTML simples 49
3.3 Exemplo de um inteiro teor resultante de uma pesquisa no portal de busca do STJ 51
3.4 Abstração do DOM de um documento na página de resposta no portal de busca do STJ 52
3.5 Exemplo de erros de agrupamento do PDFMiner em recortes de um PDF. O algoritmo identifica grupos de texto dentro de outro grupo maior. . . . 54

3.6 Distribuição das classes nos datasets por tribunal 61

5.1 Média F1 dos algoritmos validados em dados anotados manualmente (dataset VAL-M) por tribunal. 70

Lista de Tabelas

2.1 Parâmetros de estado do texto 15
2.2 Lista de atributos de metadados de processos do STJ e TRF2. 19

3.1 Características dos portais de busca de documentos dos tribunais superiores do Brasil 36
3.2 Características dos portais de busca de documentos dos Tribunais Regionais Federais do Brasil 38
3.3 Características dos portais de busca de jurisprudência dos tribunais regionais do trabalho 41
3.4 Características do portal de busca de jurisprudência dos Tribunais de Justiça 45
3.5 Quantitativo dos documentos nos *datasets* 60
3.6 Desempenho *F1-score* dos scripts por segmento com validação no *dataset* TESTE e desempenho geral *Precision*, *Recall* e *F1_micro* 60

5.1 Esquema dos dois tipos de erros: Falso Positivo e Falso Negativo 68
5.2 Resultados dos experimentos intra-tribunal. $F1$ dos modelos para cada classe e média micro. Os modelos foram treinados utilizando o TREINO MINI e validado utilizando o TESTE. 71
5.3 Desempenho geral do algoritmo LayoutLM em experimento intra-tribunal em comparação aos *scripts* de anotação automática. 73
5.4 Resultados dos experimentos inter-tribunal um-contra-um. Média micro de F1 entre todas as classes para os modelos LayoutLM em treino/teste em pares de tribunais diferentes. Observe que os resultados na diagonal (sublinhados) servem apenas como referência, já que não correspondem a modelos inter-tribunal, mas sim intra-tribunal (treinado e testado no mesmo tribunal). 74
5.5 Resultados do experimento inter-tribunal um contra um: mínimo, média e máximo $F_1 Micro$ para cada tribunal 74
5.6 Resultados dos experimentos inter-tribunal todos-contra-um. $F1$ dos modelos para cada classe e média $F1$. Os modelos foram treinados utilizando o TREINO MINI e validados no TESTE. 75

5.7 Comparação dos resultados ($F1_m$) entre as diferentes modalidades de experimentos. Importante salientar que, nos experimentos intra-tribunal e todos-contra-um, existe um único conjunto de teste por tribunal. Já no experimento um-contra-um, existem quatro conjuntos de teste para cada tribunal. Portanto, a coluna um-contra-um corresponde à média nos quatro conjuntos de teste de cada tribunal. 76

Lista de Abreviaturas

AM Aprendizado de Máquina

BERT Bidirectional Encoder Representations from Transformers

DOM Document Object Model

ISO International Organization for Standardization

JSON JavaScript Object Notation

LF Label Function

LSTM Long Short-Term Memory

MLM Masked Language Model

MLP Multilayer Perceptron

NLTK Natural Language Toolkit

NSP Next Sentence Prediction

PLN Processamento de Linguagem Natural

RNA Rede Neural Artificial

RNN Recurrent Neural Network

TF Transformation Function

STF Supremo Tribunal Federal

STJ Superior Tribunal de Justiça

STM Superior Tribunal Militar

TST Tribunal Superior do Trabalho

TSE Tribunal Superior Eleitoral

TRF1 Tribunal Regional Federal da 1ª Região

TRF2 Tribunal Regional Federal da 2ª Região

TRF3 Tribunal Regional Federal da 3ª Região

TRF4 Tribunal Regional Federal da 4ª Região

TRF5 Tribunal Regional Federal da 5ª Região

TJAC Tribunal de Justiça do Acre

TJAL Tribunal de Justiça de Alagoas

TJAP Tribunal de Justiça do Amapá

TJAM Tribunal de Justiça do Amazonas

TJBA Tribunal de Justiça da Bahia

TJCE Tribunal de Justiça do Ceará

TJDFT Tribunal de Justiça do Distrito Federal e Territórios

TJES Tribunal de Justiça do Espírito Santo

TJGO Tribunal de Justiça de Goiás

TJMA Tribunal de Justiça do Maranhão

TJMT Tribunal de Justiça de Mato Grosso

TJMS Tribunal de Justiça de Mato Grosso do Sul

TJMG Tribunal de Justiça de Minas Gerais

TJPA Tribunal de Justiça do Pará

TJPB Tribunal de Justiça da Paraíba

TJPR Tribunal de Justiça do Paraná

TJPE Tribunal de Justiça de Pernambuco

TJPI Tribunal de Justiça do Piauí

TJRJ Tribunal de Justiça do Rio de Janeiro

TJRN Tribunal de Justiça do Rio Grande do Norte

TJRS Tribunal de Justiça do Rio Grande do Sul

TJRO Tribunal de Justiça de Rondônia

TJRR Tribunal de Justiça de Roraima

TJSC Tribunal de Justiça de Santa Catarina

TJSP Tribunal de Justiça de São Paulo

TJSE Tribunal de Justiça de Sergipe

TJTO Tribunal de Justiça do Tocantins

TRE-AC Tribunal Regional Eleitoral do Acre

TRE-AL Tribunal Regional Eleitoral de Alagoas

TRE-AP Tribunal Regional Eleitoral do Amapá

TRE-AM Tribunal Regional Eleitoral do Amazonas

TRE-BA Tribunal Regional Eleitoral da Bahia

TRE-CE Tribunal Regional Eleitoral do Ceará

TRE-DF Tribunal Regional Eleitoral do Distrito Federal

TRE-ES Tribunal Regional Eleitoral do Espírito Santo

TRE-GO Tribunal Regional Eleitoral de Goiás

TRE-MA Tribunal Regional Eleitoral do Maranhão

TRE-MT Tribunal Regional Eleitoral do Mato Grosso

TRE-MS Tribunal Regional Eleitoral do Mato Grosso do Sul

TRE-MG Tribunal Regional Eleitoral de Minas Gerais

TRE-PA Tribunal Regional Eleitoral do Pará

TRE-PB Tribunal Regional Eleitoral da Paraíba

TRE-PR Tribunal Regional Eleitoral do Paraná

TRE-PE Tribunal Regional Eleitoral de Pernambuco

TRE-PI Tribunal Regional Eleitoral do Piauí

TRE-RJ Tribunal Regional Eleitoral do Rio de Janeiro

TRE-RN Tribunal Regional Eleitoral do Rio Grande do Norte

TRE-RS Tribunal Regional Eleitoral do Rio Grande do Sul

TRE-RO Tribunal Regional Eleitoral de Rondônia

TRE-RR Tribunal Regional Eleitoral de Roraima

TRE-SC Tribunal Regional Eleitoral de Santa Catarina

TRE-SP Tribunal Regional Eleitoral de São Paulo

TRE-SE Tribunal Regional Eleitoral de Sergipe

TRE-TO Tribunal Regional Eleitoral de Tocantins

TRT1 Tribunal Regional do Trabalho da 1ª Região

TRT2 Tribunal Regional do Trabalho da 2ª Região

TRT3 Tribunal Regional do Trabalho da 3ª Região

TRT4 Tribunal Regional do Trabalho da 4ª Região

TRT5 Tribunal Regional do Trabalho da 5ª Região

TRT6 Tribunal Regional do Trabalho da 6ª Região

TRT7 Tribunal Regional do Trabalho da 7ª Região

TRT8 Tribunal Regional do Trabalho da 8ª Região

TRT9 Tribunal Regional do Trabalho da 9ª Região

TRT10 Tribunal Regional do Trabalho da 10ª Região

TRT11 Tribunal Regional do Trabalho da 11ª Região

TRT12 Tribunal Regional do Trabalho da 12ª Região

TRT13 Tribunal Regional do Trabalho da 13ª Região

TRT14 Tribunal Regional do Trabalho da 14ª Região

TRT15 Tribunal Regional do Trabalho da 15ª Região

TRT16 Tribunal Regional do Trabalho da 16ª Região

TRT17 Tribunal Regional do Trabalho da 17ª Região

TRT18 Tribunal Regional do Trabalho da 18ª Região

TRT19 Tribunal Regional do Trabalho da 19ª Região

TRT20 Tribunal Regional do Trabalho da 20ª Região

TRT21 Tribunal Regional do Trabalho da 21ª Região

TRT22 Tribunal Regional do Trabalho da 22ª Região

TRT23 Tribunal Regional do Trabalho da 23ª Região

TRT24 Tribunal Regional do Trabalho da 24ª Região

TJMMG Tribunal de Justiça Militar de Minas Gerais

TJMRS Tribunal de Justiça Militar do Rio Grande do Sul

TJMSP Tribunal de Justiça Militar de São Paulo

fa

CAPÍTULO

1

Introdução

Em relatório anual publicado pelo Conselho Nacional de Justiça (CNJ, 2022), os dados referentes ao ano de 2021 apontaram um total de 77,3 milhões de processos em tramitação distribuídos entre 18.035 magistrados. Uma média de mais de 4.200 processos por magistrado. Esse número demonstra a magnitude do sistema judiciário brasileiro e o desafio de tratar esses dados manualmente. Nesse contexto, criar mecanismos automatizados para facilitar o acesso e o processamento dos documentos de um processo jurídico é altamente relevante e pode ter impacto significativo na área de processamento de documentos jurídicos, além de contribuir para o desenvolvimento de soluções tecnológicas para lidar com a demanda crescente por processos judiciais.

Para todo processo da justiça brasileira, exceto aqueles aos quais a constituição brasileira garante o direito à intimidade do interessado, a decisão proferida (sentença) deve ser publicada integralmente, acompanhada do relatório e de outras informações que completam o entendimento da sentença. Um *acórdão* é uma sentença proferida por uma turma de juízes de um tribunal, também chamada de sentença colegiada (Brasil, 2015). O documento onde o acórdão e outras informações relevantes são condensados é chamado *inteiro teor*. Na Figura 1.1, é exemplificado o inteiro teor de um acórdão com quatro seções. Neste trabalho, usaremos os termos acórdão e inteiro teor (de um acórdão) de maneira indistinta para nos referirmos ao documento contendo o inteiro teor de um acórdão.

Dada a importância do inteiro teor, é relevante desenvolver ferramentas de processamento automático deste tipo de documento. Um dos primeiros passos nesta direção consiste em identificar e classificar seções de interesse neste documento, como ilustrado na Figura 1.1, dado que cada seção inclui um tipo diferente de informação. Neste trabalho, uma seção de interesse dentro da estrutura de um documento é denominada *segmento*. A tarefa de identificar e classificar todos os segmentos de um

Poder Judiciário
TRIBUNAL REGIONAL FEDERAL DA 4ª REGIÃO
Rua Otávio Francisco Caruso da Rocha, 300, 7º andar - Bairro: Praia de Belas - CEP: 90010-035 - Fone: (51) 3213-3533 - Email: gloraci@trf4.jus.br

APELAÇÃO CRIMINAL Nº 5016592-86.2018.4.04.7200/SC

RELATOR: JUIZ FEDERAL LORACI FLORES DE LIMA
APELANTE: ADRIANA FARIAS (RÉU)
APELADO: MINISTÉRIO PÚBLICO FEDERAL (AUTOR)

RELATÓRIO

Trata-se de apelação criminal interposta por ADRIANA FARIAS em face de sentença proferida pelo Juízo Substituto da 7ª Vara Federal de Florianópolis que absolveu sumariamente a ré, revogou a suspensão condicional do processo em razão da manifestação de vontade da requerente e determinou a restituição de apenas uma parcela do benefício, quitada após a data do trânsito em julgado da absolvição proferida em favor dos codenunciados na ação penal originária.

É o relatório. À revisão.

VOTO

Trata-se de apelação criminal interposta por ADRIANA FARIAS em face de sentença proferida pelo Juízo Substituto da 7ª Vara Federal de Florianópolis que absolveu sumariamente a ré, revogou a suspensão condicional do processo em razão da manifestação de vontade da requerente e determinou a restituição de apenas uma parcela do benefício, quitada após a data do trânsito em julgado da absolvição proferida em favor dos codenunciados na ação penal originária.

Sendo assim, descabido o pedido.

Ante o exposto, voto por negar provimento à apelação.

EMENTA

PENAL. PROCESSO PENAL. APELAÇÃO CRIMINAL. PEDIDO DE DEVOLUÇÃO DOS VALORES PAGOS NA SUSPENSÃO CONDICIONAL DO PROCESSO. IMPOSSIBILIDADE.

1. O benefício da suspensão condicional do processo tem caráter meramente processual, não se vinculando ao mérito da decisão. Os valores despendidos no curso da benesse têm caráter de doação, não havendo que se falar em devolução.

2. Apelação criminal desprovida.

ACÓRDÃO

Vistos e relatados estes autos em que são partes as acima indicadas, a Egrégia 8ª Turma do Tribunal Regional Federal da 4ª Região decidiu, por unanimidade, negar provimento à apelação, nos termos do relatório, votos e notas de julgamento que ficam fazendo parte integrante do presente julgado.

Porto Alegre, 07 de novembro de 2022.

Documento eletrônico assinado por **LORACI FLORES DE LIMA, Juiz Federal Convocado**, na forma do artigo 1º, inciso III, da Lei 11.419, de 19 de dezembro de 2006 e Resolução TRF 4ª Região nº 17, de 26 de março de 2010. A conferência da **autenticidade do documento** está disponível no endereço eletrônico http://www.trf4.jus.br/trf4/processos/verifica.php, mediante o preenchimento do código verificador **40003523573v4** e do código CRC **691de8ba**.

Informações adicionais da assinatura:
Signatário (a): LORACI FLORES DE LIMA
Data e Hora: 7/11/2022, às 17:44:35

Conferência de autenticidade emitida em 08/11/2022 14:49:26.

Figura 1.1: Exemplo de um inteiro teor de um acórdão cujos segmentos de interesse estão marcados com diferentes cores. Os segmentos em destaque são: partes (verde), relatório (amarelo), voto (lilás), ementa (laranja) e acórdão (azul). Partes do conteúdo foram removidas para melhorar a exibição.

documento é denominado *segmentação*. Na Figura 1.1, é apresentado um exemplo de segmentação de um trecho de um acórdão do Superior Tribunal de Justiça. Cada cor na figura indica um segmento e as partes sem destaque (particularmente, cabeçalho e rodapé) não são de interesse pois contêm informações irrelevantes neste caso (nome do tribunal, identificação do processo, número da páginas, dentre outros).

A segmentação de acórdãos de tribunais brasileiros é o tema central deste trabalho. Mas antes de descrever a proposta completa, faz-se necessário apresentar alguns aspectos importantes deste tema central.

1.1 Layout e Estrutura de um Documento

Os tribunais de justiça brasileiros, em sua maioria, utilizam o formato PDF (*Portable Document Format*) para divulgar os inteiros teores dos acórdãos. Desde que foi concebido, em 1993, o formato de representação de documentos digitais PDF vem evoluindo e se tornou um padrão de fato. Uma de suas principais características é a fidelidade de exibição de um documento independente de dispositivo, software e plataforma (ISO 32000-1:2008, 2008). Sendo que esta independência de plataforma engloba tanto os sistemas usados para criar quanto aqueles usados para exibir ou imprimir esses documentos. O PDF pode ser visto, então, como uma linguagem para descrever a aparência gráfica exata das páginas de um documento. A maneira como os elementos (textos, figuras, tabelas, dentre outros) de um documento são dispostos em uma página é denominada *layout*. Portanto, o PDF é uma linguagem que define o layout das páginas de um documento.

Apesar das claras vantagens do formato PDF para compartilhamento e exibição, existem também algumas desvantagens desta tecnologia quando associada à manipulação de texto. Um dos principais problemas, como na maioria dos formatos utilizados para textos, é a ausência de informação explícita sobre a estrutura do conteúdo do documento, ao contrário do ocorre em formatos como XML e HTML. A estrutura do conteúdo define, em diferentes níveis, como o conteúdo de um documento é organizado. Isto envolve, por exemplo, a organização de um documento em capítulos, seções, sub-seções, parágrafos, frases e até palavras. Esta estrutura também pode englobar outros elementos de um documento, como figuras, tabelas, listas e colunas. Daqui em diante, o termo *estrutura do documento* será usado para referenciar este conceito. Sabendo que o inteiro teor dos acórdãos divulgados em formato PDF pelos tribunais brasileiros não incluem a estrutura dos documentos, a aplicação de ferramentas de processamento automático é dificultada, pois geralmente é necessário segmentar estes documentos antes de aplicar outros tipos de processamento.

1.2 Variabilidade de Layouts nos Acórdãos Brasileiros

O sistema judiciário brasileiro é composto por 90 tribunais: 4 Tribunais Superiores; 5 Tribunais Regionais Federais; 24 Tribunais Regionais do Trabalho; 27 Tribu-

nais Regionais Eleitorais; 3 Tribunais de Justiça Militar Estaduais; e 27 Tribunais de Justiça. Cada tribunal mantém pelo menos um layout próprio para publicação de seus documentos. Isto acarreta uma grande quantidade de documentos com layouts diferentes, e até tipos de segmentos variados. Essa variabilidade de layouts levanta a questão se é possível desenvolver um único método de segmentação capaz de lidar com uma grande quantidade de layouts diferentes. Pois, em princípio, cada layout pode demandar o desenvolvimento e, sobretudo, a manutenção de um *script* específico.

Algoritmos que realizam a tarefa de segmentação de documentos usando regras específicas para um domínio são fortemente sensíveis a variações no layout e na estrutura do documento, exigindo ajustes sempre que ocorre uma alteração nestes elementos. Além disso, a construção manual de regras de segmentação demanda um conhecimento que somente pode ser obtido após análise manual de um volume representativo de documentos. E, ainda que se tenha uma definição de tais padrões, o desenvolvimento de funções de segmentação é custoso e, geralmente, o código resultante é de difícil manutenção. Isto ocorre, particularmente, porque este tipo de código envolve uso extensivo de recursos como condicionais aninhadas, expressões regulares e busca por padrões específicos de um tribunal ou, até mesmo, de um magistrado.

Considerando o exposto aqui, esta última alternativa é bastante custosa. Realizar a segmentação de acórdãos de diferentes tribunais é uma tarefa desafiadora, sobretudo pela escassez de documentos segmentados que parametrizem os estudos nessa área ou sirvam como base para desenvolvimento de *scripts* e treinamento de modelos de AM. Neste trabalho, esta questão (possibilidade de um método de segmentação para diferentes tribunais) é considerada e atacada através da criação e disponibilização de um *dataset* de acórdãos de diferentes tribunais do judiciário brasileiro anotados com informações de segmentação.

1.3 Ferramentas e Métodos Existentes

Existem ferramentas para extração do texto de um PDF como, por exemplo, PDFMiner (Shinyama, 2019), QPDF (Berkenbilt, 2021) e XPDFReader (Glyph & Cog, 2021). Entretanto, estas ferramentas não realizam a segmentação completa dos documentos. Elas se restringem a identificar as estruturas mais básicas, como palavras, linhas, parágrafos e colunas. E mesmo para essas estruturas básicas, estas ferramentas falham para alguns layouts. Neste trabalho, a ferramenta pdfplumber (Singer-Vine and Jain, 2023) é usada para extrair de um PDF as estruturas de palavras e linhas. Esta ferramenta possui uma ótima acurácia para estas duas estruturas em diferentes layouts.

Uma grande variedade de estudos recentes tem produzido métodos para segmentação de PDF baseados em Aprendizado de Máquina (AM). O Corpus Conversion Service (CCS) (Staar et al., 2018) implementa um pipeline que permite que usuários analisem e anotem documentos, treinem modelos de AM, convertendo um documento PDF em uma representação estruturada do conteúdo. Neste trabalho, o layout das páginas

(*bitmap* da página do documento e posicionamento dos objetos) também é considerado para extrair informações como título, autores, resumo, tabelas, entre outros rótulos relacionados a artigos de literatura científica ou técnica. Métodos como CERMINE (Tkaczyk et al., 2015), por sua vez, usam informações internas sobre os caracteres do PDF (por exemplo, a posição exata de caracteres e palavras na página) como representação do layout do PDF de um artigo científico nato digital. (Hao et al., 2016) processam uma imagem da página do PDF na tentativa de identificar e capturar tabelas com sua posição delimitada.

Os trabalhos mencionados acima (Staar et al., 2018; Tkaczyk et al., 2015; Hao et al., 2016) utilizam não apenas o conteúdo textual do documento, mas também suas características geométricas relacionadas ao modo como o texto é exibido no arquivo PDF de origem (layout), baseando-se principalmente em aprendizado de máquina, o que aumenta sua capacidade de generalização para diferentes layouts. Embora sejam capazes de extrair metadados relacionadas ao documento, como nome do periódico, volume, emissão ou intervalo de páginas, não é usado, a princípio, para segmentar um documento baseando-se na compreensão de seu conteúdo.

Métodos ainda mais recentes, como VILA (Shen et al., 2022) e LayoutLMv3 (Huang et al., 2022), não apenas exploram as duas visões de um documento (texto e layout), mas o fazem de uma maneira holística, integrando as duas visões explicitamente na modelagem do problema através do uso de arquiteturas de redes neurais profundas como *Transformers* (Vaswani et al., 2017). Combinar informação textual e de layout para segmentação de documentos jurídicos usando aprendizado de máquina é uma ideia promissora e em constante evolução recentemente.

Todos os métodos e ferramentas citados acima foram aplicados em documentos exclusivamente em língua inglesa, treinados e testados em documentos de literatura científica e técnica. Apesar da relevância da tarefa de segmentação de documentos jurídicos, existe uma carência de bases contendo documentos segmentados deste tipo. Para a língua portuguesa, particularmente, o único recurso deste tipo é o corpus ITD (Sousa and Del Fabro, 2019) que contém mais de 40 mil acórdãos do Supremo Tribunal Federal (STF). Todos estes documentos foram segmentados automaticamente por um método criado manualmente que é baseado em expressões regulares e outros padrões simples. Este método é específico para o layout usado pelo STF na época do seu desenvolvimento. Para se ter uma ideia das suas limitações, durante a pandemia da COVID-19, o STF começou a fazer sessões virtuais para votação de acórdãos. Nestas sessões, o padrão de local e data usado no final de algumas seções dos acórdãos, que antes seguia o formato Brasília, 10 de março de 2018, foram alterados para Plenário, Sessão virtual de 18 de setembro de 2021. Com esta alteração, o método proposto por Sousa and Del Fabro já não reconhecia algumas seções dos documentos e precisou ser ajustado. Apesar do ITD ser uma importante contribuição, o método de segmentação utilizado possui limitações relevantes. Além disso, não foi realizada nenhuma anotação manual e, portanto, a qualidade da segmentação automática não foi avaliada sistematicamente.

1.4 Segmentação de Documentos Jurídicos usando Supervisão Fraca

A complexidade da tarefa de segmentação de um documento jurídico genérico depende da quantidade de layouts diferentes a serem considerados. O principal gargalo é construir um grande *dataset* para uso em treinamento de modelos de AM e aprendizado profundo. A supervisão fraca na anotação de textos é um forma de aliviar o custo de especialistas humanos em anotar uma grande quantidade de documentos, principalmente em um domínio específico. Ela substitui a rotulagem manual pela combinação de fontes ruidosas mas capazes de estimar rótulos a um baixo custo. Portanto, essa abordagem consiste em reunir as características ruidosas nos documentos, através de funções de anotação, e impor rótulos a dados ainda não rotulados, mas com qualidade suficiente para permitir seu uso para aprendizado de máquina.

Os métodos desenvolvidos neste trabalho extraem cada linha de um documento PDF e buscam por padrões que indiquem uma quebra de segmento, que podem se apresentar tanto no texto quanto no layout, possibilitando uma anotação mais precisa. Dessa forma é possível construir um grande volume de documentos segmentados que podem ser usados no treinamento de modelos de AM. Alguns documentos coletados não podem ter suas informações de texto e layout extraídas devido a problemas com o documento como arquivos digitalizados sem reconhecimento óptico de caracteres ou documentos protegidos. Como cada tribunal utiliza seu próprio layout, uma contribuição deste trabalho é analisar a dificuldade da tarefa de segmentação de diferentes tribunais usando um modelo único.

A proposta deste trabalho é criar *datasets* robustos e ricamente descritos contendo acórdãos de diferentes tribunais brasileiros de tal forma a subsidiar estudos e desenvolvimento de ferramentas de Processamento de Linguagem Natural para este domínio. Este trabalho também envolve o treinamento e validação de modelos de AM e aprendizado profundo para resolver a tarefa de segmentação de acórdãos. Os *datasets* são produzidos a partir de uma base de dados de documentos coletados dos portais dos tribunais, de onde são extraídos o conteúdo textual e também características de layout. e estruturadas em um corpus, cuja anotação e segmentação é feita utilizando supervisão fraca.

Todos os recursos desenvolvidos no decorrer deste trabalho estão disponíveis livremente.[1] Isto inclui os documentos coletados em formato PDF, os *datasets* de segmentação em formato TSV e JSON, as características de layout de todos os documentos, assim como algumas informações derivadas (estatísticas e exemplos).

1.5 Objetivos

A falta de padronização nos layouts dos documentos jurídicos pode dificultar o trabalho de processamento e análise desses documentos. A criação de uma base

[1] http://direitodigital.ufms.br:8000

de dados de acórdãos em formato PDF e a geração de *datasets* para segmentação e anotação automática dos mesmos, utilizando informações de layout, pode ajudar a solucionar esse problema. Isso pode ter impacto na área de processamento de documentos jurídicos em geral, bem como em outras áreas que lidam com a análise de documentos em formato PDF. Portanto, objetivo geral deste trabalho é fomentar a pesquisa e a inovação na área de processamento de documentos jurídicos, viabilizando o desenvolvimento de ferramentas que auxiliem juízes, desembargadores e demais envolvidos no trâmite processual, acelerando a tomada de decisões e reduzindo a sobrecarga de trabalho, o que beneficia tanto o sistema judiciário quanto os cidadãos que dependem dele. Com grande relevância prática este estudo pode ter impacto significativo na área de processamento de documentos jurídicos, além de contribuir para o desenvolvimento de soluções tecnológicas para lidar com a demanda crescente por processos judiciais. Considerando este objetivo geral, alguns objetivos específicos que se destacam são:

- *Base de dados de acórdãos em formato PDF.* Criação de uma base contendo o inteiro teor em formato PDF de acórdãos publicados por diferentes tribunais brasileiros. É importante considerar diferentes tribunais, que utilizam layouts diferentes, para permitir estudos sobre a semelhança entre estes layouts e também a viabilidade de desenvolver métodos de segmentação que funcionem bem para todos os tribunais. A construção é feita em três etapas. Primeiro é preciso definir os tribunais de coleta de acordo com a relevância, quantidade de documentos e facilidade da coleta automatizada. Segundo fazer a coleta dos metadados do acórdão e da url do arquivo PDF relacionado, através de *scripts* desenvolvidos especificamente para cada tribunal. Por último, fazer o download dos arquivos PDFs cujas urls já foram coletadas e armazena-los de maneira estruturada para fácil localização e navegação posterior.

