UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Classificação de textos de petições jurídicas com base em técnicas de Processamento de Línguas Naturais

Kelsen Henrique Rolim dos Santos

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Kelsen Henrique Rolim dos Santos

Classificação de textos de petições jurídicas com base em técnicas de Processamento de Línguas Naturais

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Thiago Alexandre Salgueiro Pardo

Versão original

São Carlos

2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi e Seção Técnica de Informática, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

Santos, Kelsen Henrique Rolim dos
S237c Classificação de textos de petições jurídicas com base em técnicas de Processamento de Línguas Naturais / Kelsen Henrique Rolim dos Santos ; orientador Thiago Alexandre Salgueiro Pardo. – São Carlos, 2023.
114 p. : il. (algumas color.)

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Processamento de Línguas Naturais. 2. Classificação de textos jurídicos. 3. Bag of words. 4. BERT. I. Pardo, Thiago Alexandre Salgueiro, orient. II. Título.

Kelsen Henrique Rolim dos Santos

Classificação de textos de petições jurídicas com base em técnicas de Processamento de Línguas Naturais

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Thiago Alexandre Salgueiro Pardo

Original version

São Carlos

2023

AGRADECIMENTOS

Devo agradecer primeiramente à minha companheira, Aline, por ser uma grande incentivadora do meu interesse de qualificação numa área tão distinta da minha formação original (em Direito) e da minha profissão de Defensor Público, que abracei para a vida; este trabalho foi produzido em uma época de intensas mudanças pessoais e profissionais, sem esse incentivo não teria conseguido vencer os momentos em que cheguei mais próximo de não prosseguir o curso. Também sou muito grato ao meu orientador, Thiago, que me ajudou a encontrar o caminho para produzir os estudos e os experimentos nessa área que eu nunca havia desbravado. Por fim, presto um agradecimento especial à Defensoria Pública do Estado de Rondônia - onde exerci o cargo de Defensor Público por quase dez anos -, que efetivamente despertou meu interesse de trabalhar com tecnologia e me proporcionou as oportunidades necessárias para aplicar e desenvolver meu conhecimento neste ramo.

RESUMO

Santos, K. H. R **Classificação de textos de petições jurídicas com base em técnicas de Processamento de Línguas Naturais**. 2023. 114p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

O interesse na aplicação de inteligência artificial no domínio jurídico tem sido crescente, ante o potencial para agilizar o trabalho de profissionais e melhorar serviços públicos, sendo o Processamento de Línguas Naturais (PLN) uma das áreas de maior utilidade. Este trabalho explora a aplicação de PLN para classificação de petições jurídicas destinadas a processos judiciais, utilizando um conjunto de dados produzidos pela Defensoria Pública do Estado de Rondônia. São experimentadas técnicas de *Bag of Words* e *Bidirectional Encoder Representations for Transformers* (BERT) para realizar classificações por Assunto e por Tipo das petições, tendo esta se destacado segundo as métricas calculadas. Os resultados demonstram a viabilidade de aplicação de modelos de linguagem para a auxiliar a rotina profissionais com automatização da classificação de documentos.

Palavras-chave: Inteligência artificial. Processamento de Línguas Naturais. Classificação de textos. Texto jurídico. Petições jurídicas processuais. *Bidirectional Encoder Representations for Transformers. Bag of words.*

ABSTRACT

Santos, K. H. R **Legal petition text classification using Natural Language Process**. 2023. 114p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

There is growing interest in using artificial intelligence (AI) in the field of law to speed up the work of legal professionals and improve public services. One useful area of AI is Natural Language Processing (NLP). This study examines the use of NLP to classify legal petitions for legal proceedings, using data from the Public Defender's Office in the State of Rondônia. The study uses Bag of Words and Bidirectional Encoder Representations for Transformers (BERT) techniques to classify petitions by Subject and Type. The results show that language models can be used to automate document classification and assist legal professionals in their daily work.

Keywords: Artificial intelligence. Natural Language Processing. Text classification. Legal text. Legal petitions. Bidirectional Encoder Representations for Transformers. Bag of words.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
BERT	*Bidirectional Encoder Representations for Transformers*
CNJ	Conselho Nacional da Justiça
GED	Gestor Eletrônico de Documentos
LLM	*Large Language Model*
NLTK	*Natural Language Toolkit*
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
PJe	Sistema de Processo Eletrônico (PJe)
PLN	Processamento de Línguas Naturais
SEEU	Sistema Eletrônico de Execução Unificado
SOLAR	Solução Avançada em Atendimento de Referência

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e motivação

O uso de técnicas de inteligência artificial para automação ou auxílio de tarefas rotineiramente executadas por profissionais jurídicos, em especial no âmbito do Poder Judiciário, tem sido crescente. Pesquisa[1] realizada em 2022 pelo Conselho Nacional de Justiça identificou 111 projetos desenvolvidos ou em desenvolvimento nos tribunais brasileiros, um crescimento de 171% em relação ao ano anterior (ORGANIZAÇÃO DAS NAÇÕES UNIDAS, 2022). Os projetos de maior aplicabilidade e relevância em geral envolvem o Processamento de Línguas Naturais (PLN), considerando que a maior parte do tempo da rotina de trabalho de profissionais jurídicos é empregado na análise e na produção de documentos textuais.

Rotineiramente, os profissionais jurídicos precisam ler, analisar, interpretar e tomar decisões sobre uma série de documentos que constituem um problema jurídico. Quando este problema torna-se um caso judicial, constitui-se um "processo", que nada mais é do que uma sequência de diferentes tipos de documentos que ora discutem o problema (apresentando argumentos sobre a aplicação da lei ao caso concreto ou decidindo sobre essas questões), ora representam atos processuais (como diligências ou audiências) e/ou fatos (podemos chamar estes de provas).

O profissional jurídico precisa reconhecer em quais documentos ele deve depositar mais atenção (e quais são apenas "barulho" naquele contexto), buscando não somente identificar que assuntos ou discussões estão sendo tratados naquele processo, mas também qual deverá ser o próximo documento da sequência. Pode ser um recurso, ou pode ser uma petição de determinado tipo, entre muitos tipos de petições que existem. O esperado é que, após essa analise, o profissional elabore um novo documento que prossiga a sequência do processo rumo a um ensejo final - quase sempre uma decisão.

Esses documentos são facilmente reconhecíveis individualmente por um profissional jurídico. No entanto, os processos costumam conter dezenas ou centenas deles. Classificar e ler tudo exige uma boa quantidade de tempo, mas depois disso, ainda é preciso elaborar o novo documento, uma tarefa que pode demandar dias de trabalho. Felizmente, a maior parte dos casos tramitando na justiça brasileira são questões rotineiras, nas quais os profissionais possivelmente já atuaram anteriormente (ou ao menos os tribunais realizaram decisões a respeito).

Por esse motivo, é muito comum que o profissional busque decisões ou documentos

[1] Painel disponível em <https://paineisanalytics.cnj.jus.br/single/?appid=9e4f18ac-e253-4893-8ca1-b81d8af59ff6&sheet=b8267e5a-1f1f-41a7-90ff-d7a2f4ed34ea>

que já trataram sobre o mesmo problema anteriormente, ou até mesmo o próprio documento que deve ser elaborado agora (apresentado em um caso anterior, obviamente). Essas decisões são chamadas de "precedentes"; e quando um documento pode ser reproduzido para vários casos (com poucas alterações contextuais), ele se torna um "modelo". Os profissionais jurídicos estão sempre produzindo ou buscando modelos para agilizar o andamento de casos rotineiros. Tarefas de classificação de textos podem auxiliar a identificar documentos legais semelhantes em casos anteriores, como também precedentes que foram aplicados ao mesmo problema. Um banco de "modelos" organizado e catalogado economizaria horas ou dias de trabalho.

Na maior parte das vezes, os profissionais jurídicos também não estão atuando individualmente em um caso. Uma pessoa específica pode ser apenas uma peça de uma organização inteira dedicada a executar determinada função no sistema jurídico. Os Tribunais, por exemplo, compostos por juízes, existem para julgar; a Defensoria Pública representa pessoas em situação de vulnerabilidade nos processos por intermédio dos Defensores Públicos.

Essas Instituições precisam se organizar para atender a demanda com a quantidade limitada de trabalhadores que possuem, e podem se beneficiar catalogando seus acervos de documentos jurídicos. A identificação analítica de documentos ou procedimentos que compõem a sua demanda auxiliam no planejamento de uma organização, com uma distribuição mais eficiente de unidades e de pessoal. Muitas vezes elas precisam atuar estrategicamente, identificando casos com determinadas características, tornando os processos mais uma vez objeto de classificação.

A classificação de textos pode identificar se determinada decisão está em consonância com as decisões proferidas anteriormente sobre o mesmo assunto. A partir disso, podem, por exemplo, serem executadas ou compartilhadas com mais eficiência estratégias processuais de litígio. Algumas classificações úteis podem envolver: as teses e os temas discutidos (classificação por tópicos); as classes ou tipos dos processos; as espécies de peças processuais.

Algumas tarefas de PLN podem ser muito úteis para otimizar o tempo e/ou melhorar a assertividade. Todavia, o desenvolvimento de modelos para PLN encontra dificuldade na escassez de dados rotulados. O domínio jurídico, especialmente, apresenta desafios adicionais, com particularidades de vocabulário, citações de precedentes ou textos normativos e documentos muito longos. Soma-se a isso o fato de a jurisdição territorial brasileira ser muito grande, adicionando complexidade ao vocabulário dado a regionalidades incorporadas à comunicação (POLO *et al.*, 2021).

Existem estudos documentados que aplicam diversas tarefas de PLN no domínio jurídico. (SERRAS; FINGER, 2022) aplica verbetização, uma tarefa que extrai grupos de palavras representativos de um documento. Já (AGUIAR *et al.*, 2022) trabalha com sumarização, ou seja, resumir o conteúdo de um texto em algumas palavras.

Neste estudo, vamos conhecer algumas dessas técnicas e seus resultados.

1.2 Hipóteses e Objetivos

Neste trabalho, exploraremos algumas técnicas clássicas e outras mais atuais aplicadas em tarefas de classificação de textos utilizando Processamento de Línguas Naturais, em especial no domínio jurídico brasileiro.

Existem modelos de linguagem - instrumentos utilizados para a máquina conseguir interpretar textos em língua natural - que foram treinados em quantidades muito grandes de dados, adquirindo alta eficiência para o desempenho em tarefas específicas. Algumas arquiteturas possuem capacidade de generalização, isto é, seus modelos, apesar de treinados para determinada tarefa, podem ser ajustados para o desempenho de novas tarefas, inclusive em outros domínios, por meio de um processo de transferência de aprendizado (SOUZA; NOGUEIRA; LOTUFO, 2020) - como fez (POLO *et al.*, 2021), criando o BERTikal para processamento no domínio legal a partir do refinamento de BERTimbau, modelo treinado pela Neuralmind (SOUZA; NOGUEIRA; LOTUFO, 2020) primariamente com dados em Português Brasileiro.

Essa capacidade é muito relevante em domínios que possuem poucos dados, catalogados ou não catalogados. É o caso do domínio jurídico em Português Brasileiro. Apesar da enorme quantidade de processos que existem no Judiciário brasileiro, a maior parte dos documentos são produzidos em formatos pouco úteis, como imagens ou PDF. A parcela destes documentos correta e devidamente catalogados é ainda menor. Adicionalmente, os textos são muito longos, dificultando o processamento ao exigir maior capacidade computacional.

A utilização de técnicas de PLN para classificação de documentos no domínio jurídico Brasileiro em aplicações que auxiliem as rotinas de profissionais da área é a justificativa deste trabalho. Desse modo, estabelecemos os seguintes objetivos:

1. Identificar as características do problema e as técnicas de PLN mais adequadas para realizar a classificação de documentos jurídicos.

2. Avaliar o desempenho das técnicas propostas.

Nossa hipótese principal é que, com aplicação de técnicas de PLN, é possível realizar a classificação de documentos jurídicos em tipos de peças processuais e por agrupamentos temáticos em áreas do Direito. Nesse sentido, levantamos as seguintes hipóteses:

1. Modelos tradicionais de classificação de textos com PLN são eficazes para a classificação de documentos no domínio jurídico em Português do Brasil.

2. Modelos distribucionais são capazes de classificar documentos no domínio jurídico em Português do Brasil com melhor desempenho do que modelos tradicionais.

3. Classes de documentos mais frequentes podem ser classificados com melhor desempenho.

4. A classificação por assunto do documento (que é mais semântica) tem desempenho superior do que a classificação por tipo do documento (que é mais estrutural).

1.3 Organização do texto

No capítulo 2 trataremos dos fundamentos teóricos dos temas. No capítulo 3 relacionaremos alguns trabalhos relacionados, que abordaram problemas no mesmo contexto de PLN no domínio jurídico do Português do Brasil. No capítulo 4 detalharemos a proposta prática do experimento: conheceremos o conjunto de dados que servirá para avaliar os modelos de classificação por aprendizado de máquina e as etapas envolvidas em cada um dos métodos aplicados. Os resultados serão apresentados e discutidos no capítulo 5, seguindo-se nossas conclusões no capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PLN, Aprendizado de Máquinas e modelos de linguagem

O Processamento de Línguas Naturais (PLN) é uma subárea de estudo da Inteligência Artificial que consiste, basicamente, nas técnicas utilizadas para a máquina ser capaz de lidar a língua. Seu objetivo área é fazer os computadores desempenharem atividades envolvendo a língua humana, tais quais possibilitar a comunicação entre a máquina e o homem, melhorar a comunicação entre humanos ou processar texto e discurso para executar tarefas úteis (JURAFSKY; MARTIN, 2023), as quais podem consistir em, por exemplo, traduzir, resumir, identificar estruturas (ou elementos), escrever e/ou classificar.

Neste trabalho, trataremos de técnicas e métodos de PLN para classificar textos. A tarefa de classificação de textos consiste no enquadramento de um documento em um conjunto de categorias pré-definidas, que podemos chamar de classes, rótulos ou *labels*. Em geral, essa classificação é um resultado estatístico da probabilidade de dado texto pertencer a determinado conjunto, isto é, de ser-lhe atribuída uma classe. A escolha das classes pode atender a diversos critérios, desde que úteis a alguma finalidade. No domínio legal, que estamos investigando neste estudo, alguns exemplos de classificação podem ser identificar o(s) tópico(s) tratados no texto ou o objetivo de um documento.

No contexto de PLN, as abordagens mais recentes tem se apoiado no Aprendizado de Máquinas (*Machine Learning*), que pode ser entendido, como explica (MITCHELL, 1997), por qualquer programação de computador que proporcione a melhoria no desempenho de alguma tarefa (segundo uma métrica definida) por meio da experiência - ou, em outras palavras, é dito que a máquina aprende por experiência quando, a partir da execução de determinadas tarefas por meio de um experimento X, a máquina atinge melhor desempenho na execução dessas tarefas segundo medido por uma métrica Y. O processamento da língua com Aprendizado de Máquinas resultou no surgimento de diversos modelos que representam a língua com base em matemática e em estatística, operacionalizados por uma série concatenada de algoritmos com vistas a executar uma tarefa de PLN específica - os modelos de linguagem.

2.2 Vetores de palavras

A princípio, a máquina não compreende um texto escrito (ou falado) e, possivelmente, as palavras para ela são apenas símbolos ou conjuntos de símbolos, sem qualquer significado. O uso das palavras compreender e entender aqui, inclusive, pode ser controverso, pois mesmo que uma aplicação seja capaz de processar a língua natural para realizar a tarefa para a qual foi arquitetada, não é possível afirmar efetivamente que

houve compreensão da linguagem. O processamento é apenas uma sequência de operações matemáticas organizadas sobre uma entrada textual (*input*) com objetivo de se obter uma saída (*output*) que resolva o problema proposto.

Como não é possível realizar operações matemáticas sobre palavras, o primeiro passo é obter uma representação numérica do texto. Uma abordagem clássica utilizada consiste em representar o texto na forma de um vetor, em que cada dimensão captura uma característica - apesar de, a princípio, não podermos identificar individualmente o que cada dimensão representa. A essa representação dá-se a designação *word embeddings* e o método para melhor obtê-los, dado o objetivo de uma tarefa, é um objeto de pesquisa da área de PLN.

Com o texto representado por vetores podemos processá-lo com operações matemáticas. Muitas pesquisas de PLN tem se dedicado às diferentes arquiteturas com as quais se pode realizar esse processamento e quais são mais vantajosas ou desvantajosas para a execução da tarefa específica que auxilia a resolução do problema posto.

2.3 *Bag of words*

Um dos métodos mais clássicos de realizar essa representação chama-se *Bag of Words* (saco de palavras, em tradução livre), que consiste em representar o documento a partir da coleção de palavras que o compõe, independentemente da sequência - como se existisse um saco em que todas as palavras do texto estivesse dentro, desordenadamente, o que resulta na expressão que dá o nome à técnica.

O texto é representado por um vetor no qual cada dimensão corresponde a uma palavra do vocabulário e possui um valor numérico. A definição deste valor é variada, podendo ser, por exemplo: binário (que indica a presença ou ausência da palavra no documento) ou quantitativo (a contagem da ocorrência da palavra no texto).

Vejamos um exemplo. Em um pequeno vocabulário, diremos que as seguintes palavras possuem estas representações: "adoro" como [1,0,0,0,0], "casa" como [0,1,0,0,0], "verde" como [0,0,1,0,0], "cachorro" como [0,0,0,1,0] e "grande" como [0,0,0,0,1]. Com esse vocabulário, a frase "adoro minha casa" seria representada pelo vetor [1,1,0,0,0]. É importante notar que nem todas as palavras precisam estar representadas no vocabulário, podendo uma palavra ausente ser simplesmente ignorada na formação do vetor. A Figura 1 ilustra um exemplo de vetor produzido com o método *Bag of Words*. Observe que cada palavra no texto (mas nem todas) corresponde a uma dimensão do vetor.

Percebe-se de antemão um problema de custo computacional para o processamento de muito texto utilizando *Bag of Words*, caso se procure representar todas as palavras que poderiam estar presentes, posto que os vetores possuem dimensionalidade proporcional ao tamanho do vocabulário - só o Vocabulário Ortográfico da Língua Portuguesa da

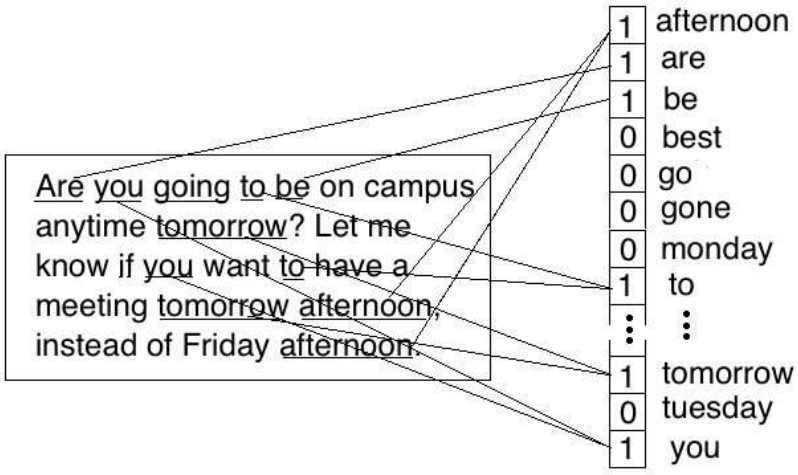

Figura 1 – Representação de *Bag of Words*. Fonte: (HOONLOR, 2011).

Academia Brasileira de Letras, em 2022, já contava com 370 mil palavras (ACADEMIA BRASILEIRA DE LETRAS, 2022).

Para reduzir o custo computacional pode-se limitar o vocabulário - por exemplo, utilizando apenas as palavras mais relevantes para o melhor resultado em determinada tarefa (as mais representativas em um dado domínio ou as mais frequentes). Outra forma de reduzir o vocabulário é a exclusão de *stopwords* do texto. Esses são vocábulos, geralmente muito repetitivos na estrutura morfossintática de uma língua natural, semanticamente irrelevantes, incapazes de representar ou distinguir contexto - como artigos ou preposições.

O método *Bag of Words* é incapaz de capturar o contexto da sequência ou variações semânticas, posto que não diferencia palavras com a mesma ortografia. Todavia, existem técnicas que procuram suavizar essas características, como o uso de *stemmers*, lematizadores ou normalizadores - que, com diferentes abordagens, realizam a representação de palavras com contexto idêntico ou parecido por um único radical, e este é incluído ao vocabulário. Por exemplo, as palavras "sabatina", "sabatinou" e "sabatinado" podem ser representadas simplesmente por "sabatin".

A grande vantagem de *Bag of Words* é que ele é um método simples de implementar e pode ser suficiente para executar determinadas tarefas, principalmente de classificação de textos. Por exemplo, é possível que a comparação entre textos de um domínio bastante específico (como a análise de laudos com um mesmo formato) possa ser realizada com resultados satisfatórios. Uma aplicação ainda muito utilizada do método é a identificação de *spam* em e-mails.

2.4 TF-IDF

A sigla TF-IDF corresponde à expressão *Term Frequency - Inverse Document Frequency*, que, em tradução livre, significa Frequência de termos - Frequência Documental Inversa. Trata-se de um aprimoramento ao método *Bag of Words* inicialmente proposta por (LUHN, 1957). que acrescenta uma abordagem estatística para atribuir importância maior ou menor a cada termo do vocabulário na representação do documento. Sua formulação parte da intuição de que termos muito frequentes no corpus (isto é, no conjunto dos documentos) são pouco significativos para diferenciar um documento de outros, enquanto termos mais raros são mais significativos.

Para atingir isso, matematicamente, o valor numérico referente à posição do termo no vetor representativo é calculado pela multiplicação dos elementos Frequência do Termo (TF) no documento e sua Frequência Documental Inversa (IDF). Enquanto o valor de TF mede a frequência da palavra no documento, IDF mede a frequência em *todos* os documentos, de modo que quanto maior a frequência do termo no *corpus*, menor é sua relevância na formação do vetor.

$$TFIDF = TF * IDF \tag{2.1}$$

Este método melhora a eficiência de *Bag of Words*, atingindo uma representação mais relevante para o documento sem adicionar muita complexidade. Entretanto, as desvantagens anteriormente tratadas permanece e, ao seu lado, é ainda mais elevado o custo computacional, dado que é necessário percorrer todo o *corpus* para calcular o IDF de cada vocábulo para, posteriormente, calcular os vetores. Além disso, com a evolução do *corpus* é necessário reprocessar o conjunto inteiro.

2.5 Word2Vec

A representação do texto por vetores é uma atividade desafiadora, pois a comunicação por língua natural não é feita pela mera sequência dos símbolos que formam frases. Existem particularidades semânticas e contextuais complexas de identificar e representar numericamente. A aplicação da técnica *Bag of Words* não distingue a semântica das palavras. Com isso, a técnica *Word2Vec* surgiu para abordar este problema. Como a expressão sugere, é um método que objetiva representar as palavras em vetores, capturando suas características semânticas e sintáticas a partir da influência de outras palavras que costumam acompanhá-las em textos.

A técnica foi proposta em (MIKOLOV *et al.*, 2013a) e (MIKOLOV *et al.*, 2013b), que demonstraram ser um método eficiente para gerar representações vetoriais de palavras com elevada qualidade capazes de capturar precisamente relações semânticas e sintáticas, atingindo estado da arte. *Word2Vec* gera um plano vetorial com centenas de dimensões,

no qual as palavras de relacionamento semântico ou sintático parecido são colocadas mais próximas umas das outras, como exemplificado na Figura 2.

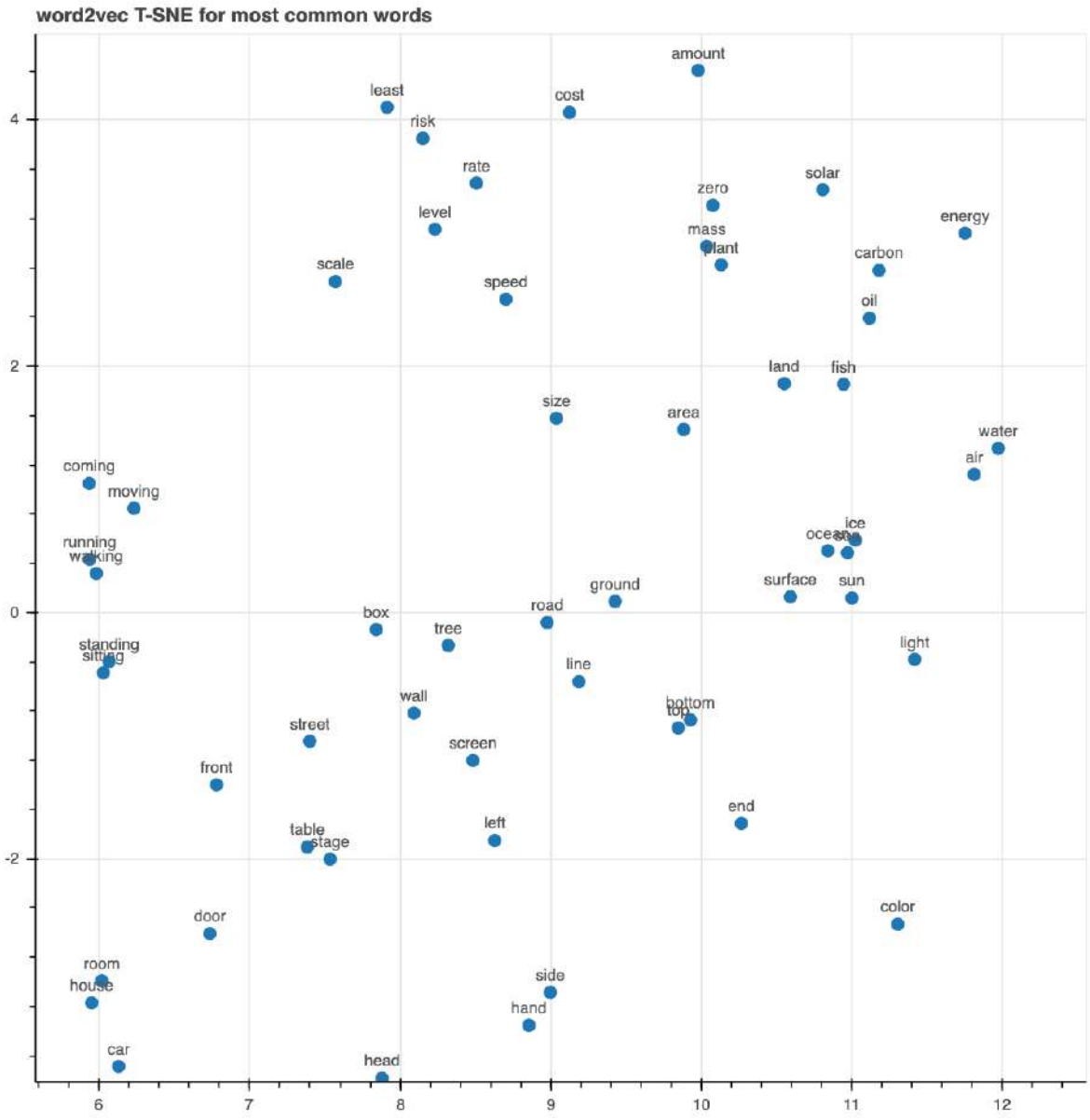

Figura 2 – Representação simplificada em duas dimensões de vetores de palavras produzidos com Word2Vec cf. (FACURE, 2017)

Podemos observar na figura, que as *coming* (vindo) e *running* (correndo) estão bastante próximas uma da outra e mais distantes de *ice* (gelo) e *ocean* (oceano), que também aparecem próximas uma da outra. A qualidade do resultado é tão elevada que é possível realizar operações matemáticas de adição e subtração com os vetores das palavras, como o clássico exemplo "rei" - "homem" + "mulher" = "rainha". Essas operações foram testadas com resultados positivos em (MIKOLOV *et al.*, 2013b).

