Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Aplicação do BERT para Classificação de Acórdãos do STF por Ramos do Direito

Kaique Kazuyoshi Komata
Yurick Yussuke Honda

Monografia Final

mac 0499 — Trabalho de Formatura Supervisionado

Supervisora: Profª. Renata Wassermann
Cossupervisor: Rafael Brito
Cossupervisor: Jackson Souza

São Paulo
22 de Dezembro de 2021

Agradecimentos

Kaique Kazuyoshi Komata

A meus amigos e família pelo apoio e incentivo durante todo o projeto e o curso.

Ao meu companheiro Yurick por todo empenho e aplicação exercidos no projeto que possibilitaram que ele fosse concluído com sucesso. Obrigado pela amizade e paciência não só neste trabalho, mas durante toda a graduação.

À minha noiva e futura esposa Elizabeth por todo o suporte e carinho que me motivam a seguir em frente e querer sempre melhorar. Muito obrigado por me apoiar e ajudar em todas as minhas escolhas e estar ao meu lado em todas as ocasiões.

À Professora Renata Wasserman e aos supervisores Rafael Brito e Jackson Souza pelas orientações e auxílio prestados durante o projeto.

Aos Professores Juliano Maranhão e Marcelo Finger pela colaboração essencial para a construção desse projeto.

Não é na ciência que está a felicidade, mas na aquisição da ciência.
— Edgar Allan Poe

Yurick Yussuke Honda

A Deus por desenhar cada detalhe da minha vida, inclusive a concretização desse projeto.

À minha família por navegar ao meu lado nesse vasto oceano que é a vida, visitando meus sonhos junto a mim. Agradeço por me acompanharem nessa aventura de trabalhar neste projeto. Em certas ocasiões da vida, a única ação necessária é acreditar, e o apoio da minha família nunca me deixou perder a fé. Sou eternamente grato a vocês.

Ao Kaique pela dedicação colocada no projeto. Sem seu trabalho sólido e constante, essa iniciativa não teria sido concluída. Obrigado por toda a amizade e parceria.

Aos supervisores Renata Wasserman, Rafael Brito e Jackson Souza que colocaram um esforço enorme para que o projeto fosse concluído.

Ao Marcelo Finger, Juliano Maranhão e alunos da Faculdade de Direito da Universidade de São Paulo por colaborarem com a execução desse projeto.

Tudo o que é bom dura o tempo necessário para ser inesquecível.

— Fernando Pessoa

Resumo

Kaique Kazuyoshi Komata Yurick Yussuke Honda. **Aplicação do BERT para Classificação de Acórdãos do STF por Ramos do Direito**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Os acórdãos documentam decisões judiciais que pertencem a um certo ramo do Direito, ou seja, uma determinada área que compreende os conceitos predominantes ao longo da decisão. Para definir a qual ramo um acórdão pertence, é preciso que se tenha um especialista que possa avaliar o conteúdo do documento. Sendo assim, em uma tentativa de tornar tal ação mais acessível, este projeto propôs o uso da arquitetura BERT abstraído ao BERTimbau, ou seja, a rede neural aplicada à Língua Portuguesa, para a construção de um classificador com a habilidade de indicar o ramo do Direito ao qual um acórdão pertence.

Palavras-chave: Acórdão. Ramo do Direito. Classificação. Rede Neural. Processamento de Linguagem Natural. BERT. BERTimbau.

Abstract

Kaique Kazuyoshi Komata Yurick Yussuke Honda. **BERT Application for Classifying Supreme Court Judgements by Field of Law**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

The judgments represent court decisions that belong to a specific area of law. In other words, it is a particular area that comprehends the main concepts highlighted throughout the decision. Usually, an expert is needed to evaluate the document with the decision to classify a judgment into an area of law. Therefore, this project proposes using the BERT architecture, more specifically the BERTimbau, which is the application of BERT for Brazilian Portuguese, to build a classifier that maps the area of law that best represents the judgment.

Keywords: Judgement. Field of Law. Classification. Neural Network. Natural Language Processing. BERT. BERTimbau.

Sumário

Capítulo 1

Introdução

Gordon E. Moore definiu, em 1965, que a velocidade de processamento dos computadores seria elevada ao dobro a cada dois anos. A chamada Lei de Moore, mesmo após mais de 50 anos, vem provando sua consistência na prática, com todos os avanços na computação e eletrônicos (Tardi, 2021).

Com esse rápido progresso no desenvolvimento de tecnologias que possibilitem a computação cada vez mais acelerada, um novo benefício é adquirido: o processamento de grande quantidade de dados. Por sua vez, os dados em abundância podem direcionar à identificação de padrões em situações cotidianas, resultando na solução de diversos problemas.

No Direito, a escrita de documentos é essencial, incluindo a definição dos acórdãos, quando decisões são realizadas. A depender da conjuntura discutida pelo acórdão, ele pode ser classificado em um certo ramo do Direito. Por exemplo, acórdãos que tratam de decisões envolvendo o voto eleitoral são mais prováveis de pertencerem ao ramo do Direito eleitoral. Em contrapartida, acórdãos que descrevem decisões envolvendo deveres de militares no Brasil são classificados como parte do Direito militar. As regras de classificação não são claras e geralmente dependem de um profissional ou alguma pessoa que tenha expertise na legislação do país.