- *Datasets de acórdãos com informações de layout.* Geração de *datasets* para segmentação de acórdãos a partir da base de dados descrita acima, contendo o texto e algumas informações de layout dos documentos. A extração de texto dos arquivos PDF da base de dados será realizada por ferramentas disponíveis. O objetivo é utilizar as funcionalidades destas ferramentas que têm boa acurácia para detectar estruturas básicas do texto como, por exemplo, palavras e linhas. Usando estas mesmas ferramentas, é trivial incluir informações de layout acerca das estruturas (basicamente sua posição na página). Atingir este objetivo é necessário para realizar a segmentação dos acórdãos, entretanto, os *datasets* produzidos não incluem ainda as anotações sobre os segmentos.

- *Segmentação e anotação automática (supervisão fraca).* Desenvolver métodos baseados em expressões regulares e outros padrões simples para segmentar automaticamente os acórdãos de um tribunal específico. O objetivo engloba criar um método específico para cada um dos tribunais contemplados na base de dados. A anotação (fraca) gerada nesta etapa será usada para treinamento de modelos de AM.

- *Revisão manual.* Revisar manualmente a segmentação e a anotação para um conjunto de acórdãos de cada tribunal. Estes conjuntos serão usados somente para validação e teste de modelos de AM, assim como dos métodos de segmentação e anotação automática mencionados acima.

- *Modelos de AM.* Treinar e avaliar modelos de AM para segmentação e anotação de acórdãos usando informações textuais e de layout. O objetivo envolve diferentes algoritmos de AM, tradicionais e modernos, com o intuito de analisar suas diferenças. Este objetivo também engloba avaliar diferentes conjuntos de treinamento e de validação, considerando situações com um único tribunal (treino e validação do mesmo tribunal) assim como vários tribunais (treino e validação com tribunais diferentes).

- *Avaliação dos resultados experimentais.* Através dos experimentos com modelos de AM mencionados acima, objetiva-se responder algumas perguntas, tais como:

  - *Intra-tribunal.* Modelos treinados e avaliados no mesmo tribunal. Estes resultados permitirão analisar quais tribunais são mais difíceis. Também permitirão compreender melhor a dificuldade das diferentes seções consideradas na segmentação.

  - *Inter-tribunal.* Modelos treinados e avaliados em tribunais diferentes. Estes resultados podem ser agrupados em dois grupos: "um-contra-um"e "todos-contra-um". No grupo "um-contra-um", um modelo treinado em um tribunal X é avaliado em dados de outro tribunal Y. Através destes resultados, será possível analisar as correlações entre os tribunais. No grupo "todos-contra-um", um modelo é treinado usando dados de todos os tribunais exceto o tribunal X e avaliado em dados somente do tribunal X. Com base nestes resultados, será possível analisar a capacidade de adaptação dos modelos para tribunais desconhecidos (não vistos durante o treinamento). Com isto, será possível responder à pergunta se um mesmo modelo pode ser usado para outros tribunais, mesmo para aqueles para os quais não existem dados anotados.

1.6 Organização do Texto

Nos capítulos seguintes serão apresentados os detalhes sobre o problema descrito, trabalhos anteriores e implementações, além da metodologia e avaliação dos resultados. No Capítulo 2 são apresentados os métodos e ferramentas utilizados para coleta, manipulação e anotação dos documentos em PDF, bem como os conceitos relacionados ao problema e a abordagem de incorporação de estruturas visuais de *layout* na segmentação e classificação de segmentos nos inteiros teores dos acórdãos em PDF. No Capítulo 3, o processo de construção de uma base de acórdãos segmentados dos tribunais brasileiros e suas etapas é demonstrado, desde a coleta, organização, extração até a produção do corpus e os *datasets* a partir dos inteiros teores. No

Capítulo 4 são discutidos trabalhos anteriores relacionados ao processamento de linguagem natural e modelos de linguagem profunda com abordagens úteis na tarefa de segmentação de documentos em geral que podem ser ajustadas ao campo de documentos jurídicos, bem como trabalhos relacionados com a segmentação baseada no conteúdo do documento. No Capítulo 5 são tratados os experimentos realizados para avaliar diferentes modelos de aprendizado de máquina, com foco em convergência e generalização, utilizando algoritmos tradicionais e modelos baseados em aprendizado profundo. Os experimentos foram divididos em três categorias e utilizaram métricas de avaliação específicas. Por fim, no Capítulo 6, é apresentada uma conclusão onde são apresentados os principais resultados da pesquisa, bem como a lista dos artefatos produzidos e sugestões de pesquisas futuras.

CAPÍTULO

2

Métodos e Ferramentas

Neste capítulo, serão descritos os métodos e as ferramentas usados neste trabalho para resolver o problema de segmentação de documentos PDF contendo o inteiro teor de acórdãos de tribunais brasileiros. Na Seção 2.1, é descrito o processo de coleta dos documentos nos sites dos tribunais e as ferramentas utilizadas neste processo. Na Seção 2.2, será apresentado o formato PDF, com suas características, seus recursos, especialmente como elementos textuais são organizados em um arquivo e como os tribunais, em geral, os aplicam em seus documentos. Também são expostos os problemas encontrados na extração de texto em arquivos PDF e as técnicas e ferramentas utilizadas para contorná-los. São expostos na Seção 2.3 os métodos adotados para armazenamento de dados coletados, especificamente para metadados dos acórdãos, arquivos coletados e dados extraídos e sua organização. Os conceitos fundamentais de redes neurais artificiais, aprendizados de máquina e representação de linguagem são apresentados na Seção 2.4. Por fim, na Seção 2.5, serão propostos métodos e modelos adaptáveis ou compatíveis com o problema de segmentação e classificação de segmentos de inteiros teores de acórdãos dos tribunais brasileiros. Ainda é discutida a estrutura hierárquica dos documentos, bem como a relação entre essa estrutura e o layout visual dos documentos, as técnicas de pré-treino auto supervisionado multimodal na análise do layout e as ferramentas e os métodos usados para incorporar as estruturas visuais do layout das linhas de texto em modelos de linguagem.

2.1 Coleta de Documentos

Como um dos objetivos é realizar o treinamento e a validação de modelos usando documentos com diferentes layouts, foram coletados arquivos de cinco tribunais brasileiros de tal forma que se possa utilizar os documentos dos diferentes tribunais para os experimentos intra-tribunal e inter-tribunal.

Os tribunais brasileiros, via de regra, devem publicar os inteiros teores dos acórdãos. Os portais dos tribunais na internet permitem, em sua maioria, localizar um determinado documento ou um conjunto de documentos por meio de filtros personalizados. No entanto, estes tribunais não disponibilizam APIs para acesso aos arquivos e nem todos permitem o livre download de documentos, implementando métodos em seus portais, como *captchas*, que limitam o número de arquivos baixados através de requisições automáticas.

O acesso a estes documentos é feito por meio formulários interativos, o que dificulta o download de uma grande quantidade de arquivos sem a utilização de ferramentas adicionais. Utilizando filtros específicos, é possível obter uma lista de acórdãos com referências para o documento PDF contendo o inteiro teor. Nesses casos é possível realizar estas consultas enviando requisições HTTP GET ou POST com os parâmetros desejados para o servidor de busca. Assim, coletar os arquivos de inteiro teor consiste em requisitar estes dados específicos no portal do tribunal. Para isto, ao solicitar o conteúdo da página, deve-se localizar o elemento dentro da página e desprezar o restante do conteúdo. Esse processo é executado em três etapas principais:

1. enviar a requisição ao servidor que hospeda o documento,
2. estruturar o HTML de resposta em um conjunto de objetos navegáveis, e
3. filtrar o conjunto de objetos de interesse, incluindo o link para download do PDF com o inteiro teor do acódão, e fazer o download do arquivo.

Os principais navegadores incluem um conjunto de ferramentas de análise da página web. Esse conjunto de ferramentas é chamado ferramentas do desenvolvedor e oferecem recursos como inspecionar elementos HTML, estilos CSS e *scripts* carregados pela página, assim como um *sniffer* do tráfego HTTP entre cliente e servidor. Dentro deste pacote de ferramentas, o inspetor de elementos é um explorador *Document Object Model* (DOM) para elementos HTML e estilos CSS. Ele perimite visualizar o resultado do HTML processado e os estilos CSS de cada elemento da árvore. O inspetor de rede é um *sniffer* que permite monitorar todo o tráfego HTTP, o que possibilita visualizar quais informações são trocadas entre cliente e servidor e, assim, sabendo como e quais dados trafegam, é possível reproduzir uma requisição desejada através de um *script*. O conjunto de informações úteis (sem metadados ou controles do protocolo) trocadas entre cliente e servidor é chamado de *payload*.

Portanto, para obter o conteúdo da página onde o download está disponível, é preciso acompanhar como ocorre a navegação na página de cada tribunal de interesse até o download do arquivo e programar a interação no site para possibilitar o download do conjunto de arquivos desejados.

2.1.1 Tratamento e Manipulação do HTML de Resposta

A biblioteca `requests`[1] do Python possibilita o envio de requisições HTTP de maneira simples. Os parâmetros de consulta podem ser adicionados facilmente às requi-

[1] https://docs.python-requests.org/

sições e o estado da conexão é mantido automaticamente. Uma vez que o conteúdo do HTML é obtido, ele pode ser analisado através do DOM, que representa este conteúdo como uma árvore.

A biblioteca `html.parser`[2] do Python é utilizada para fazer a análise de arquivos de texto no formato HTML. Os dados HTML são analisados e o tratamento é executado com base nas tags de início, tags de finalização, texto, comentários e demais elementos de marcação, formando a árvore de análise.

A biblioteca *Beautiful Soup* (Richardson, 2007) permite navegar, pesquisar e modificar um objeto DOM de forma simples. Em conjunto com o `html.parser`, ela converte um documento HTML complexo em uma árvore navegável de objetos Python. Assim, é possível separar os objetos de interesse dos demais. Também lida automaticamente com a codificação de caracteres.

2.2 Extração de Texto de Arquivos PDF

Um documento PDF é formado por um conjunto de objetos que, organizados em uma sequência de páginas, determinam sua aparência. Como todos estes elementos são representados como uma única sequência de bytes e não há informação explícita da organização desse objetos para recuperação direta e coesa dos objetos de interesse, é necessário compreender como esses objetos são organizados e como concatenar de maneira ordenada os objetos. Na Seção 2.2.1 são enumeradas as principais características de um arquivo PDF e sua construção, enquanto na Seção 2.2.2 são tratados os aspectos da extração de texto em documentos PDF e as ferramentas utilizadas, incluindo seus métodos de extração e limitações.

2.2.1 Características do Formato PDF

Considerando, por exemplo, os elementos textuais em um documento PDF, cada caractere do texto é representado como um objeto denominado *glifo*. As formas de glifo dos caracteres são descritas em uma estrutura de dados separada chamada *fonte*. Um caractere é um símbolo abstrato e um glifo, por sua vez, é uma representação gráfica específica de um caractere. Cada glifo está sujeito a todas as manipulações gráficas como transformação de coordenadas, recurso capaz de executar operações sobre glifos como mover, redimensionar, girar ou distorcer. O caractere "Y", por exemplo, pode ser representado de inúmeras formas diferentes. Os glifos *Y* e **Y** são exemplos das representações do caractere em itálico e negrito respectivamente. Uma fonte define os glifos desejados para um conjunto de caracteres específicos. Por exemplo, a fonte *Baskerville* define glifos para um conjunto de caracteres do alfabeto latino, enquanto a fonte *Noto Sans SC* define glifos para idiomas na China continental.

Um glifo, para que seja mostrado em uma página, engloba três conceitos básicos:

- *Estado do texto* é um conjunto de parâmetros que definem o estado gráfico do

[2] https://docs.python.org/3/library/html.parser.html

texto, incluindo parâmetros que definem a fonte, dimensionam os glifos para um tamanho apropriado e realizam outros efeitos gráficos sobre ele.

- *Objetos de texto e operadores* especificam os glifos a serem exibidos, representados por *strings* interpretadas como uma sequências de códigos de caracteres. Um objeto de texto inclui uma sequência de operadores de texto e seus parâmetros.

- *Estruturas de dados de fontes* são dicionários de fontes e estruturas de dados associadas que fornecem as informações que um aplicativo leitor do arquivo precisa para interpretar o texto e posicionar cada glifo. As definições dos próprios glifos devem estar contidas em programas de fontes, que podem ser incorporados ao arquivo PDF, incorporados ao aplicativo leitor ou obtidos a partir de um arquivo de fonte externo.

Na Tabela 2.1, é apresentada a lista de parâmetros de estado do texto e suas funções.

Parâmetro	Descrição
T_c	Espaço entre caracteres
T_w	Espaço entre palavras
T_h	Escala horizontal
T_l	Distancia Vertical entre linhas (leading)
T_f	Fonte do texto
T_{fs}	Tamanho da fonte
T_{mode}	Modo de renderização (preenchimento, borda e recorte do glifo)
T_{rise}	Distância do deslocamento para sobrescrita ou subscrita
T_k	Flag que determina se o glifo em um objeto de texto é tratado como um objeto elementar separado podendo, inclusive, sobrepor outro glifo (knokout);

Tabela 2.1: Parâmetros de estado do texto

Objetos de texto e operadores são formados por glifos definidos nas estrutura de dados das fontes e exibidos em uma página de um PDF dadas as características do estado do texto. Essas características, aplicáveis em cada glifo, torna a extração coesa de texto em um documento PDF uma tarefa difícil, já que não há aspectos da estrutura do texto nem mesmo da ordem de leitura.

O formato PDF permite a incorporação de algumas informações sobre a estrutura de um documento. Esta estrutura é armazenada separadamente do layout do documento mas com referências para a posição onde o conteúdo é exibido, o que pode simplificar muito a segmentação de um documento PDF. Entretanto, a maioria dos tribunais brasileiros não faz uso deste recurso para produzir o inteiro teor de seus acórdãos.

2.2.2 Análise e Extração de Objetos no Arquivo PDF

A partir do conhecimento exposto acima, conclui-se que a extração de texto em documentos PDF é um desafio peculiar, levando-se em conta que não há informação explícita da estrutura do documento no PDF. Este formato não permite fazer uma extração de textos, de forma coesa, diretamente. Para isso, é necessário aplicar métodos que concatenem os caracteres, as palavras, as frases e até os parágrafos, tornando o texto coeso.

A ferramenta PDFPlumber realiza a extração de objetos em documentos PDF, incluindo os objetos de texto. Ela é capaz de extrair todos os textos que são renderizados de forma programática, assim como metadados associados a cada parte do texto. Estes metadados incluem dados globais do documento (número de ordem, tamanho das páginas), informações sobre o layout (fonte e seu tamanho) e de posição (*bounding box*[3] e *graphic state*[4]). Para representar a posição de um ponto na página, utiliza-se um sistema de coordenadas em dois eixos cuja origem está no topo esquerdo da pagina. O eixo horizontal é chamado de x e o vertical de y.

Originalmente, no formato PDF, os dados de layout e posição são associados individualmente a cada caractere do documento. Já os objetos extraídos pelo *pdfplumber* são organizados em forma de contêineres aninhados a partir do agrupamento de objetos mais elementares, formando objetos cada vez mais estruturados e contextualizados (uma linha é uma sequência de caracteres e um bloco/parágrafo é uma sequência de linhas). O caractere, instância da classe `LTChar` do *pdfplumber*, é o objeto de texto mais elementar de um arquivo PDF. Este tipo de objeto engloba características visuais de um texto como tipo da fonte, tamanho, formatações (negrito, itálico, cor). Estruturas maiores como palavras, linhas e blocos maiores de texto (parágrafos, por exemplo) são construídos desde o agrupamento de caracteres (objetos `LTChar`). Na Figura 2.1, é mostrado como os objetos são aninhados e como se pode extrair cada característica de um documento PDF. Uma linha de texto em um inteiro teor de um acórdão em PDF é representada pela classe `LTTextLineHorizontal`, de onde se pode extrair as seguintes características:

- *bounding box*,
- altura,
- largura, e
- espaço entre as palavras.

Embora a *pdfplumber* auxilie consideravelmente na tarefa de extração de objetos, esta ferramenta extrai conteúdo de forma incorreta dependendo do layout do PDF. Por exemplo, alguns layouts levam o *pdfplumber* a reconhecer um único bloco de texto como sendo dois blocos em um layout de duas colunas, quando na verdade

[3]Conjunto de dois pontos que determinam um retângulo de tamanho mínimo no qual está inserido o objeto.

[4]Estrutura de dados que armazena a informação de como o objeto é apresentado graficamente na tela ou impressão como cores de preenchimento e bordas.

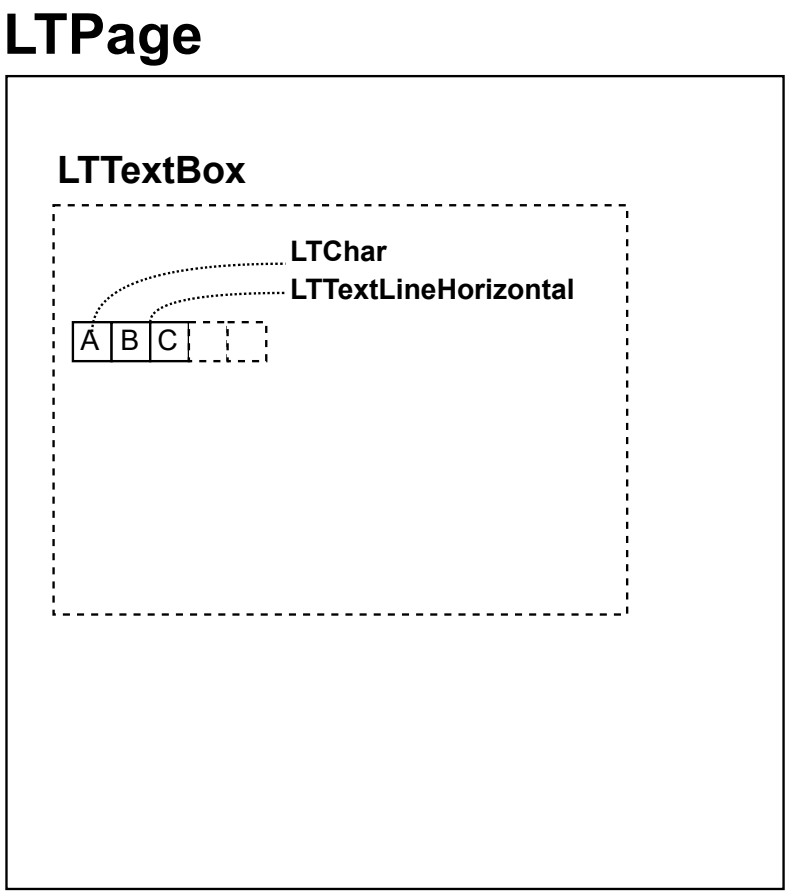


Figura 2.1: A estrutura de contêineres do *pdfplumber* com os tipos de objetos aninhados

trata-se de um único bloco. Também pode ocorrer inserção de textos originários de marcas d'água ou sobreposição em meio ao texto principal. Na Figura 2.2, é ilustrado como o *pdfplumber* reconhece o fluxo de leitura de um acórdão de maneira incorreta.

A *pdfplumber* itera as páginas do documento e, em cada página, extrai objetos de texto, juntamente com seus atributos de layout. Uma função obtain_word_tokens da biblioteca é capaz de compor os tokens em um arquivo PDF baseando-se nas propriedades de posicionamento dos glifos na página. As propriedades x_0, y_0, x_1, y_1, que determinam o posicionamento dos glifos na página, definem um fluxo legível do texto quando uma tolerância de espaçamento vertical e horizontal razoável é especificada. Essa tolerância é definida heuristicamente e, considerando que palavras e linhas normalmente tem mesmo estado do texto, a extração coesa dos tokens pode ser feita consistentemente.

Outro aspecto importante a ser considerado é quando o estado do texto dos glifos produz um sombreamento que induz a extração de um mesmo caractere mais de uma vez. Um efeito de sombreamento de um glifo pode ser entendido como um novo caractere e o texto é extraído incorretamente. Uma função da biblioteca, denominada dedupe_chars, cria uma cópia da página com caracteres duplicados removidos. Caracteres duplicados são identificados se possuem o mesmo texto, nome da fonte e tamanho, além de um posicionamento próximo, ou seja, dentro de uma tolerância

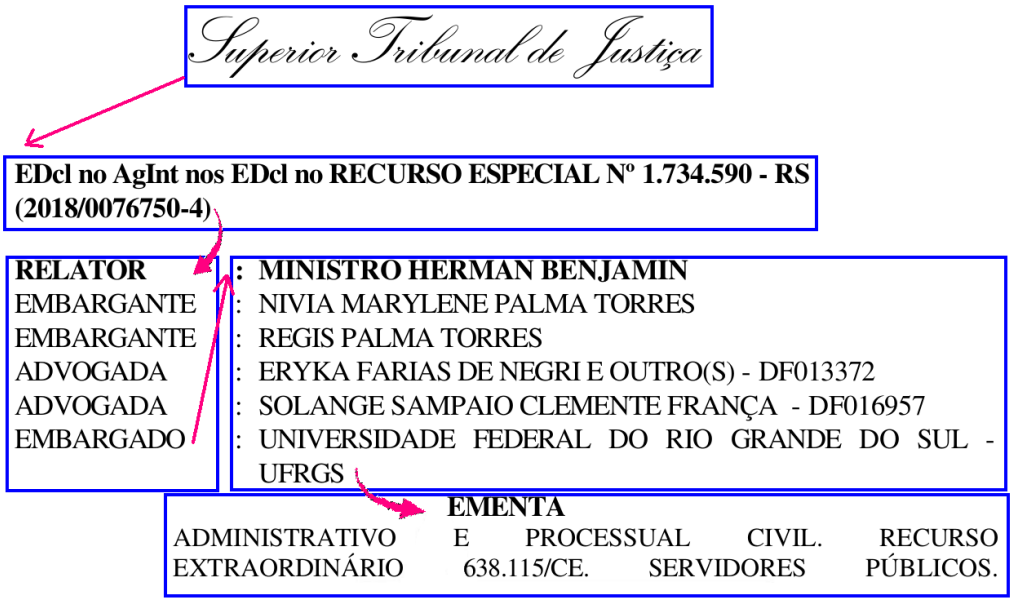


Figura 2.2: A figura mostra um problema comum na identificação da ordem de leitura de um PDF pelo *pdfplumber*, quando há uma confusão entre tabelas e colunas de leitura

x/y pré definida em relação a outro glifo.

2.3 Armazenamento de Dados

Para armazenamento dos dados extraídos dos arquivos PDF, serão utilizadas duas coleções no *MongoDB*[5]. Uma para metadados do inteiro teor e outra pra as informações de texto e layout extraídos dos documentos. O MongoDB possui uma abordagem de esquema flexível, o que significa que a modelagem de dados e estratégia de persistência são construídas para obter altas taxas de transferências tanto de leitura quanto escrita, seja para armazenamento de um nó ou dezenas deles (Banker et al., 2016). A escalabilidade é uma característica fundamental para armazenamento de dados ainda não conhecidos, e se torna ainda mais interessante por ser um banco de dados orientado a documentos, portanto de modelagem bastante intuitiva e adaptável. A importância dessa adaptabilidade pode ser notada facilmente na Tabela 2.3, onde se vê que cada tribunal mantém metadados diferentes para os seus processos.

2.4 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo matemático que tenta simular a estrutura e as funcionalidades de redes neurais biológicas (Krenker et al., 2011). A unidade básica da composição de uma RNA é o neurônio artificial, ou simplesmente neurônio, que é mais especificamente uma função matemática simples que converte

[5]https://www.mongodb.com/

STJ	TRF2
Acórdão	
Acórdãos Similares	Classe
Data da Publicação/Fonte	Data de disponibilização
Data do Julgamento	Data de decisão
Ementa	Ementa
id_processo	Pré-Visualização
Jurisprudência Citada	Órgão julgador
Órgão Julgador	pdf_file
pdf_file	pdf_url
pdf_url	Consulta processual
Processo	titulo
Referência Legislativa	Relator
Relator(a)	

Tabela 2.2: Lista de atributos de metadados de processos do STJ e TRF2.

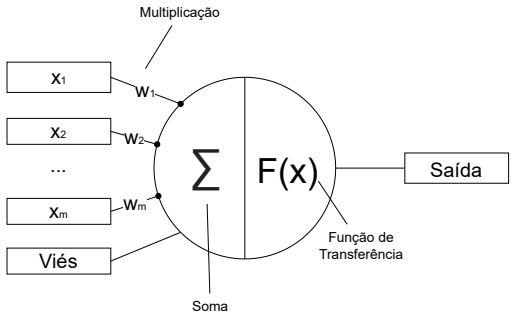


Figura 2.3: Representação de um neurônio artificial com suas operações

sinais de entrada em uma saída.

A combinação de mais de um neurônio é o que se chama de RNA. Embora o princípio de um neurônio seja simples, a medida que eles são interconectados em RNAs o poder de cálculo e o potencial em resolução de problemas é incremental e, com estas operações, é possível resolver problemas complexos. A forma como os neurônios são interconectados é chamada de topologia de uma RNA.

Como é possível observar na Figura 2.3, o neurônio é alimentado por um conjunto de valores de entrada $x = \{x_1, x_2, \ldots, x_m\}$ que são multiplicados pelos respectivos pesos $w = \{w_1, w_2, \ldots, w_m\}$. À soma das entradas ponderadas é acrescido o viés b. O resultado da soma das entradas ponderadas com o viés é passado para uma função de transferência F, que define as propriedades do neurônio artificial e pode ser qualquer função matemática que produz uma saída aos neurônios adjacentes. Em outras palavras, um neurônio é uma função do tipo:

$$y = F(\sum_{i=0}^{m} w_i.x_i + b) \tag{2.1}$$

onde x_i é um valor de entrada onde i vai de 0 a m, w_i é o valor do peso onde i vai de 0 a m, b é o viés, F é uma função de transferência e y é o valor de saída.

O ajuste dos pesos w dos neurônios em uma RNA é realizado por meio do algo-

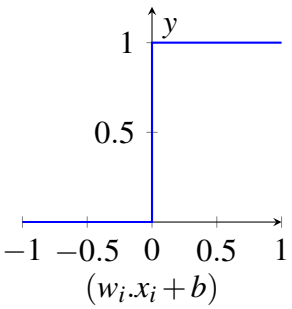


Figura 2.4: Função de degrau unitário.

ritmo de *backpropagation* (LeCun et al., 1989). Esse algoritmo utiliza uma função de perda para calcular o erro da saída e, em seguida, propaga as derivadas parciais para as camadas mais superiores da rede. O gradiente da função indica a direção em que os pesos devem ser ajustados, sinalizando se eles precisam ser aumentados ou diminuídos. Esse processo é conhecido como aprendizado de uma RNA.

A função de transferência é o item mais complexo deste modelo e deve ser escolhida cuidadosamente de acordo com o problema proposto. Existem algumas funções de transferência comuns e o tipo da função utilizada altera a precisão do resultado e influencia na eficiência do treinamento da rede (Suzuki, 2011).

Uma função de degrau unitário é uma função de ativação onde a saída do neurônio é 0 ou 1 dependendo do valor de $(w.x+b)$. Quando a soma é superior a 0 o neurônio é ativado, ou seja, sua saída é 1, caso contrário sua saída é 0, conforme mostrado na Figura 2.4.