A técnica atinge este resultado por meio modelos de rede neural simples. Podemos

resumir brevemente uma rede neural como uma sequência de operações matemáticas realizadas em paralelo sobre uma quantidade de dados numéricos de entrada. Cada camada passa o resultado à camada seguinte, enquanto a camada final entrega um resultado a partir de um cálculo probabilístico. A rede é treinada com o ajuste de pesos dessas operações com uma função obtida pela comparação do resultado previsto e o resultado correto.

Para servir o modelo na entrada, cada palavra é convertida individualmente em um token, inicialmente uma matriz numérica aleatória, cujas dimensões são atualizadas mediante o treinamento pelos modelos *Continuous Bag of Words* (CBOW) e *Skip-gram*. Em CBOW, à rede neural é dado um conjunto de N palavras posicionadas antes e depois da palavra procurada. Já em *Skip-gram*, a entrada do modelo é uma palavra, enquanto a saída são N palavras que a antecedem e sucedem. Esse treinamento ocorre em um gigantesco volume de dados não estruturados - em (MIKOLOV *et al.*, 2013b) foi utilizado um *corpus* de 33 bilhões de palavras. Ao final, cada palavra terá o seu vetor representativo, que pode ser resgatado para tarefas de PLN. Outros modelos surgiram posteriormente, que possuem resultados mais representativos em tarefas específicas, como o GloVe (PENNINGTON; SOCHER; MANNING, 2014) e FastText (BOJANOWSKI *et al.*, 2017) tendo abordagens semelhantes.

Apesar de possuir resultado relevante, a representação de palavras em vetores com esta estratégica possui suas limitações, decorrentes essencialmente da indiferença dos modelos de treinamento à ordem das palavras e à incapacidade de identificar expressões idiomáticas, como apontou (MIKOLOV *et al.*, 2013b).

As entradas e saídas dos modelos CBOW e Skip-gram não carregam informações sobre a posição das palavras; o processamento ocorre em conjuntos de palavras, mas não há um elemento que inclua a noção de sequência. Em razão de não lidar com a ordem das palavras ou do relacionamento de uma frase do texto com outra, *Word2Vec* não consegue incorporar dimensões de contexto às representações vetoriais.

Já ao processar palavras - e não sequências - *Word2Vec* não consegue identificar expressões constituídas por mais de uma palavra. Essa deficiência pode ser atenuada com a realização de um pre-processamento em que sequências de palavras que constituem expressões idiomáticas ou uma só entidade (como São Paulo) são identificadas e convertidas em um único *token* que será servido ao modelo.

2.6 Transfomers, BERT e GPT

Após *Word2Vec*, outras abordagens tentaram evoluir a capacidade de os modelos de PLN para capturar contexto - como Redes Neurais Recorrentes (RNN) e *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) -, introduzindo o processamento da sequência com base na recorrência. Essas técnicas obtiveram resul-

tados promissores, inclusive no domínio jurídico na Língua Portuguesa do Brasil, como tratamos no Capítulo 3. Entretanto, o seu custo computacional é muito grande, crescendo rapidamente com o aumento do tamanho do texto.

A arquitetura de *Transformers*, também baseada em redes neurais, surgiu como alternativa às estratégias tradicionais de recorrência, tendo sido demonstrada sua capacidade para reduzir a complexidade computacional e melhorar significativamente a possibilidade de paralelização (VASWANI *et al.*, 2017). Ela tem como princípio depender inteiramente de mecanismos de auto-atenção (*self-attention*) empilhados em multicamadas, como representado na Figura 3. Vemos representados, sobre o fundo cinza, as células principais da arquitetura: um *encoder* (à esquerda) e um *decoder* à direita. A rede completa possui várias células de *encoder* e/ou *decoder* empilhadas, onde a saída de uma fornece a entrada da próxima camada. Além disso, à entrada do modelo são adicionadas informações que remetem à posição de cada palavra na sequência.

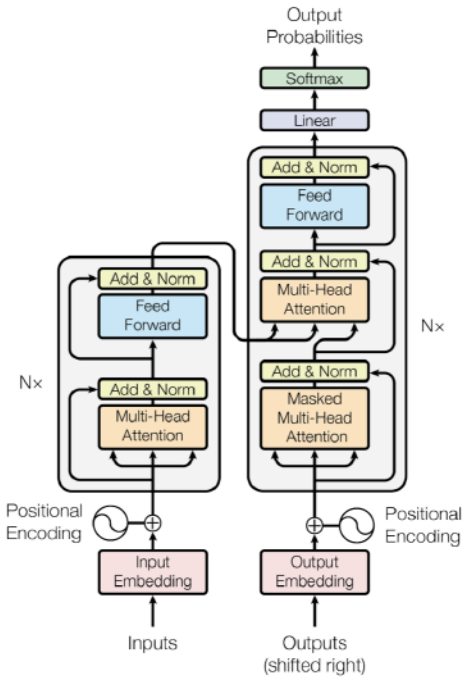

Figura 3 – Uma célula de uma rede neural com a arquitetura do modelo de Transformer. Fonte: (VASWANI *et al.*, 2017).

Esse tipo de arquitetura alcançou rapidamente estado da arte na execução de diversas tarefas de PLN, e é a base para modelos de linguagem atualmente relevantes como o *Generative Pre-trained Transformer* (GPT) - tecnologia por trás do famoso ChatGPT, da OpenAI - (RADFORD *et al.*, 2018) e o *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN *et al.*, 2019).

BERT é a sigla para *Bidirectional Encoder Representations from Transformers* - em tradução livre, Representações Codificadas Bidirecionais a partir de Transformers.

O modelo foi proposto em (DEVLIN *et al.*, 2019) com foco na camada de *Encoders* da arquitetura de *Transformers* (VASWANI *et al.*, 2017), isto é, pelo empilhamento das unidades vistas no lado esquerdo da Figura 3. É, essencialmente, uma Rede Neural com alta capacidade para produzir *embeddings* contextuais das palavras e um *embedding* adicional que representa a própria sequência. O modelo se beneficia da arquitetura de *Transformers* para processar a sequência inteira de uma vez em processamento paralelo.

Esse modelo não depende do pré-processamento para isolar as palavras mais relevantes: o texto é dado na entrada como está escrito, inclusive com suas iniciais maiúsculas e pontuações. A intuição é que todos esses elementos contribuem para a construção do contexto da sequência. O texto é quebrado em tokens com um vocabulário próprio do BERT, no qual muitas palavras são quebradas em sub-palavras (ou até em letras). O resultado é que o modelo lida melhor com palavras raras. A capacidade contextual torna o BERT muito potente para trabalhar com conjuntos de dados compostos por longos textos e palavras de uso pouco frequente na língua correspondente, como aqueles no domínio jurídico.

Ao seu lado, o BERT também favorece a transferência de aprendizado, uma estratégia que permite aproveitar etapas de um modelo de linguagem pré-treinado em uma quantidade massiva de dados, expandindo o seu treinamento ou afinamento-o para executar novas tarefas específicas em domínios com quantidade mais escassa de dados (SOUZA; NOGUEIRA; LOTUFO, 2020). Esta estratégia foi explorada por (SOUZA; NOGUEIRA; LOTUFO, 2020) para desenvolvimento do BERTimbau, o primeiro modelo de linguagem derivado do BERT treinado exclusivamente com dados na Lingua Portuguesa do Brasil - utilizando os pesos aprendidos por suas versões anteriores treinadas com uma quantidade muito superior de dados em inglês (BERT original) e em mais de cem idiomas (para o BERT multilíngue ou mBERT), nos quais os tokens relativos a estes vocabulários foram excluídos.

Os *embeddings* gerados por BERT possuem bons resultados para a realização de tarefas de classificação de textos. O fato de existir um modelo treinado com textos em Português - e aliado à sua capacidade de evoluir no treinamento com dados adicionais de um domínio especializado - tornou o modelo um objeto de muitas pesquisas para sua aplicação em tarefas de classificação de documentos jurídicos brasileiros.

2.7 Métricas

Como o emprego de ML para PLN é voltado para a execução de uma tarefa específica, é possível o desenvolvimento de medidas para calcular o seu desempenho. Essas medidas são chamadas de métricas, geralmente calculadas por expressões matemáticas de cunho estatístico ou probabilístico. Existe um grande contingente de métricas, com diversas finalidades, vantagens e desvantagens. Nessa seção apresentaremos as métricas

escolhidas para medir o desempenho do experimento, comumente utilizadas para avaliar modelos classificatórios.

Um problema de classificação se resume a atribuir um rótulo - de uma lista pré-definida (no nosso caso, tipos e de assuntos de petições processuais) - a um dado exemplo (no nosso caso, o conteúdo de uma petição processual). O modelo que enfrenta o problema só poderá apresentar um de dois resultados: uma classificação correta ou uma classificação incorreta. Portanto, a primeira métrica que se observa é a proporção de acertos sobre o conjunto total de exemplos classificados, isto é, a probabilidade geral do modelo acertar. Essa métrica é chamada de Acurácia, sendo simplesmente o total de acertos ($totalDeAcertos$) sobre o total de exemplos ($totalDeExemplos$), representados na Fórmula 2.2:

$$Acurácia = \frac{totalDeAcertos}{totalDeExemplos} \tag{2.2}$$

Todavia, quando observamos o resultado da classificação com referência a *cada classe* isoladamente, temos, na verdade quatro possíveis resultados, conforme apresentado por (MARIANO, 2021):

1. Verdadeiro positivo (VP): um exemplo predito para a classe X é realmente da classe X;

2. Verdadeiro negativo (VN): um exemplo predito como não sendo da classe X realmente não é da classe X (ou seja, é de outra classe);

3. Falso positivo (FP): um exemplo predito para a classe X, na verdade pertence a outra classe;

4. Falso negativo (FN): um exemplo predito para outra classe é, na verdade, da classe X.

Essa relação pode ser melhor compreendida na Figura 4, o que cria a chamada Matriz de Confusão. Observa-se que a soma de uma das diagonais (a de verdadeiros) resulta em todos os acertos, enquanto a outra oposta resulta nos erros (em vermelho na imagem. A matriz é melhor observada em um problema binário, mas segue os mesmos princípios para um problema com múltiplas classes, como apresentamos na Figura 5 (observada exclusivamente com referência à "Classe X").

A observação nesses moldes nos ajuda a construir outras métricas que avaliam o desempenho do modelo mais detalhadamente em cada classe, como as que utilizaremos nesse estudo:

		Classe predita	
		Classe X	Outras classes
Classe verdadeira	Classe X	Verdadeiro positivo	Falso negativo
	Outras classes	Falso positivo	Verdadeiro negativo

Figura 4 – Uma matriz de confusão explicada. Fonte: composição própria

		Classe predita			
		Classe X	Classe Y	Classe Z	Classe W
Classe verdadeira	Classe X	Verdadeiro positivo	Falso negativo	Falso negativo	Falso negativo
	Classe Y	Falso positivo	Verdadeiro negativo	Verdadeiro negativo	Verdadeiro negativo
	Classe Z	Falso positivo	Verdadeiro negativo	Verdadeiro negativo	Verdadeiro negativo
	Classe W	Falso positivo	Verdadeiro negativo	Verdadeiro negativo	Verdadeiro negativo

Figura 5 – Uma matriz de confusão com múltiplas classes observada com referência à "Classe X". Fonte: composição própria

1. Precisão: calculada pela quantidade de Verdadeiros Positivos VP sobre o total de exemplos preditos positivos (isto é, Verdadeiros Positivos VP mais Falsos Positivos FP). Representa probabilisticamente o quanto o modelo acerta ao prever que um exemplo pertence a determinada classe, ou seja, qual a chance do modelo estar correto ao prever um exemplo como integrante da classe de medição.

$$Precisão = \frac{VP}{VP + FP} \tag{2.3}$$

2. Cobertura: calculada pela quantidade de Verdadeiros Positivos VN sobre o total de exemplos reais de uma determinada classe, preditos corretamente ou não - isto é, Verdadeiros Positivos VP mais Falsos Negativos FN. Representa a proporção de exemplos de uma determinada classe que o modelo consegue identificar corretamente.

$$Cobertura = \frac{VP}{VP + FN} \tag{2.4}$$

3. Pontuação F1: essa métrica é calculada com os resultados de Precisão e de Cobertura, buscando com uma só pontuação representar o desempenho em ambas. Por esse

motivo, é comumente utilizada como medida geral de avaliação em problemas de classificação.

$$Pontuação F1 = \frac{2 * Previsão * Cobertura}{Precisão + Cobertura} \tag{2.5}$$

Em um problema onde existem múltiplas classes, como é o caso, visto que essas métricas (Precisão, Cobertura e Pontuação F1) são calculadas com referência a cada classe, existem duas formas de calcular cada uma das métricas para todo o conjunto de dados: obter a média aritmética ou a média ponderada (sobre a quantidade de exemplos na classe). Nesse estudo, utilizaremos primordialmente a média aritmética, visto que os resultados da métrica ponderada privilegiam as classes com maior ocorrência no conjunto de dados.

3 TRABALHOS RELACIONADOS

A aplicação de PLN no domínio jurídico é um tema recorrente de pesquisas acadêmicas e desenvolvimento de produtos para o público de profissionais. A partir do VICTOR (ARAUJO *et al.*, 2020), um *dataset* criado com documentos do Supremo Tribunal Federal digitalizados, (SILVA *et al.*, 2018) e (BRAZ *et al.*, 2018) demonstraram bons resultados para a classificação de documentos jurídicos utilizando RNN e LSTM.

Em um domínio mais genérico, (SOUZA; FILHO, 2022) explorou diversas técnicas para classificação de textos em Português do Brasil e concluiu que uma simples abordagem com TF-IDF em alguns casos pode ser mais vantajosa do ponto de vista prático, embora os melhores resultados obtidos tenham sido com modelos baseados em *Transformers* como o BERTimbau.

O modelo BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020) - foi o primeiro BERT treinado em Português do Brasil, para o que utilizou textos extraídos da *Wikipedia*, disponibilizados publicamente no brWaC Corpus (FILHO *et al.*, 2018). Outros modelos foram criados a partir do BERTimbau (ou seja, foram ajustados, ou *fine-tuned*) para lidar com dados no domínio jurídico, dos quais podemos citar:

1. VerBERT (SERRAS; FINGER, 2022)[1]: criado ajustando o BERTimbau para a tarefa de categorização de múltiplos rótulos, com vistas a um problema de verbetização (isto é, selecionar verbetes que representam um texto, como palavras-chave), sobre um conjunto de dados formado por textos de ementas[2] de decisões judiciais de diversas cortes brasileiras.

2. BERTikal (POLO *et al.*, 2021): criado ajustando o BERTimbau com uma das tarefas originais de treinamento de BERT - a Modelagem de Linguagem Mascarada (*Masked Language Modeling* - MLM) - em um *dataset* composto por publicações, documentos e movimentações processuais oriundos de diversas cortes brasileiras.

3. (ZANUZ; RIGO, 2022): criado a partir de BERTimbau com treinamento sobre o *dataset* LeNER-Br (LUZ DE ARAUJO *et al.*, 2018) para a tarefa de reconhecimento de entidades nomeadas (*Named Entity Recognition* - NER) em textos legais brasileiros.

Foi identificado ainda um modelo BERT treinado inicialmente com dados de textos jurídicos brasileiros, sem carregamento do modelo pré-treinado me Português do

[1] Não disponível na comunidade *Hugging Face*

[2] Um texto técnico que resume o conteúdo de ma decisão judicial, geralmente acompanhado de um cabeçalho formado por palavras chaves.

Brasil BERTimbau: o JurisBERT (VIEGAS, 2022), treinado em um *dataset* formado por documentos jurídicos diversos de obtenção própria pelo autor (a partir da mineração de textos em sites). (VIEGAS, 2022) alega que os resultados de seu experimento demonstram que pode ser melhor treinar um novo modelo com textos específicos do domínio do que ajustar modelos pré-treinados (seu trabalho comparou o modelo produzido com *BERT Multilingual* e *BERTimbau*)

4 METODOLOGIA E DESENVOLVIMENTO

4.1 Identificação do problema

O problema posto em questão consiste em uma tarefa de classificação de textos de natureza jurídica, especificamente petições processuais. O documento em si é composto basicamente por texto formatado em parágrafos, contendo uma assinatura do responsável pela petição ao final, podendo conter imagens ou tabelas - embora não frequentemente. Por meio desses documentos, as partes em um processo realizam ou se contrapõem a requerimentos, manifestam posições ou fornecem informações que, derradeiramente, impulsionarão o processo até uma decisão final. A atividade processual da Defensoria Pública consiste, em grande medida, na produção de petições enquanto representa uma das partes ou uma coletividade em uma ação judicial.

Quando uma petição processual inaugura uma ação, ela é uma petição inicial; contrariamente, será uma petição intermediária. Cada uma destas categorias possui diversas possibilidades de classificação, úteis em diferentes contextos. Neste estudo, abordaremos dois conjuntos diferentes de classificações, cujas classes apresentaremos adiante. A primeira classificação - que iremos designar de Tipo - é estrutural, relacionada com a função que determinado documento assume no processo. A segunda classificação - que chamaremos de Assunto - é mais semântica, relacionada com o tema (ou área do Direito) tratado no documento.

Cumpre compreender a utilidade das classificações postas para o contexto inserido - da Defensoria Pública. Classificações, apesar de não serem atividades estritamente jurídicas, estão sempre presentes nas atividades diárias. Recentemente, o Supremo Tribunal Federal tem divulgado programas de Inteligência Artificial voltados à classificação de processos e documentos, como vemos em (SUPREMO TRIBUNAL FEDERAL, 2023). As classificações ora estudadas são complementares entre si e atendem a propósitos funcionais e organizacionais. Com base na classificação dos documentos produzidos, o órgão público toma decisões de autogestão, em especial para a organização das atribuições dos seus órgãos internos e para a distribuição de força de trabalho com vistas a atender a demanda posta ao serviço público.

Adicionalmente, as teses e situações jurídicas processuais tendem a se repetir com bastante frequência em processos diferentes. Funcionalmente, a classificação documental auxilia a otimizar o tempo de trabalho das equipes, agrupando processos e documentos para identificar casos semelhantes que sirvam de ponto de partida para produção de novos documentos ou, ainda, para identificar precedentes, isto é, casos nos quais a mesma situação jurídica foi discutida e decidida anteriormente - buscando, assim, atingir o mesmo

tratamento.

Portanto, em resumo, o problema proposto consiste em, fornecida a íntegra do texto de uma petição jurídica produzida pela Defensoria Pública na atividade de representação judicial, proporcionar a sua classificação para o Tipo e Assunto do documento segundo um rol pré-definido de classes.

4.2 Apresentação do conjunto de dados

Os dados utilizados consistem em petições jurídicas produzidas pela Defensoria Pública do Estado de Rondônia quando do fornecimento do serviço de assistência jurídica gratuita à população hipossuficiente. Um exemplo desses documentos é apresentado na Figura 6 - no Anexo A é possível consultar outros desses documentos (com dados anonimizados) e suas classificações.

```
------------------------------------------------------------
Tipo: CIENCIA
Assunto: Família e Sucessões
------------------------------------------------------------
AO JUÍZO DA VARA CÍVEL DA COMARCA DE VILHENA-RO:
Autos nº.
A DEFENSORIA PÚBLICA DE RONDÔNIA, atuando em favor da autora, ja qualificada
nos autos em epigrafe, atraves da DEFENSORA PÚBLICA infra-assinada, vem a
presença de Vossa Excelencia, informar estar ciente da sentença.
Diante disso, pugna-se pela expediçao do formal de partilha.
Pede deferimento.
Vilhena, data certificada.

Defensora Publica

------------------------------------------------------------
```

Figura 6 – Um exemplo de petição jurídica com classificação de Tipo e Assunto.

Eles foram extraídos da base de dados da Solução Avançada em Atendimento de Referência (SOLAR), sistema que gerencia os atendimentos realizados pelo Órgão e realiza a entrega das petições produzidas para o sistema judicial por meio dos protocolos de *webservice* do Modelo Nacional de Interoperabilidade do Conselho Nacional da Justiça - no caso do Estado de Rondônia, as petições são destinadas aos Sistema de Processo Eletrônico (PJe) e ao Sistema Eletrônico de Execução Unificado (SEEU).

O usuário colaborador da Defensoria Pública de Rondônia, ao utilizar o SOLAR, tem a opção de produzir petições jurídicas por intermédio do módulo Gestor Eletrônico de Documentos (GED). Esses documentos, armazenadas pelo banco de dados em formato HTML, passaram pelo processo de limpeza descrito na seção 4.3 e compuseram o *dataset* deste estudo.