A proposta desse trabalho é reduzir o esforço necessário para classificar um acórdão no seu devido ramo do Direito. Uma rede neural devidamente treinada pode evitar a leitura do acórdão juntamente da necessidade de uma pessoa com conhecimento para classificá-lo.

O texto foi repartido em seis capítulos. No primeiro capítulo, o objetivo do projeto foi apresentado juntamente de sua motivação. No segundo capítulo, alguns conceitos preliminares à construção do trabalho são explicados. No terceiro capítulo, aborda-se tópicos técnicos que são importantes para o trabalho. O quarto capítulo apresenta e explica as etapas do desenvolvimento do projeto. O quinto capítulo exibe os resultados finais do trabalho e dos experimentos. Por fim, no sexto capítulo são evidenciadas as considerações finais do projeto.

Capítulo 2

Preliminares

2.1 Direito e seus ramos

O Direito possui uma história bastante extensa e relatos apontam que sua origem ocorreu durante a Pré-História. De acordo com Paulo Nader (Nader, 2014), a ideia seria regular a vida em sociedade tendo em vista os conceitos de ordem e justiça. O Direito também pode ser entendido como uma organização de exigências a fim de estabelecer regras de comportamento para o convívio em sociedade.

Apesar de uma origem para atender um propósito específico, o Direito passou a evoluir e ser moldado de acordo com o contexto no qual está inserido. Em outras palavras, o sistema jurídico está em sincronia com o contexto, seja social ou econômico, ao qual pertence. Dessa forma, cada país ou Estado possui liberdade para modelar suas próprias regras do Direito, por muitas das vezes tomando como inspiração as experiências bem sucedidas de outros sistemas jurídicos já consolidados.

Os vários campos do Direito, atualmente, possuem uma finalidade primariamente didática, uma vez que um especialista na área deve possuir conhecimentos que transcendem essas divisões, de forma a ter capacidade de assimilar o funcionamento da engrenagem do Direito como um todo. Seguindo o pensamento de Antonio Betioli (Betioli, 2015), o especialista em Direito privado deve, em seu cotidiano, estar familiarizado e utilizar princípios definidos no Direito público. O oposto também ocorre com frequência, reforçando ainda mais a importância do entendimento geral do Direito.

As diversas classes existentes para o Direito são altamente subjetivas e podem variar conforme a bibliografia utilizada como alicerce. Inicialmente, ramifica-se o Direito entre as esferas privada e pública. Tal ramificação permeia desde a Antiguidade com o crescimento e aplicação do Direito Romano. O principal objetivo na época seria estabelecer uma fronteira clara entre o Estado e o indivíduo. O Direito público centraliza o Estado como figura de dominância por meio das políticas públicas. Por outro lado, o Direito privado contempla os interesses individuais e a convivência de um indivíduo com o restante da sociedade.

Em conformidade com o raciocínio de Paulo Nader (Nader, 2014), de forma alguma, o Direito público deve ser visto como uma oposição ao Direito privado. Há um conjunto de

princípios e conceitos mais genéricos que permeiam o universo do Direito, sem alguma especifidade a um certo ramo ou subárea do Direito.

Por sua vez, cada uma das esferas, tanto a privada quanto a pública, compreende subdivisões que são referenciadas como os ramos do Direito. Sendo assim, Sílvio de Salvo Venosa (Salvo Venosa, 2019) menciona que a esfera do Direito público concentra os seguintes ramos:

- Direito do trabalho: ramo que trata das relações jurídicas entre empregado e empregador, e abrange normas provenientes de acordos e convenções coletivas de trabalho. Também pertencem a esse ramo as normas que cuidam da representação dos vários sindicatos;
- Direito previdenciário: ramo que cuida dos benefícios e do funcionamento dos órgãos públicos de assistência e previdência social;
- Direito constitucional: ramo que se baseia na Constituição e trata dos princípios e normas relativos à estrutura fundamental do Estado;
- Direito administrativo: ramo que se refere à administração pública, englobando a regulação jurídica do poder executivo do Estado;
- Direito financeiro: ramo que trata das finanças públicas, regulando a estrutura orçamentária das entidades públicas;
- Direito tributário: ramo que ordena a forma de arrecadação de tributos e o relacionamento entre o Fisco[1] e o contribuinte;
- Direito processual civil: ramo que regula os procedimentos jurisdicionais, administrando as relações civis;
- Direito judiciário: ramo que abrange a organização, a jurisdição e a competência do Poder Judiciário;
- Direito penal: ramo que diz respeito ao direito de punir do Estado, sendo composto pelos preceitos legais para definir os crimes e determinar as devidas penas aos seus autores;
- Direito internacional: ramo que trabalha com os tratados e acordos internacionais e rege os direitos e deveres internacionais do Estado e dos indivíduos;
- Direito eleitoral: ramo que trata das normas e procedimentos que regulam o exercício do direito ao sufrágio universal; e
- Direito militar: ramo que estuda as relações jurídicas dos militares e regula os direitos, deveres e atividades da categoria.