Portanto:

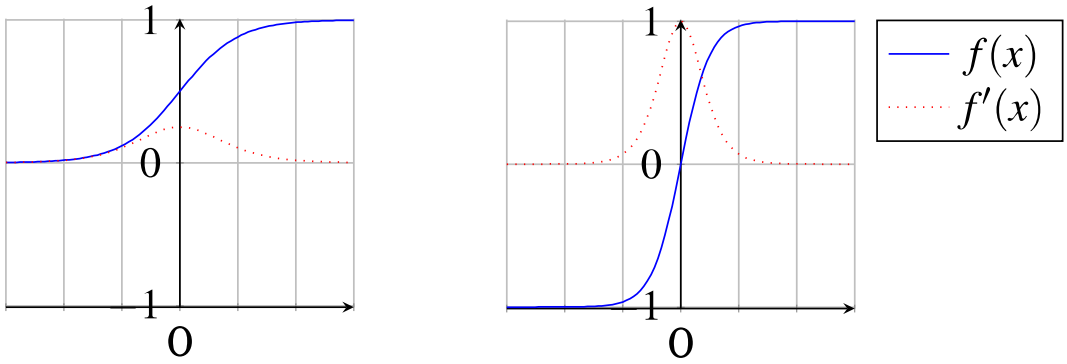
$$y = \begin{cases} 1, & \text{se } (w.x+b) \geq 0 \\ 0, & \text{se } (w.x+b) < 0 \end{cases}$$

Esse tipo de função de transferência é bastante simples e pode ser utilizada para problemas de classificação binária e geralmente é utilizada na camada final da rede de um modelo de classificação.

No entanto, na resolução de problemas complexos, a maior parte das RNAs utilizam funções continuamente diferenciáveis que limitam o resultado da soma em um intervalo reduzido (Hagan et al., 2014), seja $[0,1]$ ou $[-1,1]$. Dizemos que uma função $f(x)$ é diferenciável se a derivada $f'(x)$ existe. E se uma função $f : S \subseteq \mathbb{R} \to \mathbb{R}$ é diferenciável em $x \in S$, então f é contínua em x (Zanusso et al., 1997). Essas propriedades são fundamentais no ajuste dos pesos w da rede durante o treinamento.

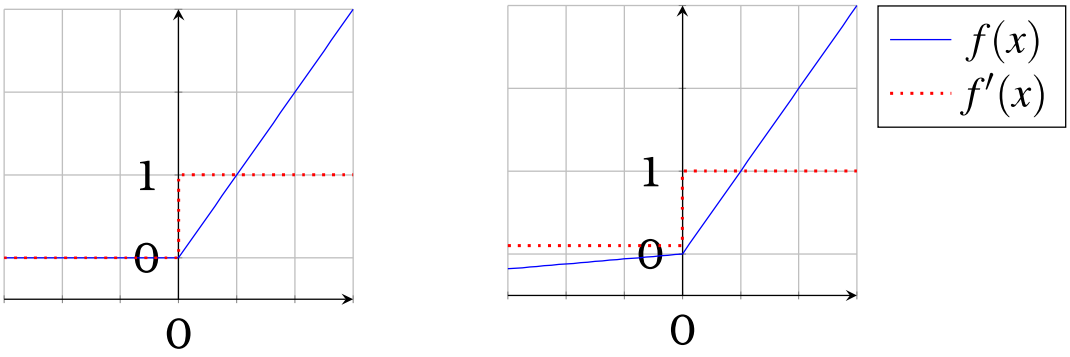
O uso de funções não lineares sigmoides potencializa o aprendizado em RNAs. Porém RNAs profundas que utilizam essas funções podem sofrer com o problema de *vanishing grandient* (Bengio et al., 1994). Os gradientes tendem a desaparecer quando as camadas inferiores de um RNA têm gradientes muito próximos de 0, já que as saídas da camada superior ficam saturadas em -1 ou 1 (Hochreiter and Schmidhuber, 1997). Dessa forma a convergência da otimização se torna lenta ou até mesmo converge para um mínimo local ruim.

Funções do tipo tangente hiperbólica, como mostrado na Figura 2.5b, são antis-



(a) Função logística (b) Tangente hiperbólica

Figura 2.5: Funções de transferência não lineares sigmoid e suas derivadas.



(a) ReLU (b) Leaky ReLU

Figura 2.6: Funções de unidade linear retificada (ReLU) e suas funções derivadas.

simétricas na proximidade da origem, dessa forma tem um gradiente mais espaçado que uma função logística, demonstrada na figura 2.5a, mas não superam o problema satisfatoriamente.

As funções ReLU (Rectified Linear Unit), ilustradas na Figura 2.6a, se tornaram uma boa escolha em aprendizado profundo, pois superam satisfatoriamente o problema de *vanishing gradient*. Isso porque para qualquer valor maior que zero, mesmo que muito alto, a derivada da função de ativação será sempre 1, possibilitando o aprendizado de qualquer forma. Por outro lado o gradiente constante em 0 no intervalo $(-\infty, 0)$ pode tornar o aprendizado lento, já que alguns neurônios nunca serão ativados. Essa característica motiva o uso da função Leaky ReLU, mostrada na Figura 2.6b

As diferentes maneiras que uma rede pode ser interconectada permite inúmeras topologias. A topologia de uma RNA é determinada pela quantidade de neurônios paralelos e como suas entradas e saídas se relacionam. Cada conjunto de neurônios paralelos é chamado de camada. Uma RNA, portanto, pode ser vista como um grafo ponderado direcionado, onde os neurônios são os vértices e as arestas, com pesos, são as conexões entre a saída e a entrada dos neurônios (Jain et al., 1996).

Baseado no padrão de conexões do grafo de uma rede neural, as RNAs podes ser

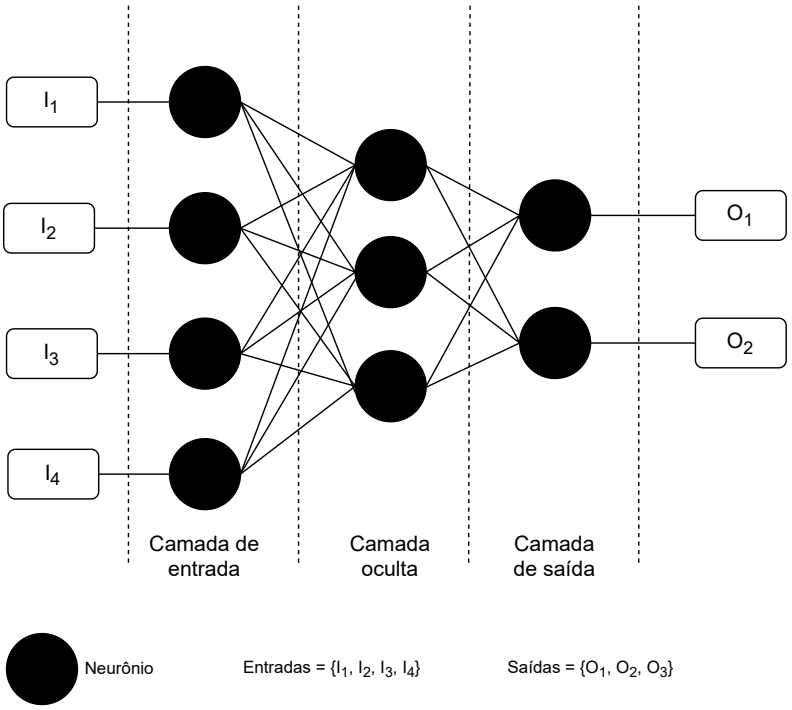


Figura 2.7: Uma rede neural artificial com uma camada oculta

categorizadas em dois tipos:

- Redes neurais diretas (feed-forward network)
- Redes neurais recorrentes (feedback network)

Na Figura 2.7 é mostrado um exemplo de uma rede neural direta composta por uma camada de entrada, uma oculta e uma camada de saída.

Os dados de entrada são consumidos pelos neurônios artificiais da primeira camada. Os valores são ponderados de acordo com os pesos w e somados juntamente com o viés b, exatamente como na Equação 2.1. As saídas da primeira camada serão os dados de entrada da próxima camada e assim sucessivamente até a produção dos valores de saída da rede.

As redes neurais recorrentes (RNN) são caracterizadas pela presença de *backloops* em sua topologia. Assim, a informação da rede não flui em apenas uma direção mas também são retropropagadas. Isso permite um comportamento temporal dinâmico e podem processar qualquer sequência de entradas utilizando sua memória interna. Na figura 2.8 é mostrado um exemplo de rede neural recorrente, onde cada um dos vértices possui uma recorrência para si. Essa topologia também é chamada de rede neural artificial totalmente recorrente.

2.4.1 Treinamento de uma Rede Neural Artificial

Existem basicamente três métodos de aprendizagem aplicáveis a praticamente qualquer arquitetura de uma RNA: a aprendizagem supervisionada, a aprendizagem não supervisionada e a aprendizagem por reforço.

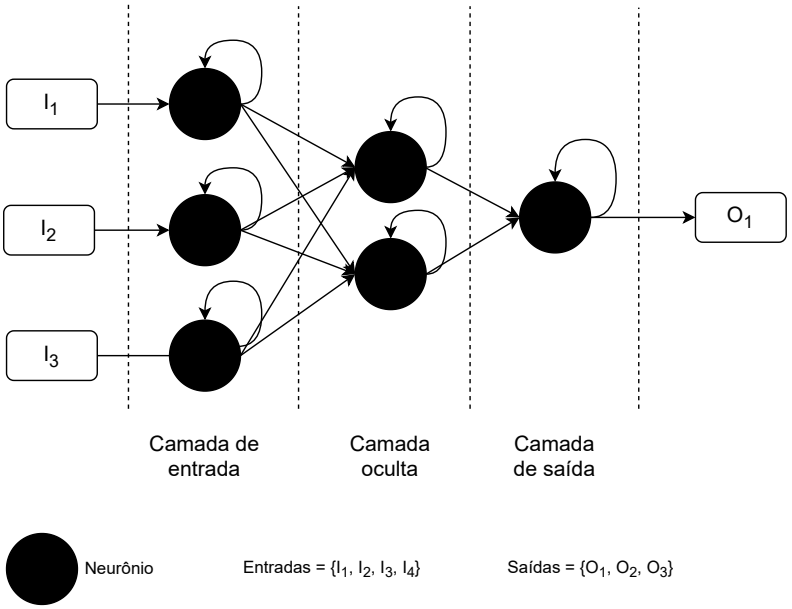


Figura 2.8: Rede neural artificial recorrente.

O aprendizado supervisionado consiste no ajuste dos parâmetros w de uma RNA a partir de um conjunto de dados de treinamento. O conjunto de dados de treinamento é dividido em pares de valores de entrada, chamado de exemplo, e saída desejados, chamado de anotação, normalmente apresentados em forma de vetores de dados.

Para realizar o treinamento de uma RNA através de aprendizagem supervisionada, um conjunto de dados de treinamento, que descreva de forma suficiente o problema e represente um subconjunto do mundo real, precisa ser reunido e formatado de forma compreensível à topologia e arquitetura da RNA. O desempenho do modelo gerado é avaliado com um conjunto de dados de teste, nunca utilizado durante o treinamento.

O ajuste dos pesos é feito com base em uma função de perda que indique a diferença entre uma saída e a anotação real fornecido. O parâmetro é ajustado para mais ou para menos, conforme o gradiente da derivada parcial e um novo exemplo é apresentado à rede, repetindo o procedimento. À medida que a rede neural é alimentada com exemplos de treinamento e os parâmetros são ajustados através de um processo iterativo, espera-se que a divergência entre a saída predita e a saída real seja gradualmente reduzida, com o objetivo ideal de convergência para zero. Assim, após o treinamento, o modelo é capaz de predizer o rótulo de um exemplo, até então desconhecido, com alta precisão.

No aprendizado não supervisionado, os parâmetros da RNA são definidos com base em dados fornecidos e uma função de perda que deve ser minimizada assim como no aprendizado supervisionado, com a diferença que os exemplos de treinamento não são anotados. Isso porque esse tipo de aprendizagem busca determinar a organização de dados. Sua principal aplicação está em problemas de estimativa, como modelagem estatística, compressão, filtragem e agrupamento, onde diferentes dados são agrupados por semelhança.

Na aprendizagem por reforço de uma RNA, os parâmetros são ajustados a longo prazo a partir de dados gerados por interações com o ambiente, e uma função de ma-

ximização de recompensa. Esta técnica visa treinar redes de sistemas que interagem com o ambiente e ajusta a sua resposta ao longo dessa interação. A aprendizagem por reforço costuma complementar um algoritmo de aprendizagem geral da RNA.

2.4.2 Modelos de Representação de Linguagem

A representação da linguagem para modelos AM é um desafio complexo, pois exige a compreensão prévia da estrutura e dos contextos da linguagem, além de um amplo conhecimento sobre o mundo em que esta linguagem é usada (Wang et al., 2014). Informações contidas no texto ou conjunto de documentos de um domínio, ou seja, informações estatísticas locais não são suficientes. Os modelos de representação tem por característica básica a capacidade de determinar, dada uma sequência de *tokens*, a probabilidade condicional de um *token* ser o seguinte à sequência.

Para adquirir essa habilidade, uma das técnicas principais é o uso de n-gramas (Bengio et al., 2003). Um n-grama é uma sequência contendo n elementos retirados de um conjunto. A principal vantagem da correspondência baseada em n-gramas é resultado de sua própria natureza, que divide cada conjunto em partes menores e limitadas, de modo que quaisquer erros presentes afetam apenas um número limitado dessas partes, deixando as demais intactas. Se contarmos n-gramas que são comuns a dois conjuntos, obteremos uma medida de sua similaridade que é resistente a uma ampla variedade de erros textuais (Cavnar et al., 1994). Modelos de representação de linguagem pré-treinados em conjuntos de dados extensos, disponíveis publicamente, são capazes de oferecer representações bastante realistas pelo conhecimento de mundo que carregam. Variações de diferentes técnicas, que realizam ajuste fino desses modelos públicos, são fundamentados no conceito de que o significado de uma palavra representada é definido pelo contexto. Modelos como Word2vec (Mikolov et al., 2013) e GloVe (Pennington et al., 2014) são treinados utilizando representações de palavras Continuous Bag-of-Words e varições do n-grama.

2.4.3 RNNs e Suas Aplicações em Dados Sequenciais

As RNNs, até pouco tempo, foram o principal método para lidar com dados sequenciais como, por exemplo, sequências de *tokens* em sentenças, uma vez que incorporam correlações entre pontos de dados que estão próximos (Schuster and Paliwal, 1997). Ela realiza composições com base nas árvores binárias, e obtém as representações vetoriais de forma ascendente. RNNs podem ser treinadas para geração de sequências processando as sequências de dados reais passo a passo para predizer o item seguinte.

Nesta estrutura, os vetores de entrada $x = (x_1, x_z, \ldots, x_t)$ são alimentados um de cada vez na RNN, onde x_t é um vetor de uma sequencia x em momento t. Essa sequência de vetores alimentam uma pilha de N camadas ocultas conectadas recorrentemente através de conexões ponderadas e calcula primeiro as sequências de vetores ocultos $\mathbf{h}^n = (h_1^n, \ldots, h_n^T)$, e então a sequência do vetor de saída $\mathbf{y} = (y_1, \ldots, y_T)$. Cada y_t serve como parâmetro de uma distribuição preditiva $Pr(x_{t+1}|y_T)$ das próximas entradas x_{t+1}. O

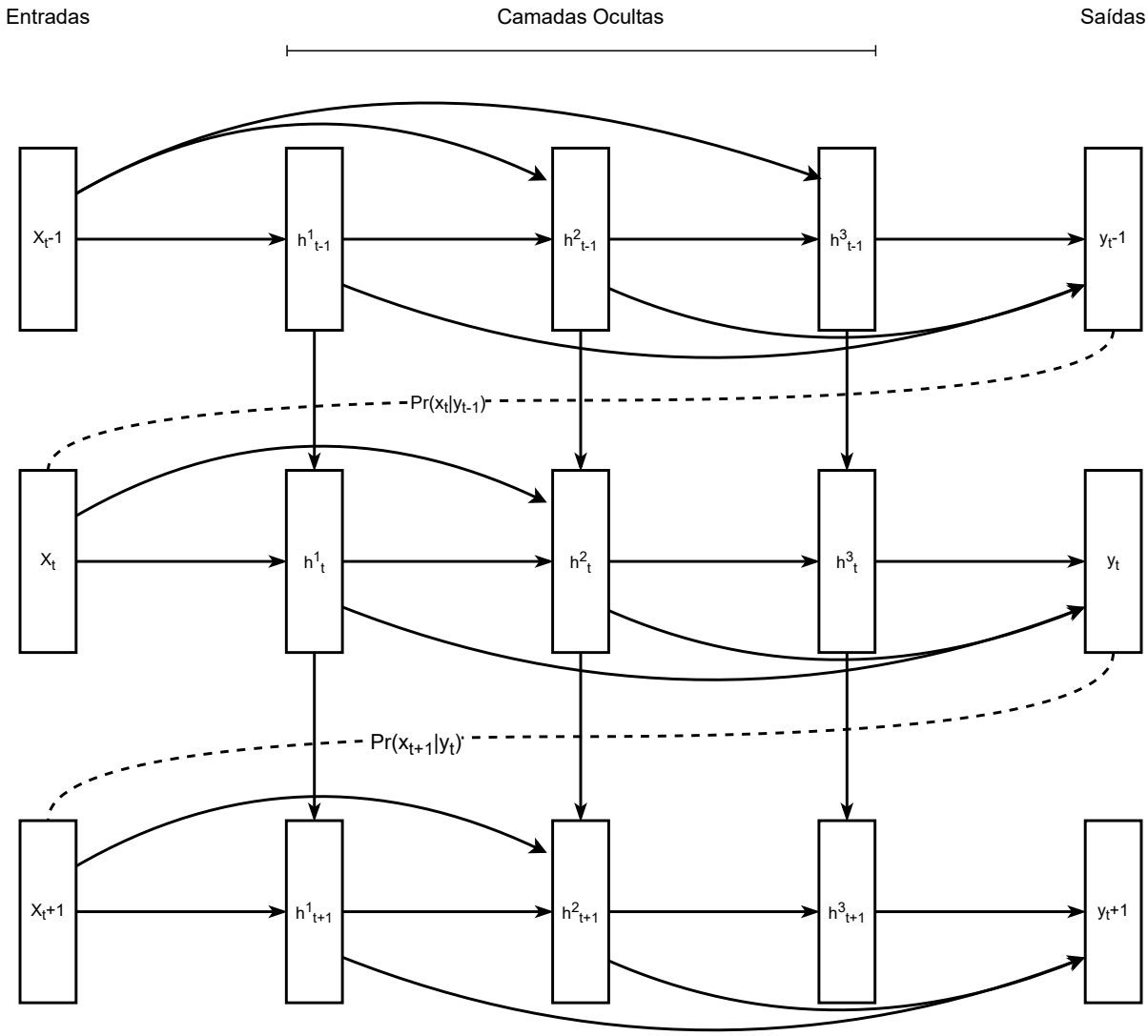


Figura 2.9: Arquitetura de predição de uma RNN profunda. Baseado em Graves (2013)

primeiro elemento de cada sequência de entrada é sempre um vetor esparso x_1. A rede, portanto, emite uma previsão para o próximo elemento x_2, a primeira entrada real, sem nenhuma informação previamente conhecida. Na figura 2.9 toda essa estrutura é representada.

Em vez de usar um número fixo de vetores de entrada como feito nas estruturas MLP, esta arquitetura pode fazer uso de todas as informações de entrada disponíveis até o período de tempo atual t_c, onde $t = \{1,2 \ldots c\}$. para predizer y_c.

Essa arquitetura, inerentemente sequencial, limita a paralelização nos exemplos de treinamento e, em sequencias longas, a exigência de uso de memória é uma obstáculo ao envio de exemplos em lote. Modelagens modernas baseadas unicamente em mecanismos de atenção (Vaswani et al., 2017) superam as exigências e limitações das RNNs e alcançaram novos estados da arte em diversas tarefas de PLN.

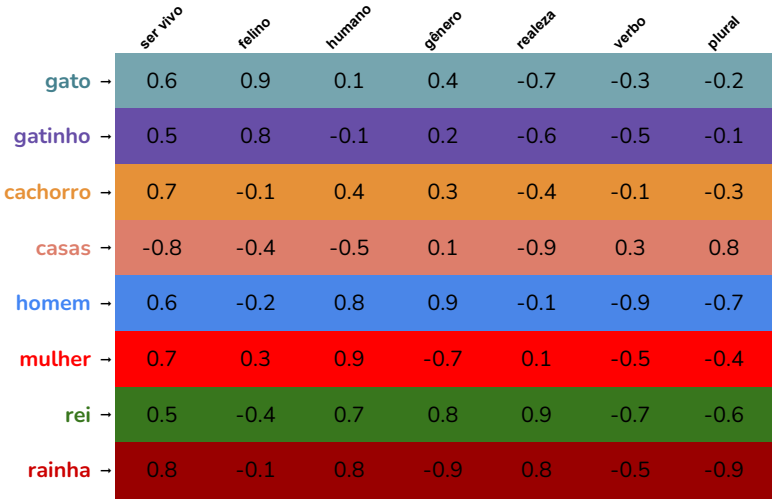


Figura 2.10: Demonstração de um grupo de palavras e sua representação de recursos

2.4.4 Embedding de Palavras

Nas primeiras abordagens de aprendizado de máquina para PLN, as palavras eram representadas como identificadores únicos referentes a vetores de recursos usando representações *one-hot*, em que cada vetor tem o mesmo comprimento que o tamanho do vocabulário com apenas uma dimensão declarada para indicar o ID da palavra. As representações distribuídas de palavras mais modernas, chamadas de *embedding* de palavras, são vetores de baixa dimensão e valor real. Cada dimensão no vetor representa um recurso de uma palavra, e o vetor pode, em teoria, capturar os recursos semânticos e sintáticos da palavra (Wang et al., 2014). *Embedding* de palavras são normalmente vetores de 50 a 200 dimensões, onde cada posição do vetor indica a influencia de uma determinada característica no significado. A Figura 2.10 é uma representação reduzida de um *embedding* de palavras.

2.4.5 Mecanismos de Atenção

Os mecanismos de atenção são técnicas avançadas e fundamentais na modelagem de sequências, que permitem a captura de correlações entre termos de entrada e saída sem a restrição da distância entre eles. Esses mecanismos podem ser definidos como funções que estabelecem uma relação entre uma consulta e um conjunto de pares chave-valor para gerar uma saída, onde cada uma dessas variáveis é representada por um vetor. A saída é obtida a partir de uma combinação linear dos valores, onde os pesos de cada valor são determinados pela compatibilidade entre a consulta e sua respectiva chave.

Em um mecanismo de atenção denominado *scaled dot-product*, a entrada é um conjunto de consultas (Q) e chaves (K) de mesma dimensão d_k e valores (V) de dimensão d_v. Os produtos escalares de Q e (V) são normalizados por uma divisão por $\sqrt{d_k}$ aos

quais são aplicados uma *softmax* para obter os pesos dos valores. Isso significa obter as funções de atenção do conjunto de consultas de uma vez, agrupadas em uma matriz Q. As chaves e os valores são condensados nas matrizes K e V. Este mecanismo é representado no esquema da Figura 2.11a, cuja matriz de saída é representado pela Equação 2.2

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

A função softmax $\sigma : \mathbb{R}^K \to [0,1]^K$ é uma função de normalização definida pela fórmula:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

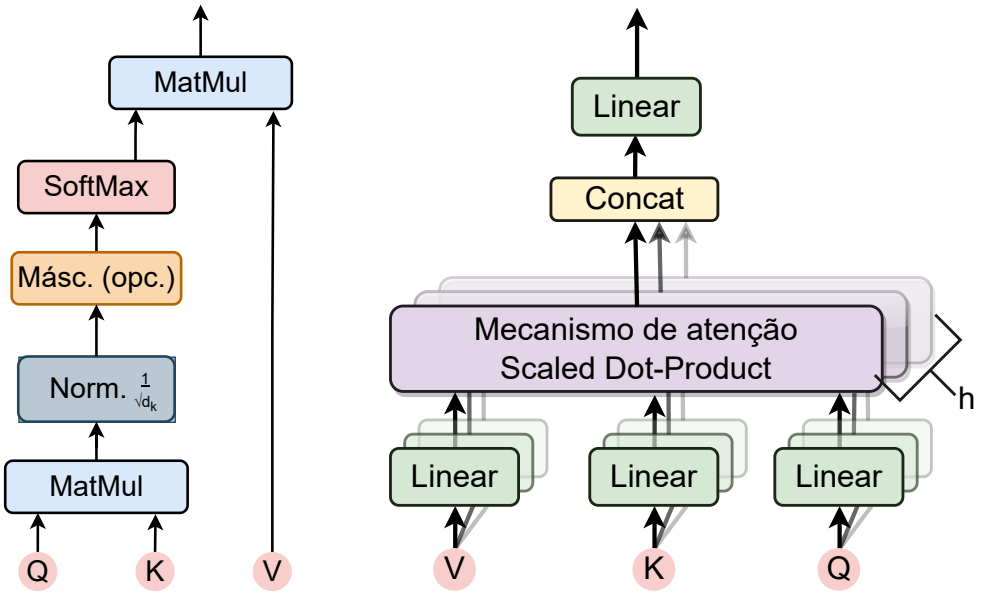
para $i = 1,\ldots,K$ e $\mathbf{z} = (z_1,\ldots,z_K) \in \mathbb{R}^K$.

Em resumo, a operação de atenção permite que o modelo se concentre em partes específicas da entrada, determinando a importância de cada elemento em relação a um determinado contexto. A atenção scaled dot-product usa uma operação matricial para calcular os pesos de atenção e produzir uma representação contextualizada da entrada.

Em outras palavras a consulta (Q) é uma representação vetorial do elemento atual que está sendo processado pelo modelo e é usada para calcular a relevância dos elementos na entrada em relação ao elemento atual. A chave (K) é uma representação vetorial de cada elemento na entrada e serve para determinar a relevância de cada elemento em relação à consulta. Cada elemento na sequência de entrada tem uma chave associada. O valor (V) é uma representação vetorial de cada elemento na entrada sendo usado para calcular a saída final do mecanismo de atenção. Cada elemento na sequência de entrada tem um valor associado. A $\sqrt{d_k}$ é utilizada para escalar a operação de multiplicação ponto-a-ponto entre as consultas e chaves, a fim de controlar o valor médio da operação e prevenir que o gradiente exploda ou desapareça durante o treinamento.

Por outro lado, o mecanismo de atenção chamado *multi-head*, em vez de realizar uma única função de atenção com chaves, valores e consultas dimensionais d_{model}, as consultas, chaves e valores são projetadas linearmente h vezes com diferentes projeções lineares aprendidas para dimensões d_k, d_q e d_v, respectivamente. Em cada uma dessas projeções, várias camadas de atenção são executada em paralelo, gerando valores de saída com dimensões de d_v. Estes são concatenados e projetados novamente, resultando nos valores finais. O mecanismo de atenção *multi-head*, cuja arquitetura é representada na Figura 2.11b, permite que o modelo atenda conjuntamente às informações de diferentes subespaços de representação em diferentes posições.

A auto-atenção é um mecanismo de atenção que relaciona diferentes posições de uma única sequência para computar uma representação da sequência. Auto-atenção tem sido usada com sucesso em uma variedade de tarefas, incluindo compreensão de leitura, resumo abstrativo, vinculação textual e aprendizagem de representações de sentenças independentes da tarefa. (Vaswani et al., 2017)



(a) Mecanismo de atenção *Multi-Head*

(b) Mecanismo de atenção *Scaled Dot-Product*

Figura 2.11: Mecanismos de atenção

2.4.6 *Transformers*

Os *Transformers* são modelos de linguagem baseados em mecanismos de atenção, capazes de computar representações abstratas de linguagem sem o uso de RNNs ou convoluções para estabelecer dependências globais entre os dados de entrada e de saída (Vaswani et al., 2017). Essa arquitetura permite uma paralelização maior e pode alcançar um novo estado da arte em tarefas de processamento de linguagem natural, como a tradução automática.