Ao destinar a petição para protocolo no sistema judicial correspondente, o colaborador realiza manualmente a sua classificação, definindo o Tipo do documento. Já o Assunto foi dado agregando a informação da área de competência do processo destinatário

da petição mediante cruzamento de dados do processo no SOLAR com dados do ProcAPI, API responsável por operacionalizar a integração do sistema com o PJe ou com o SEEU. Ao final, obtivemos um conjunto com 312.685 documentos. Para explorar o *dataset*, criamos uma estrutura de *dataframe* da biblioteca Pandas (THE PANDAS DEVELOPMENT TEAM, 2020), cujas características gerais são apresentadas na Figura 7.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 312685 entries, 1 to 286593
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   conteudo  312685 non-null  object
 1   assunto   312685 non-null  category
 2   tipo      312685 non-null  category
dtypes: category(2), object(1)
memory usage: 5.4+ MB
```

Figura 7 – Características gerais do *dataset* apresentado pelas propriedades de um *DataFrame* da biblioteca Pandas

A classificação será realizada sobre o Conteudo, isto é, a íntegra do texto do documento. As colunas Tipo e Assunto são as classes de cada documento: são 29 classes de Tipo e 9 classes de Assunto. A distribuição de documentos entre as classes pode ser vista nas Figuras 8 e 9. Existem duas classes genéricas de Tipo: "DIVERSOS" e "OUTROS". A classe "DIVERSOS" é composta por documentos que não foram classificados pelo usuário do SOLAR; já a classe "OUTROS" foi obtida com o agrupamento de classes de documentos efetivamente classificados pelos usuários, mas com quantidade pouco representativa - o que consideramos arbitrariamente menos de 250 ocorrências no *dataset*.

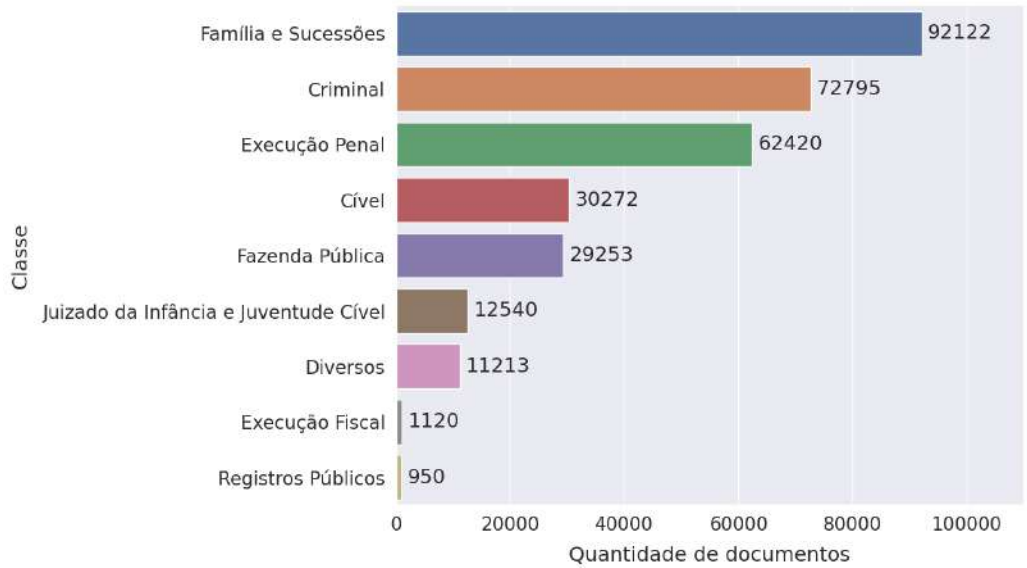

Figura 8 – Distribuição dos documentos entre as classes de <assunto>.

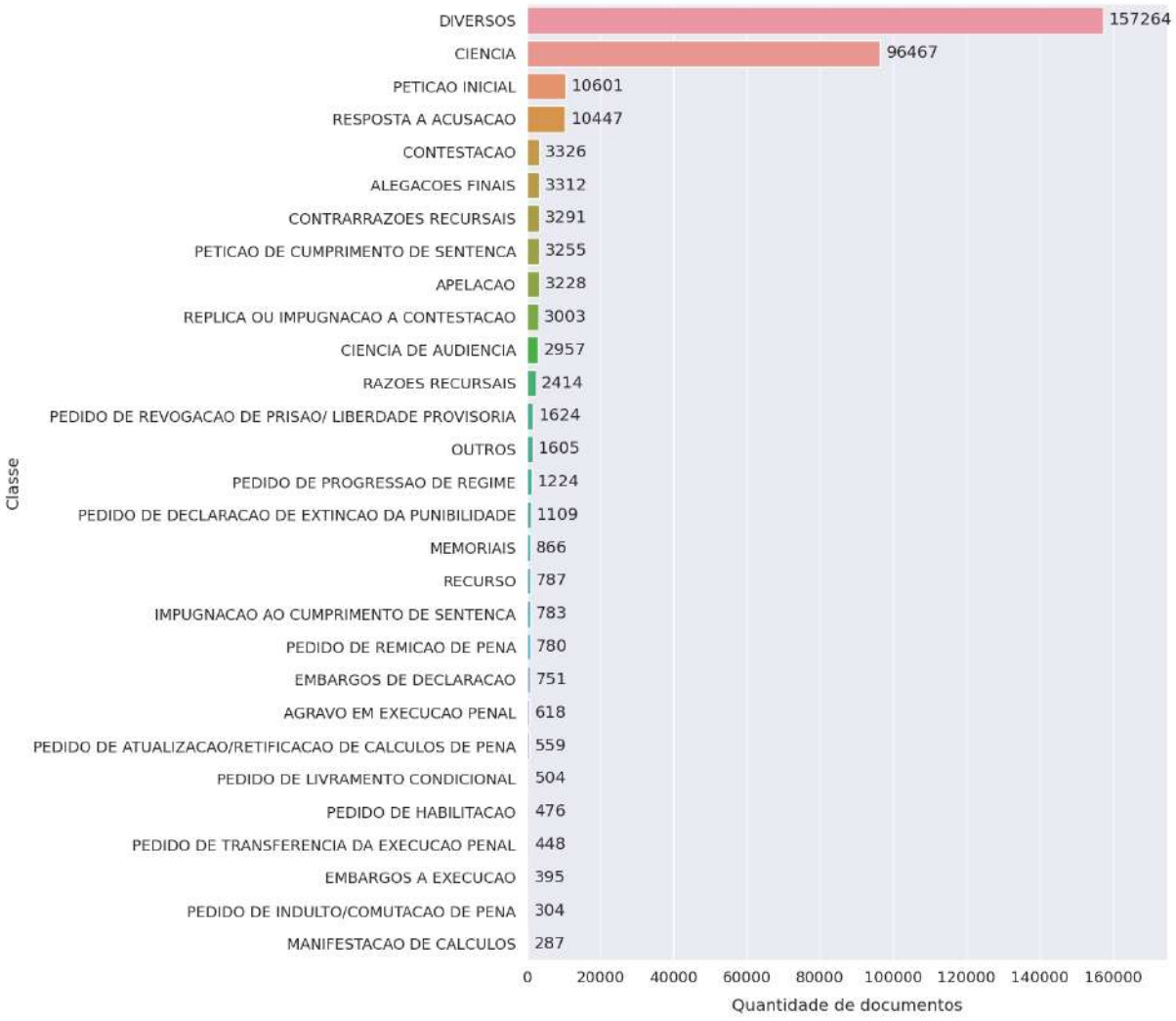

Figura 9 – Distribuição dos documentos entre as classes de <tipo>.

A Tabela 1 mostra que é comum a repetição de documentos com *exatamente* o mesmo conteúdo. Nessa tabela, a cada linha temos um conteúdo de documento que se repete a quantidade de vezes especificada na coluna "QTD" para a combinação de Tipo e Assunto. Observa-se que a repetição é mais frequente na classificação por Tipo: muitas vezes um documento com o mesmo Conteudo e Tipo se repete em diferentes classes de Assunto. Essas repetições também são mais recorrentes nas classes com maior quantidade de documentos, independentemente da classificação. Cerca de 32% dos documentos possuem conteúdo repetido, isto é, possuem conteúdo idêntico ao de outro documento do conjunto de dados. Essa conjuntura reflete a prática comum dos profissionais jurídicos utilizarem modelos de documentos para casos e situações repetitivas.

conteudo	tipo	assunto	QTD
MM. Juiz,\nCiente. \nPorto Velho, data inserid...	CIENCIA	Família e Sucessões	495
MM. Juiz,\nCiente. \nPorto Velho, data inserid...	DIVERSOS	Família e Sucessões	473
MM. Juiz,\nCiente.\nPorto Velho, data inserida...	DIVERSOS	Família e Sucessões	411
Douto Juízo:\nA Defensoria Pública do Estado d...	CIENCIA	Família e Sucessões	378
CIENTE A DPE.\n	CIENCIA	Execução Penal	327
CIENTE A DPE.\n	CIENCIA	Família e Sucessões	287
MM. Juiz,\nCiente a Defensoria Publica do Esta...	CIENCIA	Fazenda Pública	280
MM. Juíza,\nCiente da audiência designada.\nMa...	CIENCIA	Criminal	250
MM. Juiz,\nCiente.\nPorto Velho, data inserida...	CIENCIA	Família e Sucessões	244
MM. Juiz,\nCiente da decisao.	CIENCIA	Criminal	235
MM. Juíza,\nCiente da decisão.	CIENCIA	Criminal	227
Douto Juízo:\nA Defensoria Pública do Estado d...	CIENCIA	Cível	134
CIENTE A DPE.\n	CIENCIA	Juizado da Infância e Juventude Cível	122
MM. JUIZ (a)\nA Defensoria Publica do Estado d...	CIENCIA	Família e Sucessões	121
MM. Juiz,\nCiente a Defensoria Publica do Esta...	CIENCIA	Fazenda Pública	108

Tabela 1 – Distribuição, entre as classes, dos quinze documentos com o mesmo conteúdo mais frequentes. Verifica-se a grande repetição de documentos com exatamente o mesmo conteúdo e a mesma combinação das classes de Tipo e de Assunto.

4.3 Etapas de limpeza dos dados

Esta seção descreve as etapas para limpeza do conjunto de dados. O objetivo da limpeza foi viabilizar a interpretabilidade do conteúdo das decisões, descartar os dados com inconsistências que poderiam prejudicar a qualidade das classificações e agrupar classes numericamente pouco representativas de cada tipo. O processo inteiro está representado na Figura 10.

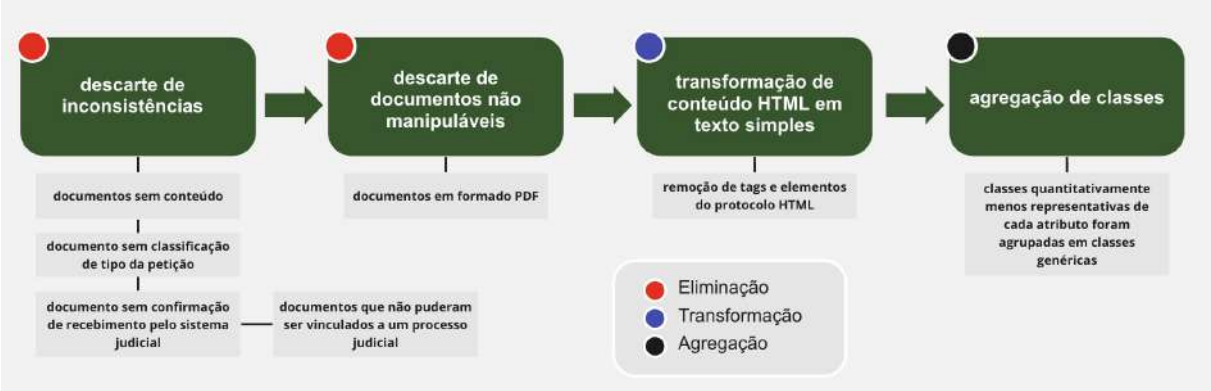

Figura 10 – Visão geral da etapa de limpeza dos dados

Inicialmente, foram descartados dados com inconsistências básicas, como documentos sem conteúdo, sem classificação de tipo da petição ou sem confirmação de recebimento pelo sistema judicial. Também foram descartados registros nos quais os colaboradores preferiram anexar um arquivo em formato PDF com a petição para protocolo ao invés de utilizar o próprio editor nativo do módulo GED do SOLAR, que armazena o conteúdo em formato HTML. Em seguida, o conteúdo HTML foi tratado para se tornar um conteúdo de texto simples com quebra de linhas, ou seja, foram removidas as tags e elementos do protocolo HTML. Após agregação com dados dos sistemas judiciais, foram descartados os registros que não puderam ser vinculados a um processo judicial, de modo que não puderam ter identificado o Assunto a que pertenciam. Finalmente, algumas classes quantitativamente menos representativas de cada classificação (assim consideradas aquelas com menos de 250 ocorrências) foram agrupadas em grupos genéricas - originando as classes "OUTROS" e "diversos" de Tipo e de Assunto, respectivamente.

No momento do processamento, para a classificação por Tipo, descartamos os documentos da classe "DIVERSOS", em razão de dela ser composta por documentos que não foram classificados manualmente pelos usuários do SOLAR; essa classe poderá ser utilizada posteriormente para testes qualitativos. Já na classificação por Assunto descartamos as classes "Execução Fiscal" e "Registros Públicos", em razão de apresentarem grande discrepância quantitativa das demais classes no experimento (menos de 10% da menor classe incluída).

Todas as etapas de extração e de limpeza foram realizadas utilizando o *Jupyter Notebook*. Para melhor reprodutibilidade, a cada etapa foi exportado um *dataset* em formato CSV utilizado na etapa seguinte para continuação.

4.4 Etapas de desenvolvimento

Vamos dividir os experimentos em duas macro etapas. Inicialmente aplicamos a técnica mais clássica pautada em *bag of words* balanceada com TF-IDF (seção 4.5). Em seguida, utilizaremos alguns modelos *open-source* pré-treinados com a tecnologia BERT e calibrados para o Português do Brasil (seção 4.6).

4.5 Aplicando *Bag of words*

4.5.1 Visão geral das etapas

Como abordamos na seção 2.3, o método *Bag of Words* consiste na representação do texto com base no seu vocabulário - isto é, a grosso modo, na contagem de ocorrência de cada palavra. Neste experimento aplicaremos a técnica com TF-IDF (seção 2.4).

Uma vez que esse método trata como item de vocábulo diferente cada token não idêntico (a de caractere), diferenciando inclusive iniciais maiúsculas ou variações verbais,

realizamos uma etapa de pré-processamento com finalidade de tratar essa questão. Após o pré-processamento, as demais etapas podem ser globalmente representadas pela Figura 11. Em suma, o *dataset* foi inicialmente dividido em duas porções, uma de treino e outra de teste, sendo esta reservada para a avaliação final, de modo que o modelo seja validado com dados inéditos, isto é, não utilizados na etapa de treinamento. Com o conjunto de treino, são testados diversos modelos em um processo de validação cruzada (cf. Seção 4.5.3); o modelo de melhor desempenho é submetido a simulações que realizam combinações de hiperparâmetros. Com as configurações de melhor desempenho (segundo uma métrica determinada), é retreinado um novo modelo com a partição inteire de treino do *dataset* e ele é finalmente validado utilizando a partição de teste. Nas próximas seções, apresentaremos melhor cada um desses passos.

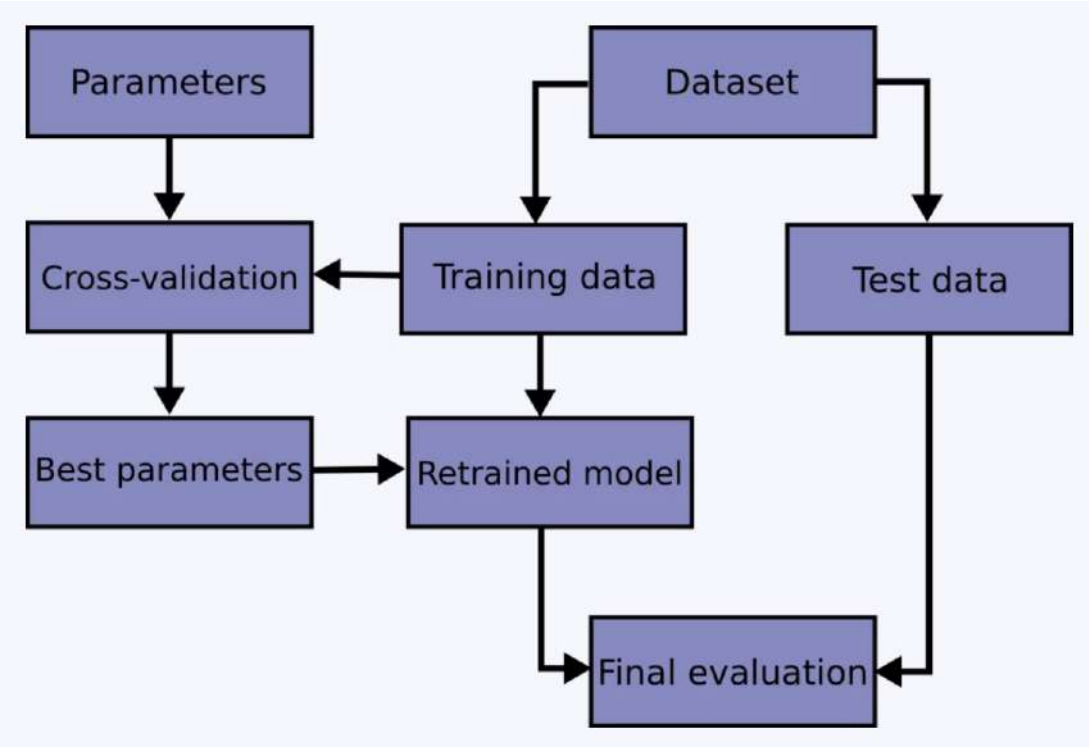

Figura 11 – Visão geral do processamento com *Bag of Words*. Fonte: (CROSS-VALIDATION. . . , 2023).

4.5.2 Pré-processamento

Inicialmente tornamos todas as letras minúsculas e substituídos os caracteres de tabulação por um espaço. Foram removidos: todos os caracteres numéricos, símbolos, pontuações, as palavras com um único caractere e *stopwords* (referenciados na seção 2.3) - para esta última remoção foi utilizada a coleção de *stopwords* em Português da biblioteca *Natural Language Toolkit* (NLTK) (LOPER; BIRD, 2002).

No passo seguinte, o texto foi dividido em uma lista de *tokens* (cada *token* sendo uma palavra), e eles foram substituídos pelos seus respectivos *lema*, utilizando o pacote *pt_*-

core_news_md da biblioteca *spaCy* (HONNIBAL *et al.*, 2020). Esse processo é conhecido por lematização: as palavras são substituídas por formas inflexíveis, isto é, uma palavra-base (chamado lema), de modo que suas diferentes variações sejam representadas pelo mesmo *token* - por exemplo, as palavras "condenação", "condeno", "condenar" e "condenado" podem ser todas representadas pelo token "condenar". Após as substituições a lista de tokens foi novamente consolidada em uma sequência de texto, com cada palavra separada da outra por um espaço.

4.5.3 Processamento

Inicialmente, as classes foram substituídas por correspondentes numéricos, de modo que cada classe foi representada por um número, ao invés do seu nome. A divisão do conjunto de dados entre treino e teste, abordada na seção 4.5.1, foi realizada na proporção 3:1, respectivamente, utilizando a função *train_test_split* da biblioteca *scikit-learn* (PEDREGOSA *et al.*, 2011). Essa divisão foi realizada com estratégia de estratificação (por meio do parâmetro *stratity* do algoritmo citado), que mantém em cada partição a mesma proporção entre as classes presente no conjunto de dados original.

Para o processamento, foi implementado um *pipeline*, isto é, uma cadeia de passos pré-definidos executados em sequência, no qual a cada passo a entrada é a saída do passo anterior. Esses passos variam segundo a estratégia de classificação. Para tanto, utilizamos a função *make_pipeline* de *imbalanced-learn*, uma biblioteca licenciada do *Massachusetts Institute of Technology* que depende do *scikit-learn* e provê ferramentas para lidar com o problema de classes desbalanceadas (LEMAîTRE; NOGUEIRA; ARIDAS, 2017). O objetivo dessa etapa foi identificar empiricamente o modelo de melhor desempenho para hiperparâmetros básicos, combinados ou não com balanceamento de classes e com redução de dimensionalidade. Os *pipelines* foram executados uma vez para cada estratégia, isto é, cada combinação de balanceamento de classes e algoritmo classificador, com uso de validação cruzada e metodologia estratificada. A execução se deu exclusivamente sobre o conjunto de dados de treinamento, preservando o recorte de teste para validação posterior.

A técnica de validação cruzada tem por objetivo reduzir o efeito de *overfitting*, que ocorre quando um algoritmo apresenta alto desempenho no conjunto de dados de treinamento e de teste, mas não possui capacidade de generalização para a aplicação em dados novos. Ela consiste em dividir o conjunto de treinamento em K porções e repetir a simulação K vezes, de modo que a cada repetição uma porção exerça papel de teste e as demais de treino, desse modo o método permite que o modelo seja testado com diferentes combinações de dados, conforme explicado por (BURMAN, 1989) e (CUNHA, 2019) e demonstrado na Figura 12. Nela, vemos que o conjunto de treinamento (*Train data*) é repartido de três modos diferentes (*"fold"*, em cinza) e a cada vez uma partição é utilizada para validação (em azul) e as outras duas (em vermelho) para realização das simulações.

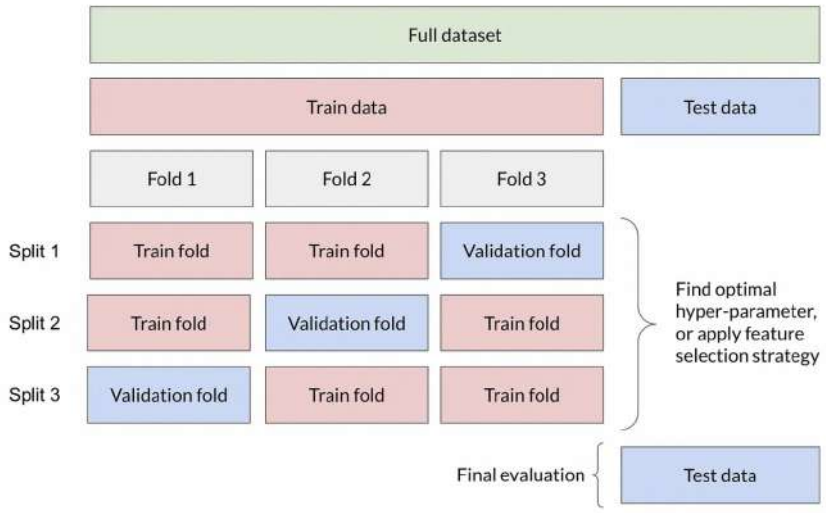

Figura 12 – Representação de *cross-validation* com K-fold, onde K = 3. Fonte: (TRANTHE, 2022)

No experimento, K foi definido com valor 10. A técnica foi aplicada com o método *cross_validate* da biblioteca *scikit-learn*. Para a divisão dos subconjuntos, aplicamos o método *StratifiedKFold* da mesma biblioteca, de modo a obter a mesma proporção entre as classes em cada recorte.

No *pipeline*, temos os seguintes passos:

1. Balanceamento de classes, se definido
2. Vetorização
3. Redução de dimensionalidade, se definida
4. Classificação

Na etapa de balanceamento das classes, foram testadas técnicas de *oversampling* e *undersampling*, além do próprio conjunto de dados sem modificações. No primeiro caso, as tuplas das classes com menos ocorrência são aleatoriamente duplicadas para se igualar à classe mais frequente, enquanto no segundo caso são descartadas tuplas das classes mais frequentes para equivalerem à classe mais rara (em termos de quantidade).

A vetorização foi realizada com o método *TfidfVectorizer* da biblioteca *scikit-learn*. Os únicos hiperparâmetros definidos foram *max_df* (eliminando os vocábulos presentes em mais de 50% dos documentos) e *min_df* (desconsiderando os vocábulos que não estejam presentes em no mínimo 10 documentos); os demais hiperparâmetros seguiram os padrões descritos na documentação da biblioteca.

Para a redução de dimensionalidade foram testadas duas estratégias. A primeira foi aplicar o algoritmo *TruncatedSVD* da biblioteca *scikit-learn*, que trunca as dimensões.

A segunda foi limitar o tamanho do vocabulário gerado pelo vetorizador com o parâmetro *max_features* do método *TfidfVectorizer*.

Foram testados os seguintes modelos de classificação, todos com os métodos disponíveis na biblioteca *scikit-learn*:

1. RidgeClassifier
2. LinearSVC
3. ComplementNB
4. RandomForestClassifier
5. LogisticRegression
6. SGDClassifier

Ao final do *pipeline*, são colhidas as métricas de acurácia, cobertura, precisão e pontuação F1 - tratadas na seção 2.7 -, que foram sendo armazenados em estruturas de dados do tipo *dataframe* da biblioteca Pandas (THE PANDAS DEVELOPMENT TEAM, 2020) e exportadas para arquivos de formato *.csv*, para posterior comparação e seleção do modelo de classificação mais performático. Todas as métricas foram computadas utilizando a biblioteca *scikit-learn*, em especial com o método "classification_report".

4.5.4 Ajuste de hiperparâmetros e validação

Como resultado da validação cruzada, teremos tabelas com métricas dos modelos de classificação sobre o conjunto de treinamento. Desse modo, identificaremos a estratégia de combinação de balanceamento de classes e classificador de melhor desempenho com base na métrica de Pontuação F1 médio entre as classes, proporcionando maior equilíbrio entre precisão e cobertura.

Identificado o *pipeline* de melhor desempenho, realizamos uma série de simulações entre combinações dos hiperparâmetros do classificador e do vetorizador. Para isso, utilizamos a função *GridSearchCV* da biblioteca *scikit-learn*. A seleção da melhor combinação de parâmetros será guiada pela métrica de Pontuação F1.

Por derradeiro, realizaremos a validação utilizando o conjunto de teste separado inicialmente na seção 4.5.3. Para isso, treinaremos um novo vetorizador e classificador (utilizando os hiperparâmetros ótimos identificados) sobre o conjunto de treinamento completo - isto é, sem validação cruzada - e o aplicaremos sobre o conjunto de dados de teste, computando as métricas finais do modelo.

4.6 Aplicando BERT

4.6.1 Visão geral das etapas

O treinamento de um novo modelo de BERT consumiria muitos recursos computacionais (e tempo). Felizmente, a arquitetura de BERT se beneficia da possibilidade de transferência de aprendizado, que tratamos na seção 2.6. Desse modo, é possível utilizar modelos pré-treinados, mesmo que para tarefas diferentes, adicionando camadas à rede neural para o desempenho da nova tarefa. A comunidade *Hugging Face* e a biblioteca *Transformers*, desenvolvida por (WOLF *et al.*, 2020), disponibilizam um enorme repositório de modelos pré-treinados com as ferramentas necessárias para utilizá-los - com os quais realizaremos o experimento.

Os modelos testados estão todos disponíveis abertamente na comunidade *Hugging Face*, tendo sido selecionados em razão da similaridade do domínio e/ou tarefa no qual foram treinados, conforme mencionado no capítulo 3. Foram os seguintes:

1. neuralmind/bert-base-portuguese-cased (SOUZA; NOGUEIRA; LOTUFO, 2020);
2. dominguesm/legal-bert-base-cased-ptbr (DOMINGUES, 2022);
3. pierreguillou/bert-base-cased-pt-lenerbr;
4. felipemaiapolo/legalnlp-bert (POLO *et al.*, 2021).

Da mesma forma que em *Bag of Words*, iniciaremos dividindo o conjunto de dados em uma repartição de treinamento e outra de teste, sendo esta resguardada exclusivamente para validação posterior. Os passos seguintes, detalhados adiante, seguem o esquema demonstrado na Figura 13, com criação de um módulo de dados, seguido de treinamento e teste dos modelos. Os logs resultantes registrados são avaliados para identificar o melhor modelo, que é carregado e validado no conjunto de dados de teste, separados inicialmente.

Todo o processo foi realizado utilizando primariamente PyTorch (PASZKE *et al.*, 2019), uma biblioteca de código livre que provê programação em *Python* para aplicação de modelos de *Machine Learning*, computando tensores com alta eficiência e compatibilidade com aceleradores, como GPUs.

4.6.2 Módulo de dados

O processamento de uma rede neural é computacionalmente muito custoso. Não é possível, com os recursos de que dispomos, processar todo o conjunto de dados de uma vez, como fizemos anteriormente com *Bag of Words*, pois não há memória suficiente para guardar os tensores. Em razão disso, criamos um módulo de dados, responsável por efetuar a entrega dos dados ao *tokenizador* e à rede neural em pacotes - neste experimento,

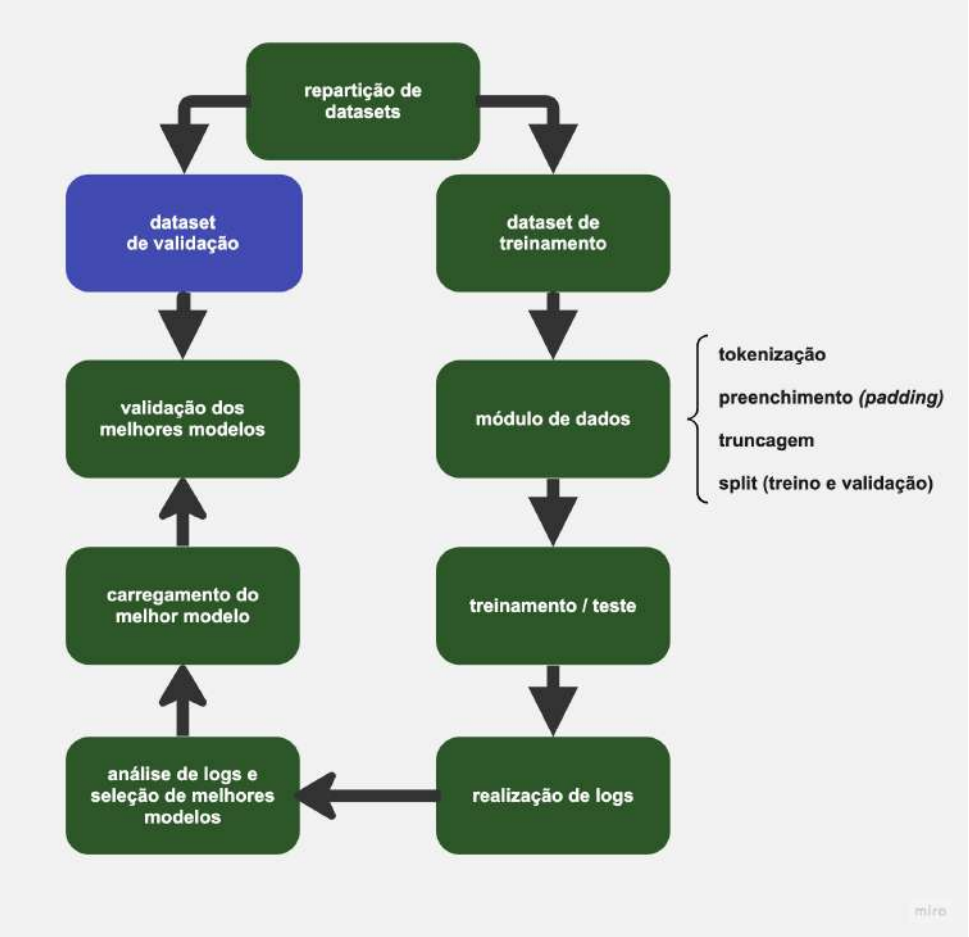

Figura 13 – Visão geral do processamento com *BERT*.

cada pacote foi composto por 16 documentos. Esse módulo foi criado utilizando a classe *LightningDataModule* (FALCON; The PyTorch Lightning team, 2019) da biblioteca *PyTorch Lightning* em conjunto com as classes *Dataset* e *DataLoader* da biblioteca PyTorch (PASZKE *et al.*, 2019). Além de gerenciar os pacotes, o módulo de dados foi responsável por realizar a *tokenização* do texto (seção 4.6.3) e por aplicar as estratégias de truncagem e de preenchimento (seção 4.6.4) dos *inputs*. Adicionalmente, o módulo também realizou a subdivisão do conjunto de dados de treinamento entre conjunto de treino e de validação, sendo aquele utilizado no treinamento em si e este para computar métricas entre as épocas.

4.6.3 Tokenização

A entrada dos modelos BERT é preparada por um *tokenizador* diretamente com o texto em formato de *string*, sem pré-processamento. O *tokenizador*, em termos gerais, representa cada *token* (que pode ser uma palavra ou parte dela) com um vetor numérico (coletado a partir de um vocabulário pré-treinado próprio para cada modelo) somado a um vetor posicional (que representa a posição daquele *token* no texto) e a um vetor do

segmento (não utilizado, nesse experimento[1], além de acrescentar *tokens* especiais - o *token* [SEP], que representa a quebra de sentenças no texto, e o *token* [CLS], representa todo o texto (sendo a soma de todos os vetores). A formação desse *input* é apresentado originalmente por (DEVLIN *et al.*, 2019) pela Figura 14.

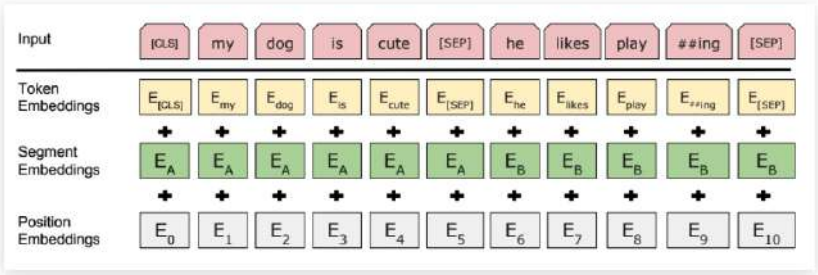

Figura 14 – Representação de uma entrada de BERT. Cada vetor da entrada é a soma dos vetores do token, segmento e posição. Fonte: (DEVLIN *et al.*, 2019).

Para realizar a *tokenização*, utilizamos o método *from_pretrained* da classe *AutoTokenizer* disponibilizada pela biblioteca *Transformers* (WOLF *et al.*, 2020), que carrega o vocabulário apropriado para o modelo a partir do repositório correspondente na comunidade *Hugging Face*. Esse mesmo algoritmo possui os parâmetros necessários para aplicar as estratégias de truncagem e preenchimento (seção 4.6.4).

4.6.4 Truncagem e preenchimento