Ainda em concordância com o raciocínio de Salvo Venosa (Salvo Venosa, 2019), a esfera do Direito privado compreende os seguintes ramos:

[1] Fisco é o órgão fiscalizador que controla os pagamentos de impostos em todas as esferas tributárias no país.

- Direito civil: ramo que trata dos conjuntos de normas jurídicas responsáveis por regular os direitos e obrigações em relação às pessoas e seus bens. É nele que se encontra o Código Civil, um conjunto de leis essenciais para o sistema judiciário;
- Direito da família: ramo que compreende o conjunto de normas marcadas pelo interesse social. É focado no convívio entre casais e seus filhos;
- Direito comercial: ramo que regula todas as relações jurídicas advindas do comércio, incluindo atividades empresariais e comerciais;
- Direito econômico: ramo que busca o ordenamento da macroeconomia e se refere à intervenção do Estado no domínio econômico;
- Direito ambiental: ramo que é composto pelos princípios e normas reguladoras das atividades humanas que possam afetar a preservação e estado do meio ambiente; e
- Direito do consumidor: ramo que trata do conjunto de regras e princípios jurídicos que regulam a relação entre o consumidor e o fornecedor de bens ou de serviços.

2.2 Jurisprudência e acórdãos do Supremo Tribunal Federal

A Constituição Federal garante a devida legitimidade do exercício do Supremo Tribunal Federal (STF). Com sua criação decorrente do Poder Constituinte originário, a Constituição define o STF como órgão do Poder Judiciário. O Supremo também se identifica pela defesa das minorias, mais especificamente pela igualdade nas decisões assegurando que todas as partes são representadas.

Segundo uma publicação do STF no portal Jusbrasil[2], decisões tomadas pelo Supremo Tribunal Federal são descritas por meio dos acórdãos. Os resultados dos processos gerados pelos ministros do STF são publicados nos acórdãos que permanecem visíveis no Diário da Justiça Eletrônico do STF (DJe). Até que seja apresentado no DJe, o acórdão percorre um longo caminho de escrita. Um desses passos seria a escrita da ementa por parte do ministro que redige o acórdão. A ementa corresponde a uma das seções que compõem o acórdão. Em especial, a ementa tem uma enorme importância, dado que ela sintetiza a decisão e define uma descrição direta do caso julgado.

A Pesquisa de Jurisprudência do STF[3] permite que os acórdãos do STF sejam visualizados no seu formato digitalizado. A pesquisa permite que o usuário preencha filtros de busca, ou utilize palavras-chave e mecanismos de linguagem natural para selecionar acórdãos, como mostrado na figura 2.1.

Utilizando uma pesquisa genérica, a pesquisa retornou um total de 323.683 acórdãos, como mostra a figura 2.2, o que demonstra a quantidade de documentos disponíveis no portal. Este número também representa um bom indicativo da quantia de acórdãos públicos do STF.

[2] https://stf.jusbrasil.com.br/noticias/100321222/acordao-do-julgamento-ate-a-publicacao-no-diario-da-justica

[3] https://jurisprudencia.stf.jus.br/pages/search

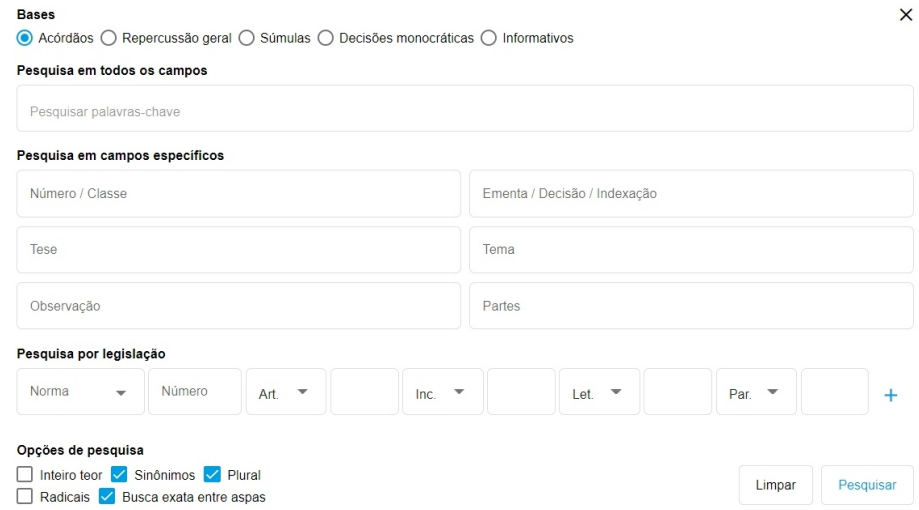


Figura 2.1: *Formulário de busca da Pesquisa de Jurisprudência do STF. Disponível em: https://jurisprudencia.stf.jus.br/pages/search.*