A arquitetura *Transformer* é composta por um *encoder* e um *decoder*. O *encoder* recebe a sequência de entrada e a processa, produzindo uma representação vetorial de cada elemento da sequência. Ele é composto por várias camadas de subcomponentes, cada uma das quais executa uma operação chamada *Multi-Head Attention* e uma camada de rede neural direta. Enquanto o *decoder* gera a sequência de saída a partir da representação vetorial gerada pelo *encoder*. Ele contém uma série de camadas que também executam operações de atenção multi-cabeça e redes neurais diretas, mas também incorpora um mecanismo de atenção adicional que ajuda o modelo a se concentrar nas partes relevantes da sequência de entrada enquanto produz a saída. O *encoder* mapeia uma sequência de embeddings de *tokens* de entrada $x = (x_1, \ldots, x_n)$ para uma sequência de embeddings contínuas $z = (z_1, \ldots, z_n)$, enquanto o *decoder* gera uma sequência de saída $y = (y_1, \ldots, y_m)$, um por um, usando os embeddings gerados pelo *encoder* como entrada. Em cada etapa, o modelo é auto-regressivo (Graves, 2013), consumindo os *tokens* gerados na etapa anterior como entrada adicional ao gerar a próxima.

O *Transformer* aproveita a arquitetura geral usando camadas de auto-atenção empilhadas e ponto-a-ponto totalmente conectadas para o *decoder* e o *decoder*, como

mostrado na Figura 2.12. As camadas de auto-atenção são capazes de ponderar diferentes partes da entrada de acordo com sua relevância para a saída, permitindo que o modelo leve em consideração as dependências entre todas as posições da entrada e saída simultaneamente. Essa abordagem permite que o *Transformer* capture relações não lineares entre as posições de entrada e saída, tornando-o eficaz em tarefas de processamento de linguagem natural.

2.4.7 BERT

O BERT (Bidirectional Encoder Representations from Transformers) é um modelo de linguagem baseado em atenção cuja arquitetura consiste em um transformer multicamadas que é capaz de produzir uma representação contextualizada final de um conjunto de tokens de entrada (Devlin et al., 2018).

A característica distintiva do BERT é que ele contextualiza cada token em ambos os sentidos do texto, em todas as camadas do modelo. Isso é possível graças à utilização de uma arquitetura de transformer bidirecional. Dessa forma, o modelo é pré-treinado em representações profundas do texto não anotado explicitamente, tornando-o capaz de lidar com uma ampla variedade de tarefas de processamento de linguagem natural, sendo considerado um dos melhores modelos pré-treinados devido ao seu alto desempenho e precisão. Uma vantagem do BERT é que ele pode ser adaptado para uma variedade de tarefas de PLN, sem a necessidade de grandes modificações em sua arquitetura. Isso o torna uma ferramenta muito valiosa para a pesquisa em PLN e aplicações práticas.

Para representar as sequências de entrada e saída, o BERT utiliza uma maneira padronizada de representação baseada em *tokens* especiais, capaz de representar inequivocamente uma sentença isolada ou um par de sentenças. Cada sentença é representada por uma sequência de *tokens*, com o primeiro *token* sempre sendo um *token* de classificação especial $[CLS]$. Para sentenças em pares de conjuntos de tokens A e B, esses conjuntos são separados por um *token* especial $[SEP]$. Adicionalmente, *embeddings* aprendidos são adicionados a cada *token*, indicando se ele pertence ao conjunto A ou ao B.

A representação de entrada de um *token* é construída somando os *embeddings* de *token*, segmento e posição na sequência correspondentes. O *embedding* de entrada é chamado de E, o vetor oculto final do *token* especial $[CLS]$ é chamado de C, onde $C \in \mathbb{R}^H$ e o vetor oculto final para o i-ésimo *token* de entrada é chamado de $T_i \in \mathbb{R}^H$. A representação da entrada do BERT é mostrada na Figura 2.13.

O modelo BERT pré treinado aprende a representação da linguagem a partir da Modelagem de Linguagem Mascarada (MLM) e Previsão da Próxima Palavra (NSP).

A técnica de pré-treinamento utilizando o MLM, consiste em, aproveitando-se do contexto bidirecional, realizar um mascaramento em uma porção aleatória dos *tokens* de entrada. Cada um desses n *tokens* w_t é substituído por um *token* especial $[MASK]$ e predito usando todos os *tokens* anteriores e posteriores a w_t, onde $W_t = (w_1, w_2, \ldots, w_{t-1}, w_{t+1}, \ldots, w_n)$, sendo sua principal vantagem diante dos modelos de

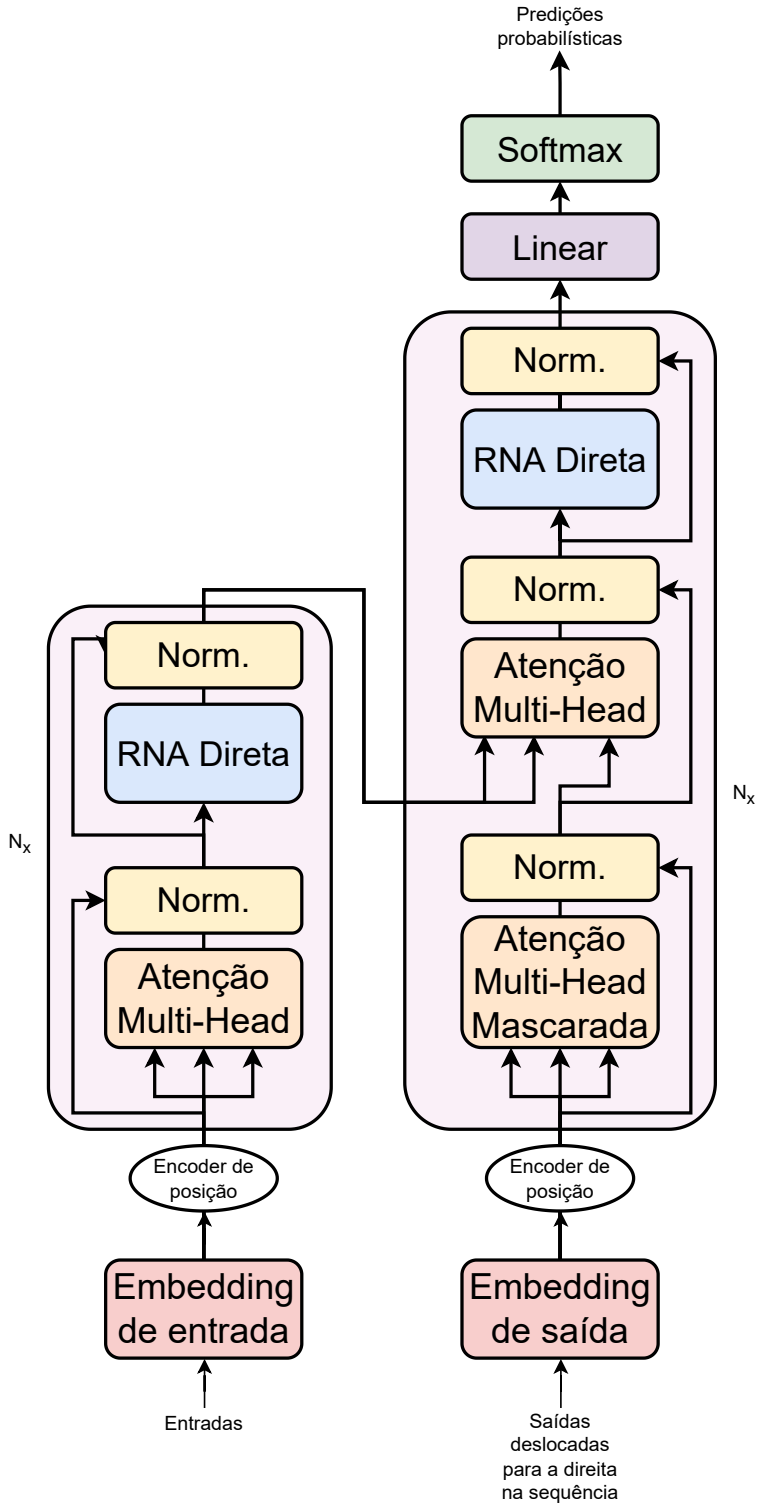


Figura 2.12: Arquitetura do modelo *Transformer* (Vaswani et al., 2017)

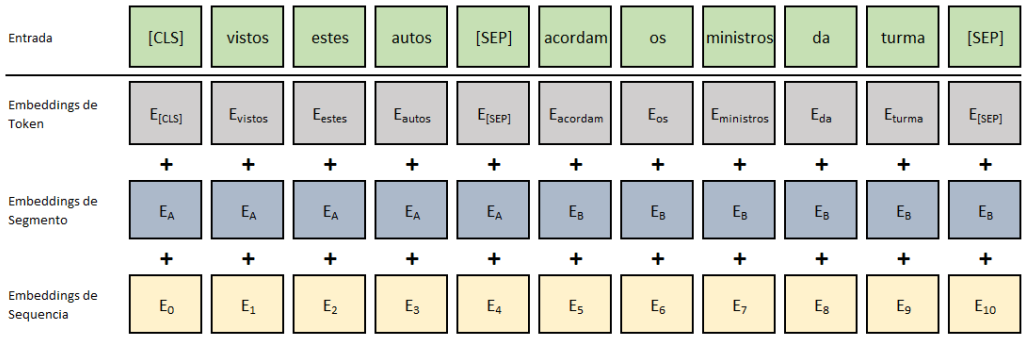


Figura 2.13: Os *embeddings* de entrada são a soma dos *embeddings* de *token*, de segmento e de sequência

linguagem convencionais, que predizem w_t usando apenas *tokens* anteriores $W_{<t} = (w_1, \ldots, w_{t-1})$. Durante a etapa do ajuste fino do modelo o *token* $[MASK]$ não está presente prejudicando essa etapa. Para atenuar essa desvantagem, 10% dos *tokens* selecionados para mascaramento é substituído por um *token* aleatório e outros 10% não são alterados, então o *token* selecionado será usado para predizer o *token* original. A técnica *next sentence prediction* (NSP) é importante para tarefas que são baseadas na compreensão da relação entre duas sentenças. Para que o modelo entenda essas relações, são escolhidas duas sentenças A e B para cada exemplo de pré-treinamento, em metade deles, B é a sentença real seguinte a A, então anotada como IsNext, na outra metade, B é uma sentença aleatória do corpus não consecutiva a A e anotada como NotNext.

O ajuste fino do modelo BERT para tarefas de um domínio específico pode ser feito facilmente adicionando uma camada de saída. O BERT faz o ajuste fino diretamente, apenas ajustando entradas e saídas apropriadas. Nesse ajuste fino, os parâmetros são atualizados de ponta a ponta. Em aplicações que envolvem pares de textos, ele unifica os estágios de codificação de pares de texto e aplicação de atenção cruzada bidirecional. Na sua entrada, os textos A e B do pré-treinamento são análogas aos pares de sentenças das tarefas que envolvem pares de textos. Para as tarefas de nível de *token*, as representações do *token* são alimentadas em uma camada de saída e o *token* $[CLS]$ é alimentada em uma camada de saída de classificação.

2.5 Abordagem Híbrida: Texto e Layout

Documentos são geralmente estruturados de maneira hierárquica. Por exemplo, um livro é composto por capítulos, um capítulo é composto por seções, uma seção é composta por parágrafos, e assim por diante. O layout de um documento tende a refletir visualmente esta estrutura. Esta característica é aproveitada por (Huang et al., 2022) para classificação de *tokens* em documentos científicos em PDF usando modelos de AM. As técnicas de pré-treino auto supervisionado multimodal (Xu et al., 2021) tem evoluído rapidamente na inteligência de documentos por conta de suas aplicações bem-sucedidas de layout de documentos e aprendizado de representação de imagem (Garncarek et al., 2021). Esta abordagem serve como base neste trabalho

para avaliar a tarefa de segmentação acórdãos jurídicos. Nas seções a seguir serão demonstradas as ferramentas e os métodos para incorporar estruturas visuais de layout das linhas de texto em modelos de linguagem. Particularmente, serão descritas a representação e a modelagem das entradas que combinam texto e layout em um classificador baseado em AM.

2.5.1 LayoutLMv3

Características visuais de layout de um documento PDF podem ser usados como recursos adicionais para melhorar modelos de linguagem baseados no BERT. Huang et al. (2022) propõem um modelo, denominado LayoutLMv3, utilizando *patch embeddings* (Dosovitskiy et al., 2020) e realizam o pré-treinamento de *transformers* multimodais para inteligência documental unificando mascaramento de texto e imagens com 3 tarefas de treino:

- Masked Language Modeling (MLM)
- Masked Image Modeling (MIM)
- Word-patch alignment (WPA)

Na figura 2.14 está representada a arquitetura do LayoutLMv3 e suas tarefas de treino. O LayoutLMv3 é um *transformer* multimodal pré-treinado para inteligência de documentos através do mascaramento unificado de texto e imagem. A entrada do modelo é basicamente uma imagem de documento, seu texto correspondente e as informações de posição de layout de cada token. O modelo faz uma projeção linear de *patches* e *tokens* de palavras como entradas e as codifica em representações vetoriais contextualizadas.

O LayoutLMv3 é pré-treinado por processos de reconstrução de tokens discretos de *Masked Language Modeling* (MLM) e *Masked Image Modeling* (MIM). Além de MLM e MIM, o LayoutLMv3 é pré-treinado com uma tarefa de *Word-Patch Alignment* (WPA) com objetivo de aprender o alinhamento modal cruzado, buscando predizer se o patch de imagem correspondente a uma palavra do texto está mascarado ou não. Os *tokens* $[Seg]$ denotam posições em nível de segmento e os *tokens* $[CLS]$, $[MASK]$, $[SEP]$ e $[SPE]$ são tokens especiais herdados do MLM do BERT.

O módulo de linguagem do LayoutLMv3 é inspirado no MLM do BERT. 30% dos *tokens* de texto são mascarados com uma estratégia de mascaramento de períodos com comprimentos de extensão extraídos de uma distribuição de Poisson (Gurgel, 1945) (λ = 3).

No módulo de imagem do LayoutLMv3, o mascaramento de imagem é alinhado ao mascaramento de linguagem, onde cerca de 40% das imagens de *tokens* são mascaradas aleatoriamente com a mesma estratégia de mascaramento de períodos do MLM. O objetivo é fazer o modelo interpretar o conteúdo visual a partir de representações contextuais de texto e imagem, direcionado por um cálculo de perda baseado em entropia cruzada para reconstruir os tokens de imagens que foram mascarados baseando-se nos *tokens* de texto e imagem em volta.

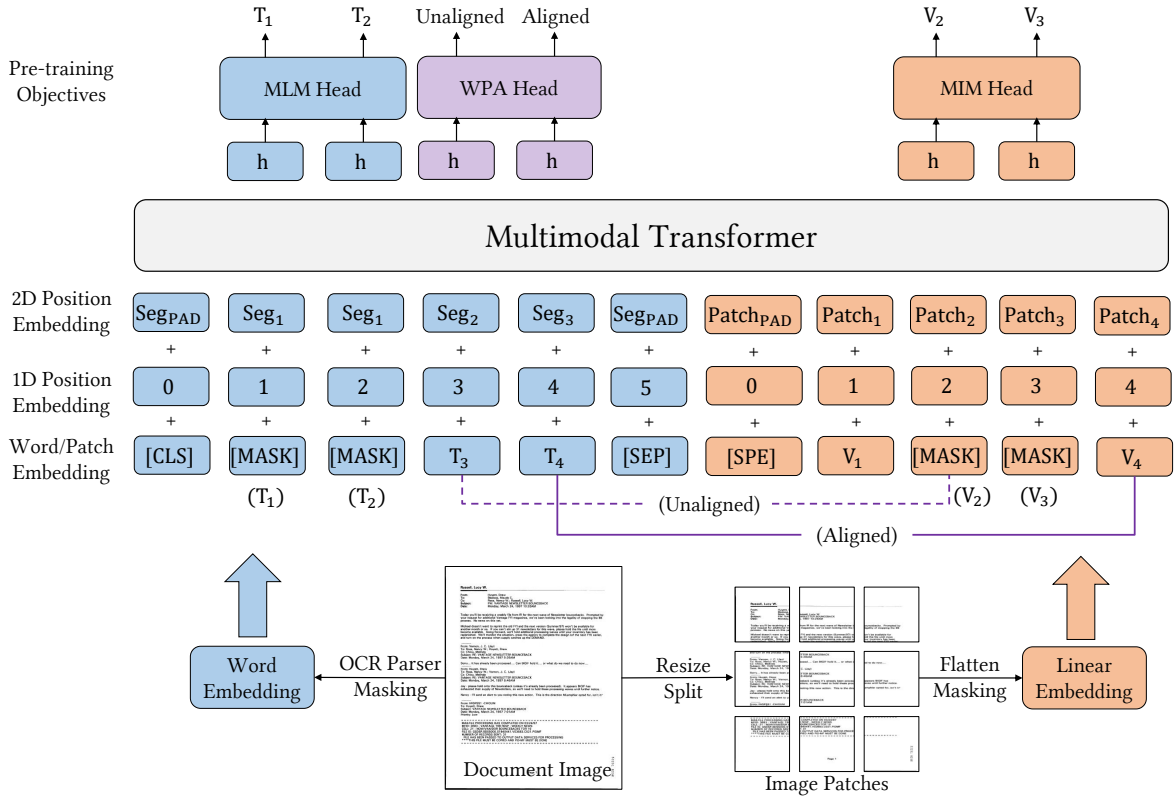


Figura 2.14: A arquitetura do LayoutLMv3 e suas tarefas de treino (Huang et al., 2022).

No módulo WPA, em um documento cada palavra extraída do texto corresponde a um patch de imagem. Como os tokens de texto e imagem são mascarados aleatoriamente pelo MLM e pelo MIM, o aprendizado nos módulos de texto e imagem não são alinhados explicitamente. Assim, um rótulo de um token de texto não mascarado é atribuído a um token de imagem correspondente também não mascarado. O objetivo do WPA, portanto, é prever se os *patches* de imagem correspondentes de uma palavra de texto estão mascarados. Um *transformer* de duas camadas insere texto e imagem contextualizadas e devolve os rótulos binários alinhados/desalinhados com a perda calculada baseada em entropia cruzada binária.

CAPÍTULO

3

Datasets de Segmentação

Neste capítulo, é descrito o processo de construção dos datasets de segmentação de acórdãos em todas as suas etapas. Na Seção 3.1, os critérios utilizados na escolha dos tribunais, cujos arquivos foram coletados, são apresentados, assim como as diversas maneiras que cada tribunal disponibiliza seus arquivos na internet. Também são apresentadas algumas características que dificultam ou facilitam a coleta dos documentos em cada tribunal. A forma como as páginas HTML são manipuladas para extrair as informações de interesse, bem como o processo de coleta dos dados efetivamente são descritos na Seção 3.2. Nesta mesma seção são detalhados como foram elaborados os filtros nos portais de busca, como fazer a busca programaticamente e as bibliotecas utilizadas para realização da coleta. Na Seção 3.3 deste capítulo, será explicado o processo de extração automática do conteúdo de texto e as informações de *layout* de um arquivo em formato PDF através do PDFPlumber. A técnica utilizada para anotação automática dos PDFs e como os scripts de anotação foram concebidos e aplicados são descritos na Seção 3.4, onde também é apresentado o formato adotado para o dataset.

Todos os scripts desenvolvidos para extração, anotação automática e anotação manual dos acórdãos estão disponíveis para download.[1]

3.1 Tribunais Considerados

Para definir quais tribunais serão usados para coleta dos documentos, algumas características foram levadas em consideração. Os tribunais escolhidos devem permitir em seus portais a coleta de todos os documentos entre 2012 e 2021 de maneira programática e os resultados devem ser determinísticos. A possibilidade de fazer a

[1] http://direitodigital.ufms.br:8000

coleta de forma automática através de scripts é primordial. A partir daí, outras características facilitadoras ou limitantes são consideradas.

A coleta automática requer a premissa do portal aceitar requisições HTTP com os parâmetros da busca. Como a intenção é coletar todos os documentos do período, o único filtro desejado é a data de publicação ou, quando não disponível, a data de julgamento. Não sendo desejável qualquer outro filtro ou critério obrigatório como, por exemplo, nome do juiz, nome de uma das partes ou número do processo.

Quando filtros adicionais simples forem requeridos, as diferentes combinações de filtros devem produzir conjuntos de resultados mutuamente exclusivos, para evitar duplicidade de documentos. Na página de pesquisa do TJMG, por exemplo, é obrigado selecionar um Órgão Julgador dentre os 49 existentes no tribunal. Ao fazer uma pesquisa para cada um deles é possível obter todos os documentos disponíveis sem resultados repetidos. Na página do TJMMG (Tribunal de Justiça Militar de MG), é obrigado selecionar a matéria cível ou criminal, bastando uma pesquisa para cada uma delas para obter todos os resultados desejados.

Tribunais que possuem sistemas de validação de captcha ou necessidade de cadastro prévio foram desconsiderados. Uma navegação pelo portal do tribunal foi realizada para avaliar a fluidez e estabilidade do sistema, bem como as características que determinam os objetos de interesse na página. Portanto, um portal de pesquisa elegível para coleta deve possuir filtros ou combinações de filtros que produzam as seguintes características:

- pesquisa apenas por data,
- URL estática para acesso ao inteiro teor,
- inteiro teor em PDF,
- não exigência de captcha, cadastro, ou outras validações,
- não possuir filtros de pesquisa obrigatórios que limitam a busca.

Dentre os tribunais que apresentam todas as características acima, considerou-se, em ordem, a esfera a qual pertence, a quantidade de documentos disponíveis e a disponibilidade do serviço durante a navegação.

As características dos portais de busca dos tribunais do sistema judiciário brasileiro são apresentadas a seguir, onde inclui-se características como: filtro de resultados por período, quantidade de registros por página de resultados, em qual formato o inteiro teor é disponibilizado e se o link para o arquivo é estático. Também são discriminadas características não desejáveis como validação com captcha, logon no sistema e cadastro prévio.

Inicialmente, na Tabela 3.1, são apresentados os tribunais superiores:

- STF: Superior Tribunal Federal
- STJ: Superior Tribunal de Justiça
- STM: Superior Tribunal Militar

- TST: Tribunal Superior do Trabalho
- TSE: Tribunal Superior Eleitoral

Tribunais Superiores							
Tribunal	Busca por período	Total consulta	Registros por página	Atributos	Formato inteiro teor	Captcha	Link estático
STF	SIM	SIM	250	Órgão julgador relator data julgamento data publicação ementa decisão Observações indexação Legislação publicação diário oficial partes	PDF	NÃO	SIM
STJ	SIM	SIM	50	processo relator órgão julgador data do julgamento data publicação ementa acórdão	PDF	SIM	SIM
STM	SIM	SIM	100	Relator(a) Assuntos Data de Autuação Data de Julgamento Data de Publicação EMENTA	PDF	NÃO	NÃO
TST	SIM	NÃO	100	Processo Órgão Judicante Relator Julgamento Publicação Tipo de Documento	HTML DOC PDF	NÃO	NÃO
TSE	SIM	SIM	20	processo data do Acórdão Relator(a) Publicação Ementa Decisão Partes Ver Também Observação	PDF	NÃO	NÃO

Tabela 3.1: Características dos portais de busca de documentos dos tribunais superiores do Brasil

Dentre estes tribunais, apenas STF e TST possuem portais facilmente navegáveis e aptos à coleta sem complicações. No entanto, destaca-se que o STJ, embora exija a validação de captcha após algumas páginas de navegação, por ser o tribunal com a maior quantidade de documentos, foi estudada uma forma de superar essa limitação. Verificou-se que ao reiniciar a sessão o contador para solicitação da validação de captcha era reiniciado, possibilitando a coleta dos arquivos desse tribunal. Além do STJ, dentre os tribunais superiores, o STF foi selecionado. Esta escolha se deu, principalmente, por dois motivos: é a corte máxima do judiciário brasileiro e existe um trabalho anterior que disponibiliza uma grande base de PDFs e um script de segmentação (Sousa and Del Fabro, 2019).

Existe um sistema de consulta de jurisprudência que unifica a pesquisa de acórdãos de todos os tribunais da Justiça Eleitoral no portal do TSE. No entanto ele limita a coleta programática por não fornecer uma URL estática para o arquivo do inteiro teor do acórdão, como apontado na Tabela 3.1. A URL do PDF é gerada durante a sessão e perde a validade ao encerrar a sessão. Como o sistema dá acesso a acórdãos de 28 tribunais distintos, uma forma de superar essa limitação também foi estudada. Devido à complexidade apresentada e a necessidade de utilização de ferramentas mais avançadas, além da instabilidade do sistema, que esteve frequentemente fora do ar durante o desenvolvimento deste trabalho, os tribunais regionais da Justiça Eleitoral foram preteridos. Dentre os tribunais regionais federais (listados na Tabela 3.2) e do trabalho (Tabela 3.3), apenas o TRF2 e o TRT1 apresentam as características desejadas. Entre eles, o TRF2 foi selecionado por apresentar todos os resultados em uma única consulta, enquanto o TRT1 limita o intervalo de pesquisa em, no máximo, um ano.

Dentre os Tribunais de justiça, que são apresentados na Tabela 3.4, foram selecionados o TJMG, por ter uma quantidade de documentos maior, e o TJPB, por apresentar uma variação grande de layouts.

Depois de escolhidos alguns tribunais, outros problemas foram identificados durante o desenvolvimento dos scripts de coleta. No TJMG, por exemplo, foram encontradas dificuldades em relação ao tempo para carregar as páginas e também na fluidez das pesquisas e navegação. Foram necessários 34 dias para coletar o link de apenas 44.812 documentos, bem menos do que os 2.235.572 documentos existentes no período.

O TJPB permite acesso a um HTTP *directory* com 96.683 documentos, entre acórdãos e decisões monocráticas. Pelo esforço empregado para separação de decisões monocráticas e acórdãos, além de aproveitar os metadados que a consulta na web fornece, decidiu-se utilizar o *web scrapping*.