Os modelos de BERT utilizados somente suportam uma entrada de até 512 *tokens*. Este é um hiperparâmetro definido quando do treinamento inicial do modelo e, uma vez que partimos de modelos pré-treinados, a nossa entrada está também limitada a esse valor. Nosso conjunto de dados possui muitos documentos que superam esse limite. Para tratar a questão, simulamos três estratégias: (a) eliminar os últimos *tokens*, utilizando somente os primeiros 512; (b) eliminar os primeiros *tokens*, utilizando somente os últimos 512; (c) mesclar, concatenando os primeiros 256 *tokens* aos últimos 256 *tokens*. O corte foi realizado após a tokenização e, portanto, após a inclusão dos tokens especiais de BERT. Os *tokens* que superam o limite são simplesmente descartados.

Além disso, o processamento em paralelo realizado por BERT exige que todos os *inputs* tenham a mesmo tamanho. Desse modo, aplicamos estratégia de preenchimento (*padding*) ao limite máximo de *tokens*, isto é, as entradas com menos de 512 *tokens* foram preenchidas com um vetor nulo até essa quantidade, para que todas as entradas tenham o mesmo cumprimento. Essa operação é exemplificada pela Figura 15, onde observamos que

[1] BERT aceita receber como *input* até duas *sequências de texto*, identificando-as com o vetor de segmento. Isso é útil para a execução de uma das tarefas com o qual foi treinado originalmente, a Predição de Próxima Frase)

os textos com menos de quatro *tokens* (limite hipotético do exemplo) são preenchidos com *tokens* nulos para que todos tenham o mesmo formato.

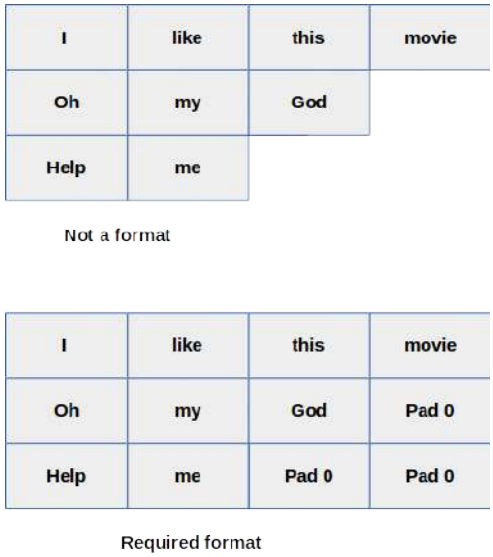

Figura 15 – Representação de preenchimento (*padding*) de *inputs* de BERT para que todos tenham o mesmo formato. Acima os textos sem preenchimento e abaixo os textos preenchidos. Fonte: (PRAKASH, 2021).

4.6.5 Ajustando BERT para a tarefa de classificação (*fine-tuning*)

Como já estabelecemos, aplicaremos modelos pré-treinados de BERT em nosso experimento. Contudo, nenhum deles foi treinado para a tarefa de classificação de textos, mas para as tarefas de reconhecimento de entidades nomeadas, descoberta de *tokens* mascarados e predição da próxima frase - tarefas geralmente utilizadas para treinar os vetores dos *tokens* de BERT no vocabulário de um domínio específico, preparando o modelo para ser ajustado e utilizado em outras tarefas.

O modelo *neuralmind/bert-base-portuguese-cased* - apelidado pelos seus criadores de BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020) - foi o primeiro modelo BERT treinado em Português do Brasil, para o que utilizou textos extraídos da *Wikipedia*. Os demais modelos testados foram treinados a partir do próprio BERTimbau executando tarefas em textos jurídicos[2] - legislações e documentos processuais (na maior parte movimentos, decisões ou despachos judiciais). Não identificamos um modelo BERT treinado especificamente com petições processuais, como é o caso do conjunto de dados do experimento.

Para poder executar a tarefa de classificação é necessário realizar um ajuste do modelo, processo conhecido como *fine tuning*, que consiste em descartar a última camada

[2] Os *datasets* utilizados por esses modelos estão disponíveis em <https://www.kaggle.com/datasets/felipepolo/brazilian-legal-proceedings> (POLO; CIOCHETTI; BERTOLO, 2021), <https://ailab.unb.br/victor/lrec2020> (ARAUJO *et al.*, 2020) e <https://github.com/peluz/lener-br> (LUZ DE ARAUJO *et al.*, 2018)

da rede neural (que realizaria a tarefa na qual o modelo foi treinado) e adicionar uma nova, a qual executa a tarefa alvo. Como demonstrado na Figura 16, a camada aqui acoplada utiliza o *token* [CLS] - formado pela soma de todos os *tokens* para representar todo o texto - e aplica uma função linear sobre a saída (*Pooler Output*).

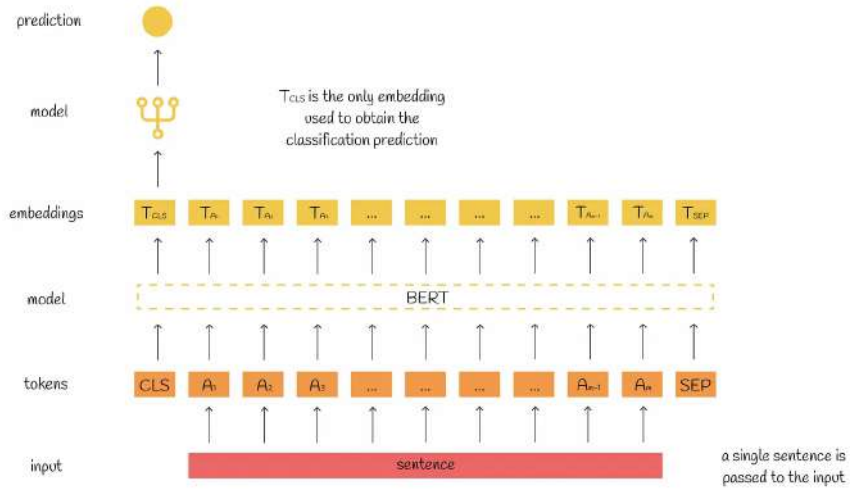

Figura 16 – Representação do processo de *fine tuning* de BERT para a tarefa de classificação de texto. Fonte: (EFIMOV, 2023).

A implementação de *fine tuning* foi realizada por meio do método *from_pretrained* da classe *BertForSequenceClassification* já disponibilizado com esse propósito pela *Transformers* (WOLF *et al.*, 2020), especificando o tipo de problema *single_label_classification*.

4.6.6 Treinamento da rede neural

Acoplar a camada de rede neural necessária para o modelo conseguir realizar a tarefa de classificação não é suficiente. Os pesos do modelo não estão naturalmente ajustados para produzir os *embeddings* que apresentem bons resultados de classificação para as classes em análise. Desse modo, deve ser realizado o treinamento do modelo durante algumas épocas para que esses parâmetros se ajustem. A cada passo do treinamento é calculada uma perda comparando o valor predito com o valor real, que mede a distância entre as distribuições probabilísticas das predições e dos verdadeiros rótulos, de modo que quanto menor o valor de perda melhor (ou mais próxima da classe correta) será a classificação realizada pela rede (KOMATA; HONDA, 2021).

O algoritmo *BertForSequenceClassification* utilizado calcula a perda com a função de entropia cruzada, recomendada para classificações com mais de uma classe (GORDON-RODRIGUEZ *et al.*, 2020) computada pelo método *CrossEntropyLoss* da biblioteca *PyTorch* (PASZKE *et al.*, 2019). A entropia cruzada é calculada pela Fórmula 4.1, onde M é o número de classes, y é um indicador binário (0 ou 1) se a classe c está correta ou incorreta para a observação o e p é a probabilidade prevista de que a observação o seja da classe c.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.1}$$

Nesse experimento, utilizamos a função AdamW como otimizador, operacionalizado pelo algoritmo de mesmo nome da biblioteca *PyTorch* (PASZKE *et al.*, 2019). O otimizador atualiza os parâmetros do modelo com base no resultado da função de perda. Esse método, em especial, é uma otimização estocástica que modifica a implementação típica de redução de peso em Adam (KINGMA; BA, 2014), desacoplando a redução de peso da atualização do gradiente (LOSHCHILOV; HUTTER, 2017).

Foram utilizados os seguintes hiperparâmetros:

1. Número de tokens por input: 512;
2. Tamanho de cada pacote de dados: 16;
3. Total de épocas: 5;
4. Taxa de aprendizado do otimizador: 2e-5;
5. Sem passos de aquecimento

O processo de treinamento foi facilitado com a implementação disponibilizada pelas classes *LightningModule* e *Trainer* da biblioteca *PyTorch Lightning* (FALCON; The PyTorch Lightning team, 2019). Essas classes manejam o processo de treinamento, aplicando nos passos e valores adequados a função de perda e otimizador, segundo os hiperparâmetros definidos. O treinamento irá processar todo o conjunto de dados de treino (o que compreende o encerramento de uma época), e, a seguir, processará o conjunto de dados de teste (sem que o otimizador atualize os parâmetros) para colher métricas de validação.

4.6.7 Logs

A realização de logs é essencial para ser possível examinar as métricas que avaliam o experimento. A aplicação das classes *LightningModule* e *Trainer* da biblioteca *PyTorch Lightning* (FALCON; The PyTorch Lightning team, 2019) também facilita esse registro. Nesse experimento, utilizaremos a classe *TensorBoardLogger* da mesma biblioteca, a qual registra os logs em formato próprio para serem explorados na plataforma *Tensorboard*[3], uma ferramenta de código livre parte do ecossistema TensorFlow, feita para a visualização métricas, com sua publicação e compartilhamento.

Ao final do processamento de cada pacote (*batch*), isto é, a cada passo do treinamento, o resultado da função de perda é registrado no log. Após o processamento de

[3] <https://tensorboard.dev/>

todo o conjunto de dados de treinamento, é realizado um processamento completo sobre o conjunto de teste, computando as métricas de validação somente ao final da época. Adicionalmente, o método *ModelCheckpoint* da biblioteca *PyTorch Lightning* (FALCON; The PyTorch Lightning team, 2019) (configurado como parâmetro de *Trainer*) salva em disco um *backup* do modelo na época atual se o resultado da métrica de Pontuação F1 da validação for superior ao das épocas anteriores.

5 AVALIAÇÃO EXPERIMENTAL

Rememorando, o projeto executa uma tarefa de classificação de documentos jurídicos (especificamente petições processuais) com PLN. Apresentamos agora os resultados desse processamento, iniciando com modelo de *Bag of words* com TF-IDF. A seguir, apresentamos os resultados do uso de BERT e ao final comparamos ambos.

5.1 Resultados de *Bag of words*

5.1.1 Resultados da validação cruzada

Enquanto para a vetorização (isto é, para a extração de vetores numéricos a partir do texto dos documentos) utilizamos uma única técnica (*Bag of Words* com TF-IDF) computada pelo algoritmo (*TfidfVectorizer* da biblioteca *scikit-learn*), para a classificação aplicamos uma série de modelos estatísticos. O objetivo dessa etapa foi identificar empiricamente o modelo de melhor desempenho (para hiperparâmetros básicos), combinados ou não com balanceamento de classes e com redução de dimensionalidade (para o que foram utilizados os algoritmos *TruncatedSVD* da biblioteca *scikit-learn* e o parâmetro *max_features* do algoritmo *TfidfVectorizer*), conforme tratamos na Subseção 4.5.3.

Nas Tabelas 2 e 3, temos os resultados das cinco melhores combinações para cada conjuntos de dados (um no qual as classes compõem o atributo Tipo e outro onde as classes estão no atributo Assunto, conforme exposto na Seção 4.2) - tabelas com resultados completos podem ser consultadas no Apêndice A. Para comparação consideramos, na ordem, as métricas de pontuação F1 média (isto é, média do resultado para cada classe) e acurácia mais elevados (métricas abordadas na Seção 2.7), além do menor tempo de treinamento do modelo.

Tabela 2 – Resultados de *cross_validation* para o conjunto de dados com classes de Tipo

Balanceamento de classes	**Método para redução de dimensionalidade**	**Modelo de classificação**	**Tempo de treinamento (s)**	**Acurácia**	**Pontuação F1 média**
sem alterações	Não utilizado	LinearSVC	23.67	0.96	0.89
sem alterações	Não utilizado	LogisticRegression	1470.2	0.96	0.88
sem alterações	Parâmetro *max_features*	LinearSVC	21.71	0.96	0.87
oversampled	TruncatedSVD	LinearSVC	343.95	0.96	0.87
sem alterações	TruncatedSVD	LinearSVC	356.5	0.96	0.87

Para ambos os tipos de classificação, o modelo estatístico por vetores de suporte linear - aplicado com o algoritmo *LinearSVC* da biblioteca *scikit-learn* - apresentou melhor

Tabela 3 – Resultados de *cross_validation* para o conjunto de dados com classes de Assunto

Balanceamento de classes	**Método para redução de dimensionalidade**	**Modelo de classificação**	**Tempo de treinamento (s)**	**Acurácia**	**Pontuação F1 média**
sem alterações	Não utilizado	LinearSVC	52.67	0.86	0.78
sem alterações	Não utilizado	LinearSVC (max_iter 100)	53.25	0.86	0.78
sem alterações	Não utilizado	LogisticRegression	518.42	0.86	0.78
sem alterações	Não utilizado	RandomForestClassifier)	5513.34	0.85	0.77
sem alterações	TruncatedSVD	RandomForestClassifier	558.57	0.84	0.76

desempenho. No caso da classificação por Assunto, outros modelos tiveram resultados similares, contudo a custos de computação mais elevados.

5.1.2 Resultados do modelo com hiperparâmetros ótimos

Observados os resultados da validação cruzada, selecionamos o modelo *LinearSVC* para prosseguir e realizamos uma sequência de simulações exaustivos com objetivo de pesquisar e determinar os hiperparâmetros mais eficientes; para tanto, utilizamos o algoritmo *GridSearchCV*, também da biblioteca *scikit-learn*, tendo a pontuação F1 média como estratégia de avaliação, chegando aos hiperparâmetros ótimos apresentados na Tabela 4.

Tabela 4 – Parâmetros ótimos identificados por *GridSearchCV*

Parâmetro	**Classificação por tipo**	**Classificação por assunto**
linearsvc__C	5	5
linearsvc__max_iter	1000	1000
tfidfvectorizer__max_df	0.4	0.3
tfidfvectorizer__max_features	None	None
tfidfvectorizer__min_df	50	10
tfidfvectorizer__ngram_range	(1, 2)	(1, 2)
tfidfvectorizer__strip_accents	None	None
tfidfvectorizer__sublinear_tf	True	True

Ambos os tipos de classificação apresentaram uma melhora sensível no desempenho geral dos modelos quanto estes foram treinados utilizando os hiperparâmetros ótimos, conforme comparado na Figura 17. Na validação dos resultados, seguindo a estratégia descrita nas seções 4.5.3 e 4.5.4 - isto é, treinando um novo vetorizador e classificador sobre o conjunto de treinamento completo utilizando os hiperparâmetros ótimos identificados e aplicado-os ao conjunto de dados de validação -, obtivemos as métricas Acurácia e Pontuação F1 (média e ponderada) apresentadas na Figura 18; nos Apêndices B e C é possível consultar detalhadamente as métricas Precisão, Cobertura e Pontuação F1 médias, ponderadas e para cada classe.

No geral, houve um bom desempenho do modelo baseado na formação de *Bag of Words*, sendo um pouco mais eficiente para a classificação dos documentos por tipo

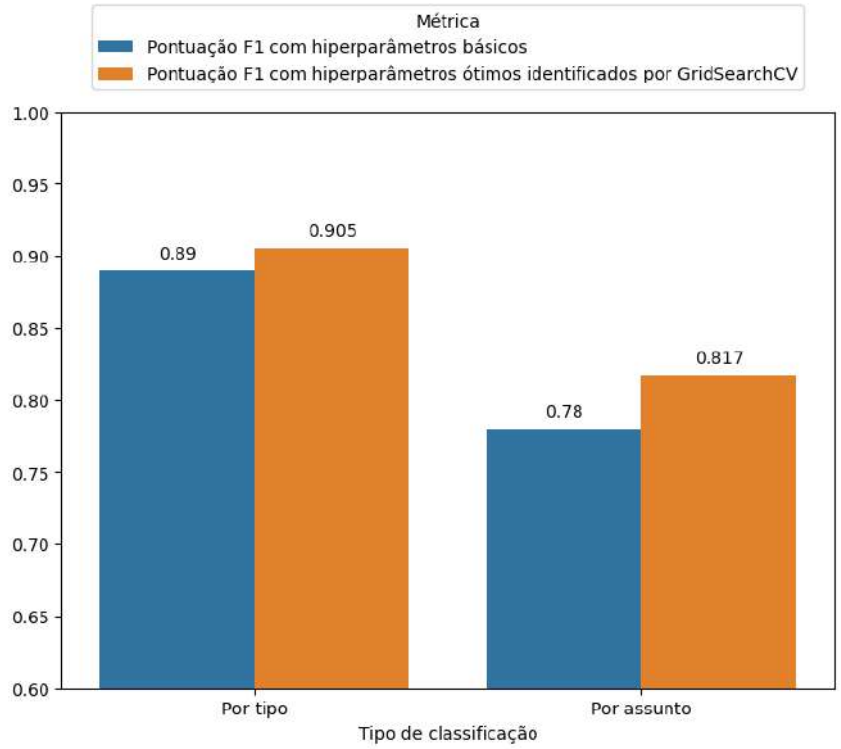

Figura 17 – Comparação da métrica Pontuação F1 com os hiperparâmetros básicos e ótimos identificados

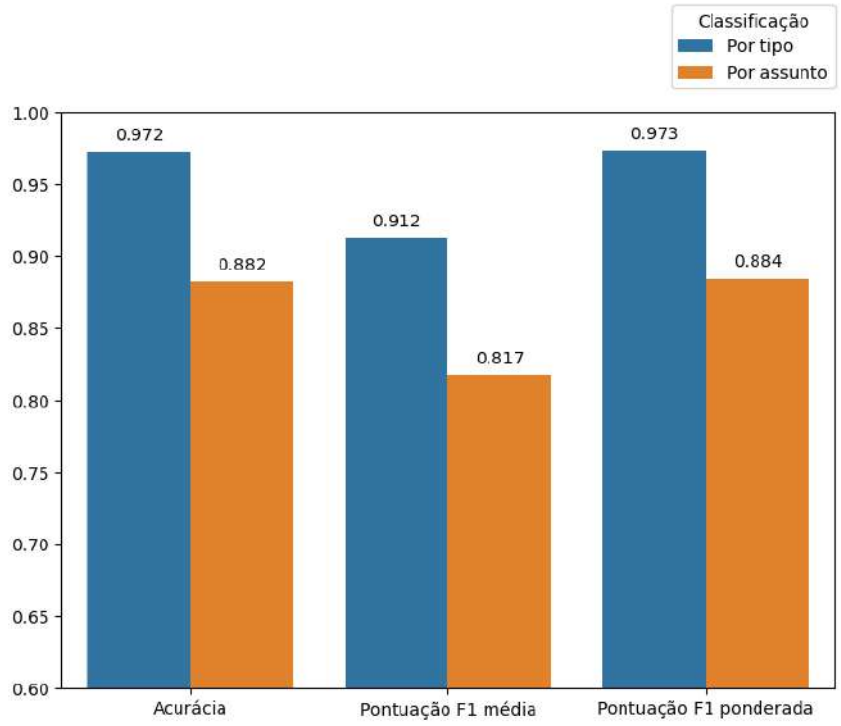

Figura 18 – Métricas gerais sobre o conjunto de validação do modelo reconstruído com hiperparâmetros ótimos identificados

(estrutura ou finalidade da petição) do que por assunto (semântica). Acreditamos que a dificuldade de classificação semântica se dá porque alguns tipos de documentos que compartilham uma estrutura ou finalidade comum não utilizam expressões específicas para os diferentes assuntos.

Observamos que nem a existência de uma menor quantidade de classes e nem o uso de uma maior quantidade de documentos para o treinamento importaram em maior eficiência do modelo, visto que a classificação por Assunto obteve desempenho inferior à classificação por Tipo, apesar daquela conter significativamente menos classes e possuir um conjunto de treinamento expressivamente maior (mais do que o dobro de documentos), como abordado na Seção 4.2 (apesar de ambos os *datasets* conterem os mesmos documentos, para o treinamento do conjunto com classificação por Tipo, foram inicialmente descartados os documentos da classe "DIVERSOS"). Além disso, não foi possível verificar uma correlação direta entre a quantidade de documentos em cada classe e um melhor desempenho do modelo (segundo a métrica de Pontuação F1), como se verifica no Apêndice D.

Finalmente, com a biblioteca WordCloud, desenvolvida por (MUELLER, 2023), criamos as visualizações em nuvens de palavras (onde os vocábulos de maior frequência são apresentados em maior tamanho) disponibilizadas nos Apêndices F e G, que nos ajudam a identificar as principais palavras utilizadas para realizar as classificações pelos modelos de *Bag of Words*.

5.2 Resultados de *BERT*

5.2.1 Comparação de resultados dos modelos

Os logs de processamento de cada modelo foram armazenados utilizando *TensorBoardLogger* da biblioteca *PyTorch Lightning* (FALCON; The PyTorch Lightning team, 2019). A cada passo do treinamento (isto é, ao final do processamento de cada pacote de dados) foi armazenado o resultado da função de perda e, ao final de cada época, um relatório com as métricas calculadas sobre o conjunto de dados de validação (Acurácia, Precisão, Cobertura e Pontuação F1), como descrito na seção 4.6.7. A relação completa de modelos e os respectivos "ids" que referenciaremos nas próximas Tabelas e Figuras podem ser consultados no Apêndice H.

A seguir, nas Figuras 19 e 20, apresentamos as métricas dos cinco melhores modelos (com avaliação pela Pontuação F1 média) de cada tipo de classificação. No Apêndice I é possível consultar métricas de mais modelos e, caso seja de interesse, as métricas de todos os modelos, juntamente como os valores de perda por passo, podem ser exploradas na plataforma *TensorBoard* [1].

[1] Para a classificação por Tipo acesse o link <https://tensorboard.dev/experiment/GaVJI5orT42xmh4TAKf3vQ> e para a classificação por Assunto acesse o link <https:

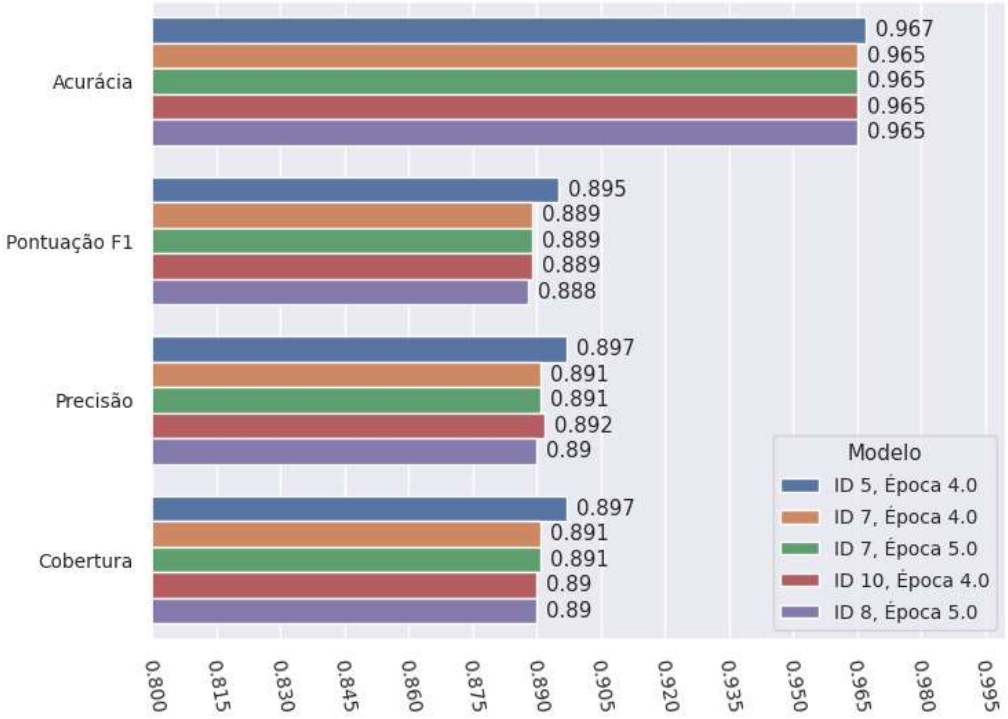

Figura 19 – Métricas comparadas dos melhores cinco modelos (avaliados pela Pontuação F1 média) para a classificação por Tipo.

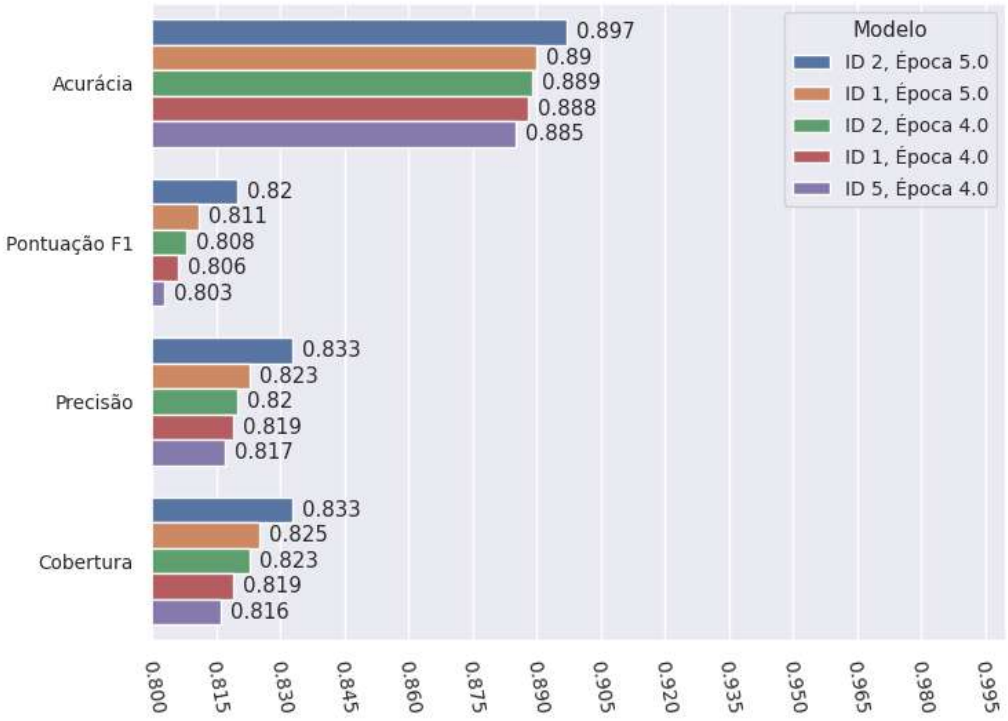

Figura 20 – Métricas comparadas dos melhores cinco modelos (avaliados pela Pontuação F1 média) para a classificação por Assunto.

//tensorboard.dev/experiment/aN73ec5aTmKOZGcrMR9Alw>

Observa-se que geralmente, mas nem sempre, os modelos começam a perder desempenho após a quarta época de treinamento. Além disso, modelos diferentes apresentaram melhores métricas em cada uma das tarefas de classificação, embora o modelo *felipemaiapolo/legalnlp-bert* com estratégia de truncagem combinando os tokens iniciais e finais tenha se destacado. A combinação de tokens na truncagem, inclusive, revelou-se melhor estratégia em ambos os casos.

5.2.2 Validação dos melhores modelos

A validação do modelo é realizada com dados inéditos - o conjunto de dados separado inicialmente para teste, conforme tratado na seção 4.6.1. Para cada tipo de classificação, realizamos uma época de validação utilizando o modelo pré-treinado (sem ajustes finos) e em seguida realizamos também uma época sobre os mesmos dados com o modelo ajustado que apresentou a melhor métrica Pontuação F1 média. As métricas computadas na validação do modelo estão apresentadas nas Figuras 21 e 22

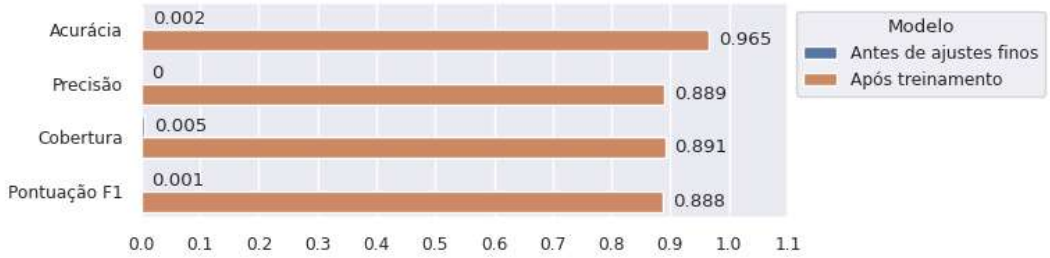

Figura 21 – Validação do modelo para classificação por Tipo: comparação de métricas do modelo com e sem ajustes finos aplicado sobre conjunto de dados inédito.

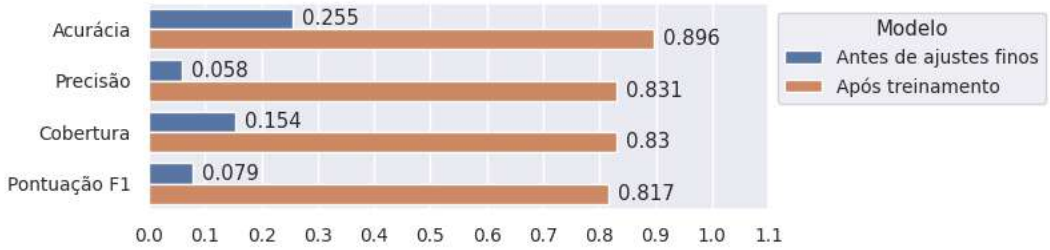

Figura 22 – Validação do modelo para classificação por Assunto: comparação de métricas do modelo com e sem ajustes finos aplicado sobre conjunto de dados inédito.

O modelo pré-treinado, sem realização de treinamento adicional - isto é ajustes finos (*fine tuning*) -, demonstrou desempenho praticamente nulo. Contudo, com poucas épocas de treinamento, os resultados se tornaram relevantes. As métricas calculadas com detalhamento para cada classe podem ser consultadas nos Apêndices J e K. Do mesmo modo que em *Bag of Words* (seção 5.1.2, não ficou demonstrada uma relação direta entre a quantidade de exemplos disponíveis na classe e um melhor desempenho representado pela Pontuação F1, como observado no Apêndice L.

5.3 Comparação de resultados de *Bag of Words* e *BERT*

As Figuras 23 e 24 comparam as métricas obtidas na validação dos modelos identificados como de melhor desempenho segundo a Pontuação F1 para cada tipo de classificação. No geral, o modelo baseado em *Bag of Words* obteve melhores resultados para a classificação dos documentos por Tipo, enquanto o modelo baseado em BERT apresentou melhores resultados para a classificação por Assunto.

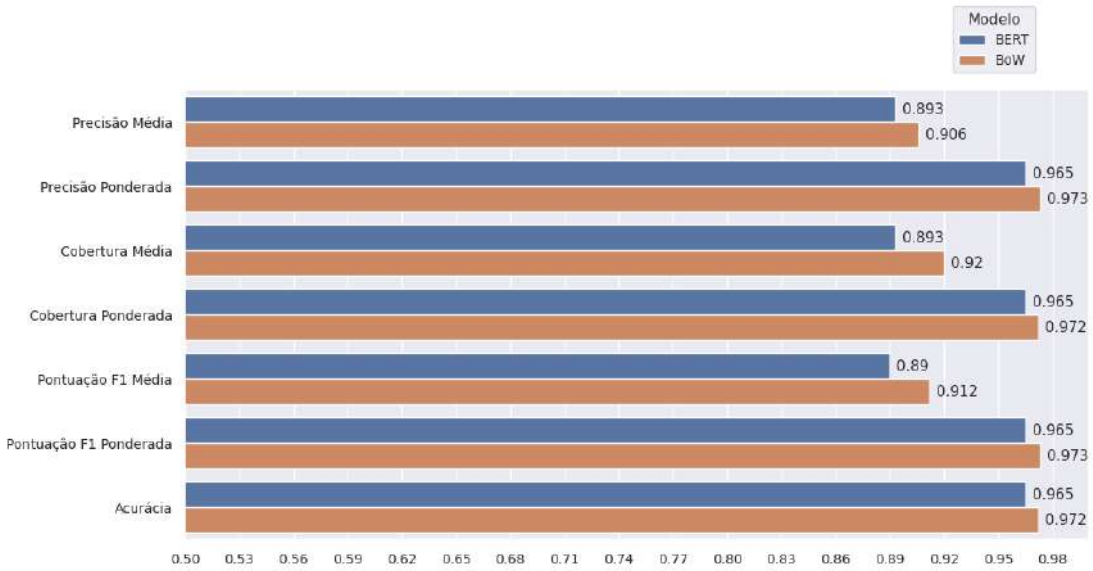

Figura 23 – Comparação de métricas dos modelos de *Bag of Words* e BERT de melhor desempenho para a classificação por Tipo.

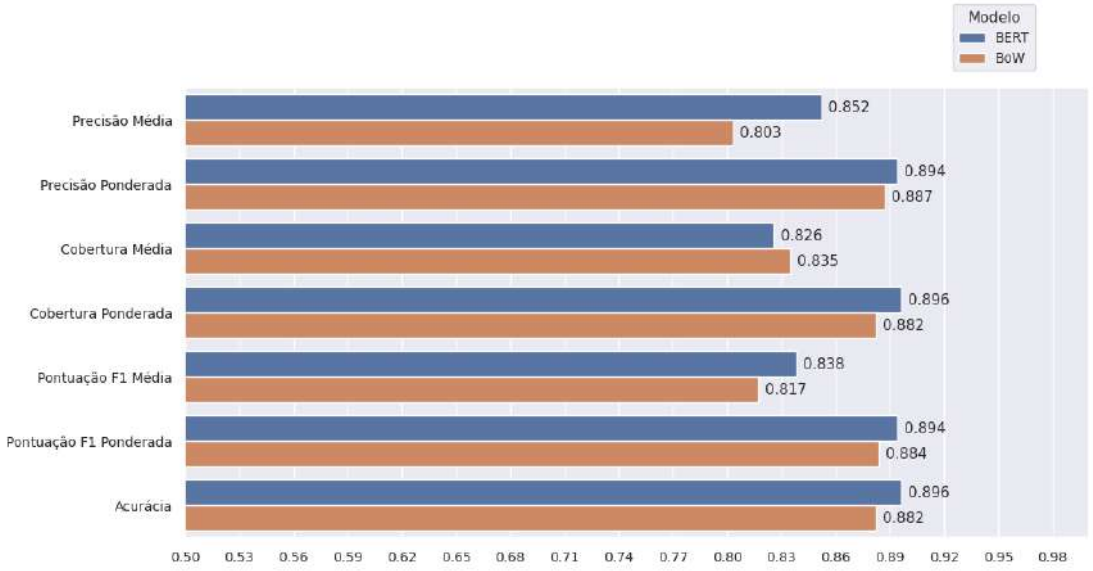

Figura 24 – Comparação de métricas dos modelos de *Bag of Words* e BERT de melhor desempenho para a classificação por Assunto.

5.4 Análise qualitativa

Esta seção consiste numa exploração de amostas dos documentos classificados pelos algoritmos do experimento, tendo por finalidade avaliar a classificação e identificar possíveis motivos para erros e/ou divergências.

5.4.1 Classificação por Tipo

Nesse ponto, devemos recordar que as classificações reais foram realizadas por colaboradores da Defensoria Pública do Estado de Rondônia no momento da apresentação do documento para protocolo em um sistema de processo judicial eletrônico. O rol de classificações disponível para o usuário é uma decisão de Governança e não necessariamente intrínseca a documentos jurídicos. Não se deve descartar a ocorrência de dúvidas e erros dos colaboradores, a deficiência da parametrização ou a possibilidade de documentos que se enquadrariam perfeitamente em mais de uma classe (embora somente uma pudesse ser selecionada pelo usuário).

Exploramos os resultados nos quais ao menos uma das classificações preditas (por *Bag of Words* ou por *BERT* estava em desacordo com a classificação real. Alguns exemplos podem ser observados no Anexo B. Observamos, como evidenciam as matrizes de confusão nos Apêndices E e M, uma maior incidência de divergência entre classes que são muito similares e cuja diferença só existe por decisão de Governança.

O maior grau de confusão ocorre entre as classes "memoriais" e "alegações finais" - que, em verdade, são o mesmo tipo de documento e, possivelmente, derivam de um erro de parametrização.