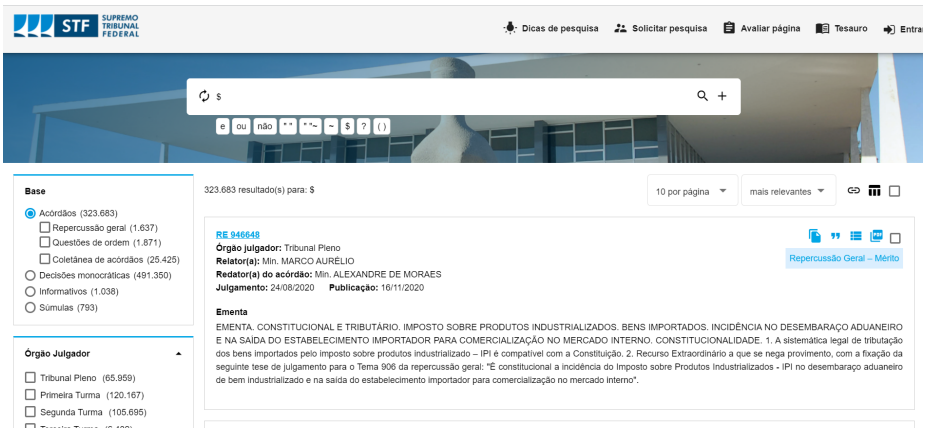


Figura 2.2: *Resultado de uma busca genérica na Pesquisa de Jurisprudência do STF. Imagem retirada em Novembro de 2021.*
Disponível em: https://jurisprudencia.stf.jus.br/pages/search?...&queryString=$&sort=_score&....

Por meio de uma pesquisa mais detalhada, a Pesquisa de Jurisprudência recupera os acórdãos relacionados no formato de uma lista. É possível acessar cada um dos acórdãos recuperados de forma a visualizar informações detalhadas, como mostrado na figura 2.3. Em geral, todos os acórdãos possuem a ementa contendo um resumo essencial do que está sendo discutido no acórdão. Um outro dado disponível nas informações dos acórdãos é um conjunto de termos e expressões-chave que estão relacionados com os documentos, também chamado de indexação.

Para obter todas as informações disponíveis sobre um determinado acórdão, o usuário também pode optar por acessar o inteiro teor[4]. Nesse documento, em formato *Portable Document Format* (pdf), a decisão está detalhada de forma integral, contendo o texto completo da decisão, bem como os relatórios e votos dos membros do julgamento. Um

[4] https://www.tre-al.jus.br/jurisprudencia/perguntas-frequentes

Resultado completo | Ementa sem formatação

RE 865401 / MG - MINAS GERAIS
RECURSO EXTRAORDINÁRIO
Relator(a): Min. DIAS TOFFOLI
Julgamento: 25/04/2018 Publicação: 19/10/2018
Órgão julgador: Tribunal Pleno

Repercussão Geral – Mérito

Publicação
ACÓRDÃO ELETRÔNICO
REPERCUSSÃO GERAL - MÉRITO
DJe-223 DIVULG 18-10-2018 PUBLIC 19-10-2018

Partes
RECTE.(S) : MARCOS ANTÔNIO RIBEIRO FERRAZ
ADV.(A/S) : DAVI LEONARD BARBIERI E OUTRO(A/S)
RECDO.(A/S) : ANTÔNIO VAZ DE MELO
ADV.(A/S) : JÉSUS IRINEU RIBEIRO FILHO E OUTRO(A/S)
RECDO.(A/S) : PREFEITO MUNICIPAL DE GUIRICEMA
PROC.(A/S)(ES) : PROCURADOR-GERAL DO MUNICÍPIO DE GUIRICEMA

Ementa
EMENTA Direito Constitucional. Direito fundamental de acesso à informação de interesse coletivo ou geral. Recurso extraordinário que se funda na violação do art. 5º, inciso XXXIII, da Constituição Federal. Pedido de vereador, como parlamentar e cidadão, formulado diretamente ao chefe do Poder Executivo solicitando informações e documentos sobre a gestão municipal. Pleito indeferido. Invocação do direito fundamental de acesso à informação, do dever do poder público de transparência e dos princípios republicano e da publicidade. Tese da municipalidade fundada na separação dos poderes e na diferença entre prerrogativas da casa legislativa e dos parlamentares. Repercussão geral reconhecida. 1. O tribunal de origem acolheu a tese de que o pedido do vereador para que informações e documentos fossem requisitados pela Casa Legislativa foi, de fato, analisado e negado por decisão do colegiado do parlamento. 2. O jogo político há de ser jogado coletivamente, devendo suas regras ser respeitadas, sob pena de se violar a institucionalidade das relações e o princípio previsto no art. 2º da Carta da República. Entretanto, o controle político não pode ser resultado apenas da decisão da maioria. 3. O parlamentar não se despe de sua condição de cidadão no exercício do direito de acesso a informações de interesse pessoal ou coletivo. Não há como se autorizar que seja o parlamentar transformado em cidadão de segunda categoria. 4. Distinguishing em relação ao caso julgado na ADI nº 3.046, Relator o Ministro Sepúlveda Pertence. 5. Fixada a seguinte tese de repercussão geral: o parlamentar, na condição de cidadão, pode exercer plenamente seu direito fundamental de acesso a informações de interesse pessoal ou coletivo, nos termos do art. 5º, inciso XXXIII, da CF e das normas de regência desse direito. 6. Recurso extraordinário a que se dá provimento.