No decorrer da execução do cronograma, as páginas do STJ, STF e TRF2 foram modificadas, exigindo ajustes para que os crawlers continuassem a funcionar. Como a base ITD disponibiliza documentos PDF com o inteiro teor de processos do STF, esta base foi utilizada neste trabalho. Os demais tribunais tiveram seus scripts de coleta ajustados aos novos portais.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TRF1	Não		-	Tipo Acórdão Número Classe Relator(a) Origem Órgão julgador Data Data da publicação Fonte da publicação Ementa Decisão Texto Inteiro teor	HTML segmentado na consulta pública do pje.	recaptcha	-
TRF2	Sim	sim	10	Ementa Classe Órgão julgador Data de decisão Data de disponibilização Relator pdf_url	HTML ou PDF	nao	sim
TRF3	Não		Não	Processo Relator(a) Órgão Julgador Data do Julgamento Data da Publicação/Fonte Ementa Acórdão	HTML	não	-
TRF4	Não		Não	Classe Processo Data da Decisão Órgão Julgador Inteiro Teor Citação Relator Relator para Acórdão Decisão Ementa	HTML	NÃO	-
TRF5	Não		Não	Processo Órgão Julgado Classe Data Julgamento Relator Data Autuação EMENTA	-	sim	-

Tabela 3.2: Características dos portais de busca de documentos dos Tribunais Regionais Federais do Brasil

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TRT1	SIM	SIM	100	Data de Acesso Data de Disponibilização Data de Publicação URL Título Tipo de Documento Data do Julgamento Órgão Julgador Tipo de Processo Juiz / Relator / Redator designado Tipo de Relator Número do Documento Ementa Identificador do Documento Sistema Processual Aparece nas coleções (ano)	PDF	NÃO	SIM
TRT2	SIM	SIM	100	Título Classe Pólo passivo Instância Data distribuição Assuntos Acórdão NÃO Data de publicação Relator(a) Órgão julgador	RTF HTML PDf	SIM	NÃO
TRT3	SIM	SIM	10	Título Disponibilização Órgão Julgador Relator Ementa	HTML	SIM	NÃO
TRT4	SIM	SIM	10	Data Órgão Julgador Redator EMENTA	HTML	NÃO	NÃO
TRT5	Ano	NÃO	entre 5 e 10	Processo/Recurso Sistema Autuado em Polo Ativo Advogado(s) Polo Passivo Advogado(s) Local do Processo Data da Consulta	HTML	SIM	NÃO
TRT6	SIM	SIM	100	Processo Classe Processual Redator Orgão Colegiado Data da Assinatura Data de Julgamento Ementa	HTML RTF	SIM	NÃO
TRT7	SIM	SIM	100	Título Classe Pólo passivo Instância Data distribuição Assuntos Acórdão NÃO Data de publicação Relator(a) Órgão julgador	RTF HTML PDF na consulta publica do pje	SIM	NÃO
TRT8	SIM	SIM	10	Titulo Ementa Numero Data Relator Classe	HTML PDF na consulta publica do pje	SIM	NÃO

Continua na próxima página.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TRT9	SIM	SIM	100	Título Classe Pólo passivo Instância Data distribuição Assuntos Acórdão NÃO Data de publicação Relator(a) Órgão julgador	RTF HTML PDF na consulta publica do pje	SIM	NÃO
TRT10	SIM	SIM	20	Número Processo Redator Data de Julgamento Data de publicação Tipo de Documento Ementa Ementa/Sentença/Decisão/Verbete Dispositivo	HTML PDF	SIM	NÃO
TRT11	SIM	SIM	50	Data Disponibilização Desembargador(a) / Juiz(a) Ementa Inteiro Teor	PDF	SIM	NÃO
TRT12	SIM	SIM	100	Nº Publicação Órgão Julgador Magistrado Ementa	HTML PDF	SIM	NÃO
TRT13	SIM	SIM	10	Processo Publicação Julgamento Redator(a) Ementa	html	SIM	NÃO
TRT14	SIM	SIM	100	Classe Pólo passivo Instância Data distribuição Órgão Julgado Magistrado Assuntos	HTML RTF PDF	SIM	NÃO
TRT15	SIM	SIM	10	Data publicação Ano do processo Órgão Julgador Relator	HTML PDF	SIM	NÃO
TRT16	SIM	SIM	up to 10	Número Relator(a) Assinatura Origem EMENTA	HTML PDF	SIM	NÃO
TRT17	SIM	SIM	100	Classe Pólo passivo Instância Data distribuição Órgão Julgado Magistrado Assuntos	HTML RTF PDF	SIM	NÃO
TRT18	NÃO	SIM	TXT PDF	DOCUMENTO PROCESSO ÓRGÃO JULGADOR DECISÃO RELATOR(A): Ementa(extraida automaticamente) Inteiro teor	TXT PDF	SIM	SIM
TRT19	NÃO	SIM	20	Publicação Processo: Relator(a) Ementa	html	NÃO	

Continua na próxima página.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TRT20	SIM	SIM	10	Número Relator(a) publicação EMENTA	HTML PDF	SIM	NÃO
TRT21	SIM	NÃO	100	Número do processo Magistrado(a) Julgamento Publicação	html	SIM	NÃO
TRT22	SIM	NÃO	10	processo Relator Orgao Julgador Data de julgamento Classe Data de publicação ementa Inteiro Tero	TXT HTML	SIM	NÃO
TRT23	SIM	SIM	100	Classe Pólo passivo Instância Data distribuição Órgão Julgado Magistrado Assuntos	HTML RTF PDF	SIM	NÃO
TRT24	SIM	SIM	100	Classe Pólo passivo Instância Data distribuição Órgão Julgado Magistrado Assuntos	HTML RTF PDF	SIM	NÃO

Tabela 3.3: Características dos portais de busca de jurisprudência dos tribunais regionais do trabalho

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TJAC	SIM	SIM	20	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	TXT PDF	SIM	SIM
TJAL	SIM	SIM	20	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	TXT PDF	SIM	SIM
TJAP	NÃO	SIM	20/mais resultados	Numero Acórdão Numero Processo Relator Ementa Acórdão Teor do Ato RELATÓRIO VOTOS MÉRITO DECISÃO	HTML parcialmente segmentado	NÃO	NÃO
TJAM	SIM	SIM	20	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	TXT PDF	SIM	SIM
TJBA	SIM	SIM	10	Número Processo Jurisdição Data da Distribuição Órgão Julgador Classe Judicial Assunto Partes Ementa	HTML	NÃO	SIM
TJCE	SIM			Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	TXT PDF	SIM	SIM
TJDFT	SIM	SIM	20	Classe do Processo Registro do Acórdão Número Data de Julgamento Órgão Julgador Relator Relator Designado Data da Intimação ou da Publicação Ementa Decisão Jurisprudência em Temas	PDF	NÃO	NÃO
TJES	NÃO	SIM	10	Numero Classe Relator Órgão Julgador Data do Julgamento	HTML	NÃO	NÃO

Continua na próxima página.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TJGO	SIM	SIM	1	ORIGEM ACÓRDÃO PROCESSO RELATOR REDATOR PROC./REC EMENTA PARTES REF. LEG REF. DOUT	PDF	NÃO	SIM
TJMA	SIM	NÃO	lazy loading	Sistema Número do Processo Data do registro do acórdão Relator Data de abertura Data do ementário Órgão Ementa	NÃO	SIM	NÃO
TJMT	NÃO	SIM	CONFI-GURA-VEL	Processo Julgado em Publicado em Órgão Julgador Classe Classe Feito Relator Ação Tipo do Processo Assunto Tipo de julgamento Ementa	HTML/PDF	SIM	NÃO
TJMS	SIM	SIM	100	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	TXT/PDF	SIM	SIM
TJMG	SIM	SIM	50	Processo Relator(a) Órgão Julgador / Câmara Súmula Comarca de Origem Data de Julgamento Data da publicação da súmula Ementa	PDF	NÃO	SIM
TJPA	SIM	SIM	10	Número do processo CNJ Número do documento Número do acórdão Tipo de Processo Órgão Julgador Decisão Relator Seção Data de Julgamento	HTML ou RTF	NÃO	NÃO
TJPB	SIM	SIM	10	Número Relator Órgão Julgador Data de Julgamento Ementa	PDF	NÃO	SIM

Continua na próxima página.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TJPR	SIM	SIM	10	Processo Segredo de Justiça Relator(a) Desembargador Órgão Julgador Comarca Data do Julgamento Fonte/Data da Publicação Ementa DECISÃO Íntegra do Acórdão	PDF com-pactado .zip/HTML	NÃO	SIM
TJPE	SIM	SIM	5	Processo Classe CNJ Assunto CNJ Relator(a) Órgão Julgador Data de Julgamento Data da Publicação/Fonte Ementa Acórdão Meio de Tramitação	PDF	NÃO	SIM
TJPI	SIM	SIM	45	Relator Classe Julgamento Órgão Ementa DECISÃO	PDF ou HTML ou texto	NÃO	SIM
TJRJ	NÂO	SIM	10	Numero Classe Assunto Localização Órgão Julgador Relator APELANTE APELADO	HTML, RTF e PDF	SIM	NÃO
TJRN	SIM	SIM	10	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	RTF E HTML	SIM	SIM
TJRS	SIM	SIM	10	Núm. Inteiro teor Tipo de processo Tribunal Classe CNJ Relator Órgão Julgador Comarca de Origem Seção Assunto CNJ Decisão Ementa Data de Julgamento	DOC E HTML	NÃO	NÃO
TJRO	SIM	SIM	10	Processo nº Data do julgamento Data publicação Ementa Inteiro Teor Decisão Decisão Acórdão Relatório Voto	PDF e no HTML	NÃO	SIM

Continua na próxima página.

Tribunais Regionais Federais							
Tribunal	Busca por período	Total consulta	Registros por página	atributos	Formato inteiro teor	Captcha	Link estático
TJRR	SIM	SIM	10	Processo número Orgão Julgador classe AGRAVANTE AGRAVADOS RELATOR	PDF E NO HTML	NÃO	SIM
TJSC	SIM	SIM	50	Processo Relator Origem Orgão Julgador Julgado em Classe Início do documento Citações	HTML e RTF	NÃO	NÃO
TJSP	SIM	SIM		Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	RTF E HTML	SIM	NÃO
TJSE	SIM	-	-	Numero Classe/Assunto Relator(a) Comarca Órgão julgador Data do julgamento Data de publicação Ementa	HTML	SIM	NÃO
TJTO	seleção por mes	SIM	20	Numero Classe Assunto(s) Competência Relator Data Autuação ementa inteiro teor	HTML	NÃO	NÃO
TJMMG	SIM	SIM	100	Número Referência Relator Julgamento Publicação Ementa	PDF	NÃO	SIM
TJMRS	por ano	NÃO	todos	Classe Numero Relator Ementa	PDF	NÃO	SIM
TJMSP	por ano	SIM	lazy loading	Numero Ementa Indexação Decisão	PDF	NÃO	SIM

Tabela 3.4: Características do portal de busca de jurisprudência dos Tribunais de Justiça

Por fim, os tribunais utilizados para compor a base de dados deste trabalho são:

1. STF – Superior Tribunal Federal:
    - possui um conjunto de documentos já baixados,
    - possui scripts[a] de segmentação e download prontos, que pode ser utilizado como base,
    - possui volume satisfatório de documentos,
    - é a corte Suprema do Brasil.
2. STJ – Superior Tribunal de Justiça:
    - versões preliminares de scripts de segmentação e download puderam ser adaptados para novo portal,
    - possui enorme quantidade de documentos no período,
    - é o principal e maior tribunal entre os superiores.
3. TRF2 – Tribunal Regional Federal da 2ª Região:
    - versões preliminares de scripts de segmentação e download puderam ser estudados e utilizados com base,
    - possui quantidade satisfatória de documentos,
    - portal de busca de fácil navegação.
4. TJMG – Tribunal de Justiça de Minas Gerais:
    - um dos maiores Tribunais de Justiça em quantidade de processos,
    - portal de fácil navegação.
5. TJPB – Tribunal de Justiça da Paraíba:
    - possui uma grande quantidade de layouts distintos,
    - representa um grande desafio para a anotação automática,
    - possui quantidade satisfatória de documentos.

[a] https://github.com/awillsousa/BaseITD.git

Alguns órgãos incluem segmentos não convencionais em seus documentos. O STJ, por exemplo, inclui um segmento chamado *Certidão de Julgamento*, que não está presente em acórdãos de outros tribunais. O número de segmentos considerados em todos os documentos de todos os órgãos é muito grande e, até certo ponto, desnecessário. Dessa forma, é preciso determinar quais segmentos são comuns a todos os órgãos. Também é preciso considerar aqueles que são definidos por lei como obrigatórios. Então poderá ser definido quais segmentos serão considerados neste trabalho. Os demais tipos podem ser classificados como uma classe extra, assim como cabeçalhos, rodapés e outros elementos secundários. Isso mantém a uniformização dos segmentos definindo um padrão para as classes de interesse em todos dos tribunais.

3.2 Coleta de PDFs de Acórdãos

A coleta dos dados dos tribunais é feita em duas etapas: coleta de metadados do PDF com inteiro teor do acórdão (incluindo a URL do PDF do documento) e download dos PDFs. A etapa de coleta de metadados consiste em localizar as informações referentes ao acórdão disponíveis diretamente no portal de busca do tribunal e armazenar essas informações em uma coleção MongoDB para consultas posteriores. Dentre os metadados considerados estão atributos como número do processo, relator, data de julgamento, ementa e, principalmente, a *url* do inteiro teor do acórdão. Estes atributos podem variar entre os tribunais, porém todos fornecem dados suficientes para localização posterior do acórdão no portal.

A etapa de *download* é composta, é claro, pelo download e armazenamento do arquivo.

Uma página de internet, acessada através de um navegador, é a renderização, de forma compreensível e estruturada, de um código HTML (*HyperText Markup Language*). Um arquivo HTML é a resposta para uma requisição a um servidor HTTP, onde a requisição ao servidor é feita através de um método GET ou POST, com os parâmetros definidos nesta requisição. Para capturar o método da requisição, é verificado o cabeçalho da requisição utilizando a ferramenta de inspeção de atividade de rede, presente nas ferramentas de desenvolvedor, onde é exibido o método utilizado. Por outro lado, para capturar os parâmetros da pesquisa desejada, na mesma ferramenta, é analisado o *payload* da requisição da pesquisa, onde os exatos parâmetros enviados na requisição são exibidos, como mostrado na Figura 3.1. Os elementos

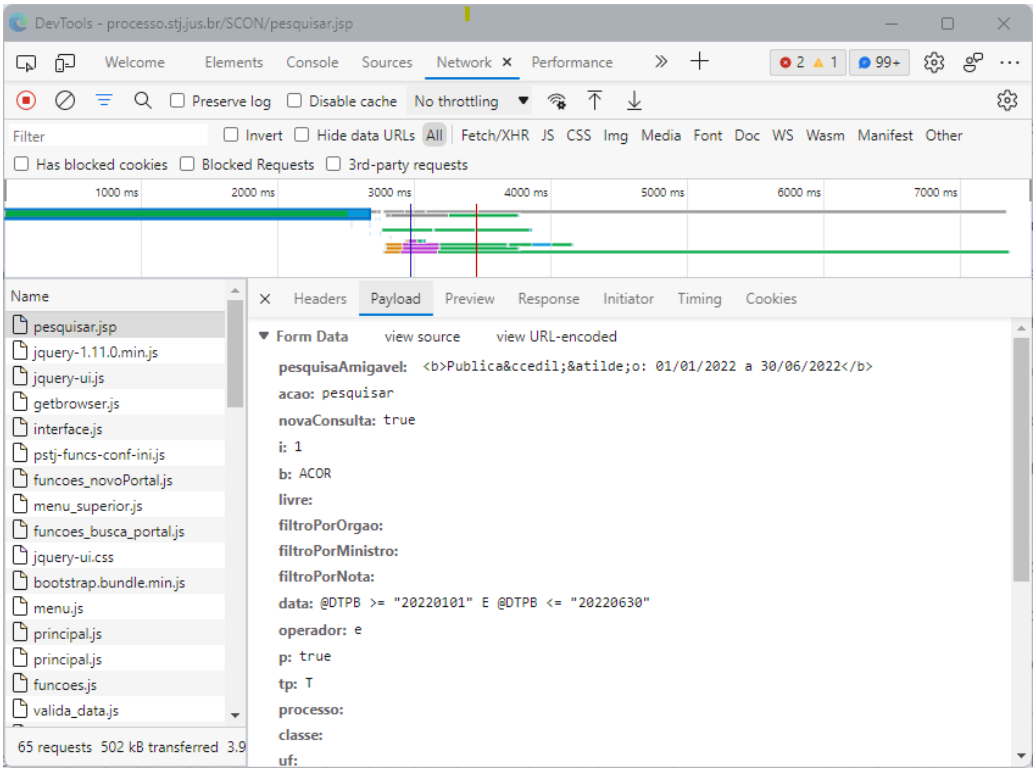


Figura 3.1: Sniffer de rede do protocolo HTTP nas ferramentas do desenvolvedor

da página, como textos, figuras e formulários, são dispostos por meio de tags que a linguagem oferece. Um exemplo simples de um documento HTML é exibido no Código 3.1.

```
1   <!DOCTYPE html>
2   <head>
3       <title>STJ Jurisprudencia do STJ</title>
4   </head>
5   <body>
6     <div class="listadocumentos">
7       <div class="documento">
8         <div class="paragrafoBRS">
9           <div class="docTitulo">Relator</div>
10          <div class="docTexto">Ministro FRANCISCO FALCAO</div>
11        </div>
12      </div>
13    </div>
14  </body>
```

Código 3.1: Parte do código HTML de um resultado de uma busca no portal do STJ

Uma tag HTML possui atributos que complementam sua função. Os atributos são propriedades que adicionam ou modificam características do elemento definido pela tag. Um documento HTML, então, pode ser entendido como uma árvore de elementos, cuja construção é denominada DOM (*Document Object Model*). Na Figura 3.2, é ilustrado o DOM do código HTML apresentado acima.

Para manipular ou verificar em que consiste um determinado elemento numa página HTML, é preciso, antes de tudo, localizar o objeto no DOM e analisar o trecho do código HTML que o representa. Para localizar o trecho do código HTML que define um elemento, é possível inspecionar o elemento da página e esta ação varia conforme o navegador e sistema operacional utilizado. A ferramenta de inspeção da página, presente nas ferramentas de desenvolvedor dos navegadores, é usada para analisar o DOM do HTML de resposta à requisição. Os padrões dos elementos no arquivo HTML, ou mesmo os estilos CSS aplicados, determinam a forma como os metadados dos acórdãos podem ser identificados. Assim, os critérios para localização dos elementos são definidos.

Uma vez identificado os padrões de funcionamento das requisições e respostas HTTP de uma página de pesquisa de um tribunal, é necessário desenvolver um script de coleta de metadados que realiza a seguinte sequência de ações:

- requisição ao portal de busca,
- análise do HTML de resposta,
- extração de metadados e
- armazenamento dos dados coletados.

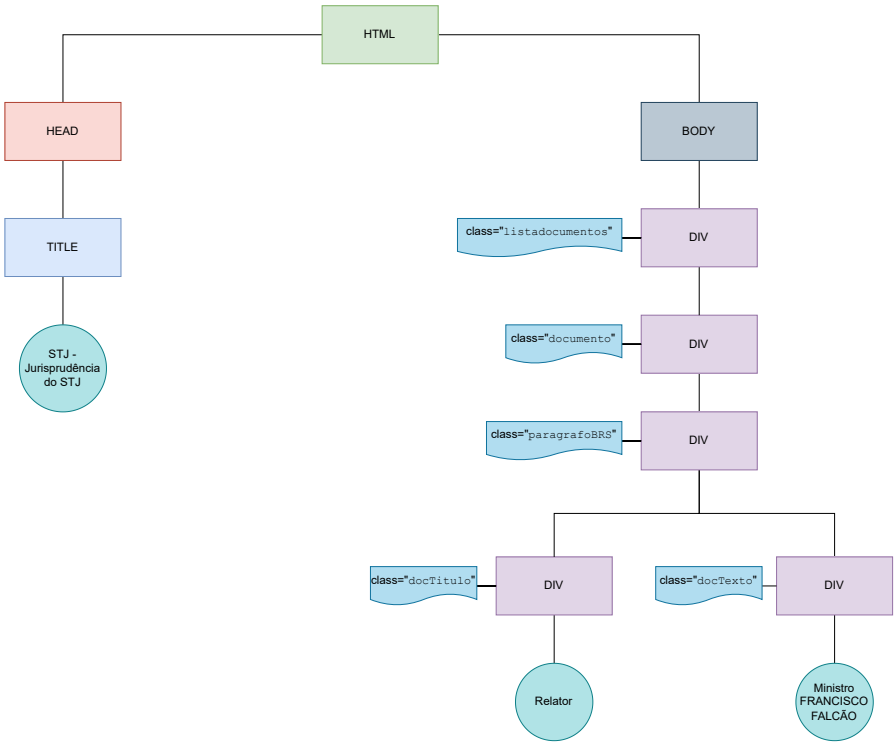


Figura 3.2: Estrutura de um arquivo HTML simples

A requisição à página de busca é feita utilizando a biblioteca `requests` do Python. Inicialmente uma requisição HTTP é enviada utilizando um método GET ou POST e indicando os parâmetros da requisição. Para reproduzir a requisição em um script utilizando a biblioteca `requests`, estes parâmetros não precisam ser fornecidos como pares de chave/valor no URL após um ponto de interrogação, como de costume. Esses argumentos podem ser formatados como um dicionário de strings, usando o argumento `params`.

No script, o conteúdo da resposta do servidor é analisado por um parser HTML. O parser retorna um DOM HTML válido com todo o conteúdo analisado. O DOM HTML, é manipulado como um objeto da biblioteca *Beatiful Soup*, que permite a navegação, busca e alteração em um DOM HTML, tornando possível localizar e separar os elementos de interesse na página facilmente com base nos padrões previamente identificados. Tomando como exemplo o STJ, utilizando o intervalo de busca de 01/01/2022 a 30/06/2022, a pesquisa é feita através do portal utilizando a data de julgamento como filtro. Para melhorar a navegação, troca-se o número de documentos por página para o maior número possível, neste caso 50. Ao inspecionar o cabeçalho da requisição, identificamos a Request URL[2] e o *Request Method: POST* e que o payload é formado pelos atributos:

- numDocsPagina: 50. É a quantidade de documentos por página.
- p: true. Indica que os termos de busca também serão buscados no plural.

[2] https://processo.stj.jus.br/SCON/jurisprudencia/toc.jsp

- b: ACOR. Representa o tipo de documento buscado, neste caso, acórdão.
- data: @DTDE >= "20220101"E @DTDE <= "20220630".Com a devida formatação, é o período de julgamento dos documentos desejados.
- pesquisaAmigavel: Julgamento: 01/01/2022 a 30/06/2022. Também é o período de julgamento dos documentos desejados.
- i: 1. É o número sequencial do documento que inicia a página, ou seja, na página 2 i é igual a 51, na 3 é igual a 105 portanto, em regra geral, o numero da página é (i-1)/l + 1.
- l: 50. É semelhante a numDocsPaginas e sempre ambos tem sempre o mesmo valor;
- tp: T. É o tipo da pesquisa, onde T é uma pesquisa por termo e P é uma pesquisa por número do processo.
- operador: e. É o operador lógico para os termos da pesquisa, neste caso o operador e indica que os resultados devem obedecer a todos os critérios especificados

É fácil identificar pelo nome a que se refere alguns atributos. Por outro lado, a função de outros atributos não é tão clara. Nestes casos, pode-se fazer uma alteração manual controlada e observar as mudanças na resposta. De todo modo, não interessa ao script alterar a maior parte desses atributos. Uma vez identificado o atributo correspondente ao filtro desejado e os controles de navegação, tem-se o suficiente para buscar todos os documentos de interesse.

Uma vez que a pesquisa e seus parâmetros foram determinados e se tem uma página de resposta a essa pesquisa, é preciso encontrar os objetos de interesse nesse HTML de resposta. Especificamente no STJ, os metadados visíveis nos resultados da pesquisa são: processo, relator, órgão julgador, data do julgamento, data publicação, ementa, acórdão e notas (são os mesmos listados na Tabela 3.1). Estes metadados são apresentados na página web, como ilustrado na Figura 3.3. Ao inspecionar os objetos da página é possível visualizar a estrutura da árvore, mostrada parcialmente na Figura 3.4, os elementos HTML e seus atributos.

Atributos HTML proveem informações adicionais sobre os elementos HTML. Essas informações podem ser determinantes na identificação do elemento de interesse. Conhecendo a estrutura dos elementos e seus atributos, pode-se isolar informações de interesse manipulando os objetos com *beautiful soup*. Os métodos find e find_all da biblioteca *beautiful soup* localiza elementos na árvore, dadas as características do elemento e seus atributos. Ao especificar uma tag HTML e um valor de atributo é possível localizar os elementos que atendem à característica. No STJ, os documentos retornados pela pesquisa estão localizados em uma tag DIV cujo atributo *class* é igual a "listadocumentos". Cada documento nesta DIV é outra div cuja *class* é "documento". Neles há outras *DIVs* da *class "paragrafoBRS"* que contém os atributos de

Figura 3.3: Exemplo de um inteiro teor resultante de uma pesquisa no portal de busca do STJ

metadados do documento, enquanto o PDF do inteiro teor é localizado no elemento *<a>*, cujo atributo *title* é "Exibir o inteiro teor do acórdão."

Uma vez coletados os metadados do processo juntamente com a URL do PDF do inteiro teor, um documento é armazenado em uma *collection* do MongoDB. Esse documento será chamado, então, documento do acórdão. Ao fim, o documento do acórdão possui os seguintes atributos:

- `_id`: identificador único do documento na coleção MongoDB.
- `pdf_url`: URL estática para download do documento PDF.
- `Processo`: identificador do processo no tribunal.
- `Relator(a)`: relator do processo.
- `Órgão Julgador`: denominação do órgão julgador, que é o conjunto de juízes responsáveis por julgar um processo.
- `Data do Julgamento`: data em que o processo foi julgado pelo órgão julgador.
- `Data da Publicação/Fonte`: data em que a decisão é tornada pública por meio de diário oficial.
- `Ementa`: resume os pontos principais e fundamentais de um acórdão.
- `Acórdão`: decisão final proferida pelo outro órgão julgador.
- `Referência Legislativa`: citação da norma jurídica que fundamenta ou é aplicável ao acórdão.
- `Acórdãos Similares`: decisões proferidas por tribunais em processos semelhantes.

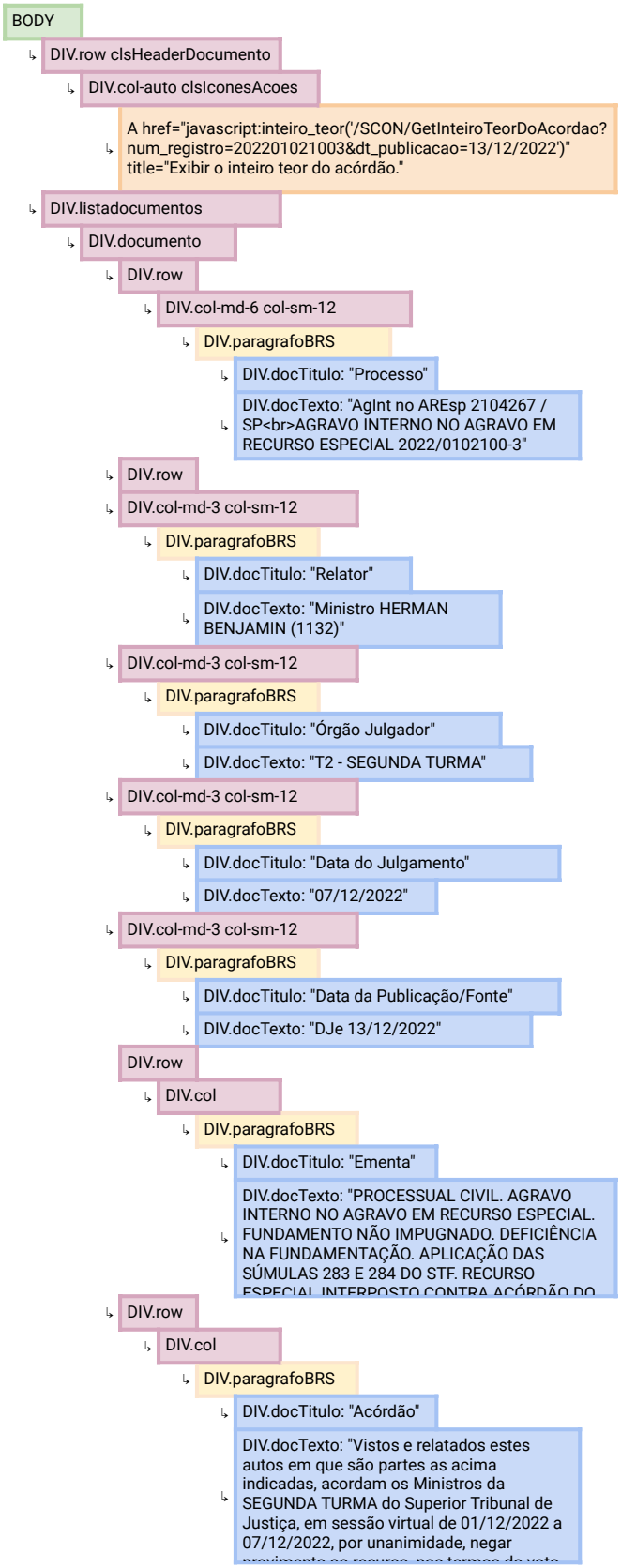


Figura 3.4: Abstração do DOM de um documento na página de resposta no portal de busca do STJ

Outros atributos de controle são adicionados conforme os dados são manipulados.