Em seguida, está a classe "ciência de audiência" muitas vezes confundida com "ciência" - de fato, toda ciência de audiência é também uma ciência e as duas classes só existem por uma decisão de Governança. O mesmo acontece com as classes "apelação" e "recurso" e as classes "embargos à execução" e "petição inicial" - onde os primeiros são espécies jurídicas dos segundos. Ao observar exemplos desses documentos e suas classificações, identificamos algumas situações nas quais o usuário classificou com o gênero ("ciência", "recurso" ou "petição inicial"), enquanto o modelo identificou a subespécie aplicável ("ciência de audiência", "apelação" ou "embargos à execução") e vice-versa.

A nosso ver, os resultados apontam à hipótese de que a aplicação dos modelos em produção poderá auxiliar na identificação de deficiências de parametrização e de erros comuns de usuários, melhorando a capacidade de tomada de decisão da alta administração para Governança de sistemas.

5.4.2 Classificação por Assunto

Diferentemente da classificação por Tipo, a classificação por Assunto é menos suscetível a erros de usuários porque esta é obtida a partir de dados oficiais do sistema judicial eletrônico que comandam a distribuição de competência do processo.

Realizando a exploração, observamos que a dificuldade do modelo em realizar essa classificação (que apresenta métricas gerais bem inferiores à classificação por Tipo) pode advir da inexistência - nos documentos mais frequentes - de elementos de distinção semântica do assunto. Diversos tipos de petição possuem a mesma apresentação textual, apesar de serem de assuntos diversos, porque voltados a uma mesma finalidade no processo (como declarar ciência de uma decisão judicial, sem manifestar de qualquer modo o assunto tratado no processo).

Com isso, levantamos a hipótese de que os resultados podem ser melhores ao restringir o treinamento e aplicação em momentos processuais em que a descoberta do assunto é mais relevante, como na confecção de documentos que inauguram processos - quando a informação de assunto é relevante para apontar a competência (que, por sua vez, define o órgão responsável por julgar o processo).

6 CONCLUSÕES

Neste estudo, nós investigamos aplicação de técnicas de Processamento de Línguas Naturais (PLN) baseadas na formação de *Bag of Words* e na aplicação de BERT para a classificação de petições jurídicas por Tipo (finalidade processual) e por Assunto (competência processual). O conjunto de dados consistiu de cerca de 350 mil petições produzidas por colaboradores da Defensoria Pública do Estado de Rondônia com utilização do SOLAR. Nos resultados gerais de ambos os modelos, as métricas para a classificação por Tipo (que obteve uma Acurácia de 97,2% e Pontuação F1 média de 91,2%) foram mais satisfatórias do que a classificação por Assunto (que obteve uma Acurácia de 89,6% e Pontuação F1 média de 83,8%). O resultado do processamento apontou um desempenho superior do modelo com formação de *Bag of Words* para a classificação por Tipo, enquanto o modelo com aplicação de BERT foi superior para a classificação por Assunto.

Quanto às hipóteses levantadas (conforme seção 1.2), concluímos que modelos tradicionais de classificação (como *Bag of Words*) podem ser eficazes para a classificação de textos do domínio jurídico e modelos distribucionais (como BERT) não necessariamente apresentarão desempenho superior. Concluímos ainda que não existe uma relação direta entre a quantidade de documentos de uma determinada classe e um melhor desempenho dos modelos para classificá-la. Não pudemos confirmar a hipótese de que os modelos estudados são mais eficientes na classificação semântica (por Assunto) do que na classificação estrutural (por Tipo).

A execução dos experimentos foi satisfatória, resultando na criação de quatro modelos de classificação com PLN, que foram disponibilizados em repositórios públicos[1], assim como os códigos[2] necessários para produzi-los e para treinar novos modelos com a mesma sistemática. Avaliamos que os modelos podem ser colocados em produção para auxiliar os usuários do SOLAR, agilizando o fluxo de peticionamento do sistema com a sugestão de classificação automatizada, além de minimizar a ocorrência de erros e a ausência de catalogação. A aplicação do modelo e estudo dos seus resultados também pode auxiliar os administradores na tomada de decisões de Governança.

Em trabalhos futuros, esperamos testar o desempenho dos modelos BERT com outros hiperparâmetros e investigar a aplicação de novas tarefas de PLN em petições processuais, como o reconhecimento de entidades nomeadas (visando a identificar e mascarar dados pessoais presentes nos documentos), a modelagem de tópicos e a similaridade entre documentos - essas tarefas podem auxiliar o desenvolvimento de um banco de petições que proporcione proteção à privacidade e ferramentas eficientes de pesquisa. Esperamos

[1] Repositório no *Hugging Face*: <https://huggingface.co/kelsensantos>
[2] Repositório no *GitHub*: <https://github.com/kelsensantos/juspln>

também evoluir a técnica de ajuste de modelos pré-treinados para classificação por Assunto, utilizando documentos hipoteticamente mais apropriados, onde a relevância semântica é mais acentuada (como petições iniciais e recursos). Existe ainda oportunidade para afinamento de modelos BERT pré-treinados para aplicação especializada em vocabulário jurídico específico de petições processuais - com tarefas de maior capacidade de abstração, como a descoberta de palavras mascaradas e/ou similaridade semântica entre sentenças.

REFERÊNCIAS

ACADEMIA BRASILEIRA DE LETRAS. **Vocabulário Ortográfico da Língua Portuguesa**. 2022. Disponível em: <https://www.academia.org.br/nossa-lingua/busca-no-vocabulario>. Acesso em: 11 de março de 2022.

AGUIAR, A. *et al.* Using topic modeling in classification of brazilian lawsuits. *In*: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. [*S.l.: s.n.*], 2022. p. 233–242.

ARAUJO, P. H. Luz de *et al.* VICTOR: a dataset for Brazilian legal documents classification. *In*: **Proceedings of the Twelfth Language Resources and Evaluation Conference**. Marseille, France: European Language Resources Association, 2020. p. 1449–1458. ISBN 979-10-95546-34-4. Disponível em: <https://aclanthology.org/2020.lrec-1.181>.

BOJANOWSKI, P. *et al.* Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, v. 5, p. 135–146, 2017. ISSN 2307-387X.

BRAZ, F. A. *et al.* Document classification using a bi-lstm to unclog brazil's supreme court. **CoRR**, abs/1811.11569, 2018. Disponível em: <http://arxiv.org/abs/1811.11569>. Acesso em: 24 de setembro de 2023.

BURMAN, P. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. **Biometrika**, v. 76, p. 503–514, 09 1989. Disponível em: <https://doi.org/10.1093/biomet/76.3.503>. Acesso em: 24 de setembro de 2023.

CROSS-VALIDATION: evaluating estimator performance. 2023. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 28 de maio de 2023.

CUNHA, J. P. Z. **Um estudo comparativo das técnicas de validação cruzada aplicadas a modelos mistos**. 2019. Tese (Doutorado) — Universidade de São Paulo, 2019.

DEVLIN, J. *et al.* BERT: Pre-training of deep bidirectional transformers for language understanding. *In*: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>.

DOMINGUES, M. **Language Model in the legal domain in Portuguese**. 2022. <https://huggingface.co/dominguesm/legal-bert-base-cased-ptbr/>.

EFIMOV, V. **Large Language Models: BERT**. 2023. Disponível em: <https://towardsdatascience.com/bert-3d1bf880386a>. Acesso em: 14 de setembro de 2023.

FACURE, M. **Aprendendo Representações de Palavras**. 2017. Disponível em: <https://matheusfacure.github.io/2017/03/20/word2vec/>. Acesso em: 17 de março de 2023.

FALCON, W.; The PyTorch Lightning team. **PyTorch Lightning**. 2019. Disponível em: <https://github.com/Lightning-AI/lightning>.

FILHO, J. A. W. *et al.* The brwac corpus: a new open resource for brazilian portuguese. *In*: **Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)**. [*S.l.: s.n.*], 2018.

GORDON-RODRIGUEZ, E. *et al.* Uses and abuses of the cross-entropy loss: Case studies in modern deep learning. *In*: FORDE, J. Z. *et al.* (ed.). **Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops**. PMLR, 2020. (Proceedings of Machine Learning Research, v. 137), p. 1–10. Disponível em: <https://proceedings.mlr.press/v137/gordon-rodriguez20a.html>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

HONNIBAL, M. *et al.* **spaCy: Industrial-strength Natural Language Processing in Python**. [*S.l.: s.n.*]: Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.1212303>. Acesso em: 24 de setembro de 2023.

HOONLOR, A. **Sequential patterns and temporal patterns for text mining**. [*S.l.: s.n.*]: Rensselaer Polytechnic Institute, 2011.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing**. 2023. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Acesso em: 24 de setembro de 2023.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KOMATA, K. K.; HONDA, Y. Y. **Aplicação do BERT para Classificação de Acórdãos do STF por Ramos do Direito**. 2021. Disponível em: <https://linux.ime.usp.br/~yurickyh/mac0499/MAC0499_Monografia.pdf>.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017. Disponível em: <http://jmlr.org/papers/v18/16-365.html>. Acesso em: 24 de setembro de 2023.

LOPER, E.; BIRD, S. **Nltk: The natural language toolkit**. 2002. Disponível em: <https://doi.org/10.48550/arXiv.cs/0205028>. Acesso em: 24 de setembro de 2023.

LOSHCHILOV, I.; HUTTER, F. Fixing weight decay regularization in adam. **CoRR**, abs/1711.05101, 2017. Disponível em: <http://arxiv.org/abs/1711.05101>.

LUHN, H. P. A statistical approach to mechanized encoding and searching of literary information. **IBM Journal of Research and Development**, v. 1, n. 4, p. 309–317, 1957.

LUZ DE ARAUJO, P. H. *et al.* LeNER-Br: a dataset for named entity recognition in Brazilian legal text. *In*: **International Conference on the Computational Processing of Portuguese (PROPOR)**. Canela, RS, Brazil: Springer, 2018. (Lecture Notes on Computer Science (LNCS)), p. 313–323. Disponível em: <https://teodecampos.github.io/LeNER-Br/>.

MARIANO, D. Métricas de avaliação em machine learning: acurácia, sensibilidade, precisão, especificidade e f-score. *In*: _____. [*S.l.: s.n.*], 2021. ISBN 9786599275326.

MIKOLOV, T. *et al.* **Efficient Estimation of Word Representations in Vector Space**. arXiv, 2013. Disponível em: <https://arxiv.org/abs/1301.3781>.

MIKOLOV, T. *et al.* **Distributed Representations of Words and Phrases and their Compositionality**. arXiv, 2013. Disponível em: <https://arxiv.org/abs/1310.4546>.

MITCHELL, T. M. **Machine Learning**. [*S.l.: s.n.*]: McGraw-Hill Science/Engineering/Math, 1997. Disponível em: <https://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html>. Acesso em: 24 de setembro de 2023. (McGraw-Hill International Editions). ISBN 0070428077.

MUELLER, A. C. **Wordcloud**. 2023. Disponível em: <https://github.com/amueller/word_cloud>. Acesso em: 24 de setembro de 2023.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. **Pesquisa identifica 111 projetos de inteligência artificial no Judiciário**. 2022. Disponível em: <https://brasil.un.org/pt-br/188306-pesquisa-identifica-111-projetos-de-inteligencia-artificial-no-judiciario>. Acesso em: 21 de fevereiro 2023.

PASZKE, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. *In*: WALLACH, H. *et al.* (ed.). **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. *In*: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [*S.l.: s.n.*], 2014. p. 1532–1543.

POLO, F. *et al.* Legalnlp - natural language processing methods for the brazilian legal language. *In*: **Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2021. p. 763–774. ISSN 2763-9061. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/18301>.

POLO, F. M.; CIOCHETTI, I.; BERTOLO, E. Predicting legal proceedings status: approaches based on sequential text data. *In*: **Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law**. [*S.l.: s.n.*], 2021. p. 264–265.

PRAKASH, P. **An Explanatory Guide to BERT Tokenizer**. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/09/an-explanatory-guide-to-bert-tokenizer/>. Acesso em: 14 de setembro de 2023.

RADFORD, A. *et al.* Improving language understanding by generative pre-training. 2018. Disponível em: <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>. Acesso em: 24 de setembro de 2023.

SERRAS, F. R.; FINGER, M. verbert: Automating brazilian case law document multi-label categorization using bert. **arXiv preprint arXiv:2203.06224**, 2022.

SILVA, N. C. D. *et al.* Document type classification for brazil's supreme court using a convolutional neural network. *In*: **10th International Conference on Forensic Computer Science and Cyber Law (ICoFCS), Sao Paulo, Brazil**. [*S.l.: s.n.*], 2018. p. 29–30.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. *In*: CERRI, R.; PRATI, R. C. (ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8.

SOUZA, F. D.; FILHO, J. B. d. O. e. S. Embedding generation for text classification of brazilian portuguese user reviews: from bag-of-words to transformers. **Neural Computing and Applications**, Springer, p. 1–14, 2022.

SUPREMO TRIBUNAL FEDERAL. **Inteligência artificial permitirá classificação dos processos do STF sob a ótica dos direitos humanos**. 2023. Disponível em: <https://portal.stf.jus.br/noticias/verNoticiaDetalhe.asp?idConteudo=487134&ori=1>, Acesso em: 28 de maio 2023.

THE PANDAS DEVELOPMENT TEAM. **pandas-dev/pandas: Pandas**. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Acesso em: 24 de setembro de 2023.

TRAN-THE, T. D. **How cross-validation can go wrong and how to fix it**. 2022. Disponível em: https://towardsdatascience.com/how-cross-validation-can-go-wrong-and-how-to-fix-it-feature-selection-use-case-with-sample-code-abf928be9080 Acesso em: 28 de maio 2023.

VASWANI, A. *et al.* Attention is all you need. *In*: GUYON, I. *et al.* (ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.

VIEGAS, C. F. O. Jurisbert: Transformer-based model for embedding legal texts. Fundação Universidade Federal de Mato Grosso do Sul, 2022. Disponível em: <https://repositorio.ufms.br/handle/123456789/5119>. Acesso em: 24 de setembro de 2023.

WOLF, T. *et al.* Transformers: State-of-the-art natural language processing. *In*: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**. Association for Computational Linguistics, 2020. p. 38–45. Disponível em: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>. Acesso em: 24 de setembro de 2023.

ZANUZ, L.; RIGO, S. J. Fostering judiciary applications with new fine-tuned models for legal named entity recognition in portuguese. *In*: PINHEIRO, V. *et al.* (ed.).

Computational Processing of the Portuguese Language. Cham: Springer International Publishing, 2022. p. 219–229. ISBN 978-3-030-98305-5.

APÊNDICES

APÊNDICE A – RESULTADOS DE CROSS_VALIDATION DOS MODELOS DE CLASSIFICAÇÃO POR FORMAÇÃO DE *BAG OF WORDS* SOBRE OS CONJUNTOS DE DADOS DE TREINAMENTO E DE TESTE PARA AS CLASSIFICAÇÕES POR TIPO E POR ASSUNTO

Ordem	Balanceamento de classes	Método para redução de dimensionalidade	Modelo de classificação	Tempo de treinamento (s)	Acurácia média	Pontuação F1 média
1	sem alterações	Não utilizado	LinearSVC(C=1)	52.67	0.86	0.78
2	sem alterações	Não utilizado	LinearSVC(C=1, max_iter=100)	53.25	0.86	0.78
3	sem alterações	Não utilizado	LogisticRegression(C=5, max_iter=1000)	518.42	0.86	0.78
4	sem alterações	Não utilizado	RandomForestClassifier()	5513.34	0.85	0.77
5	sem alterações	TruncatedSVD(n_components=100)	RandomForestClassifier()	558.57	0.84	0.76
6	sem alterações	TruncatedSVD(n_components=1000)	LogisticRegression(C=5, max_iter=1000)	1825.70	0.84	0.75
7	sem alterações	TruncatedSVD(n_components=1000)	LinearSVC(C=1, max_iter=100)	885.40	0.84	0.74
8	sem alterações	TruncatedSVD(n_components=1000)	RandomForestClassifier()	2283.74	0.83	0.74
9	undersampled	Não utilizado	LinearSVC(C=1, max_iter=100)	45.56	0.80	0.74
10	undersampled	Não utilizado	LogisticRegression(C=5, max_iter=1000)	167.74	0.80	0.74
11	undersampled	Não utilizado	LinearSVC(C=1)	28.62	0.80	0.73
12	undersampled	Não utilizado	RidgeClassifier(solver='sparse_cg', tol=0.01)	52.38	0.80	0.73
13	undersampled	Não utilizado	RandomForestClassifier()	914.39	0.79	0.73
14	sem alterações	Parâmetro 'max_features'	LinearSVC(C=1)	37.69	0.82	0.71
15	sem alterações	TruncatedSVD(n_components=100)	KNeighborsClassifier(n_neighbors=20)	88.54	0.81	0.70
16	sem alterações	TruncatedSVD(n_components=10)	RandomForestClassifier()	185.23	0.80	0.70
17	undersampled	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	36.92	0.77	0.70
18	sem alterações	TruncatedSVD(n_components=1000)	RidgeClassifier(solver='sparse_cg', tol=0.01)	788.55	0.82	0.69
19	oversampled	RandomOverSampler(random_state=0)	KNeighborsClassifier(n_neighbors=20)	134.43	0.76	0.68
20	sem alterações	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	41.61	0.81	0.67
21	undersampled	Não utilizado	ComplementNB(alpha=0.1)	36.75	0.75	0.67
22	sem alterações	TruncatedSVD(n_components=100)	LogisticRegression(C=5, max_iter=1000)	206.73	0.79	0.66
23	sem alterações	TruncatedSVD(n_components=1000)	SGDClassifier(early_stopping=True, loss='log_l...	793.11	0.80	0.65
24	sem alterações	Não utilizado	ComplementNB(alpha=0.1)	32.66	0.78	0.64
25	sem alterações	TruncatedSVD(n_components=100)	LinearSVC(C=1, max_iter=100)	119.75	0.78	0.62
26	sem alterações	TruncatedSVD(n_components=100)	LinearSVC(C=1, max_iter=100)	171.82	0.78	0.62
27	oversampled	RandomOverSampler(random_state=0)	KNeighborsClassifier(n_neighbors=20)	73.43	0.68	0.62
28	sem alterações	TruncatedSVD(n_components=100)	RidgeClassifier(solver='sparse_cg', tol=0.01)	88.58	0.77	0.59
29	sem alterações	TruncatedSVD(n_components=100)	SGDClassifier(early_stopping=True, loss='log_l...	125.69	0.77	0.59
30	sem alterações	TruncatedSVD(n_components=100)	RidgeClassifier(solver='sparse_cg', tol=0.01)	131.92	0.77	0.59
31	sem alterações	TruncatedSVD(n_components=10)	LogisticRegression(C=5, max_iter=1000)	77.63	0.59	0.35
32	sem alterações	TruncatedSVD(n_components=10)	SGDClassifier(early_stopping=True, loss='log_l...	62.88	0.58	0.32
33	sem alterações	TruncatedSVD(n_components=10)	LinearSVC(C=1, max_iter=100)	81.87	0.57	0.32
34	sem alterações	TruncatedSVD(n_components=10)	RidgeClassifier(solver='sparse_cg', tol=0.01)	60.62	0.56	0.29

Tabela 5 – Métricas obtidas por Cross_Validation para classificação por Assunto.

Ordem	Balanceamento de classes	Método para redução de dimensionalidade	Modelo de classificação	Tempo de treinamento (s)	Acurácia média	Pontuação F1 média
1	sem alterações	Não utilizado	LinearSVC(C=1, max_iter=100)	23.67	0.96	0.89
2	sem alterações	Não utilizado	LogisticRegression(C=5, max_iter=1000)	1470.20	0.96	0.88
3	sem alterações	Parâmetro 'max_features'	LinearSVC(C=1)	21.71	0.96	0.87
4	oversampled	TruncatedSVD(n_components=1000)	LinearSVC(C=1, max_iter=100)	343.95	0.96	0.87
5	sem alterações	TruncatedSVD(n_components=1000)	LinearSVC(C=1, max_iter=100)	356.50	0.96	0.87
6	sem alterações	TruncatedSVD(n_components=1000)	LinearSVC(C=1, max_iter=100)	374.09	0.96	0.87
7	sem alterações	TruncatedSVD(n_components=1000)	LogisticRegression(C=5, max_iter=1000)	409.06	0.96	0.87
8	sem alterações	TruncatedSVD(n_components=1000)	LogisticRegression(C=5, max_iter=1000)	449.72	0.96	0.87
9	sem alterações	TruncatedSVD(n_components=1000)	LogisticRegression(C=5, max_iter=1000)	457.01	0.96	0.87
10	oversampled	Não utilizado	LinearSVC(C=1, max_iter=100)	390.25	0.94	0.87
11	undersampled	TruncatedSVD(n_components=1000)	LinearSVC(C=1)	111.82	0.96	0.86
12	sem alterações	Não utilizado	RandomForestClassifier()	238.65	0.95	0.85
13	sem alterações	Parâmetro 'max_features'	LinearSVC(C=1)	22.14	0.95	0.83
14	sem alterações	Não utilizado	RidgeClassifier(solver='sparse_cg', tol=0.01)	58.36	0.95	0.82
15	oversampled	Não utilizado	RidgeClassifier(solver='sparse_cg', tol=0.01)	5430.39	0.90	0.82
16	sem alterações	TruncatedSVD(n_components=1000)	RandomForestClassifier()	1397.98	0.94	0.80
17	sem alterações	TruncatedSVD(n_components=1000)	RandomForestClassifier()	1407.81	0.94	0.80
18	sem alterações	TruncatedSVD(n_components=1000)	RidgeClassifier(solver='sparse_cg', tol=0.01)	316.04	0.93	0.74