Decisão
O Tribunal, apreciando o tema 832 da repercussão geral, por unanimidade e nos termos do voto do Ministro Dias Toffoli (Relator), deu provimento ao recurso e fixou a seguinte tese: "O parlamentar, na condição de cidadão, pode exercer plenamente seu direito fundamental de acesso a informações de interesse pessoal ou coletivo, nos termos do art. 5º, inciso XXXIII, da CF e das normas de regência desse direito". Presidiu o julgamento a Ministra Cármen Lúcia. Plenário, 25.4.2018.

Indexação
- DIREITO, BRASILEIRO, ESTRANGEIRO DOMICILIADO NO BRASIL, PESSOA JURÍDICA, RECEBIMENTO, INFORMAÇÃO, ÓRGÃO PÚBLICO, RISCO, RESPONSABILIDADE. EXCEÇÃO, DIREITO À INFORMAÇÃO, SIGILO, SEGURANÇA, SOCIEDADE, PODER PÚBLICO. RESTRIÇÃO, DIREITO FUNDAMENTAL. CONSTITUIÇÃO FEDERAL; LEI INFRACONSTITUCIONAL; DIREITO FUNDAMENTAL, TERCEIRO. FUNÇÃO, PODER LEGISLATIVO, AUXÍLIO, TRIBUNAL DE CONTAS, FISCALIZAÇÃO CONTÁBIL, FISCALIZAÇÃO ORÇAMENTÁRIA, FISCALIZAÇÃO OPERACIONAL, FISCALIZAÇÃO PATRIMONIAL, ADMINISTRAÇÃO PÚBLICA, ÂMBITO FEDERAL, ÂMBITO MUNICIPAL. PRINCÍPIO DA SIMETRIA. FISCALIZAÇÃO, CONGRESSO NACIONAL, CONVOCAÇÃO, MINISTRO DE ESTADO, FORNECIMENTO, INFORMAÇÃO; REMESSA, PEDIDO, INFORMAÇÃO, MESA DIRETORA, CÂMARA DOS DEPUTADOS, SENADO FEDERAL; INSTALAÇÃO, COMISSÃO PARLAMENTAR DE INQUÉRITO (CPI), APURAÇÃO, FATO. PRINCÍPIO DA COLEGIALIDADE. JURISPRUDÊNCIA, STF, ILEGITIMIDADE, PARLAMENTAR, IMPETRAÇÃO, MANDADO DE SEGURANÇA, DEFESA, PRERROGATIVA, CASA LEGISLATIVA, CABIMENTO, PARLAMENTAR, REPRESENTAÇÃO ADMINISTRATIVA, ATO, IMPROBIDADE ADMINISTRATIVA, SOLICITAÇÃO, INVESTIGAÇÃO, MINISTÉRIO PÚBLICO.

Figura 2.3: *Visualização das informações detalhadas de um acórdão recuperado. Disponível em: https:// jurisprudencia.stf.jus.br/ pages/ search/ sjur392966/ false.*

exemplo de uma página do inteiro teor de um acórdão pode ser visto na figura 2.4.

Mesmo com a enorme quantidade de informações fornecidas por essa pesquisa, ela ainda não é capaz de detalhar o ramo do Direito ao qual uma decisão pertence. O maior motivo para essa situação seria a necessidade de uma classificação manual, ou seja, uma pessoa com conhecimento suficiente para abstrair o conteúdo do acórdão ao seu ramo do Direito.

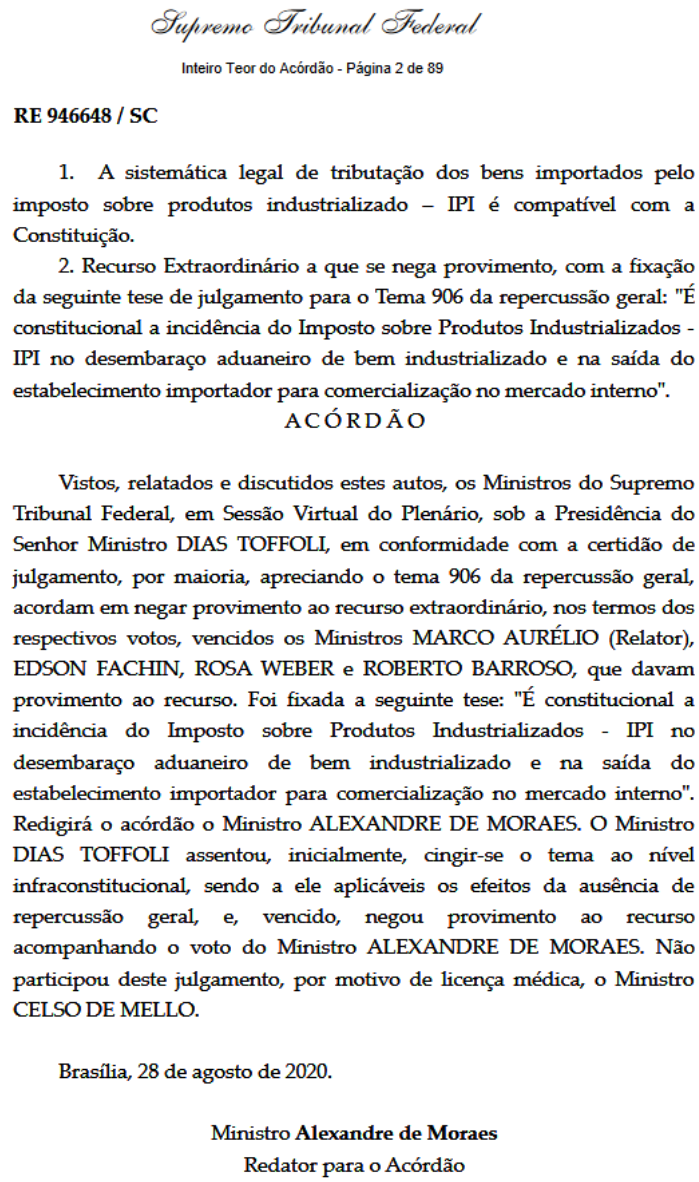
O processo de classificação consiste em, a partir dos conceitos tratados em um acórdão, identificar o ramo do Direito ao qual ele pertence. No caso, um acórdão é classificado a um único ramo do Direito, de forma que as classes sejam mutuamente excludentes.