Após a coleta de todos os documentos dos acórdãos, o download dos inteiros teores é feito. Os documentos baixados são organizados de maneira a evitar uma grande quantidade de arquivos num mesmo diretório. A estratégia adotada é separar os arquivos, primeiramente por ano, em seguida por mês e, por último, por dia do mês. Um atributo de controle denominado `pdf_file` é adicionado com o local do arquivo PDF no disco.

3.3 Extração de Texto e Layout a partir de um PDF

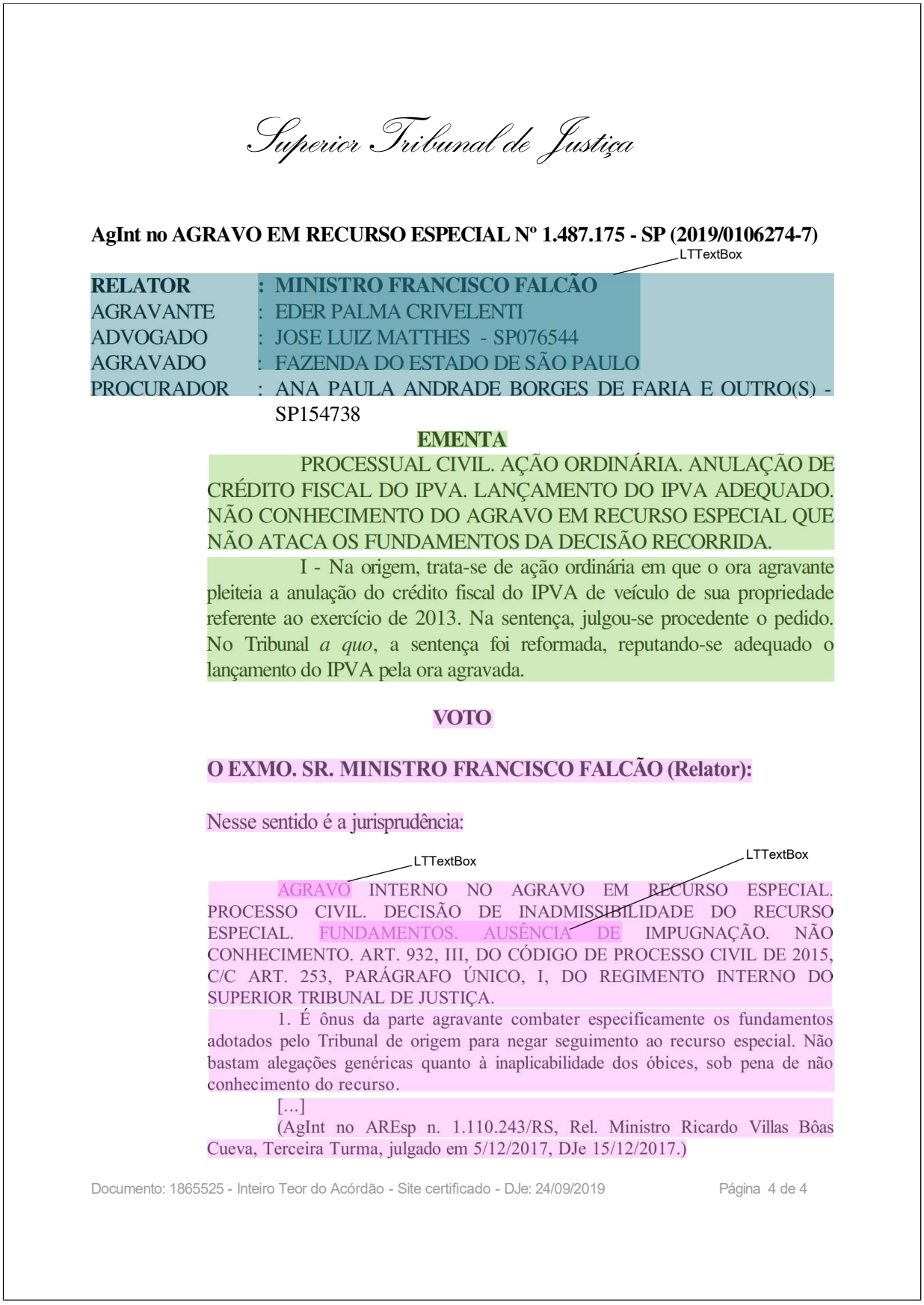
Para extrair os atributos do arquivo PDF, foi utilizada a biblioteca *pdfplumber*. Nesta biblioteca, um arquivo PDF é representado por uma lista de páginas. Uma página possui uma lista de glifos além de propriedades indicando seu número, largura e altura. A combinação estruturada das propriedades dos glifos em um objeto define uma associação coesa entre eles formando um texto legível.

Embora a *PDFPlumber* ofereça funcionalidade para extrair blocos de texto (algo similar a um parágrafo), muitos problemas nesses agrupamentos foram encontrados nos layouts dos acórdãos. Um exemplo é quando há uma detecção de um bloco de texto sobreposto a um bloco já identificado, normalmente quando a disposição das palavras fica semelhante a uma tabela ou quando a justificação do texto na largura da linha provoca um espaçamento um pouco maior entre as palavras. Na Figura 3.5 é apresentado um exemplo de documento com problemas na identificação de grupos de texto. Por isso, as informações de texto e layout não será feita em blocos de texto, apenas em linhas.

Uma linha de texto de um documento tem sua posição na página delimitada por uma *bounding box* retangular construída a partir de dois pontos extremos: um no canto superior esquerdo da linha (x_0, y_0) e outro no canto inferior direito (x_1, y_1). Além da posição, a fonte utilizada no texto e o tamanho são outros atributos de layout disponíveis. Sendo assim, o dataset é construído a partir da extração de cada uma das linhas juntamente com os atributos de layout, que são: x_0 e largura da linha, cuja soma é o valor de x_1; y_0 e altura da linha, cuja soma é o valor de y_1; número da página e fonte, todos armazenadas em coleção MongoDB, por ordem de leitura dentro do arquivo. As linhas são armazenadas como um vetor de palavras com suas informações de layout individuais também.

Considerando as características, ferramentas e dificuldades apresentadas, se destacam nos tribunais:

- STF: Os inteiros teores dos acórdãos do STF são bastante uniformes em seu layout, mesmo no decorrer de muitos anos e a organização textual não se altera em todo o período considerado. Por outro lado, em muitos documentos, efeitos visuais de sombreamento produziram erros como duplicação de caracteres e linhas na extração. A função dedupe_chars com tolerância 0.1 resolveu o problema de caracteres duplicados e a função obtain_word_tokens limita a extração apenas para texto. Além disso, nem todos os arquivos são nato-digitais



Superior Tribunal de Justiça

AgInt no AGRAVO EM RECURSO ESPECIAL Nº 1.487.175 - SP (2019/0106274-7)

LTTextBox

RELATOR	:	**MINISTRO FRANCISCO FALCÃO**
AGRAVANTE	:	EDER PALMA CRIVELENTI
ADVOGADO	:	JOSE LUIZ MATTHES - SP076544
AGRAVADO	:	FAZENDA DO ESTADO DE SÃO PAULO
PROCURADOR	:	ANA PAULA ANDRADE BORGES DE FARIA E OUTRO(S) - SP154738

EMENTA

PROCESSUAL CIVIL. AÇÃO ORDINÁRIA. ANULAÇÃO DE CRÉDITO FISCAL DO IPVA. LANÇAMENTO DO IPVA ADEQUADO. NÃO CONHECIMENTO DO AGRAVO EM RECURSO ESPECIAL QUE NÃO ATACA OS FUNDAMENTOS DA DECISÃO RECORRIDA.

I - Na origem, trata-se de ação ordinária em que o ora agravante pleiteia a anulação do crédito fiscal do IPVA de veículo de sua propriedade referente ao exercício de 2013. Na sentença, julgou-se procedente o pedido. No Tribunal *a quo*, a sentença foi reformada, reputando-se adequado o lançamento do IPVA pela ora agravada.

VOTO

O EXMO. SR. MINISTRO FRANCISCO FALCÃO (Relator):

Nesse sentido é a jurisprudência:

LTTextBox

LTTextBox

AGRAVO INTERNO NO AGRAVO EM RECURSO ESPECIAL. PROCESSO CIVIL. DECISÃO DE INADMISSIBILIDADE DO RECURSO ESPECIAL. FUNDAMENTOS. AUSÊNCIA DE IMPUGNAÇÃO. NÃO CONHECIMENTO. ART. 932, III, DO CÓDIGO DE PROCESSO CIVIL DE 2015, C/C ART. 253, PARÁGRAFO ÚNICO, I, DO REGIMENTO INTERNO DO SUPERIOR TRIBUNAL DE JUSTIÇA.

1. É ônus da parte agravante combater especificamente os fundamentos adotados pelo Tribunal de origem para negar seguimento ao recurso especial. Não bastam alegações genéricas quanto à inaplicabilidade dos óbices, sob pena de não conhecimento do recurso.

[...]

(AgInt no AREsp n. 1.110.243/RS, Rel. Ministro Ricardo Villas Bôas Cueva, Terceira Turma, julgado em 5/12/2017, DJe 15/12/2017.)

Figura 3.5: Exemplo de erros de agrupamento do PDFMiner em recortes de um PDF. O algoritmo identifica grupos de texto dentro de outro grupo maior.

e não é possível isolar estes arquivos. Não há um ponto em que os digitalizados deixam de existir. Para este problema não há uma solução viável e estes arquivos serão considerados ruídos a base de doumentos. Alguns documentos possuem diferenças sutis em breakpoins. Todas as diferenças detectadas foram cobertas pelos breakpoints do script de anotação automática. Além dos segmentos padrão, este tribunal apresenta um segmento extra em seus inteiros teores denominado *EXTRATO DE ATA*.

- STJ: Os inteiros teores dos acórdãos do STJ são bastante uniformes em seu layout, mesmo no decorrer de muitos anos. Os PDFs possuem objetos para efeitos visuais e sem função informativa. Ao fazer a extração é preciso lidar com esses objetos (sombras, códigos e marcações), anotando como *OUTROS*. Não foram localizados aquivos não nato-digitais neste tribunal.

  Os inteiros teores possuem os segmentos *Certidão*, *Certidão de Julgamento* e *Autuação*, não comuns em outros tribunais. Apesar de ser fácil identifica-las no script e esta anotação estar presente no *dataset*, para os modelos produzidos, serão consideradas como *OUTROS*.

- TRF2 Os documentos tem variações no layout que não seguem um padrão. Documentos diversificam espaçamento entre linhas e caracteres, fonte, alinhamento. Outras características como sublinhado e negrito em delimitadores de segmentos também variam. Possui os segmentos padrão de todos os tribunais e todos os documentos são nato digitais.

- TJMG Os inteiros teores dos acórdãos são bastante uniformes em seu layout, mesmo no decorrer de muitos anos. No entanto, a organização textual se altera bastante em todo o período considerado. Delimitadores(*breakpoints*) de seção mudam ou deixam de existir, o que dificulta bastante a identificação de determinados segmentos, mesmo em anotação manual para um leitor leigo. Em grande parte dos documentos não há uma identificação expressa do início do segmento VOTO após o RELATÓRIO, onde, na maioria desses casos apenas uma leitura consciente é capaz de identificar onde termina o RELATÓRIO e começa o VOTO. Algumas expressões podem ser utilizadas para fazer a identificação mas isso não é possível em todos os casos, pois o texto é livre. Isso se reflete no desempenho do script de anotação automática na identificação do segmento *VOTO*. Isso poderia ser melhorado com algum pre processamento baseado em IA, como análise de similaridade, mas a ideia é que os scripts de anotação automática não tomem tempo nem se demande muito esforço para manutenção deles.

  Não foram identificados arquivos não nato-digitais

- TJPB Os inteiros teores dos acórdãos são completamente variados em seu layout. Cada gabinete de desembargador utiliza um layout próprio. Portanto são dezenas de layouts diferentes para este tribunal. Certamente há layouts que sequer foram identificados em meio aos milhares de documentos. Como não há um padrão estabelecido, cada arquivo pode ter um layout diferente. Entre os

layouts identificados, há, pelo menos, os mesmos segmentos. Foram utilizados *breakpoints* mais genéricos possível Também há arquivos digitalizados com ocr (*Optical Character Recognition*) e sem ocr.

Após os ajustes de extração, os atributos extraídos dos arquivos PDF são:

- Local do arquivo
- Lista de páginas com:
  - Número da página
  - Lista de linhas com:
    - x
    - y
    - Largura
    - Altura
    - Lista de tokens com:
      - Texto
      - x
      - y
      - Nome da fonte
      - Largura
      - Altura

Cada documento corresponde a um arquivo JSON com este formato que é armazenado na coleção de documentos do MongoDB.

3.4 Anotação Automática

A anotação automática de um acórdão é uma etapa posterior ao processo de extração de texto e layout de um arquivo PDF. Essa tarefa é feita através de um script em Python que identifica padrões específicos do acórdão. O objetivo da anotação automática é reduzir a necessidade de anotação manual e possibilitar a rápida anotação de uma grande quantidade de documentos. A entrada do problema é uma coleção de n documentos $D = (d_1, d_2, \ldots, d_n)$, composto por m linhas $L = (l_1, l_2, \ldots, l_m)$, de 5 tribunais $T \in$ {STF, STJ, TRF2, TJMG, TJPB}. O objetivo nesta etapa é construir um anotador automático de documentos com alta precisão a partir dessas entradas, atribuindo uma anotação s_i, onde $s_i \in S =$ {PARTES, EMENTA, ACORDAO, VOTO, RELATORIO, ..., OUTROS}, para cada linha $l_i \in L$.

Para anotação automática dos documentos foi adotado um sistema de *breakpoints*, onde características esperadas no texto determinam uma troca de segmento. Um *breakpoint* é um ponto em d onde se estima uma troca de segmento. A cada tribunal t é atribuído um conjunto de p *breakoints* $B = (b_1, b_2, \ldots, b_p)$ onde b pode ser identificado

através de características de texto, posicionamento (x ou y), largura ou altura da linha, uma *feature* por vez. O documento d é lido a partir de l_1 até l_m. Cada linha l_i é analisada e comparada a B. Um rótulo de segmento inicial é definido (geralmente *OUTROS*) que vai sendo atribuído às linhas até que l_i seja correspondente a b_j, então a anotação do segmento é alterada. Um breakpoint b_j define se a anotação será atribuída somente a l_i, a $(l_i, l_{i+1}, \ldots)$ ou a $(l_{i+1}, l_{i+2}, \ldots)$.

Um *breakpoint* então é composto por

- Feature: caraterística da linha do arquivo pdf que deseja-se comparar
- Valor esperado: valor da *feature* que determina a correspondência verdadeira
- Segmento atual: rótulo que vinha sendo atribuído à linha corrente. Um coringa (*) pode ser usado.
- Segmento: rótulo que vai ser atribuído a partir de então
- Anotação única: indica se a anotação será apenas para l_i ou para as linhas seguintes também.
- Inclui atual: se a anotação única é falsa, indica se a anotação ocorre na linha atual ou apenas a partir das próximas.
- Funçao: função de anotação que compara a *feature* da entrada com o valor esperado.

A combinação das correspondências verdadeiras do valor esperado com o segmento atual determinam a correspondência de l_i e b_j. Quando a *feature* é o texto, a comparação é feita através de expressões regulares que casem com o texto que indica a quebra de segmento. Por exemplo: se o *breakpoint* indica o início do segmento VOTO e o texto pode ser VOTO, voto, v o t o, Voto ou Voto: uma expressão regular (V ?O ?T ?O ?:?) não sensível a maiúsculas e minúsculas, deve indicar bem essa quebra. Portanto:

- feature: text
- valor esperado: r'(V ?O ?T ?O ?:?)'
- segmento atual: *
- Segmento: VOTO
- Anotação única: FALSO
- inclui atual: VERDADEIRO
- função: re.compile(valor esperado, re.IGNORECASE).match(l_i.text)

Assim, para anotar documentos de um novo tribunal, basta identificar os *breakpoints* e executar o *scprit* sobre o PDF utilizando a lista de *breakpoints*. Apesar desta estrutura usada para anotação automática ser a mesma para todos os tribunais, a definição de *breakpoints* é específica para cada tribunal. Além disso, definir *breakpoints* com boa acurácia não é uma tarefa simples.

3.5 Anotação Manual

Para avaliar o desempenho, tanto de classificadores baseados em AM quanto da anotação automática, é necessário usar um dataset com anotações de alta qualidade. Esse tipo de anotação é conhecido como *gold labels* (Raykar et al., 2010) e, geralmente, precisa ser realizada manualmente. Algoritmos de segmentação baseados em regras e desenvolvidos especificamente para um único layout de documento podem ter uma qualidade alta, até próxima da anotação manual, mas é preciso garantir que um dataset produzido dessa forma é bom o suficiente para as etapas de treinamento dos modelos. Os ruídos de um dataset anotado automaticamente podem ser aprendidos pela rede e não poderão ser detectados se validados em um dataset com os mesmos padrões de ruído. Assim, para cada tribunal, um conjunto de documentos foi revisado manualmente.

A anotação manual de documentos foi realizada revisando documentos anotados automaticamente e corrigindo os erros do script de anotação através de um script Python escrito para esta finalidade. Uma página do PDF é exibida com cada linha anotada pelo script de anotação automática e o revisor seleciona as linhas anotadas incorretamente pelo script e corrige a classificação. Os arquivos anotados manualmente tem o objetivo de validar modelos e scripts a partir de um pequeno *dataset* com *golden labels*. Dois conjuntos de arquivos foram anotados manualmente: um conjunto de validação e um conjunto de teste.

3.6 Estatísticas dos Datasets

Os documentos adquiridos para cada tribunal foram subdivididos nos datasets a seguir com diferentes finalidades:

- **TESTE**: Documentos com anotação manual (*golden labels*) para teste final dos modelos produzidos. Usado para avaliar a capacidade do modelo em generalizar para novos dados e importante para garantir que o modelo seja preciso, robusto e eficaz em termos de suas métricas de desempenho.

- **VAL-M**: Documentos com anotação manual (*golden labels*) para avaliação dos modelos em desenvolvimento. E usado para avaliar o desempenho do modelo durante o treinamento. A validação é usada para ajustar os hiper parâmetros do modelo, garantindo que ele não esteja superajustando ou subajustando aos dados de treinamento.

- **VAL-A**: Documentos com anotação automática baseada em heurísticas para avaliação dos modelos em desenvolvimento. Embora este conjunto não tenha sido usado em nenhum experimento deste trabalho, é considerado importante mantê-lo. A validação em um dataset igualmente ruidoso em relação ao treino permite uma melhor compreensão do modelo e suas limitações permitindo o ajuste do modelo para lidar com o ruído e outras variações nos dados. O mo-

delo, por exemplo, pode ter sido superajustado (*overfitted*) ao conjunto de treinamento ruidoso e não ser capaz de lidar com o ruído em novos dados. Além disso, a validação em um dataset ruidoso pode ajudar a identificar padrões de erro específicos do modelo, o que pode fornecer percepções valiosas para melhorar a qualidade do conjunto de dados e ajustar o modelo para lidar melhor com dados ruidosos.

- **TREINO MINI**: Documentos com anotação *automática* com a finalidade de treinar os modelos de AM. A quantidade de documentos neste *dataset* é reduzida para rodadas rápidas de treinamento a fim de possibilitar a experimentação e prototipagem de ideias de forma ágil. Com um dataset menor, o processo de modelagem pode ser mais fácil e rápido, o que permite iterar com maior rapidez sobre as soluções e testar diferentes abordagens de modelagem. Isso pode ser especialmente útil em fases de experimentos, quando se busca validar a viabilidade de uma solução ou explorar alternativas para problemas complexos de maneira mais eficiente.

- **TREINO**: Documentos com anotação automática baseada em heurísticas com a finalidade de treinar os modelos de AM. A quantidade de documentos neste *dataset* é grande para rodadas de treinamento completas quando se deseja obter o melhor desempenho do modelo e melhor generalização.

Os documentos de cada subdivisão foram sumarizados e a composição de cada uma em páginas e linhas dos documentos. Foram calculados também os desempenhos dos scripts de anotação automática, bem como a distribuição dos segmentos identificados pelos scripts nos *datasets* TREINO, TESTE e VAL-A, onde a classificação é dada pelos scripts. A anotação dos segmentos nos *datasets* TESTE e *VAL-M* é feita manualmente, portanto é uma anotação altamente confiável.

Na Tabela 3.5, são apresentados os quantitativos de arquivos dos cinco *datasets* desenvolvidos, onde são destinados 80 arquivos para compor o *dataset* TESTE, 20 arquivos para compor o VAL-M, 400 para VAL-A, 1000 para o TREINO MINI e o restante dos arquivos disponíveis para o TREINO. Como cada arquivo contém quantidades de páginas e linhas diferentes, é exibido também seus quantitativos, onde é possível observar que a proporção de páginas por arquivo e linhas por páginas é equilibrada e a quantidade de linhas, que é a quantidade final de exemplos, em cada *dataset* nos diferentes tribunais não se distancia muito. A proporção de exemplos de cada segmento nos *datasets* também é equilibrada, sendo o destaque para o segmento *OUTROS* nos tribunais STF e STJ acima da média dos outros tribunais. Isso se dá pelo fato desses tribunais possuir segmentos adicionais, que foram sumarizados, neste caso, como *OUTROS*. Vale ressaltar também que o segmento *PARTES* tem uma representação maior no STF porque este segmento é repetido ao iniciar um novo segmento do inteiro teor e não apenas na primeira página como nos demais tribunais. Na Figura 3.6, são apresentas as proporções dos segmentos em cada *dataset*.

Na Tabela 3.6, por sua vez, o desempenho *f1-score* por segmento dos scripts de anotação automática avaliado é demonstrado. Nestes resultados é claramente percep-

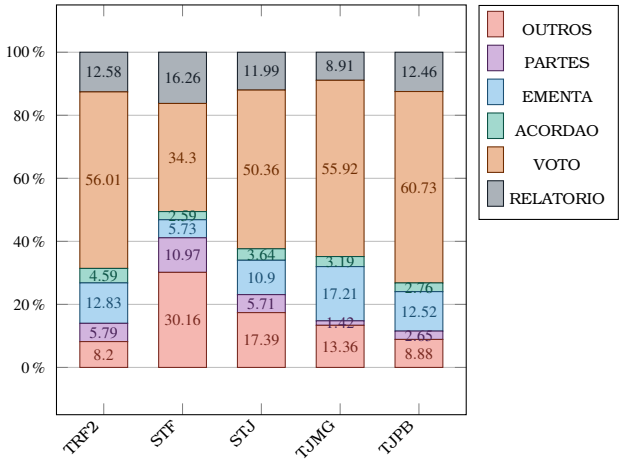
Split	Documentos	Páginas	Linhas
		STF	
TESTE	80	787 (9,84/doc)	22.320 (28,36/pág)
VAL-M	20	306 (15,30/doc)	9.962 (32,56/pág)
VAL-A	400	4.192 (10,48/doc)	128.348 (30,62/pág)
TREINO MINI	1.000	10.255 (10,26/doc)	309.938 (30,22/pág)
TREINO	48.843	514.393 (10,53/doc)	15.380.477 (29,90/pág)
		STJ	
TESTE	80	832 (10,40/doc)	29.259 (35,17/pág)
VAL-M	20	220 (11,00/doc)	7.511 (34,14/pág)
VAL-A	400	4.203 (10,51/doc)	146.775 (34,92/pág)
TREINO MINI	1.000	10.064 (10,06/doc)	349.695 (34,75/pág)
TREINO	396.020	3.944.159 (9,96/doc)	136.910.911 (34,71/pág)
		TRF2	
TESTE	80	545 (6,81/doc)	18.177 (33,35/pág)
VAL-M	20	160 (8,00/doc)	5.507 (34,42/pág)
VAL-A	400	2.788 (6,97/doc)	92.046 (33,02/pág)
TREINO MINI	1.000	7.086 (7,09/doc)	233.309 (32,93/pág)
TREINO	116.570	795.984 (6,83/doc)	26.239.121 (32,96/pág)
		TJPB	
TESTE	80	639 (7,99/doc)	23.506 (36,79/pág)
VAL-M	20	167 (8,35/doc)	6.278 (37,59/pág)
VAL-A	400	3.325 (8,31/doc)	124.149 (37,34/pág)
TREINO MINI	1.000	8.629 (8,63/doc)	317.277 (36,77/pág)
TREINO	12.309	100.358 (8,15/doc)	3.686.559 (36,73/pág)
		TJMG	
TESTE	80	705 (8,81/doc)	18.856 (26,75/pág)
VAL-M	20	196 (9,80/doc)	5.425 (27,68/pág)
VAL-A	400	3.526 (8,81/doc)	93.434 (26,50/pág)
TREINO MINI	1.000	8.708 (8,71/doc)	232.422 (26,69/pág)
TREINO	42.919	377.877 (8,80/doc)	10.044.126 (26,58/pág)
TOTAL	**624.161**	**5.800.104 (9,29/doc)**	**194.435.388 (35,94/pág)**

Tabela 3.5: Quantitativo dos documentos nos *datasets*

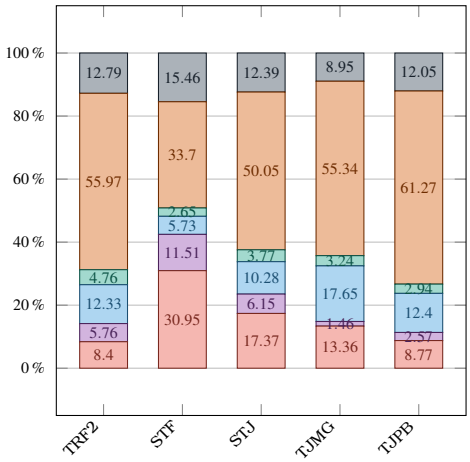
tível o impacto das dificuldades de identificação de alguns segmentos já mencionados nesta Seção. No STF, a presença dos arquivos digitalizados a partir de documentos físicos pode ter prejudicado o desempenho do script. Além disso o uso comum de citações de ementas de outros processos em segmentos como *VOTO* e *RELATORIO*, também pode ter causado confusão na anotação automática. Quanto ao TJPB, a grande variação de *layouts* certamente fez com que a cobertura dos *breakpoints* definidos para este tribunal não fosse suficiente pra cobrir os diferentes documentos satisfatoriamente. Já no TJMG, a ausência de um *breakpoint* entre os segmentos *RELATORIO* e *VOTO* em boa parte dos documentos prejudicou muito a anotação deste segmento.