19	sem alterações	TruncatedSVD(n_components=1000)	RidgeClassifier(solver='sparse_cg', tol=0.01)	328.29	0.93	0.74
20	undersampled	Não utilizado	RandomForestClassifier()	14.59	0.84	0.73
21	undersampled	Não utilizado	LogisticRegression(C=5, max_iter=1000)	21.19	0.83	0.73
22	oversampled	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	188.82	0.84	0.72
23	undersampled	Não utilizado	LinearSVC(C=1, max_iter=100)	10.87	0.82	0.72
24	undersampled	Não utilizado	RidgeClassifier(solver='sparse_cg', tol=0.01)	42.12	0.80	0.71
25	sem alterações	TruncatedSVD(n_components=100)	LinearSVC(C=1)	31.63	0.92	0.70
26	undersampled	TruncatedSVD(n_components=10000)	LinearSVC(C=1)	1671.62	0.81	0.70
27	undersampled	TruncatedSVD(n_components=1000)	LinearSVC(C=1)	83.69	0.80	0.68
28	undersampled	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	10.66	0.79	0.68
29	sem alterações	Parâmetro 'max_features'	LinearSVC(C=1)	21.00	0.90	0.67
30	undersampled	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	7.58	0.79	0.66
31	sem alterações	Não utilizado	SGDClassifier(early_stopping=True, loss='log_l...	209.75	0.90	0.63
32	oversampled	TruncatedSVD(n_components=100)	LinearSVC(C=1)	381.68	0.79	0.62
33	sem alterações	TruncatedSVD(n_components=1000)	SGDClassifier(early_stopping=True, loss='log_l...	314.08	0.90	0.60
34	oversampled	TruncatedSVD(n_components=1000)	SGDClassifier(early_stopping=True, loss='log_l...	317.84	0.90	0.60
35	sem alterações	TruncatedSVD(n_components=1000)	SGDClassifier(early_stopping=True, loss='log_l...	347.83	0.90	0.60
36	sem alterações	TruncatedSVD(n_components=1000)	SGDClassifier(early_stopping=True, loss='log_l...	352.47	0.90	0.60
37	undersampled	StandardScaler(with_mean=False)	LinearSVC(C=1, max_iter=100)	8.96	0.71	0.60
38	undersampled	TruncatedSVD(n_components=100)	LinearSVC(C=1)	14.56	0.71	0.55
39	undersampled	TruncatedSVD(n_components=100)	LinearSVC(C=1)	14.44	0.71	0.54
40	oversampled	Não utilizado	ComplementNB(alpha=0.1)	52.36	0.57	0.48
41	sem alterações	Não utilizado	ComplementNB(alpha=0.1)	10.99	0.84	0.45
42	undersampled	Não utilizado	ComplementNB(alpha=0.1)	9.92	0.53	0.44
43	undersampled	TruncatedSVD()	LinearSVC(C=1, max_iter=100)	8.81	0.63	0.09

Tabela 6 – Métricas obtidas por Cross_Validation para classificação por Tipo.

APÊNDICE B – MÉTRICAS DO MODELO DE CLASSIFICAÇÃO POR FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR TIPO

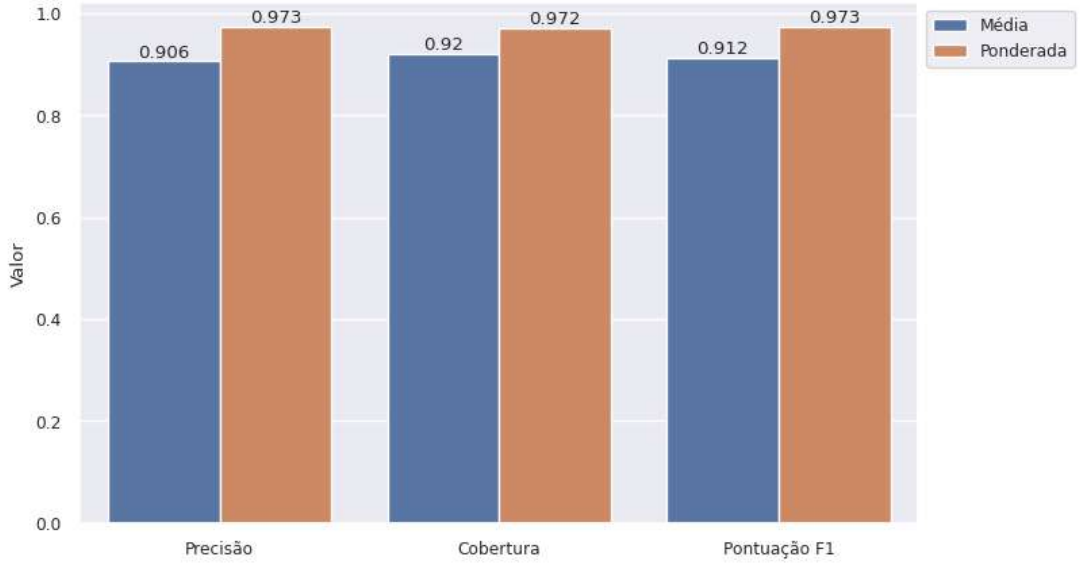

Figura 25 – Métricas médias e ponderadas para a classificação por Tipo com a formação de *Bag of Words*.

Classe	Precisão	Cobertura	Pontuação F1	Quantidade	%
CIENCIA	0.995	0.988	0.992	24265	62.5
PETICAO INICIAL	0.973	0.968	0.971	2663	6.85
RESPOSTA A ACUSACAO	0.994	0.988	0.991	2627	6.76
ALEGACOES FINAIS	0.946	0.918	0.932	853	2.2
CONTRARRAZOES RECURSAIS	0.984	0.972	0.978	833	2.14
CONTESTACAO	0.971	0.975	0.973	829	2.13
APELACAO	0.912	0.922	0.917	798	2.05
PETICAO DE CUMPRIMENTO DE SENTENCA	0.903	0.926	0.914	794	2.04
REPLICA OU IMPUGNACAO A CONTESTACAO	0.981	0.983	0.982	750	1.93
CIENCIA DE AUDIENCIA	0.792	0.888	0.837	659	1.7
RAZOES RECURSAIS	0.882	0.869	0.876	612	1.58
PEDIDO DE REVOGACAO DE PRISAO/ LIBERDADE PROVISORIA	0.98	0.978	0.979	407	1.05
OUTROS	0.666	0.845	0.745	316	0.81
PEDIDO DE PROGRESSAO DE REGIME	0.944	0.938	0.941	308	0.79
PEDIDO DE DECLARACAO DE EXTINCAO DA PUNIBILIDADE	0.931	0.931	0.931	277	0.71
IMPUGNACAO AO CUMPRIMENTO DE SENTENCA	0.867	0.872	0.87	195	0.5
PEDIDO DE REMICAO DE PENA	0.913	0.922	0.918	193	0.5
EMBARGOS DE DECLARACAO	0.979	0.968	0.974	190	0.49
RECURSO	0.848	0.903	0.874	185	0.48
MEMORIAIS	0.681	0.84	0.752	175	0.45
AGRAVO EM EXECUCAO PENAL	0.87	0.876	0.873	153	0.39
PEDIDO DE ATUALIZACAO/RETIFICACAO DE CALCULOS DE PENA	0.829	0.784	0.806	148	0.38
PEDIDO DE HABILITACAO	1.0	0.96	0.979	124	0.32
PEDIDO DE LIVRAMENTO CONDICIONAL	0.929	0.959	0.944	122	0.31
PEDIDO DE TRANSFERENCIA DA EXECUCAO PENAL	0.911	0.919	0.915	111	0.29
EMBARGOS A EXECUCAO	0.838	0.806	0.822	103	0.27
PEDIDO DE INDULTO/COMUTACAO DE PENA	0.961	0.961	0.961	76	0.2
MANIFESTACAO DE CALCULOS	0.875	0.9	0.887	70	0.18

Tabela 7 – Métricas por classe da classificação por Tipo.

APÊNDICE C – MÉTRICAS DO MODELO DE CLASSIFICAÇÃO POR FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR ASSUNTO

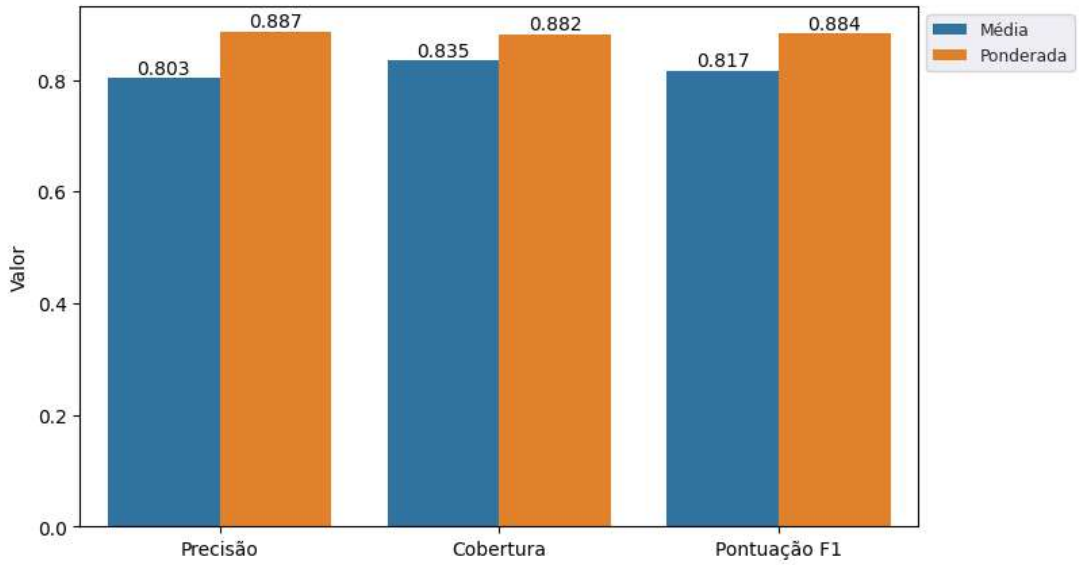

Figura 26 – Métricas médias e ponderadas para a classificação por Assunto com a formação de *Bag of Words*.

Classe	Precisão	Cobertura	Pontuação F1	Quantidade	%
Família e Sucessões	0.936	0.882	0.909	24438	31.47
Criminal	0.953	0.916	0.934	18933	24.38
Execução Penal	0.902	0.935	0.918	15053	19.38
Cível	0.692	0.735	0.713	7128	9.18
Fazenda Pública	0.893	0.931	0.912	7018	9.04
Juizado da Infância e Juventude Cível	0.743	0.846	0.791	2753	3.55
Diversos	0.499	0.6	0.545	2331	3.0

Tabela 8 – Métricas por classe da classificação por Assunto.

APÊNDICE D – VISUALIZAÇÃO DE COMPARAÇÃO ENTRE MÉTRICA PONTUAÇÃO F1 E QUANTIDADE DE DOCUMENTOS PARA A CLASSIFICAÇÃO POR TIPO E POR ASSUNTO DOS MODELOS DE CLASSIFICAÇÃO POR FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO

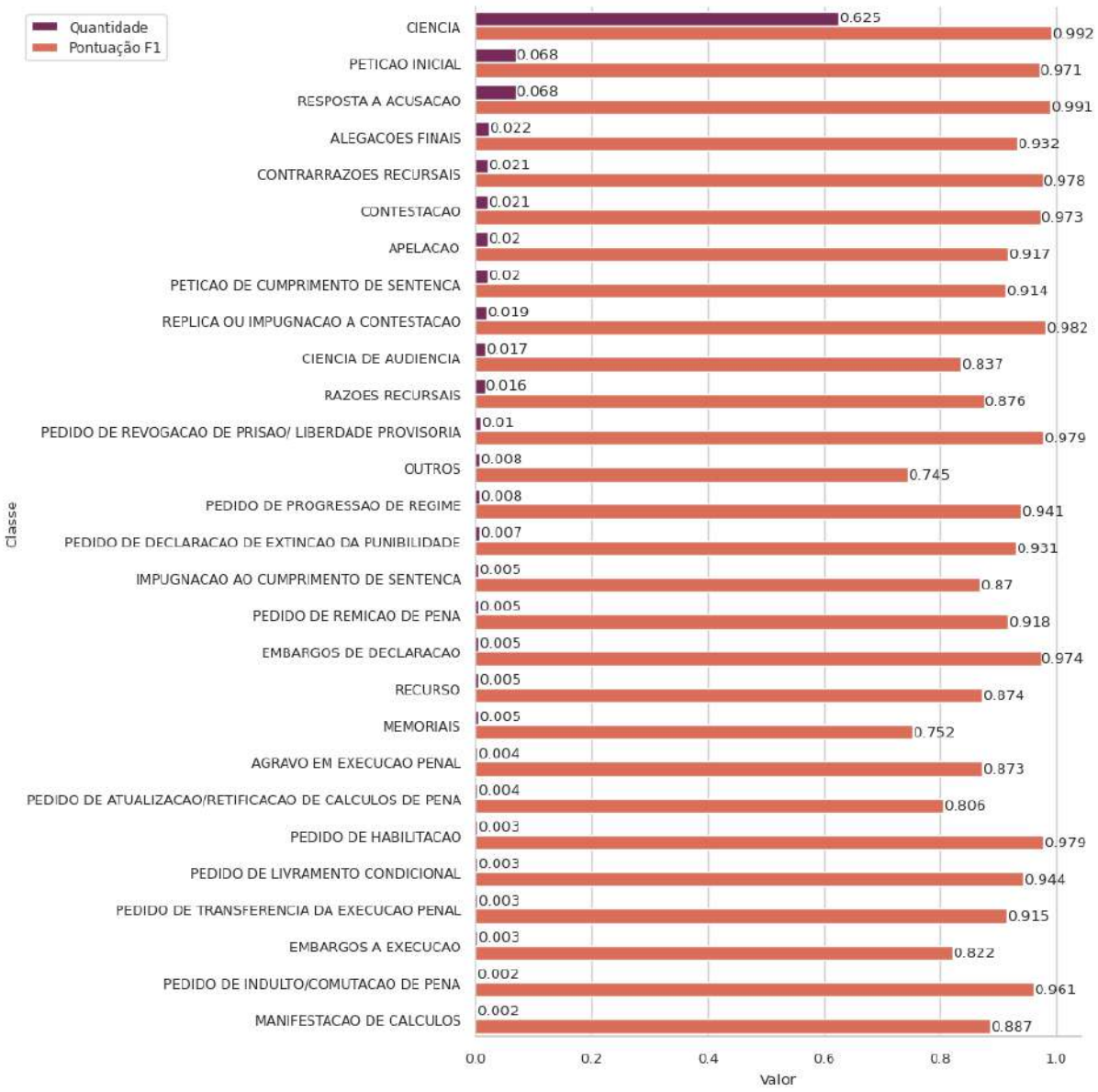

Figura 27 – Visualização de comparação entre métrica Pontuação F1 e quantidade de documentos por classe (representação percentual em número inteiro) para a classificação por Tipo.

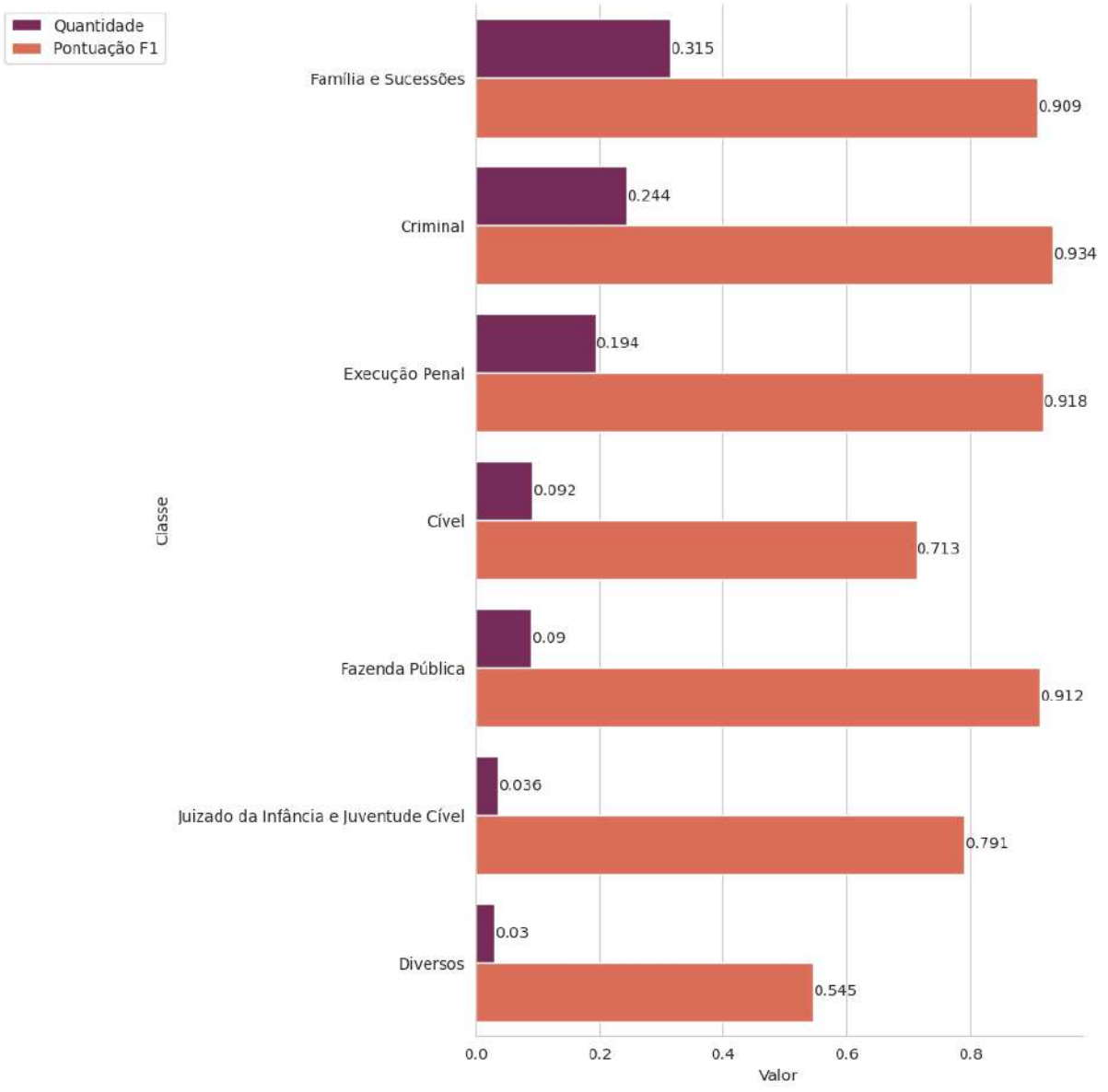

Figura 28 – Visualização de comparação entre métrica Pontuação F1 e quantidade de documentos por classe (representação percentual em número inteiro) para a classificação por Assunto.

APÊNDICE E – MATRIZES DE CONFUSÃO PARA CLASSIFICAÇÕES DO MODELO BASEADO EM FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR TIPO E POR ASSUNTO

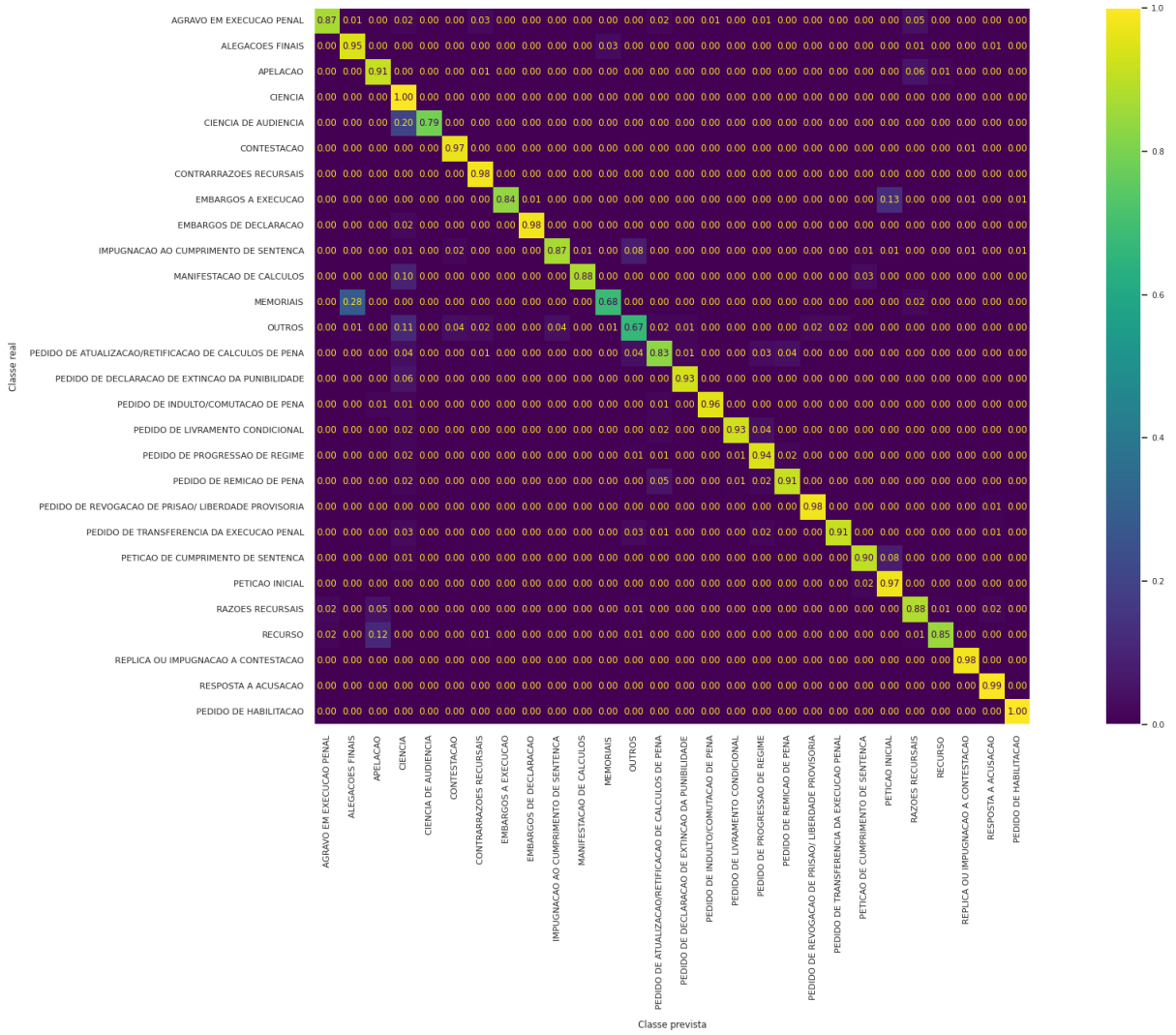

Figura 29 – Matriz de confusão para classificação por Tipo, com valores normalizados sobre o total de amostras para cada classe, isto é, a soma dos valores de cada linha totaliza um inteiro.

Figura 30 – Matriz de confusão para classificação por Assunto, com valores normalizados sobre o total de amostras para cada classe, isto é, a soma dos valores de cada linha totaliza um inteiro.

APÊNDICE F – NUVENS DE PALAVRAS PARA CLASSIFICAÇÕES DO MODELO BASEADO EM FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR TIPO

Figura 31 – Nuvem de palavras para a classe "CIENCIA".

Figura 32 – Nuvem de palavras para a classe "PETICAO INICIAL".

Figura 33 – Nuvem de palavras para a classe "RECURSO".

Figura 34 – Nuvem de palavras para a classe "PEDIDO DE REVOGACAO DE PRISAO/ LIBERDADE PROVISORIA".

APÊNDICE G – NUVENS DE PALAVRAS PARA CLASSIFICAÇÕES DO MODELO BASEADO EM FORMAÇÃO DE *BAG OF WORDS* COM HIPERPARÂMETROS ÓTIMOS SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR ASSUNTO

Figura 35 – Nuvem de palavras para a classe "Cível".

Figura 36 – Nuvem de palavras para a classe "Criminal".

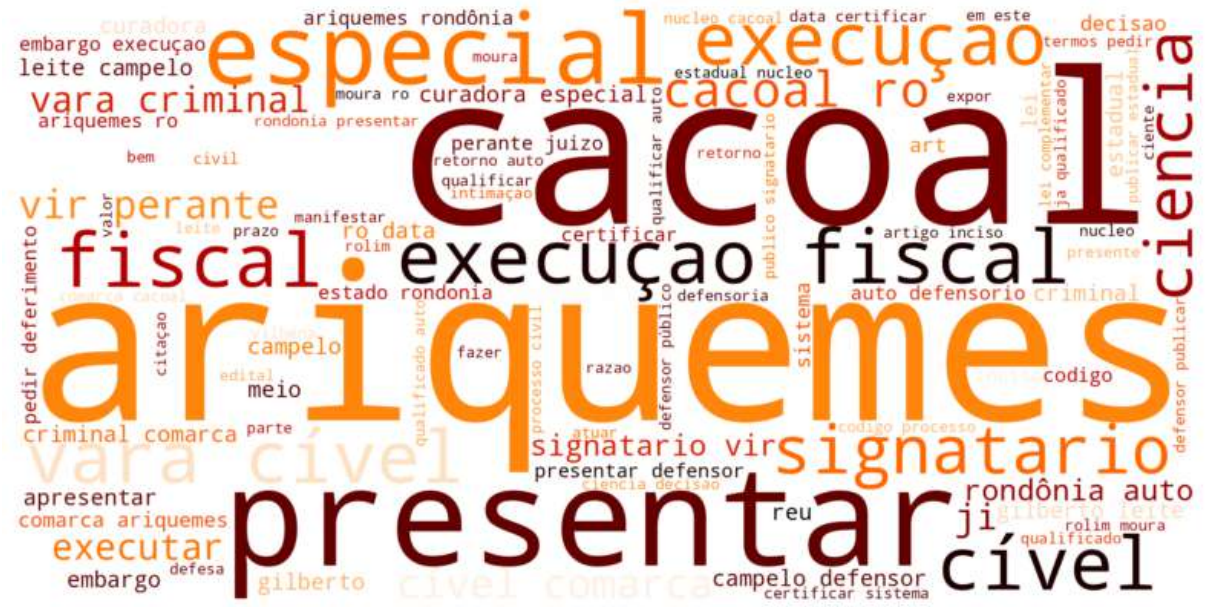

Figura 37 – Nuvem de palavras para a classe "Diversos".

Figura 38 – Nuvem de palavras para a classe "Família e Sucessões".

APÊNDICE H – RELAÇÃO DE MODELOS BERT TESTADOS

ID	MODELO	ESTRATÉGIA PARA TRUNCAR
0	Dominguesm-Legal-Bert-Base-Cased-Ptbr	Usar Primeiros Tokens
1	Dominguesm-Legal-Bert-Base-Cased-Ptbr	Usar Últimos Tokens
2	Dominguesm-Legal-Bert-Base-Cased-Ptbr	Combinar Primeiros E Últimos
3	Felipemaiapolo-Legalnlp-Bert	Usar Primeiros Tokens
4	Felipemaiapolo-Legalnlp-Bert	Usar Últimos Tokens
5	Felipemaiapolo-Legalnlp-Bert	Combinar Primeiros E Últimos
6	Neuralmind-Bert-Base-Portuguese-Cased	Usar Primeiros Tokens
7	Neuralmind-Bert-Base-Portuguese-Cased	Usar Últimos Tokens
8	Neuralmind-Bert-Base-Portuguese-Cased	Combinar Primeiros E Últimos
9	Pierreguillou-Bert-Base-Cased-Pt-Lenerbr	Usar Primeiros Tokens
10	Pierreguillou-Bert-Base-Cased-Pt-Lenerbr	Usar Últimos Tokens
11	Pierreguillou-Bert-Base-Cased-Pt-Lenerbr	Combinar Primeiros E Últimos

Tabela 9 – Relação de modelos BERT para consulta de ID

APÊNDICE I – MELHORES MÉTRICAS DOS MODELOS DE BERT PARA CADA CLASSIFICAÇÃO, COM COMPARAÇÃO PELA PONTUAÇÃO F1 MÉDIA

model_id	Época	Acurácia	Pontuação F1	Precisão	Cobertura
5	4.0	0.967363	0.895120	0.897112	0.897005
7	4.0	0.964942	0.889218	0.890965	0.891430
7	5.0	0.965282	0.888995	0.891155	0.890849
10	4.0	0.965358	0.888674	0.891607	0.889886
8	5.0	0.965282	0.888245	0.890436	0.890208
11	4.0	0.964904	0.887009	0.889472	0.888520
11	5.0	0.964979	0.886660	0.889101	0.888249
2	5.0	0.964412	0.886645	0.888944	0.888415
5	5.0	0.964336	0.886300	0.888669	0.888166
4	5.0	0.963693	0.885107	0.887625	0.886842
1	5.0	0.964185	0.884588	0.886212	0.887034
8	4.0	0.963844	0.884358	0.886382	0.886508
10	5.0	0.964185	0.883284	0.885767	0.885063
1	4.0	0.963353	0.882302	0.885278	0.883682
4	4.0	0.962293	0.881068	0.883184	0.883066
8	3.0	0.962672	0.879709	0.881792	0.882007
0	5.0	0.962445	0.878267	0.882048	0.879312
1	3.0	0.961953	0.877714	0.881084	0.878656
2	4.0	0.961045	0.876360	0.879545	0.877503
10	3.0	0.961007	0.876348	0.879345	0.877641
11	3.0	0.961272	0.875419	0.877829	0.877396
5	3.0	0.961196	0.875271	0.877504	0.877617
0	4.0	0.961423	0.875218	0.878509	0.876726
6	4.0	0.961234	0.874576	0.876891	0.877030
6	3.0	0.960704	0.874511	0.876748	0.877187

Tabela 10 – Vinte e cinco melhores modelos da classificação por Tipo. Avaliação segundo a métrica Pontuação F1 média.

model_id	Época	Acurácia	Pontuação F1	Precisão	Cobertura
2	5.0	0.897470	0.820421	0.833178	0.832694
1	5.0	0.890124	0.810548	0.822607	0.824678
2	4.0	0.888894	0.807876	0.819582	0.822983
1	4.0	0.888004	0.805942	0.819189	0.819240
5	4.0	0.884955	0.802685	0.816868	0.815601
4	4.0	0.886432	0.802549	0.815280	0.816969
4	5.0	0.884539	0.802241	0.815813	0.816086
5	5.0	0.884274	0.802016	0.815639	0.815465
7	5.0	0.883763	0.799805	0.813537	0.813679
0	5.0	0.882608	0.797303	0.810440	0.812230
11	5.0	0.879730	0.795346	0.808154	0.811072
5	3.0	0.881282	0.794458	0.807096	0.809238
4	3.0	0.882191	0.794302	0.808564	0.808485
2	3.0	0.882873	0.794282	0.808411	0.807758
7	4.0	0.880866	0.793730	0.808148	0.807557
1	3.0	0.878404	0.792566	0.806778	0.806395
11	4.0	0.880809	0.792218	0.807236	0.805468
8	4.0	0.880051	0.791734	0.807618	0.804692
0	4.0	0.880241	0.791493	0.805248	0.806213
8	5.0	0.877590	0.791360	0.803674	0.807720
10	3.0	0.877571	0.790944	0.803425	0.806977
8	3.0	0.877022	0.787575	0.801225	0.802638
7	3.0	0.877287	0.787512	0.802729	0.801734
10	2.0	0.873235	0.784862	0.797870	0.801233
1	2.0	0.875886	0.784240	0.798502	0.798434

Tabela 11 – Vinte e cinco melhores modelos da classificação por Assunto. Avaliação segundo a métrica Pontuação F1 média

APÊNDICE J – DETALHAMENTO DAS MÉTRICAS DO MODELO BERT DE MELHOR DESEMPENHO SEGUNDO AVALIAÇÃO DA PONTUAÇÃO F1 MÉDIA SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR TIPO

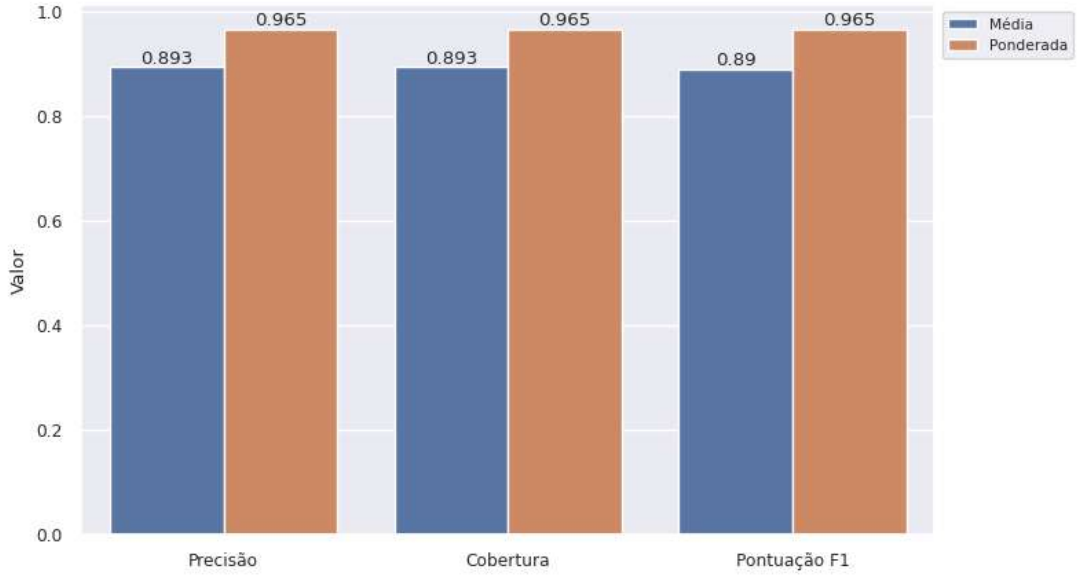

Figura 39 – Métricas médias e ponderadas para a classificação por Tipo

Classe	Precisão	Cobertura	Pontuação F1	Quantidade	%
CIENCIA	0.988	0.992	0.99	14471	62.07
PETICAO INICIAL	0.974	0.958	0.966	1590	6.82
RESPOSTA A ACUSACAO	0.984	0.996	0.99	1567	6.72
CONTESTACAO	0.966	0.954	0.96	499	2.14
ALEGACOES FINAIS	0.887	0.93	0.908	497	2.13
CONTRARRAZOES RECURSAIS	0.957	0.984	0.97	494	2.12
PETICAO DE CUMPRIMENTO DE SENTENCA	0.909	0.898	0.903	488	2.09
APELACAO	0.857	0.88	0.869	484	2.08
REPLICA OU IMPUGNACAO A CONTESTACAO	0.98	0.969	0.974	450	1.93
CIENCIA DE AUDIENCIA	0.856	0.767	0.81	443	1.9
RAZOES RECURSAIS	0.816	0.859	0.837	362	1.55
PEDIDO DE REVOGACAO DE PRISAO/ LIBERDADE PROVISORIA	0.971	0.963	0.967	244	1.05
OUTROS	0.824	0.643	0.723	241	1.03
PEDIDO DE PROGRESSAO DE REGIME	0.972	0.946	0.959	184	0.79
PEDIDO DE DECLARACAO DE EXTINCAO DA PUNIBILIDADE	0.888	0.958	0.922	166	0.71
MEMORIAIS	0.75	0.646	0.694	130	0.56
RECURSO	0.817	0.644	0.72	118	0.51
PEDIDO DE REMICAO DE PENA	0.873	0.88	0.877	117	0.5
IMPUGNACAO AO CUMPRIMENTO DE SENTENCA	0.832	0.846	0.839	117	0.5
EMBARGOS DE DECLARACAO	0.965	0.973	0.969	113	0.48
AGRAVO EM EXECUCAO PENAL	0.948	0.785	0.859	93	0.4
PEDIDO DE ATUALIZACAO/RETIFICACAO DE CALCULOS DE PENA	0.787	0.881	0.831	84	0.36
PEDIDO DE LIVRAMENTO CONDICIONAL	0.892	0.974	0.931	76	0.33
PEDIDO DE HABILITACAO	0.946	0.986	0.966	71	0.3
PEDIDO DE TRANSFERENCIA DA EXECUCAO PENAL	0.952	0.896	0.923	67	0.29

Tabela 12 – Métricas por classe da classificação por Tipo.

APÊNDICE K – DETALHAMENTO DAS MÉTRICAS DO MODELO BERT DE MELHOR DESEMPENHO SEGUNDO AVALIAÇÃO DA PONTUAÇÃO F1 MÉDIA SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO PARA AS CLASSIFICAÇÕES POR ASSUNTO

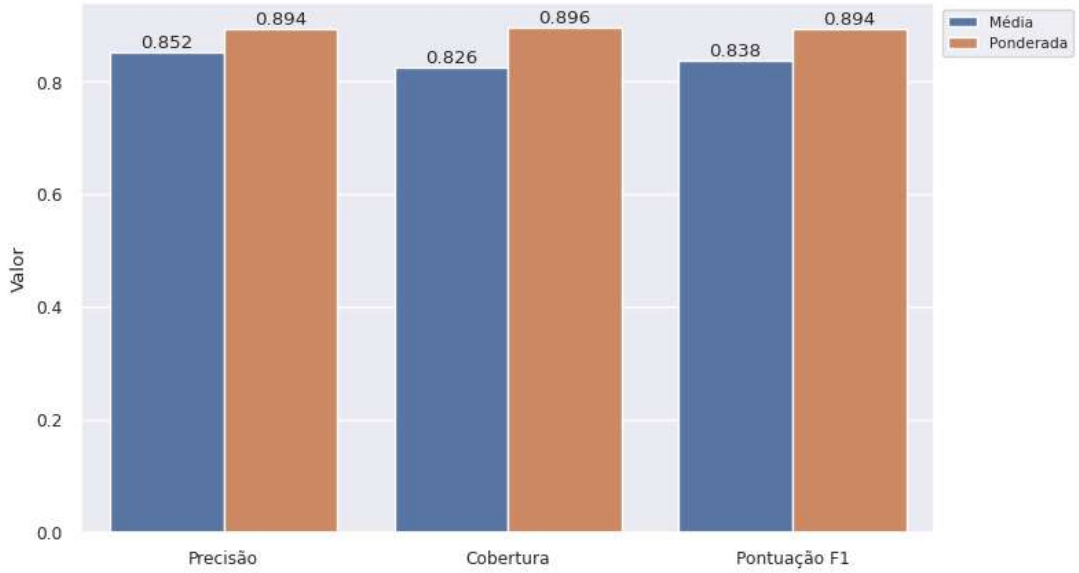

Figura 40 – Métricas médias e ponderadas para a classificação por Assunto

Classe	Precisão	Cobertura	Pontuação F1	Quantidade	%
Família e Sucessões	0.895	0.945	0.919	13819	29.66
Criminal	0.931	0.962	0.946	10919	23.43
Execução Penal	0.947	0.908	0.927	9363	20.1
Cível	0.784	0.722	0.752	4541	9.75
Fazenda Pública	0.915	0.915	0.915	4388	9.42
Juizado da Infância e Juventude Cível	0.818	0.767	0.792	1881	4.04
Diversos	0.678	0.561	0.614	1682	3.61

Tabela 13 – Métricas por classe da classificação por Assunto.

APÊNDICE L – VISUALIZAÇÃO DE COMPARAÇÃO ENTRE MÉTRICA PONTUAÇÃO F1 E QUANTIDADE DE DOCUMENTOS PARA A CLASSIFICAÇÃO POR TIPO E POR ASSUNTO DO MODELO BERT DE MELHOR DESEMPENHO SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO

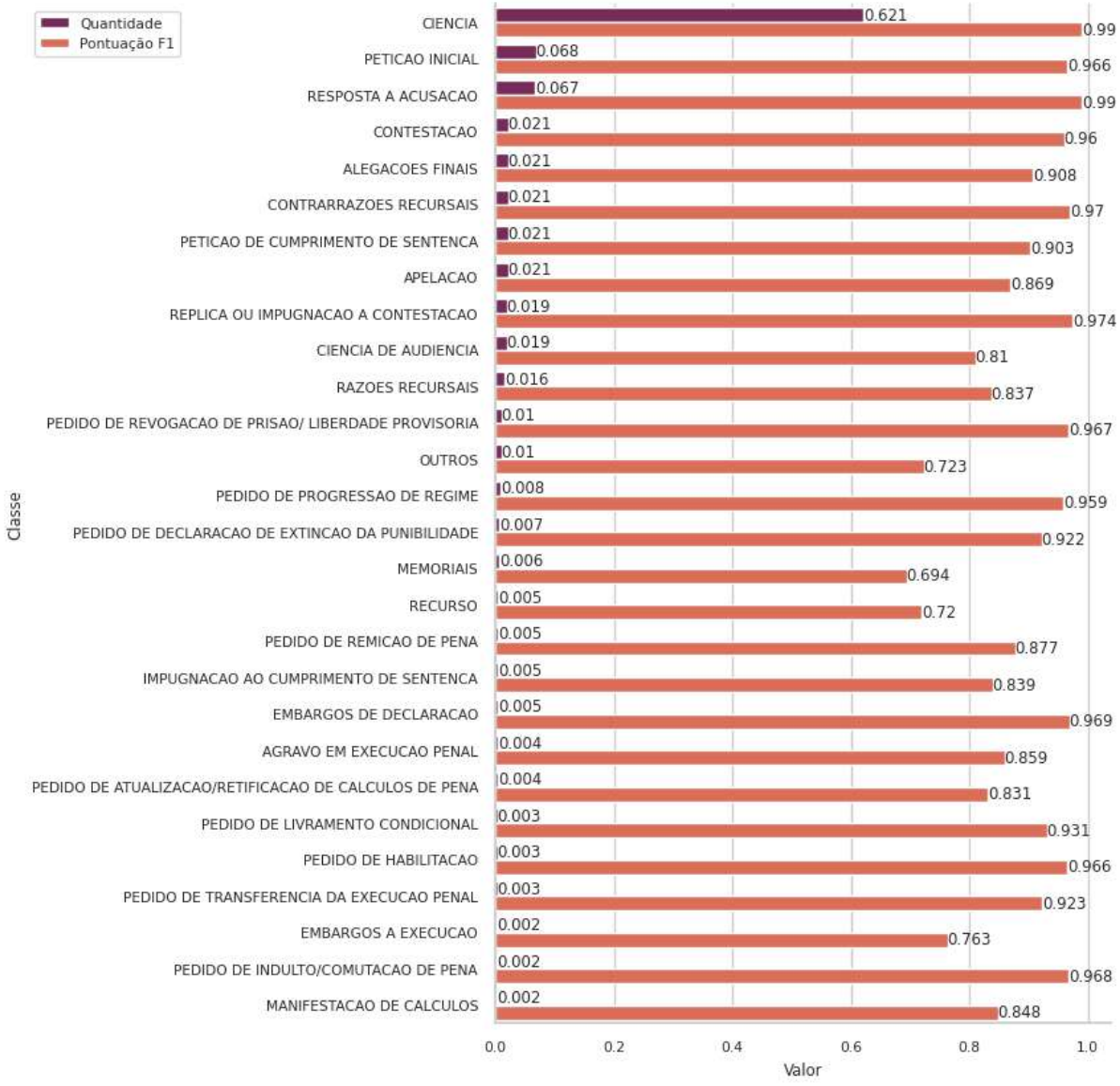

Figura 41 – Visualização de comparação entre métrica Pontuação F1 e quantidade de documentos por classe (representação percentual em número inteiro) para a classificação por Tipo.

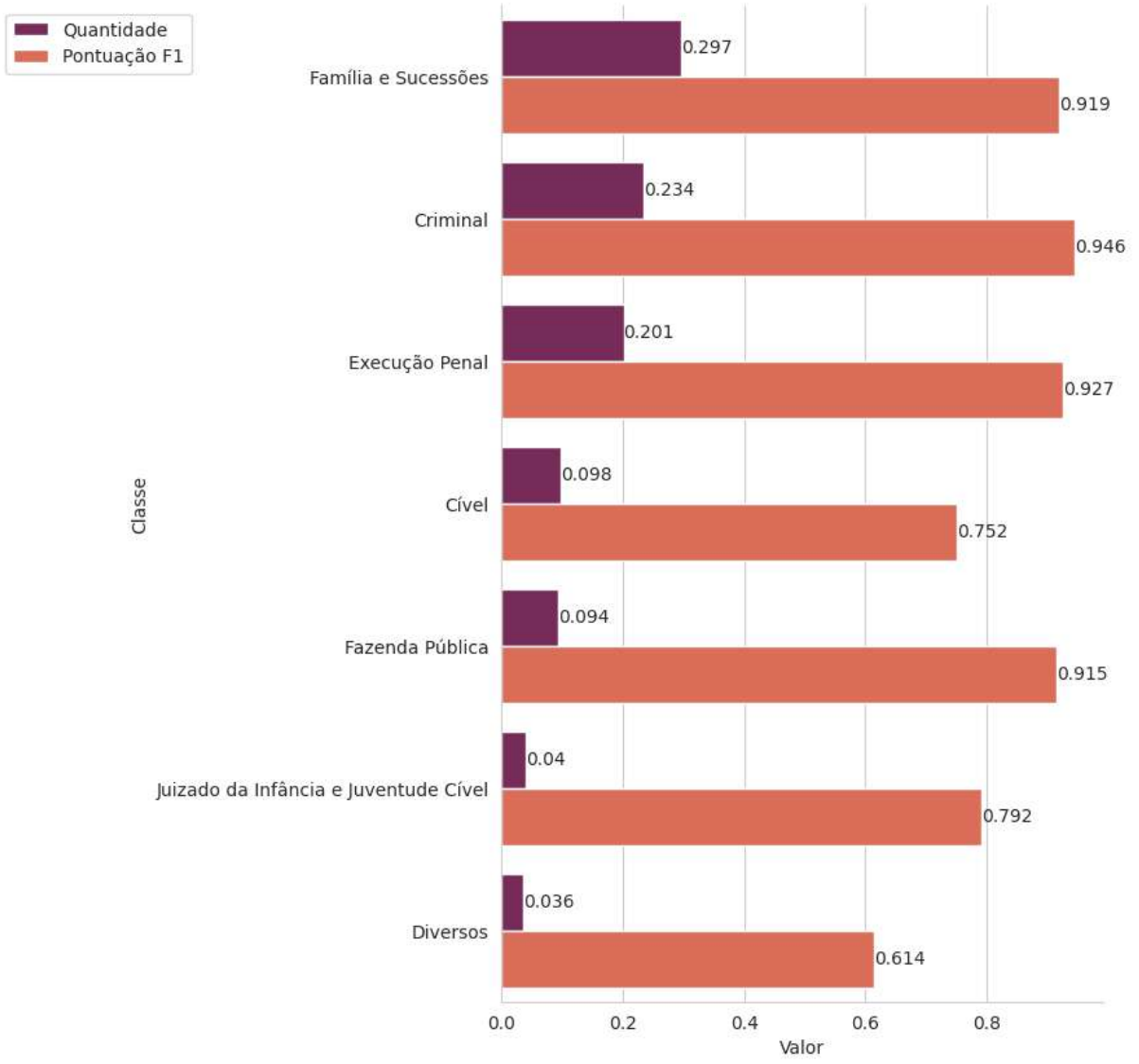

Figura 42 – Visualização de comparação entre métrica Pontuação F1 e quantidade de documentos por classe (representação percentual em número inteiro) para a classificação por Assunto.

APÊNDICE M – MATRIZES DE CONFUSÃO PARA AS CLASSIFICAÇÕES POR TIPO E POR ASSUNTO PELO MODELO BERT DE MELHOR DESEMPENHO SEGUNDO AVALIAÇÃO DA PONTUAÇÃO F1 MÉDIA SOBRE O CONJUNTO DE DADOS DE VALIDAÇÃO

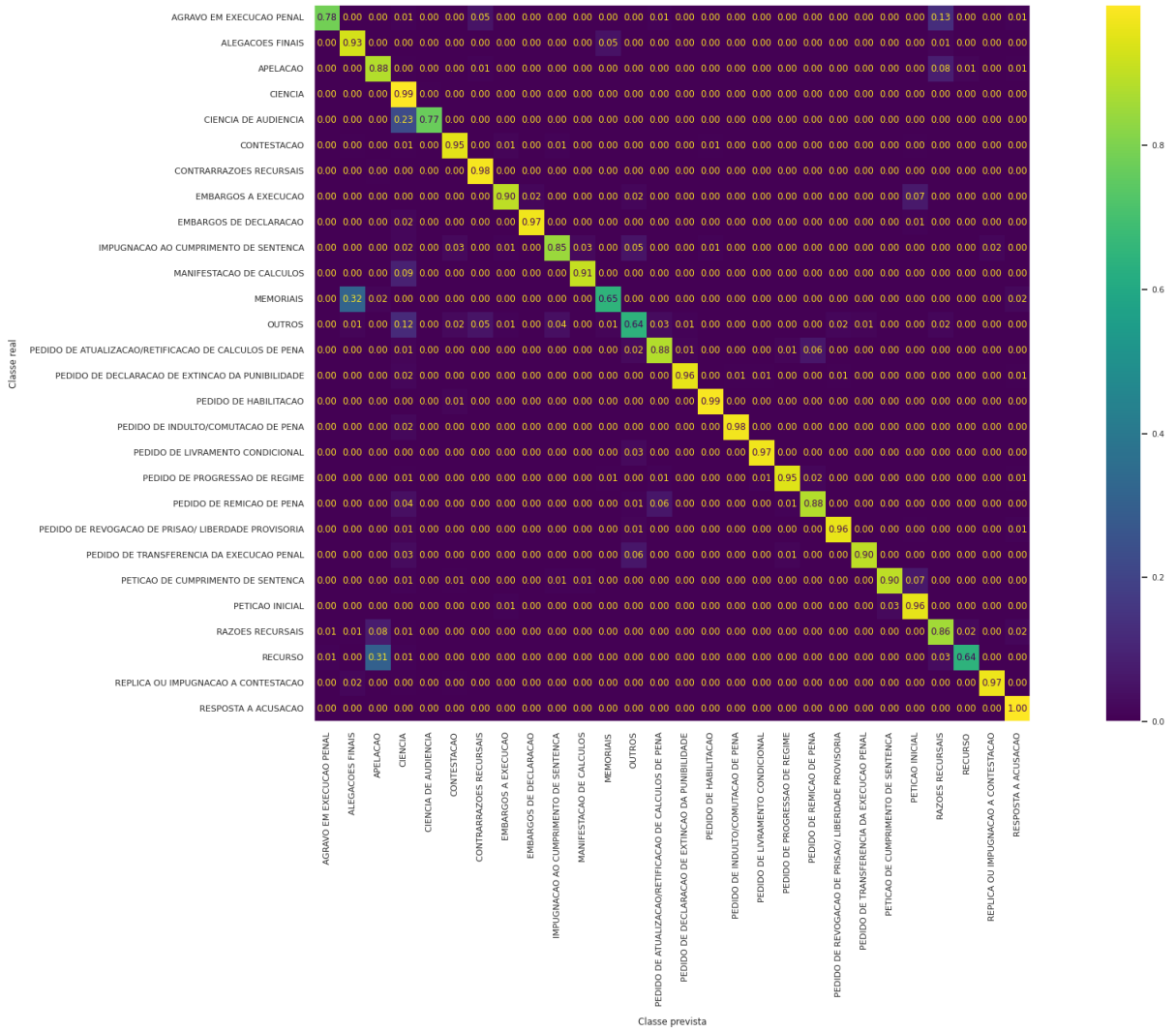

Figura 43 – Matriz de confusão para classificação por Tipo, com valores normalizados sobre o total de amostras para cada classe, isto é, a soma dos valores de cada linha totaliza um inteiro.

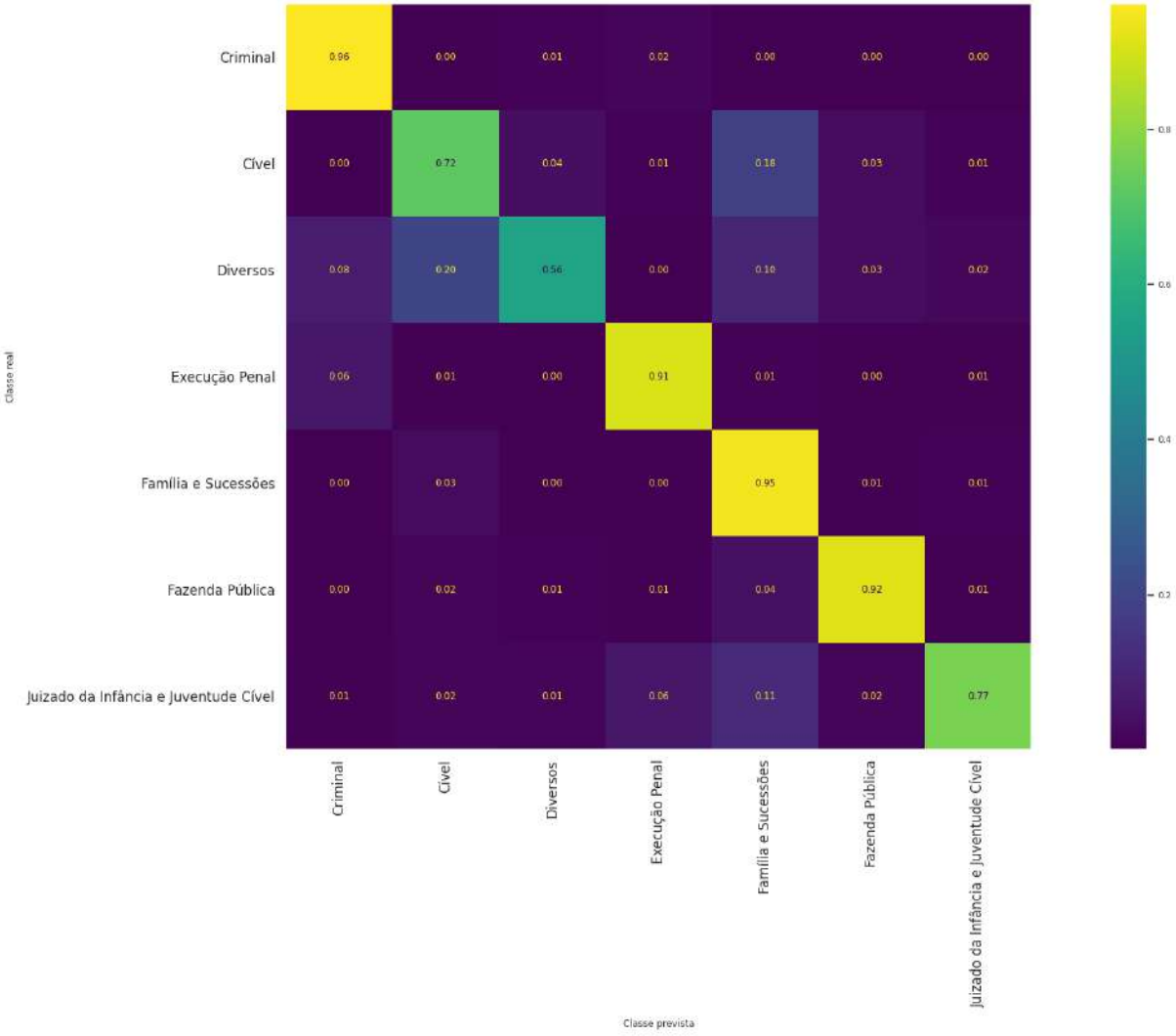

Figura 44 – Matriz de confusão para classificação por Assunto, com valores normalizados sobre o total de amostras para cada classe, isto é, a soma dos valores de cada linha totaliza um inteiro.

ANEXOS

ANEXO A – EXEMPLOS DE DOCUMENTOS DO CONJUNTO DE DADOS

Tipo: PETICAO INICIAL
Assunto: Família e Sucessões

EXCELENTÍSSIMO(A) SENHOR(A) DOUTOR(A) JUIZ(A) DA __ VARA CÍVEL DA COMARCA DE JI-PARANÁ - RONDÔNIA
Guarda, visitas e alimentos
, brasileiro(a), , residente e domiciliado(a) na
, fone , e
, residente e domiciliado(a) na Principal - Residencial
, fone None, ambos representados pela Defensoria Publica do Estado de Rondonia, cujo Defensor Publico abaixo subscreve, vem, respeitosamente, na forma do artigo 731 do CPC, requerer a HOMOLOGAÇÃO JUDICIAL DE ACORDO EXTRAJUDICIAL, conforme documento em anexo por ambos assinados.
De antemao, as partes pleiteiam a concessao do beneficio da gratuidade da justiça, na forma do artigo 98 do CPC, eis que nao dispoem de recursos para arcar com o pagamento das custas e despesas processuais e nem com o pagamento de honorarios advocaticios, tanto que estao sendo assistidos pela Defensoria Publica de Rondonia neste pedido de homologaçao judicial de acordo extrajudicial.
Os termos do acordo estao detalhados no documento em anexo, o qual foi elaborado em conjunto com as partes em sessao de conciliaçao realizada na Defensoria Publica do Estado de Rondonia e assinado por todos os envolvidos, o qual, por disposiçao legal, necessita de homologaçao judicial apos a oitiva do Ministerio Publico, ja que envolve interesse de menores, nos moldes do artigo 178, II, do CPC.
Ante o exposto, requer:
1\. A concessao dos beneficios da gratuidade da justiça, eis que nao dispoem de recursos para arcar com o pagamento das custas e despesas processuais e nem com os honorarios advocaticios;
2\. A homologaçao do acordo extrajudicial acostado em anexo nos moldes ali descritos, com a previa oitiva do Ministerio Publico, eis que envolve interesse de menor incapaz, expedindo os oficios necessarios para a averbaçao cartoraria.
Da-se a causa o valor de
Termos em que,
Pede deferimento.
De Porto Velho para Ji-Parana, .

Defensor Publico
Coordenador do NUREC

Tipo: CIENCIA
Assunto: Criminal

AO JUÍZO DA 3ª VARA CRIMINAL DA COMARCA DE JI-PARANÁ/RO
Autos nº
A DEFENSORIA PÚBLICA DO ESTADO DE RONDÔNIA, atraves da Defensora Publica signataria, no prazo em dobro que lhe e conferido pelo artigo 128, inciso I da Lei Complementar Federal nº 80/94, bem como pelo artigo 69, inciso XI da Lei Complementar Estadual de Rondonia nº 117/94, vem, perante este Juizo, tomar CIÊNCIA da decisao acostada ao ID que homologou Acordo de Nao Persecuçao Penal celebrado por Lariessa Mariane Pereira da Luz.
Ji-Parana/RO, 11 de abril de 2022.

Defensora Publica
rtec

Tipo: PEDIDO DE PROGRESSAO DE REGIME
Assunto: Execução Penal

EXCELENTÍSSIMO SENHOR JUIZ DE DIREITO DA VARA DE EXECUÇÕES E CONTRAVENÇÕES PENAIS DA COMARCA DE PORTO VELHO.
Execuçao:
Custodiado:
A Defensoria Publica do Estado de Rondonia, no uso de suas atribuiçoes constitucionais, vem se manifestar seguintes termos.
Conforme calculo atualizado na aba informaçoes adicionais do SEEU, verifica-se que o sentenciado cumpriu o requisito temporal necessario para obter o direito a progressao de regime aberto em .
Quanto ao requisito subjetivo, verificou-se que a certidao de conduta carceraria, juntada aos autos no item 147.1, atribuiu comportamento neutro ao assistido e que nao pode ser interpretada em seu desfavor.
Ante o exposto, a Defesa requer seja concedida em favor do reeducando a progressao de regime para o cumprimento da pena no regime aberto, a partir da citada data.
Porto Velho/RO, datado e assinado digitalmente.

Tipo: CONTRARRAZOES RECURSAIS
Assunto: Execução Penal

EXCELENTÍSSIMO SENHOR JUIZ DE DIREITO DA VARA DE EXECUÇÕES E CONTRAVENÇÕES PENAIS DA COMARCA DE PORTO VELHO.
Execuçao:
Assistido:
, devidamente qualificado nos autos em epigrafe, regularmente assistido pela DEFENSORIA PÚBLICA DO ESTADO DE RONDÔNIA, com fulcro no artigo 134 da Constituiçao Federal e artigo 3º, §1º da LC/RO 117/94, vem, respeitosamente, a presença de Vossa Excelencia, apresentar, em anexo, CONTRARRAZÕES AO AGRAVO EM EXECUÇÃO interposto pelo Ministerio Publico contra a decisao proferida no item 163.1 dos autos supracitados.
Pugna pela manutençao da decisao agravada e posterior remessa ao Egregio Tribunal de Justiça do Estado de Rondonia.
Termos em que pede deferimento.
Porto Velho/RO, datado e assinado digitalmente.

Defensor Publico

Tipo: CIENCIA
Assunto: Família e Sucessões

AO JUÍZO DA VARA CÍVEL DA COMARCA DE VILHENA-RO:
Autos nº.
A DEFENSORIA PÚBLICA DE RONDÔNIA, atuando em favor da autora, ja qualificada nos autos em epigrafe, atraves da DEFENSORA PÚBLICA infra-assinada, vem a presença de Vossa Excelencia, informar estar ciente da sentença.
Diante disso, pugna-se pela expediçao do formal de partilha.
Pede deferimento.
Vilhena, data certificada.

Defensora Publica

Tipo: RECURSO
Assunto: Criminal

EXCELENTÍSSIMO SENHOR DOUTOR JUIZ DE DIREITO DA VARA CRIMINAL DE ALTA FLORESTA D'OESTE/RO

Autos n.º Autos nº
Recorrente:
Recorrido: Ministerio Publico do Estado de Rondonia
INTERPOSIÇÃO DE RECURSO DE APELAÇÃO
, ja devidamente qualificados nos autos do processo em epigrafe, por intermedio da Defensora Publica abaixo firmada, vem respeitosamente perante Vossa Excelencia, interpor recurso de APELAÇÃO contra sentença condenatoria prolatada nos autos.
Desta forma, requer seja conhecido o presente recurso, vez que atende aos pressupostos processuais de admissibilidade, sobretudo por ter a Defensoria Publica prazo em dobro.
Esclarece que o presente recurso e tempestivo, pois como se pode observar o prazo dos processos fisicos estavam suspensos em razao da pandemia do Covid e esta foi a primeira vez que o feito foi remetido pessoalmente a Defensoria Publica.
Alem disso, o apelante foi preso e ate o momento nao foi intimado pessoalmente da sentença, o que de igual modo tem o condao de reabrir o prazo recursal.
Deste modo, apos o recebimento e certificada a tempestividade, requer-se tambem, que seja oportunizada a Defensoria Publica a apresentaçao das Razoes Recursais.
Termos em que pede e espera deferimento.
Alta Floresta D'Oeste, .

Defensora Publica

ANEXO B – EXEMPLOS DE CLASSIFICAÇÕES PREDITA E REAL DIVERGENTES