2.3 Projetos anteriores

Rafael Brito de Oliveira, ex-aluno do Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP), escreveu uma dissertação sobre seu trabalho desenvolvido com base no conjunto de dados dos acórdãos do STF. O resultado final foi uma ferramenta de consulta de acórdãos para possibilitar uma pesquisa mais intuitiva para seus usuários (Oliveira, 2017).

Via de regra, para realizar uma consulta por meio da Pesquisa de Jurisprudência, um formulário complexo deve ser preenchido. Em consequência disso, usuários sem conhecimento técnico da área do Direito veem uma certa limitação para utilizar a ferramenta adequadamente.

Visto isso, Oliveira sugeriu uma ferramenta em que as consultas pudessem assumir um formato mais simples recorrendo ao uso da linguagem natural. Com isso, sentenças próximas às questões cotidianas do universo do Direito poderiam ser facilmente respondidas como: "Quais são os ministros mais desafiadores?".



Supremo Tribunal Federal

Inteiro Teor do Acórdão - Página 2 de 89

RE 946648 / SC

1. A sistemática legal de tributação dos bens importados pelo imposto sobre produtos industrializado – IPI é compatível com a Constituição.

2. Recurso Extraordinário a que se nega provimento, com a fixação da seguinte tese de julgamento para o Tema 906 da repercussão geral: "É constitucional a incidência do Imposto sobre Produtos Industrializados - IPI no desembaraço aduaneiro de bem industrializado e na saída do estabelecimento importador para comercialização no mercado interno".

ACÓRDÃO

Vistos, relatados e discutidos estes autos, os Ministros do Supremo Tribunal Federal, em Sessão Virtual do Plenário, sob a Presidência do Senhor Ministro DIAS TOFFOLI, em conformidade com a certidão de julgamento, por maioria, apreciando o tema 906 da repercussão geral, acordam em negar provimento ao recurso extraordinário, nos termos dos respectivos votos, vencidos os Ministros MARCO AURÉLIO (Relator), EDSON FACHIN, ROSA WEBER e ROBERTO BARROSO, que davam provimento ao recurso. Foi fixada a seguinte tese: "É constitucional a incidência do Imposto sobre Produtos Industrializados - IPI no desembaraço aduaneiro de bem industrializado e na saída do estabelecimento importador para comercialização no mercado interno". Redigirá o acórdão o Ministro ALEXANDRE DE MORAES. O Ministro DIAS TOFFOLI assentou, inicialmente, cingir-se o tema ao nível infraconstitucional, sendo a ele aplicáveis os efeitos da ausência de repercussão geral, e, vencido, negou provimento ao recurso acompanhando o voto do Ministro ALEXANDRE DE MORAES. Não participou deste julgamento, por motivo de licença médica, o Ministro CELSO DE MELLO.

Brasília, 28 de agosto de 2020.

Ministro **Alexandre de Moraes**
Redator para o Acórdão

Figura 2.4: *Visualização de uma página do inteiro teor do acórdão RE 946648. Disponível em: https://redir.stf.jus.br/paginadorpub/paginador.jsp?docTP=TP&docID=754380405.*

Para a construção do sistema, Oliveira projetou uma aplicação *web* para fornecer o acesso apropriado aos usuários. Ele ainda propôs metodologias para representar o conhecimento abrangido pelo acórdão de maneira mais eficiente para a construção de um mecanismo de consulta. Um exemplo de visualização da página da aplicação de Oliveira pode ser visto na figura 2.5.

Figura 2.5: *Aplicação de Oliveira.*

Capítulo 3

Fundamentação Teórica

Esta seção será responsável por abordar tópicos que irão embasar as discussões sobre o desenvolvimento do projeto.