Tribunal	Segmento						Métricas		
	PARTES	EMENTA	ACORDAO	RELATORIO	VOTO	OUTROS	Precision	Recall	F1-micro
STF	0.992	0.780	0.961	0.735	0.990	0.908	0.93	0.92	0.92
STJ	0.919	0.993	0.969	0.926	0.983	0.992	0.98	0.98	0.98
TRF2	0.880	0.996	0.905	0.937	0.890	0.985	0.97	0.96	0.96
TJPB	0.680	0.880	0.751	0.782	0.882	0.946	0.91	0.90	0.90
TJMG	0.994	0.922	0.960	0.938	0.547	0.823	0.90	0.79	0.85

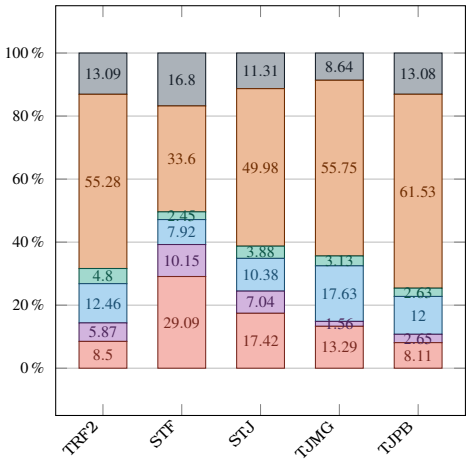
Tabela 3.6: Desempenho *F1-score* dos scripts por segmento com validação no *dataset* TESTE e desempenho geral *Precision*, *Recall* e *F1_micro*



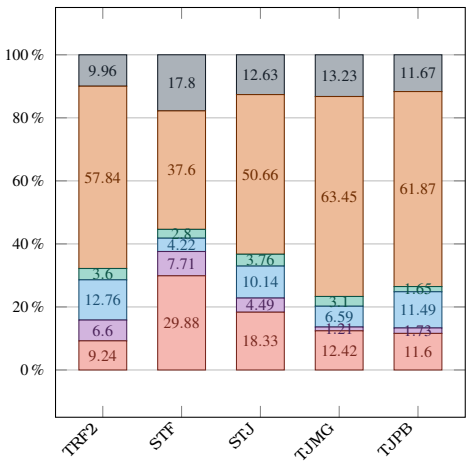
(a) *Dataset* : **TREINO**



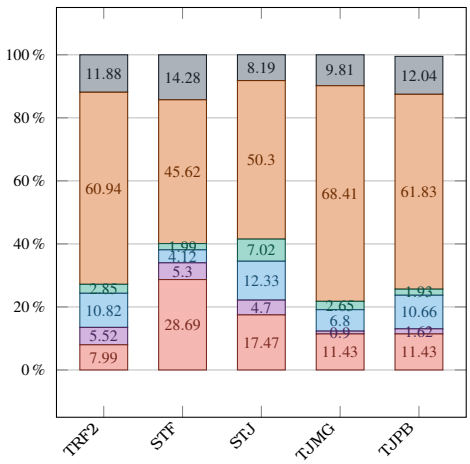
(b) *Dataset*: **TREINO MINI**



(c) *Dataset*: **VAL-A**



(d) *Dataset*: **TESTE**



(e) *Dataset*: **VAL-M**

Figura 3.6: Distribuição das classes nos datasets por tribunal

CAPÍTULO

4

Trabalhos Relacionados

Neste capítulo, são discutidos alguns trabalhos que tratam do problema de segmentação de documentos em PDF. Os trabalhos relacionados listados neste capítulo são relacionados à área de processamento de linguagem natural (NLP) e incorporação de informações adicionais em modelos profundos. No entanto, esses trabalhos não estão diretamente relacionados com a segmentação de documentos jurídicos baseada no conteúdo do documento, com a exceção do ITD, discutido na Seção 4.1. Eles se concentram mais em separar o texto em unidades lógicas e extrair metadados comuns em documentos científicos do que na segmentação baseada na compreensão do significado do texto em si. Eles abordam tópicos diferentes e possuem uma relação fraca com este. Isso dificulta uma análise crítica mais contundente, pois não há um conjunto comum de questões a serem avaliadas. Ainda assim, esses trabalhos fornecem boas abordagens para o processamento de linguagem natural e incorporação de informações adicionais em modelos de linguagem profunda, como *Layout 2D*, detalhada na Seção 4.2, imagem, detalhada na Seção 4.4 ou ambos, detalhada na Seção 4.3 e que podem ser úteis para aprimorar a segmentação de documentos em geral, incluindo documentos jurídicos.

4.1 Segmentação de Documentos Jurídicos

O ITD (Iudicium Textum Dataset) é uma base de dados de acórdãos do STF (Supremo Tribunal Federal) construída a partir de mais de 50 mil acórdãos publicados entre 2010 a 2018. O estudo considerou apenas documentos nato digitais e a informação da estrutura, incluída no documento, que indica a página de início de cada segmento. Essa informação de estrutura foi verificada na maior parte dos documentos do período.

O programa de segmentação foi desenvolvido utilizando a biblioteca Apache PDF-

Box[1] na linguagem Java para ler o arquivo PDF e convertê-lo em texto no formato HTML. Utilizando a estrutura embutida no arquivo e regras heurísticas, o inteiro teor é segmentado gerando arquivos separados para cada segmento. O trabalho considerou apenas seções típicas do STF, sendo elas: dados do acórdão, relatório, votos e extrato da ata, onde o que se chama "dados do acórdão"corresponde aos segmentos Partes, Ementa e Acórdão. O programa, no entanto, exclui partes do documento não se enquadram em nenhuma das seções consideradas. Isso impossibilita a recuperação dessas informações no dataset, mesmo que tenham sido excluídas por erro do script.

O arquivo gerado é nomeado de forma a identificar qual é o segmento e a qual inteiro teor pertence, bem como a sua localização no arquivo PDF original. O formato do nome do arquivo favorece a localização do inteiro teor do acórdão além de consultas combinadas através de comparação ou agrupamento de seções correspondente ao mesmo acórdão.

4.2 Modelos Profundos de Linguagem com Incorporação de Layout 2D

O ScienceParse (Ammar et al., 2018) executa um pré-processamento em documentos PDF a fim de fornecer subsídios na resolução do problema de extração de dados estruturados de documentos científicos, para que sejam utilizados em tarefas de PLN ou para otimizar a classificação de resultados em pesquisa acadêmica. A tarefa é desenvolvida através de um grafo de literatura, que é um grafo de propriedade com arestas direcionadas. Em grafos de propriedades, nós e arestas têm uma estrutura interna que é mais adequada para representar tipos de dados complexos, como papéis e entidades. Os nós do grafo podem ser do tipo: documento, autor, entidade ou menção à entidade. Os documentos são associados a um conjunto de atributos como título, resumo, texto completo, locais e ano de publicação.

Embora algumas das fontes dos documentos forneçam esses atributos como metadados, em outras é necessário extraí-los diretamente do PDF. O campo autor é único, com atributos como nome e sobrenome. Cada nó do tipo entidade representa um conceito científico único discutido na literatura, com atributos como nome canônico, alias e descrição. As arestas podem ser do tipo citação, autoria, ligação de entidades, menção-menção e entidade-entidade.

O ScienceParse extrai os metadados não oferecidos diretamente dos arquivos PDF. O texto do PDF é extraído utilizando a biblioteca Apache PDFBox e convertido para uma sequencia de *tokens*, cada um composto de suas características, como texto, tamanho da fonte, largura do espaço e a posição na página. As características são normalizadas antes de servir como entrada para um modelo de AM. Para cada característica tem-se três valores normalizados: um em relação à página, um em relação ao documento e, por fim, um em relação ao corpus, cada um variando entre -0.5 e

[1]https://pdfbox.apache.org/

0.5. A posição do token na página é representada por pontos de coordenadas com eixos X e Y, escalados linearmente para variar no intervalo entre os pontos (-0,5, 0,5) no canto superior esquerdo da página a (0,5, 0,5) no canto inferior direito.

Para obter as informações de maiúsculas e minúsculas, sete características numéricas são adicionadas à representação de entrada de cada token: se a primeira ou segunda letra é maiúscula ou minúscula, a proporção de letras maiúsculas e minúsculas e a proporção de dígitos. Isso ajuda o modelo a fazer previsões corretas para metadados que tendem a aparecer no início ou no final dos artigos.

Embedding de palavras são inicializados com o GloVe (Pennington et al., 2014). Essas entradas são inseridas em um modelo com uma camada totalmente conectada e, em seguida, alimenta um LSTM bidirecional de duas camadas. O modelo é treinado em um snapshot dos dados no PubMed Central. Consiste em 1,4 milhões de PDFs e seus metadados associados, que especificam os títulos, autores e bibliografias corretos. Também foi utilizado um processo de anotação heurística que encontra as strings dos metadados nos PDFs tokenizados para produzir tokens anotados.

Ainda assim o Science Parse analisa apenas seções genéricas em documentos científicos. Basicamente identifica os metadados título, autor e Bibliografias, onde cada entrada de bibliografia tem os seguintes campos: Título, Autores, Local e Ano.

4.3 Modelos Profundos de Linguagem com Incorporação de Imagem

Corpus Conversion Service (Staar et al., 2018) é um sistema que permite aos usuários analisar e anotar documentos, treinar algoritmos de classificação de AM e converter um arquivo PDF ou documentos de bitmap em um formato de representação de conteúdo estruturado. O sistema consiste em um conjunto de microsserviços organizados em cinco componentes principais. Cada um desses microsserviços pode ser consumido por sua própria API REST. Este pipeline de processamento é formado por cinco componentes:

1. conversão do documentos em um formato interno otimizado para AM: Encontrar as caixas delimitadoras de todos os fragmentos de texto que aparecem em cada página do PDF.

2. Anotação dos rótulos em documentos analisados: Neste componente, são coletadas verdades fundamentais através de anotação humana sobre os modelos de aprendizado de máquina personalizados para treinamento.

3. Treinar modelos de AM com as anotações adquiridas: Um modelo projetado para funcionar independentemente do layout do documento. São capazes, a partir de uma imagem da página, identificar e localizar objetos básicos, como tabelas, figuras, fórmulas

4. Aplicação do modelo de AM personalizado: O modelo dá suporte ao treinamento de modelos personalizados para layouts específicos, que são projetados para se

especializar em um modelo de layout específico e nos permite converter e extrair os dados de documentos com alta precisão. Eles classificarão cada fragmento da página de acordo com seu rótulo semântico de layout.

5. Montagem do documento em um formato de dados estruturados como JSON ou XML: Neste componente, um arquivo de dados estruturados em formato JSON ou XML é construído, contendo o texto e objetos do documento original, mantendo a semântica do layout. Esse arquivo é formado pela combinação de cada fragmento do arquivo analisado e seus rótulos semânticos de layout, preditos ou reais, associados. Nenhum aprendizado de máquina é usado neste componente. O processamento é baseado em regras heurísticas e, portanto, completamente determinístico.

Estes modelos de AM, treinados para modelar os textos de entrada, são então usados para classificar as categorias do *token*. Esse tipo de abordagem baseada apenas em texto obtém menor precisão quando comparado aos modelos que consideram aspectos visuais do documento.

Hao et al. (2016) processam uma imagem do PDF usando um modelo de visão computacional na tentativa de identificar e capturar, a partir de uma imagem da página de um documento, tabelas com sua posição delimitada, mas não são usados para segmentar um documento. Seu método consiste em três procedimentos principais: identificar as áreas semelhantes a tabelas, redes neurais convolucionais e adição de informações contidas em documentos PDF originais. As redes convolucionais usam a imagem das áreas identificadas como entrada. Além das imagens, informações de layout do documentos PDF original, como as coordenadas dos caracteres e o ponto inicial e final de uma linha nas tabelas são extraídos e transformados em vetores e utilizados para melhorar ainda mais os resultados.

Os métodos de AM baseados em texto, em geral, tratam problemas de classificação e dependem de um processo prévio de extração que converte o arquivo PDF em texto. Posteriormente, a sequência de tokens é utilizada como entrada.

4.4 Modelos profundos de Linguagem com Incorporação de Layout 2D e Imagem

LayoutLMv3 (Huang et al., 2022) tem sua arquitetura baseada em *Transformers*, LayoutLM (Xu et al., 2020), *BERT Pre-Training of Image Transformers* (Bao et al., 2021), *Document Image Transformer(DiT)* (Li et al., 2022) e *Detectron2* (Yuxin et al., 2019). Seu diferencial é o acréscimo de 2 novos embeddings de entrada em relação a modelos de linguagem como o BERT: um embedding de posição 2D e um embedding de imagem. A posição 2D da imagem é definida por 2 pares ordenados x e y que correspondem à localização exata de um *token* na página do arquivo. O ponto (x_0, y_0) corresponde ao ponto mais acima e a esquerda enquanto o ponto (x_1, y_1) corresponde ao ponto mais abaixo e à direita da caixa delimitadora da palavra. Com a posição de

cada palavra delimitada, é possível extrair uma 'foto' da palavra correlacionando cada uma à sua correspondente no texto, formando um embedding de imagem na representação da linguagem, além de embeddings usando a imagem inteira do documento como a região de interesse correspondendo ao *token* $[CLS]$.

O LayoutLMv3 aplica um transformer multimodal de imagem de texto para aprender representações unificadas. O *transformer* tem uma arquitetura multicamadas e cada camada consiste principalmente em redes de *multihead attention* e redes neurais diretas totalmente conectadas em posição. A entrada do *transformer* é uma concatenação de sequências de *embeddings* de texto $Y = y_{1:M}$ e *embeddings* de imagem $X = x_{1:L}$, onde L e M são o comprimento da sequência de texto e imagem, respectivamente. Através do *transformer*, a última camada do sistema produz as representações contextualizadas de texto e imagem.

As *embeddings* de texto, na verdade, são uma combinações de *embeddings* de palavras e *embeddings* de posição. As *embeddings* de posição somam *embeddings* de posição 1D, que é a posição da palavra na sequência da sentença, e posição de layout 2D que é referente às coordenadas x e y da caixa delimitadora do texto. As coordenadas x e y são normalizadas de acordo com o tamanho da imagem. LayoutLMv3 adota posições de layout em nível de segmento, onde as palavras em um segmento compartilham a mesma posição 2D, pois pressupõe-se que as palavras em um mesmo bloco de texto expressam o mesmo significado semântico.

As imagens de documentos são representadas como *patches* de imagem com recursos de projeção linear antes de alimentar um *transformer* multimodal. Essas imagens são redimensionadas em um formato $H \times W$ e referenciada por $\mathbf{I} \in \mathbb{R}^{CxRxW}$, onde C, H e W são o tamanho do canal, largura e altura da imagem, respectivamente. Em seguida a imagem é fracionada em uma sequência de *patches* uniformes P x P que são projetados linearmente em D dimensões e as alinhadas em uma sequência de vetores, cujo comprimento resultante é $M = HW/P^2$. Por fim, os *embeddings* de posição 1D a cada *patch* são adicionados. Assim, de acordo com os resultados experimentais, o LayoutLMv3 alcança o estado da arte tanto em tarefas centradas em texto quanto em tarefas centradas em imagem em inteligência de documentos.

Segmentação de documentos PDF é um problema que vem sendo amplamente estudado, seja com extração baseadas em regras ou heurísticas ou técnicas de aprendizado de máquina. No entanto, a maioria destes estudos concentra-se na extração de metadados de artigos e outros documentos científicos, quase sempre em língua inglesa. Segmentação de documentos jurídicos em língua portuguesa deve considerar as peculiaridades do domínio. As diferentes abordagens para o problema diferem no escopo da solução, formatos de arquivo suportados e métodos e algoritmos usados.

CAPÍTULO

5

Experimentos

De uma maneira geral, o que se espera de um modelo de aprendizado de máquina são duas características: convergência e generalização. Convergência é a capacidade de um modelo aprender os padrões do conjunto de treinamento. Enquanto a generalização é a capacidade do modelo funcionar corretamente para padrões que não estavam presentes no treinamento (Feitosa et al., 2000). Para avaliar estas capacidades, os experimentos deste trabalho foram divididos em 3 categorias:

- **Intra-tribunal** (Seção 5.3). Nestes experimentos, são analisados os resultados de modelos treinados em dados de um tribunal e testados em dados deste mesmo tribunal. Objetivo principal é identificar tribunais que respondem bem às soluções de AM e incorporação de *layout* e imagem. Também busca-se responder quais tribunais oferecem maior dificuldade de aprendizado e identificar possíveis problemas que levem a um desempenho ruim. Além disso, estes resultados também fornecem uma referência para comparação dos modelos apresentados nos experimentos das seções seguintes.

- **Inter-tribunal um-contra-um** (Seção 5.4.1). Nesta categoria, os modelos são treinados em dados de um tribunal e testados em dados de um outro tribunal. Nestes experimentos, busca-se identificar quais tribunais se correlacionam bem e, adicionalmente, se modelos treinados em dados de um tribunal oferecem a possibilidade de bons resultados na segmentação de documentos de um outro tribunal.

- **Inter-tribunal todos-contra-um** (Seção 5.4.2). Neste caso, os modelos foram treinados em dados provenientes de quatro tribunais (dentre os cinco tribunais disponíveis) e testados em dados do quinto tribunal. O objetivo é analisar se a variedade de layouts nos dados de treinamento proporciona ganhos em comparação aos modelos da categoria um-contra-um.

Os experimentos para avaliar o desempenho de diferentes modelos de aprendizado de máquina foram realizados com diferentes algoritmos treinados no dataset TREINO MINI. Foram considerados, como modelos de referência (*baselines*), quatro algoritmos tradicionais de AM: *Logistic Regression* (LR) (Dreiseitl and Ohno-Machado, 2002), *Random Forest* (RF) (Schonlau and Zou, 2020), *Decision Tree* (DT) (Rokach and Maimon, 2005) e *K-nearest Neighbors* (KNN) (Bijalwan et al., 2014). Dentre estes, foram selecionados aqueles com melhor desempenho para a comparação com os modelos baseados em aprendizado profundo. Os resultados dos modelos de referência são apresentados na Seção 5.2. Com relação aos modelos baseados em aprendizado profundo, foram considerados: *BERTimbau* (Souza et al., 2020) e *LayoutLMv3* (Huang et al., 2022). O modelo *BERTimbau* foi pré-treinado para o português do Brasil e representa um modelo de aprendizado profundo que é baseado exclusivamente em texto. O modelo *LayoutLMv3* (ou simplesmente *LayoutLM*) incorpora também características de layout e imagem à análise do texto. Cada experimento terá seus desempenhos medidos utilizando as métricas de avaliação descritas na seção a seguir.

Todos os scripts desenvolvidos para treinamento e avaliação dos modelos estão disponíveis para download.[1]

5.1 Métricas de Avaliação

Para medir a qualidade dos modelos de segmentação, foram utilizadas métricas clássicas para problemas de classificação, dado que a segmentação foi modelada como um problema de classificação de linhas do arquivo PDF. Em tarefas de classificação, o principal objetivo é predizer a qual classe um determinado exemplo pertence. No caso deste trabalho, isto equivale a predizer a qual segmento uma linha de um documento PDF faz parte. Obviamente, considerando um exemplo específico (uma linha do PDF), um determinado modelo pode acertar ou errar a sua classificação. No entanto, os erros podem ser divididos em dois tipos: falsos positivos (FP) e falsos negativos (FN). Estes dois tipos de erro são representados na Tabela 5.1, juntamente com os verdadeiros positivos e negativos (acertos). A título de exemplo, considere que um exemplo é da classe EMENTA (classe correta ou real). Um erro do tipo FN com relação a este exemplo significa que o modelo não classificou este exemplo da maneira correta, ou seja, o modelo classificou o exemplo como sendo de classe diferente de EMENTA. Já um erro do tipo FP, significa que o modelo classificou um exemplo de outra classe (diferente de EMENTA) como sendo da classe ementa. A quantidade de erros do tipo

Real \ Predito	Positivo	Negativo
Positivo	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Tabela 5.1: Esquema dos dois tipos de erros: Falso Positivo e Falso Negativo

[1] http://direitodigital.ufms.br:8000

FP e FN são, obviamente, muito relevantes para avaliar a qualidade de um modelo. Entretanto, estes são números absolutos, o que dificulta sua interpretação. Por isso, é comum utilizar duas medidas relativas: *precision* (P) e *recall* (R) (Manning et al., 2008). *Precision* é a proporção de exemplos classificados como positivos e que são realmente positivos. Este valor é definido como:

$$P = \frac{VP}{VP+FP}.$$

Já a medida *recall* é a proporção de *todos* os exemplos positivos que foram classificados como positivos e é definida como:

$$R = \frac{VP}{VP+FN}$$

Uma importante métrica que combina *precision* e *recall* em único valor é o F1-Score ($F1$), que corresponde à média harmônica entre *precision* e *recall*, e é definida como:

$$F1 = \frac{2PR}{P+R}.$$

As medidas P, R e $F1$ são definidas para problemas de classificação binária, ou seja, quando consideramos apenas uma classe. Sempre é possível calcular estas medidas para cada classe. Entretanto, é importante considerar também algum tipo de média entre as classes para permitir comparações entre diferentes modelos. Uma maneira de se obter a média destas medidas entre múltiplas classes é denominada *micro averaging*. Este método consiste em somar os contadores de verdadeiros positivos (VP), falsos positivos (FP) e falsos negativos (FN) para cada classe e utilizar os valores agregados para calcular o valor médio desejado. Mais especificamente, as médias micro para P, R e $F1$ são definidas como:

$$P_m = \frac{\sum_c VP_c}{\sum_c VP_c + FP_c}, \tag{5.1}$$

$$R_m = \sum_c \frac{\sum_c VP_c}{\sum_c VP_c + FN_c}, \tag{5.2}$$

$$F1_m = \frac{2P_m R_m}{P_m + R_m}, \tag{5.3}$$

onde VP_c, FP_c e FN_c são os contadores correspondentes a uma classe específica c. No decorrer deste documento, por simplicidade, o termo $F1$ será usado tanto para indicar $F1$ por classe quanto a média micro $F1_m$ definida acima. O mesmo vale para os valores P e R. Em todas as situações, o texto deixa esta diferença evidente.

5.2 Modelos de Referência

Para avaliar a capacidade de aprendizado de um modelo em um conjunto de dados, modelos de aprendizado supervisionado foram treinados utilizando o conjunto de dados de treinamento reduzido (TREINO MINI), permitindo a criação de modelos

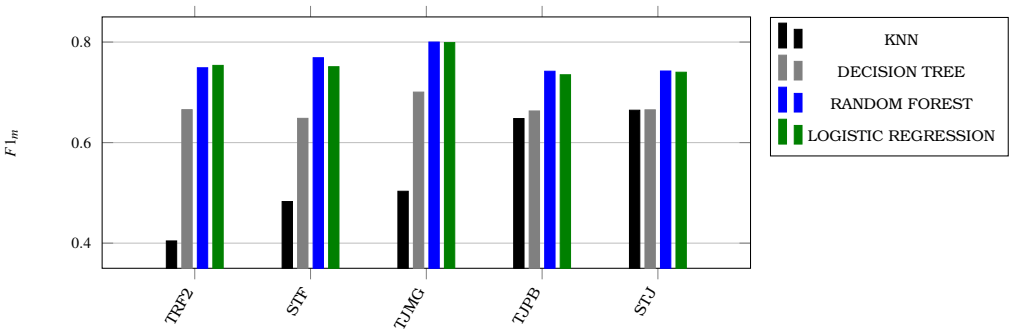


Figura 5.1: Média F1 dos algoritmos validados em dados anotados manualmente (dataset VAL-M) por tribunal.

de referência (*baselines*) para comparação. Inicialmente o treinamento foi realizado em cada tribunal e posteriormente avaliado em um conjunto de dados de validação (VAL-M), com o objetivo de mensurar a capacidade de convergência do modelo. Essa abordagem é uma prática comum em aprendizado de máquina, permitindo a identificação de problemas de sobre-ajuste e subajuste do modelo. Os algoritmos *k nearest neighbors* (KNN), *decision tree* (DT), *random forest* (RF) e *logistic regression* (LR) foram utilizados nesta avaliação, cujos hiper-parâmetros foram otimizados por meio de *Random Search*. Os resultados de desempenho $F1_m$ de cada um desses algoritmos são apresentados na Figura 5.1.

Os algoritmos *RF* e *LR* obtiveram os melhores resultados em todos os tribunais. *RF* é um modelo de conjunto que combina várias árvores de decisão para gerar previsões mais precisas. Este modelo é capaz de lidar com conjuntos de dados complexos e com muitas variáveis, além disso, *RF* é menos suscetível a sobre-ajuste do que *DT* (Breiman, 2001), o que o torna uma opção mais eficiente para classificação de textos em relação a *DT*. Não foi definido limite de profundidade para DT e o número mínimo de amostras necessárias para dividir um nó interno foi definido em 2. Para RF, neste experimento, foram 100 árvores do mesmo tipo.

LR, por sua vez, é uma técnica de classificação linear e como, em muitos casos, as características de um conjunto de dados de texto são linearmente separáveis (Bishop, 2006), significa que é possível traçar uma linha ou plano em um espaço de características para separar diferentes classes de texto.

Por outro lado, o *KNN* é um algoritmo de aprendizado de máquina simples que se baseia na similaridade entre amostras para classificá-las, mas pode ser bastante lento e suscetível a sobre-ajuste em conjuntos de dados de texto maiores (Zhang and Zha, 2004). Neste experimento, a escolha do parâmetro número de vizinhos igual a 1, obtida por meio de uma avaliação com busca aleatória (Bergstra and Bengio, 2012) dentro do intervalo de 1 a 5, pode indicar a possibilidade de sobre-ajuste. *DT* pode sofrer do mesmo problema, uma vez que, considerando que não foi definida uma profundidade máxima e o número mínimo de amostras necessárias para dividir um nó interno foi definido em 2, a árvore de decisão continuará crescendo até que tenha se ajustado perfeitamente os dados de treinamento ou até que não haja mais amostras para dividir. Isto pode levar ao sobre-ajuste, onde o modelo tem um bom desempenho

nos dados de treinamento, mas mal nos dados novos e não vistos. Além de ser menos flexível em relação a conjuntos de dados de texto complexos e com muitas variáveis (Breiman, 2001). Portanto *RF* e *LR* foram os modelos escolhidos para comparativos com os modelos de aprendizado profundo.

5.3 Experimentos Intra-Tribunal

Os experimentos denominados intra-tribunal consistem no treinamento e avaliação de modelos cujos dados pertencem a um mesmo domínio, ou seja, os mesmos tribunais utilizados no treinamento do modelo são utilizados no teste. Em geral, os modelos foram treinados e validados em um único tribunal apenas, mas um experimento de treinamento e teste em todos os tribunais também foi executado. Portanto, nesta modalidade, seis modelos são treinados por algoritmo: um pra cada tribunal e um em todos os tribunais. Uma visão geral dos diferentes modelos treinados é apresentada na Tabela 5.2, onde se pode comparar os resultados dos algoritmos *Logistic Regression*, *BERTimbau* e *LayoutLM*. Nesta visão é possível identificar em quais segmentos cada algoritmo demonstra melhor desempenho e também onde as *features* de layout apontam melhor ganho.

	PARTES			**EMENTA**			**RELATORIO**			**VOTO**			**ACORDAO**			**Média Micro**		
	LR	BERT	LayoutLM	LR	BERT	LayoutLM	LR	BERT	LayoutLM	LR	BERT	LayoutLM	LR	BERT	LayoutLM	LR	BERT	LayoutLM
STF	0.15	**0.92**	0.81	0.89	0.38	**0.91**	0.35	0.42	**0.99**	0.80	0.84	**0.93**	0.66	0.90	**0.98**	0.75	0.82	**0.95**
STJ	0.10	0.60	**0.89**	0.81	0.14	**0.99**	0.43	0.53	**0.90**	0.84	0.85	**0.96**	0.51	**0.82**	0.58	0.74	0.78	**0.95**
TRF2	0.14	**0.96**	**0.96**	0.76	0.52	**0.97**	0.36	0.39	**0.79**	0.84	0.87	**0.96**	**0.90**	0.74	0.76	0.75	0.78	**0.94**
TJPB	0.11	**0.94**	0.85	0.77	0.36	**0.93**	0.44	0.60	**0.69**	0.82	0.83	**0.88**	**0.75**	0.68	0.32	0.73	0.77	**0.83**
TJMG	0.30	**0.89**	0.83	**0.86**	0.38	0.74	0.35	0.57	**0.68**	0.87	0.89	**0.91**	0.77	0.96	**0.98**	0.80	0.85	**0.89**
TODOS	0.10	0.85	**0.89**	0.81	0.23	**0.90**	0.34	0.44	**0.88**	0.83	0.85	**0.93**	0.65	**0.82**	0.70	0.75	0.79	**0.92**

Tabela 5.2: Resultados dos experimentos intra-tribunal. $F1$ dos modelos para cada classe e média micro. Os modelos foram treinados utilizando o TREINO MINI e validado utilizando o TESTE.