```
================================================================================================================
ID:  78
exemplo_id:  68703
================================================================================================================
AO JUÍZO DA VARA CRIMINAL E EXECUÇÕES DE PENA DA COMARCA DE
AUTOS Nº.
Reeducando(a):
A DEFENSORIA PÚBLICA DO ESTADO DE RONDONIA, por intermedio do Defensor Publico
signatario, no exercicio das atribuiçoes previstas no art. 134, da
Constituiçao Federal e art. 61, III, art. 81-A da Lei 7.210/84, com redaçao
dada pela Lei 12.313/2010, em favor de:
                       , qualificado nos autos de execuçao em epigrafe, vem a
presença de Vossa Excelencia se manifestar acerca da falta disciplinar, por
ter empreendido fugo do trabalho externo.
Em audiencia de justificativa, a reeducanda          confessou ter cometido a
fuga, aduzindo que apos ser expulsa da residencia que estava morando de
aluguel, ficou apavorada, pois seus filhos que a sustentavam estavam presos e
nao teria como mais arcar com as despesas basicas para sobrevivencia.
Pois bem.
Pelo exposto, a Defensoria Publica pugna, pelo acolhimento da justificativa da
Apenada, e, via de consequencia, o nao reconhecimento de eventual falta grave.
Por fim, caso entendimento seja diverso, em atençao aos principios da
razoabilidade e proporcionalidade, requer que seja aplicada uma penalidade
disciplinar branda, a teor do que dispoe o art. 53, I, II, e III da LEP e que
eventual perda dos dias remidos NÃO se de na fraçao maxima de 1/3.
Pede deferimento.
                , data constante na assinatura digital.