3.1 Rede Neural

Com o avanço das pesquisas no ramo da Inteligência Artificial, expressões como "rede neural" vêm se tornando cada vez mais frequentes. Nas palavras de Kevin Gurney, uma rede neural é uma interconexão de elementos de processamento, unidades ou nós, em que a funcionalidade foi levemente inspirada no neurônio animal. O poder de processamento da rede neural se dá pela força das conexões dos neurônios, ou seja, os parâmetros que são obtidos por um processo de adaptação e aprendizado de padrões de treinamento (Gurney, 1997).

De forma a se ter um rigor matemático, uma rede neural pode também ser vista como:

$$y = f_{RN}(x) = f_{nl}(...(f_3(f_2(f_1(x)))$$

Assim sendo, uma rede neural é simplesmente uma função matemática que, ao ser decomposta, corresponde ao aninhamento de um conjunto de funções em uma determinada sequência. O número de camadas na rede neural é dado pelo número nl.

Por sua vez, cada uma das funções aninhadas também possui um propósito e modelagem bastante específicos:

$$f_l(z) \stackrel{def}{=} g_l(W_l * z + b_l)$$

Na definição acima, l denota a camada de rede neural que está sendo referida, logo o valor pode variar de 1 até nl. Por outro lado, g_l é chamada de função de ativação e geralmente é não linear. Além disso, outros dois parâmetros são mencionados. O primeiro parâmetro, W_l, representa uma matriz contendo os pesos. O último, b_l, representa um

vetor de viés. Ambos são ajustados de acordo com o treinamento da rede neural seguindo alguma estratégia de otimização bem definida, como o gradiente descendente[1] (Burkov, 2019).

De modo geral, a estrutura de uma rede neural, que é esboçada na figura 3.1, consiste em três tipos de camadas: entrada, saída e oculta. A camada de entrada é responsável por receber os dados a serem processados pela rede. A camada oculta é capaz de transformar o dado de entrada e ajustá-lo corretamente até que se atinja o formato e valor esperado para a camada de saída. A camada de saída, de forma intuitiva, representa os dados resultantes depois do processamento das camadas ocultas.

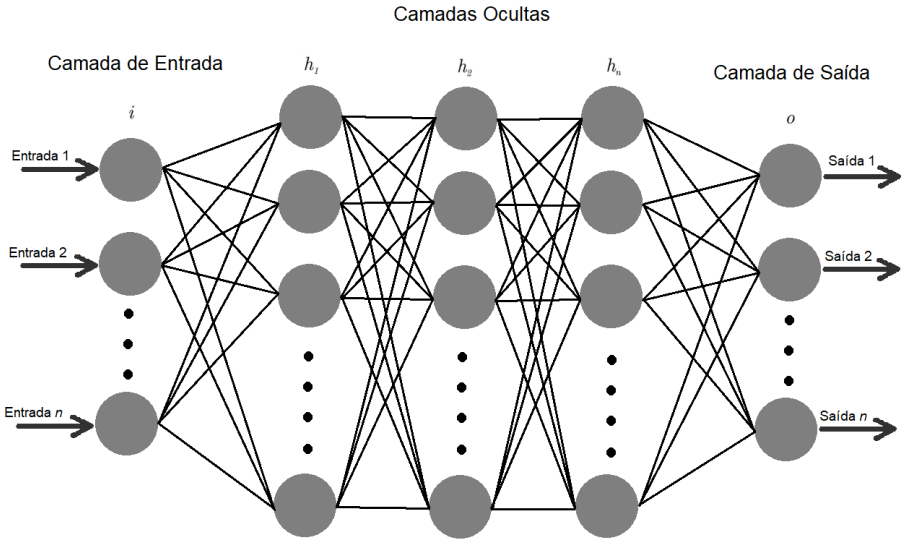


Figura 3.1: *Estrutura de uma rede neural.*

3.2 BERT

O nome BERT é uma sigla baseada nos termos em inglês *Bidirectional Encoder Representations from Transformers*, ou seja, Representações de Codificadores Bidirecionais a partir de Transformadores. O BERT foi planejado para pré-treinamento de representações bidirecionais complexas a partir de porções de textos ainda não classificadas. Sendo assim, o BERT é capaz de contextualizar todas as camadas da rede neural pelo conteúdo localizado à direita e à esquerda do texto. O estudo que possibilitou a criação do BERT foi realizado pela divisão de linguagens do Google: *Google AI Language* (Devlin *et al.*, 2019).

O BERT propõe o uso do modelo de linguagem mascarada (em inglês, *masked language model*), também chamado de MLM. A MLM, de forma aleatória, mascara alguns *tokens* do texto de forma a obter uma fusão de contexto recolhendo conteúdo à esquerda e à direita do texto. O resultado é o pré-treinamento bidirecional do Transformador, assim como mencionado anteriormente.

[1] Gradiente descendente é um algoritmo de otimização em que o objetivo é minimizar uma função utilizando um esquema iterativo (Trehan, 2020).

O BERT surgiu para propor uma solução visando assumir o estado da arte para diversas tarefas envolvendo o Processamento de Linguagem Natural (PLN), sendo uma delas a classificação de um trecho, ou seja, mapear uma porção textual a um determinado rótulo.