Como esperado para esta categoria de experimento, o LayoutLM consegue desempenho médio superior em todos os tribunais. Além disso, no geral, de 30 comparações (tribunal x classe), o LayoutLM se sobressai em 22 delas. Analisando os resultados por segmento, o modelo LayoutLM tem desempenho superior em todos os tribunais para os segmentos RELATORIO e VOTO. No segmento EMENTA, observa-se que o LR obtém desempenho alto, superior ao BERT em todas os tribunais e maior que o LayoutLM no TJMG. Uma possível explicação para esse fenômeno é que as ementas apresentam frases muito curtas. Na realidade, geralmente, as ementas são compostas por uma lista de tópicos. Especialmente no STF e no TJMG, os acórdãos são frequentemente muito concisos. Textos muito curtos são um desafio para modelos profundos que se beneficiam de contexto.

Os modelos LR podem apresentar melhores resultados na análise de textos curtos por alguns motivos. Frases curtas tendem a ter menos ruído ou informações irrelevantes, o que permite que o modelo se concentre melhor nas características importantes do texto e, consequentemente, obtenha resultados mais precisos. Frases curtas também são mais representativas e específicas. As palavras usadas no segmento EMENTA costumam ser muito específicas e representativas, tornando mais

fácil para o modelo aprender a associar certas palavras ou frases a determinados tópicos ou conceitos. O BERT, por outro lado, é projetado para capturar o contexto em textos longos. Em frases curtas, o contexto é mais limitado do que em textos longos, o que dificulta o uso eficiente do BERT, onde a análise pode depender mais de características superficiais, como as palavras individuais. Como resultado, o desempenho do BERT costuma não ser tão bom em segmentos curtos. Ao mesmo tempo, o layout da ementa dos acórdãos traz características marcantes que podem ser exploradas pelo LayoutLM. Na maioria dos tribunais, a ementa é apresentada com um deslocamento à direita, destacando e separando claramente esse segmento dos demais. Essa formatação privilegia o desempenho do modelo LayoutLM. A exceção é o TJMG, que apresenta o segmento EMENTA com o mesmo alinhamento do restante do documento, o que afeta claramente o desempenho deste modelo.

O segmento PARTES, embora também apresente textos curtos, é composto basicamente por nomes próprios de pessoas ou entidades. Os modelos de LR podem ser limitados na modelagem de dependências contextuais em um texto, ou seja, eles não conseguem capturar a relação entre palavras e frases em um texto, o que é essencial para a classificação de textos cujo destaque é um nome próprio. Eles também podem ser sensíveis a características irrelevantes, o que pode afetar negativamente seu desempenho nesse cenário. No entanto, o BERT pode ter um melhor desempenho nesse caso, já que o pré-treinamento do modelo possivelmente o tenha tornado capaz de reconhecer esses nomes, independentemente do contexto.

O segmento ACORDAO combina algumas das características citadas anteriormente e é difícil afirmar os motivos dos resultados neste segmento, mas certas características podem trazer algumas elucidações. Embora também seja um segmento curto, o texto é consistente e uniforme em sua redação, notadamente sob aplicação de um modelo e contendo, na maioria dos acórdãos, as mesmas palavras, alterando apenas os sujeitos e os objetos das orações. Isto faz com que o BERT consiga contextualizar melhor e obter desempenho muito superior que no segmento EMENTA. Por outro lado, a característica marcante de layout que se observa na EMENTA (deslocamento) não está presente no segmento ACORDAO. Além disso, segmento curto pode estar disposto em posições diferentes da página em diferentes documentos. Isto leva o LayoutLM a uma degradação no desempenho quando comparado à EMENTA. Outro ponto que vale destacar é a representatividade deste segmento e a sensibilidade ao tamanho da amostra de treinamento. Pelo número reduzido de exemplos deste segmento disponíveis para treinar o modelo, há uma maior chance de ocorrer *overfitting*, ou seja, o modelo pode se tornar muito especializado para os exemplos específicos de treinamento e, assim, não generalizar bem para novos exemplos. Como resultado, o desempenho do LayoutLM pode ser prejudicado.

Na Tabela 5.3, o modelo LayoutLM é comparado com os scripts de segmentação automática desenvolvidos especificamente para cada tribunal. Na última linha da tabela, assim como nos experimentos acima, são apresentados os resultados de modelos treinados e testados em dados de todos os tribunais. Estes experimentos demonstram uma tendência de aproximação do desempenho dos modelos ao *script* de

anotação automática. Um modelo genérico apresenta resultados equivalentes ou até superiores a modelos específicos dos tribunais STF e TJMG quando avaliados através da métrica F1. Isto indica que, mesmo para os tribunais que apresentam maior complexidade na anotação dos dados, os modelos apresentam uma boa convergência, e que a etapa de anotação pode ser o fator limitante para a obtenção de resultados ainda mais precisos. Essa informação sugere que o modelo genérico pode ser utilizado

Tribunal	**PRECISION**		**RECALL**		**Média F1 Micro**	
	LayoutLM	*Script*	LayoutLM	*Script*	LayoutLM	*Script*
STF	0.91	0.93	0.95	0.92	0.95	0.92
STJ	0.92	0.98	0.94	0.98	0.95	0.98
TRF2	0.92	0.97	0.95	0.96	0.94	0.96
TJPB	0.80	0.91	0.85	0.90	0.83	0.90
TJMG	0.86	0.90	0.88	0.79	0.89	0.85
TODOS	0.90	0.93	0.92	0.91	0.92	0.92

Tabela 5.3: Desempenho geral do algoritmo LayoutLM em experimento intra-tribunal em comparação aos *scripts* de anotação automática.

como uma alternativa viável para segmentar acórdãos de diferentes tribunais conhecidos, podendo até ser evitada a necessidade de treinar modelos específicos para cada tribunal. Fica evidente também que a qualidade da anotação dos dados é fundamental para a obtenção de resultados precisos em qualquer modelo de aprendizado de máquina, uma vez que os modelos são tão bons quanto os dados que utilizam para aprendizagem.

5.4 Experimentos Inter-Tribunal

Nestes experimentos, denominados aqui inter-tribunal, os dados de treinamento são de tribunais diferentes daqueles usados nos dados de teste. Eles são divididos em experimentos do tipo um-contra-um e todos-contra-um. Na Seção 5.4.2, serão detalhados os experimentos um-contra-um, onde os modelos são treinados em um tribunal e testados em outro tribunal diferente. Na Seção 5.4.1, serão detalhados os experimentos todos-contra-um, onde os modelos são treinados em quatro tribunais e testados no quinto tribunal que não está presente nos dados de treinamento.

5.4.1 Experimentos um-contra-um

Os experimentos na modalidade um-contra-um são aqueles onde os modelos são treinados em um tribunal e testados em um tribunal diferente. Estes experimentos buscam avaliar a capacidade de generalização de um modelo treinado em um tribunal específico e responder se existe uma correlação forte o suficiente entre documentos de tribunais diferentes capaz de adaptar um modelo de um tribunal em outro tribunal. O desempenho destes modelos são apresentados na Tabela 5.4. Pode-se observar que o modelo treinado no STF apresenta um desempenho significativamente inferior

Treino \ Teste	STF	STJ	TRF2	TJPB	TJMG
STF	0.95	0.37	0.21	0.56	0.17
STJ	0.39	0.95	0.88	0.60	0.54
TRF2	0.58	0.81	0.94	0.61	0.66
TJPB	0.49	0.63	0.62	0.83	0.69
TJMG	0.54	0.50	0.58	0.63	0.89

Tabela 5.4: Resultados dos experimentos inter-tribunal um-contra-um. Média micro de F1 entre todas as classes para os modelos LayoutLM em treino/-teste em pares de tribunais diferentes. Observe que os resultados na diagonal (sublinhados) servem apenas como referência, já que não correspondem a modelos inter-tribunal, mas sim intra-tribunal (treinado e testado no mesmo tribunal).

em comparação aos outros tribunais, evidenciando a peculiaridade deste tribunal em relação aos demais. Tanto os modelos treinados no STF quanto aqueles treinados em outros tribunais apresentam desempenho médio mais baixo quando validados no STF, indicando a dificuldade em generalizar para este tribunal específico.

Também é possível observar na Tabela 5.4 uma leve correlação entre os tribunais TJMG e TJPB e uma correlação mais forte entre os tribunais TRF2 e STJ. Um modelo treinado no TJMG tem seu melhor desempenho quando validado no TJPB e vice-versa. Da mesma forma entre STJ e TRF2. O modelo treinado no STJ apresentou um desempenho superior quando avaliado no TRF2, superando, inclusive, o desempenho do TJPB em experimento intra-domínio, o que reforça a forte correlação entre esses tribunais. Portanto, apesar da presença de tribunais com padrões distintos, há ainda uma correlação significativa entre alguns tribunais, como o TRF2 e o STJ. Esses resultados sugerem que, embora haja peculiaridades nos diferentes tribunais, ainda é possível encontrar relações fortes entre eles, indicando que um modelo treinado em uma variedade maior de tribunais pode tornar casos como o do STF cada vez mais raros. Essa hipótese será testada nos experimentos inter-tribunal apresentados na seção a seguir, onde serão realizados experimentos todos-contra-um. Na Tabela 5.5 são condensados os resultados mínimos, médios e máximos para o experimento um-contra-um para cada tribunal.

Tribunal	Mínimo	Média	Máximo
STF	0.17	0.33	0.56
STJ	0.39	0.60	0.88
TRF2	0.58	0.67	0.81
TJPB	0.49	0.61	0.69
TJMG	0.50	0.56	0.63

Tabela 5.5: Resultados do experimento inter-tribunal um contra um: mínimo, média e máximo $F_1 Micro$ para cada tribunal

	PARTES				EMENTA				RELATORIO				VOTO				ACORDAO				$F1_m$			
	LR	RF	BERT	LayoutLM	LR	RF	BERT	LayoutLM	LR	RF	BERT	LayoutLM	LR	RF	BERT	LayoutLM	LR	RF	BERT	LayoutLM	LR	RF	BERT	LayoutLM
STF	0.04	0.05	**0.48**	0.37	**0.44**	0.26	0.07	0.30	0.18	0.12	0.25	**0.42**	0.70	0.67	**0.74**	0.72	0.03	0.03	0.21	**0.48**	0.56	0.51	0.61	**0.70**
STJ	0.09	0.11	**0.52**	0.47	0.45	0.29	0.12	**0.95**	0.33	0.22	0.37	**0.84**	0.77	0.75	0.79	**0.96**	0.49	0.45	0.62	**0.66**	0.64	0.61	0.64	**0.88**
TRF2	0.07	0.06	**0.75**	0.63	0.47	0.48	0.16	**0.85**	0.31	0.21	0.44	**0.86**	0.81	0.80	0.84	**0.96**	0.64	0.59	0.40	**0.66**	0.68	0.67	0.74	**0.91**
TJPB	0.03	0.05	**0.66**	0.08	**0.47**	0.29	0.07	0.46	0.27	0.19	**0.37**	0.32	0.79	0.79	**0.81**	0.76	0.37	**0.50**	0.05	0.22	0.66	0.64	**0.68**	0.67
TJMG	0.10	0.02	**0.49**	0.12	0.70	0.45	0.09	**0.72**	0.46	0.30	**0.56**	0.31	0.87	0.84	**0.89**	0.79	0.34	0.53	0.50	**0.60**	0.77	0.72	**0.81**	0.73

Tabela 5.6: Resultados dos experimentos inter-tribunal todos-contra-um. $F1$ dos modelos para cada classe e média $F1$. Os modelos foram treinados utilizando o TREINO MINI e validados no TESTE.

5.4.2 Experimentos todos-contra-um

Os experimentos na modalidade todos-contra-um são aqueles onde os modelos são treinados em dados de quatro tribunais (dentre os cinco considerados) e testados no tribunal que não está presente nos dados de treinamento. São realizados experimentos considerando cada um dos tribunais como teste. O objetivo é analisar se um modelo tem capacidade de generalização para tribunais desconhecidos durante o treinamento e responder se um mesmo modelo pode ser usado para tribunais diferentes ou novos layouts e estruturas de documentos aplicados pelos tribunais, mesmo para aqueles tribunais para os quais não haja dados anotados.

Na Tabela 5.6 são exibidos os resultados gerais por algoritmo em experimento inter-tribunal em diferentes métricas. Nesses dados é possível notar uma capacidade de adaptação dos modelos para tribunais não utilizados durante o treinamento. Os testes no TRF2 e STJ, por exemplo, atingiram $F1_m$ superior ao TJPB em teste intra-tribunal no LayoutLM, comprovando a existência de fortes correlações entre tribunais diferentes já percebida em experimentos um-contra-um e que uma maior variação de layouts de documentos apresentados durante o treinamento, de fato, melhora o desempenho da segmentação em tribunais desconhecidos.

Essa tendência de melhora no desempenho, conforme aumento da variedade de tribunais durante o treinamento, é melhor observada na Tabela 5.7. Por meio destes resultados, é evidente que o aumento da variedade de tribunais usados no treinamento impacta positivamente o desempenho em outros tribunais. O melhor exemplo é o TRF2, que mesmo quando o modelo não conhece nenhum exemplo deste tribunal durante o treinamento, consegue generalizar bem e se aproximar do desempenho do modelo intra-tribunal.

Tribunal	Intra Tribunal	Inter Tribunal	
		um-contra-um	todos-contra-um
STF	0.95	0.50	0.70
STJ	0.95	0.58	0.88
TRF2	0.94	0.57	0.91
TJPB	0.83	0.60	0.67
TJMG	0.89	0.52	0.73

Tabela 5.7: Comparação dos resultados ($F1_m$) entre as diferentes modalidades de experimentos. Importante salientar que, nos experimentos intra-tribunal e todos-contra-um, existe um único conjunto de teste por tribunal. Já no experimento um-contra-um, existem quatro conjuntos de teste para cada tribunal. Portanto, a coluna um-contra-um corresponde à média nos quatro conjuntos de teste de cada tribunal.

CAPÍTULO

6

Conclusão

Neste trabalho realizamos um estudo sobre a estrutura do sistema judiciário brasileiro, que reuniu informações sobre como os 90 tribunais produzem e publicam seus documentos na internet. Também fizemos uma análise da estrutura dos inteiros teores de acórdãos produzidos por 5 desses tribunais e identificamos padrões na estrutura dos documentos, a partir dos quais definimos de um método de segmentação automática aplicável a qualquer tribunal. Realizamos o download de 960.503 acórdãos dos cinco tribunais e extraímos informações de texto e layout de 624.161 acórdãos utilizando ferramentas de manipulação de PDFs. Executamos a anotação automática dos acórdãos cujas informações puderam ser extraídas e realizamos uma revisão manual em 500 desse acórdãos, 100 de cada tribunal. Os documentos anotados resultaram em cinco conjuntos menores com propósitos específicos de treino, validação e teste gerados a partir da coleta e anotação.

Os experimentos realizados possibilitaram uma análise sobre a possibilidade de se desenvolver ferramentas para segmentação automática de documentos baseadas em AM. Os experimentos intra-tribunal, onde modelos treinados e testados em dados do mesmo tribunal, os experimentos inter-tribunal um-contra-um, onde modelos foram treinados em dados de um tribunal e validados em dados dos outros tribunais e os experimentos inter-tribunal todos-contra-um, onde modelos foram treinados em dados de quatro tribunais e validados em dados do quinto tribunal permitiram a avaliação de desempenho e dificuldades de aprendizado, bem como a comparação de um modelo geral com um modelo específico. Também possibilitaram a identificação de correlações entre tribunais e avaliação da eficácia de modelos treinados em diferentes tribunais. A comparação de resultados dos experimentos todos-contra-um em relação aos experimentos um-contra-um permitiram avaliar se a variedade de layouts apresentada durante o treinamento melhora o desempenho dos modelos.

Modelos de AM e aprendizado profundo foram treinados e avaliados sobre estes

datasets, onde obtivemos pontuações $F1_m$ superiores a 90% em modelos específicos intra-tribunal em vários tribunais e em modelos gerais todos-contra-um obtivemos pontuação superior a 90% em um tribunal, como também obtivemos pontuação inferior a 70% em outro. Esses modelos se generalizam melhor com o aumento da variedade de layouts conhecidos no treinamento. Portanto, treinar com uma variedade maior de tribunais pode melhorar o desempenho para aplicação prática. Os modelos produzidos nestes experimentos foram treinados em uma parcela reduzida do *dataset* e resultados melhores podem ser obtidos ao treinar em um *dataset* maior.

Constatamos também que, apesar de modelos do tipo BERT alcançarem resultados excelentes em várias tarefas complexas de PLN, estes modelos são baseados exclusivamente em informações de texto. Quando se observa, em domínio jurídico, especificamente o inteiro teor dos acórdãos, as variações de layout e o valor que essas características podem prover, é válido introduzir tais informações codificadas no modelo de AM. Foi possível observar que o BERT (bertimbau) obtém alguns bons resultados para a tarefa de segmentação. Isso sugere que a modalidade unicamente de linguagem, sem incluir informações de layout, oferece uma função vital na compreensão do documento, mas ainda assim os resultados nem sempre são satisfatórios. Pode-se notar que alguns segmentos não podem ser bem classificados apenas pela compreensão do texto e a incorporação das *features* de layout e imagem melhoraram significativamente o resultado da classificação nestes segmentos.

Ao comparar os resultados dos modelos de AM com o desempenho dos *scripts* de anotação automática, foi constatado que a variabilidade de layouts dentro do mesmo tribunal e a falta de clareza na quebra de segmentos prejudicam o processo de anotação automática. Isso resultou na criação de *datasets* ruidosos, o que, por sua vez, afetou negativamente o desempenho dos modelos treinados nesses *datasets*, indicando que no processo de anotação está o gargalo do desempenho no cenário atual.

Modelos de segmentação específicos conseguem obter desempenhos superiores até aos *scripts* de anotação automática em alguns casos. Isto comprova que modelos de aprendizado profundo, com incorporação de informações de *layout* e imagem, garantem um altíssimo desempenho mesmo quando comparados a modelos de linguagem de ponta. Modelos de segmentação gerais, quando aplicados sobre documentos de tribunais desconhecidos, apresentam tendência de generalização conforme aumento dos layouts conhecidos no treinamento, o que sugere que ao expandir a variedade de tribunais em treinamento pode-se obter desempenho satisfatório para aplicação prática.

Os *scripts* de anotação automática apresentam um desempenho satisfatório de forma geral, porém, é importante destacar que há limitações relevantes em seu uso. A criação e manutenção desses *scripts* podem ser desafiadoras, e pequenas variações de layout podem inviabilizar seu uso, enquanto grandes variações podem tornar seu desenvolvimento impraticável. Embora sejam importantes para o treinamento de modelos, é preciso considerar as limitações dos *scripts* de anotação automática na sua aplicação sobre dados mutáveis ou projetos mais longos.

Como propostas de trabalhos futuros, podemos aumentar a quantidade de tribu-

nais treinados em modelos gerais para melhorar o desempenho neste cenário. Assim como melhorar o método de anotação automática, uma vez que ficou comprovado que este foi um limitador de desempenho neste trabalho. Métodos de *active learning* podem ser utilizados para selecionar acórdãos específicos para treinamento e validação do modelo, em vez de utilizar uma seleção aleatória. Isso pode aumentar a eficiência do treinamento e melhorar a qualidade do modelo resultante. Além disso, a técnica de *active learning* também pode ser usada para ajudar na anotação manual dos acórdãos. Em vez de anotar aleatoriamente alguns dos acórdãos, o modelo pode ser alimentado com exemplos específicos para anotação, selecionados com base na incerteza do modelo em relação a esses exemplos. Isso pode reduzir significativamente o esforço necessário para anotar manualmente um grande número dos acórdãos. Futuramente podemos extrair conhecimento dos segmentos dos acórdãos, como padrões de votos ou decisões baseados em uma ementa, por exemplo, na tentativa de auxiliar tribunais em julgamentos e consolidação de jurisprudência. Além disso, os desdobramentos mais impactantes deste trabalho, provavelmente, serão os trabalhos que somente serão possíveis pela disponibilização para a comunidade dos recursos desenvolvidos neste trabalho.

Referências Bibliográficas

Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., Dunkelberger, J., Elgohary, A., Feldman, S., Ha, V., et al. (2018). Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*. Citado na página 63.

Banker, K., Garrett, D., Bakkum, P., e Verch, S. (2016). *MongoDB in action: covers MongoDB version 3.0*. Simon and Schuster. Citado na página 18.

Bao, H., Dong, L., e Wei, F. (2021). Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*. Citado na página 65.

Bengio, Y., Ducharme, R., Vincent, P., e Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155. Citado na página 24.

Bengio, Y., Simard, P., e Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166. Citado na página 20.

Bergstra, J. e Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305. Citado na página 70.

Berkenbilt, J. (2021). Qpdf. Citado na página 6.

Bijalwan, V., Kumar, V., Kumari, P., e Pascual, J. (2014). Knn based machine learning approach for text and document mining. *International Journal of Database Theory and Application*, 7(1):61–70. Citado na página 68.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer. Citado na página 70.

Brasil (2015). lei nº 13.105, de 16 de março de 2015. código de processo civil. *Diário Oficial [da] República Federativa do Brasil*. Citado na página 3.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32. Citado nas páginas 70 e 71.

Cavnar, W. B., Trenkle, J. M., et al. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer. Citado na página 24.

CNJ, C. N. d. J. (2022). Justiça em números. https://www.cnj.jus.br/wp-content/uploads/2022/09/sumario-executivo-jn-v3-2022-2022-09-15.pdf. Citado na página 3.

Devlin, J., Chang, M.-W., Lee, K., e Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. Citado na página 29.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., e Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929. Citado na página 32.

Dreiseitl, S. e Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359. Citado na página 68.

Feitosa, R., Vellasco, M., Oliveira, D., Andrade, D., e Maffra, S. (2000). Facial expression classification using rbf and back-propagation neural networks. In *Proc. 6th Int. Conf. on on Information Systems Analysis and Synthesis*, páginas 73–77. Citado na página 67.

Garncarek, Ł., Powalski, R., Stanisławek, T., Topolski, B., Halama, P., Turski, M., e Graliński, F. (2021). Lambert: layout-aware language modeling for information extraction. In *International Conference on Document Analysis and Recognition*, páginas 532–547. Springer. Citado na página 31.

Glyph & Cog, L. (2021). pdftohtml: Portable document format (pdf) to html converter. Citado na página 6.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. Citado nas páginas vii, 25, e 28.

Gurgel, J. (1945). Análise estatística da distribuição de poisson. *Anais da Escola Superior de Agricultura Luiz de Queiroz*, 2:299–320. Citado na página 32.

Hagan, M., Demuth, H., Beale, M., e De Jesús, O. (2014). *Neural Network Design*. Martin Hagan. Citado na página 20.

Hao, L., Gao, L., Yi, X., e Tang, Z. (2016). A table detection method for pdf documents based on convolutional neural networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, páginas 287–292. IEEE. Citado nas páginas 7 e 65.

Hochreiter, S. e Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. Citado na página 20.

Huang, Y., Lv, T., Cui, L., Lu, Y., e Wei, F. (2022). Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387*. Citado nas páginas vii, 7, 31, 32, 33, 65, e 68.

ISO 32000-1:2008 (2008). Document management — Portable Document Format — Part 1: PDF 1.7. Standard, International Organization for Standardization. Citado na página 5.

Jain, A. K., Mao, J., e Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3):31–44. Citado na página 21.

Krenker, A., Bešter, J., e Kos, A. (2011). Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, páginas 1–18. Citado na página 18.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551. Citado na página 20.

Li, J., Xu, Y., Lv, T., Cui, L., Zhang, C., e Wei, F. (2022). Dit: Self-supervised pre-training for document image transformer. *arXiv preprint arXiv:2203.02378*. Citado na página 65.

Manning, C., Raghavan, P., e Schutze, H. (2008). Introduction to information retrieval (vol. 1). cambridge: Cambridge university press. Citado na página 69.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., e Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, páginas 3111–3119. Citado na página 24.

Pennington, J., Socher, R., e Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, páginas 1532–1543. Citado nas páginas 24 e 64.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., e Moy, L. (2010). Learning from crowds. *Journal of machine learning research*, 11(4). Citado na página 58.

Richardson, L. (2007). Beautiful soup documentation. *April*. Citado na página 14.

Rokach, L. e Maimon, O. (2005). Decision trees. In *Data mining and knowledge discovery handbook*, páginas 165–192. Springer. Citado na página 68.

Schonlau, M. e Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, 20(1):3–29. Citado na página 68.

Schuster, M. e Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681. Citado na página 24.

Shen, Z., Lo, K., Wang, L. L., Kuehl, B., Weld, D. S., e Downey, D. (2022). Vila: Improving structured content extraction from scientific pdfs using visual layout groups. *Transactions of the Association for Computational Linguistics*, 10:376–392. Citado na página 7.

Shinyama, Y. (2019). Pdfminer: Python pdf parser and analyzer. 11. Citado na página 6.

Singer-Vine, J. e Jain, S. (2023). PDFPlumber. https://github.com/jsvine/pdfplumber (version: 0.8.0). Citado na página 6.

Sousa, A. W. e Del Fabro, M. D. (2019). Iudicium textum dataset uma base de textos jurídicos para nlp. In *XXXIV Simpósio Brasileiro de Banco de Dados: Dataset Showcase Workshop, SBBD*. Citado nas páginas 7 e 36.

Souza, F., Nogueira, R., e Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*. Citado na página 68.

Staar, P. W., Dolfi, M., Auer, C., e Bekas, C. (2018). Corpus conversion service: A machine learning platform to ingest documents at scale. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, páginas 774–782. Citado nas páginas 6, 7, e 64.

Suzuki, K. (2011). *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand. Citado na página 20.

Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P. J., e Bolikowski, Ł. (2015). Cermine: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335. Citado na página 7.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., e Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, páginas 5998–6008. Citado nas páginas vii, 7, 25, 27, 28, e 30.

Wang, R., Liu, W., e McDonald, C. (2014). Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, volume 39, páginas 1–8. Citado nas páginas 24 e 26.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., e Zhou, M. (2020). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, páginas 1192–1200. Citado na página 65.

Xu, Y., Lv, T., Cui, L., Wang, G., Lu, Y., Florencio, D., Zhang, C., e Wei, F. (2021). Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding. *arXiv preprint arXiv:2104.08836*. Citado na página 31.

Yuxin, W., Kirillov, A., Massa, F., Wan-Yen, L., e Girshick, R. (2019). Detectron2. Citado na página 65.

Zanusso, M. B. et al. (1997). Familias de t-normas diferenciaveis, funções de pertinencia relacionadas e aplicações. Citado na página 20.

Zhang, Y. e Zha, H. (2004). K-nearest neighbor and locally linear embedding for nonlinear dimensionality reduction. In *Advances in neural information processing systems*. Citado na página 70.