Defensor Publico

================================================================================================================
Real : ALEGACOES FINAIS
BoW : ALEGACOES FINAIS
BERT : OUTROS
================================================================================================================
================================================================================================================
ID:  266
exemplo_id:  26776
================================================================================================================
MM. Juiza,
Em que pese o acusado ter afirmado em Juizo que ingeriu bebida alcoolica,
esclareceu que isso se deu no dia anterior e tinha acabado de assumir a
direçao do veiculo com o objetivo de retornar para sua casa, que fica em

Dessa forma, considerando que o artigo 306 do Codigo de Transito Brasileiro
exige, alem da ingestao de bebida, a perda da capacidade psicomotora, o que
nao restou devidamente comprovado, requer a absolviçao, nos termos do artigo
386, inciso VII, do Codigo de Processo Penal.
Pede deferimento.


================================================================================================================
Real : MEMORIAIS
BoW : PEDIDO DE REVOGACAO DE PRISAO/ LIBERDADE PROVISORIA
BERT : MEMORIAIS
================================================================================================================
```

```
================================================================================================
ID:  386
exemplo_id:  267882
================================================================================================
MM JUIZO DE DIREITO DA VARA CRIMINAL DA COMARCA DE
Autos nº
Acusado:
A DEFENSORIA PÚBLICA DO ESTADO DE RONDÔNIA, pela Defensora Publica signataria,
vem, atestar ciencia da redesignaçao de audiencia de instruçao e julgamento
              para o dia                          , as 10h00min.
                     /RO, data do protocolo.

Defensora Publica

================================================================================================
Real : CIENCIA
BoW : CIENCIA DE AUDIENCIA
BERT : CIENCIA DE AUDIENCIA
================================================================================================
 ==============================================================================================
 ID:  276
 exemplo_id:  13321
 ==============================================================================================
 AO JUIZADO ESPECIAL CRIMINAL DE
 Autos n.
 Promovido(a)(s):
 A DEFENSORIA PÚBLICA DO ESTADO DE RONDÔNIA, presentada pela Defensora Publica
 subscritora, vem a presença de Vossa Excelencia manifestar-se nos termos
 seguintes.
 Sobreio noticia de cumprimento integral do acordo                .
 Desta forma, a Defensoria Publica requer a declaraçao de extinçao da
 punibilidade do(a) promovido(a).
                  , data assinalada pelo sistema.

 Defensora Publica

 ==============================================================================================
 Real : CIENCIA
 BoW : CIENCIA
 BERT : PEDIDO DE DECLARACAO DE EXTINCAO DA PUNIBILIDADE
 ==============================================================================================
```