A criação do BERT é fundamentada em dois processos: pré-treino (no inglês, *pre-training*) e os ajustes finos (no inglês, *fine tuning*). O pré-treinamento é feito com base em dados não rotulados que são submetidos a tarefas específicas. O BERT oferece duas opções de modelos que diferem basicamente no tamanho:

- $BERT_{BASE}$: 12 camadas, 768 nós nas camadas ocultas e 110 milhões de parâmetros; e
- $BERT_{LARGE}$: 24 camadas, 1024 nós nas camadas ocultas e 340 milhões de parâmetros.

A entrada dos modelos BERT pode ser constituída de uma sentença ou um par de sentenças. O segundo caso é essencial para situações em que é preciso receber um par (questão, resposta). É importante ressaltar que o significado do termo "sentença" empregado no BERT não corresponde ao mesmo da Língua Portuguesa. No caso do BERT, uma sentença representa uma porção de texto.

Como parte do processo de *tokenização*[2], um *token* especial é adicionado ao início de cada sentença: [CLS]. O BERT também adiciona um outro *token* especial para delimitar uma separação entre as sentenças: [SEP]. Um exemplo do *tokenizador* em ação pode ser visto na figura 3.2.

Depois disso, o BERT executa duas tarefas específicas:

- MLM: tarefa responsável por mascarar uma determinada porcentagem dos *tokens* disponíveis na sentença. Dessa forma, o BERT deve trabalhar em predizer os *tokens* mascarados, por isso, o nome atribuído é MLM; e
- Predição da próxima sentença: tarefa cujo objetivo é facilitar a compreensão do BERT sobre a conexão entre duas sentenças. O modelo é treinado para identificar quando duas sentenças A e B são subsequentes, bem como quando não são.

Quando o foco é a língua inglesa, os treinamentos, nos quais o modelo BERT é exposto inicialmente, são baseados em dois vocabulários: *BooksCorpus* (contendo 800 milhões de palavras) e *English Wikipedia* (2.5 bilhões de palavras).

Sendo assim, o BERT surge como uma arquitetura de modelo, para se tornar o estado da arte quando o assunto é resolução de problemas envolvendo processamento de linguagem natural.

3.2.1 BERTimbau

As rotinas de pré-treinamento do BERT são sempre baseadas em um certo idioma. Acima, é possível perceber que a arquitetura inicialmente proposta foi exercitada através

[2] *Tokenização* é o processo de separação de um texto em unidades de menor tamanho chamadas *tokens*, podendo estes serem palavras, subpalavras ou caracteres. É um processo essencial para tarefas de processamento de linguagem natural pois permite aos modelos interpretar melhor o contexto das sequências de texto a serem analisadas (Chakravarthy, 2020).

```
tokenizer = BertTokenizer.from_pretrained('neuralmind/bert-large-portuguese-cased')
ementa = 'ementa: Extradição Instrutória. Regularidade Formal. Requisitos Legais Atendid
tokenized = tokenizer.encode(ementa)
pp = pprint.PrettyPrinter(indent=4)
pp.pprint(tokenizer.convert_ids_to_tokens(tokenized))
```

```
[   '[CLS]',
    'em',
    '##enta',
    ':',
    'Extra',
    '##dição',
    'Instru',
    '##tória',
    '.',
    'Reg',
    '##ular',
    '##idade',
    .
    .
    .
    '##5',
    '/',
    '1980',
    '.',
    '[SEP]']
```

Figura 3.2: *Exemplo de tokenização executada pelo tokenizador do BERTimbau.*

de vocabulários em inglês. Para o propósito desse trabalho, um modelo com base na língua inglesa não seria passível de utilização, uma vez que as ementas dos acórdãos do STF estão escritas em português.

Por essa razão, os modelos utilizados para esse projeto são resultado de uma pesquisa denominada BERTimbau (Souza *et al.*, 2019). O BERTimbau surgiu com um propósito específico de adaptar a aplicação do BERT para a Língua Portuguesa.

Não há diferenças significativas entre o BERT e o BERTimbau a nível arquitetural, porém o pré-treino do modelo é realizado utilizando um vocabulário diferente que é o BrWaC (no inglês, *Brazilian Web as Corpus*). Assim como o BERT, o BERTimbau fornece um modelo padrão, também dito como $BERT_{BASE}$, e outro mais robusto, também referido como $BERT_{LARGE}$.

Os modelos definidos no projeto BERTimbau são *case sensitive*, isso significa que palavras são diferenciadas caso sejam escritas com letras maiúsculas ou minúsculas. Esse tipo de diferenciação torna-se essencial em alguns contextos cuja forma de escrita influencia no comportamento do modelo. De forma geral, a análise tipográfica das ementas dos acórdãos não influenciará no resultado do modelo.

3.3 Transferência de aprendizagem

O conceito de transferência de aprendizagem (*transfer learning*, no inglês) surgiu da capacidade do ser humano de se adaptar a tarefas similares àquelas já aprendidas. No contexto da Inteligência Artificial, a transferência de aprendizagem busca aproveitar o conhecimento já adquirido em um certo domínio para obter uma melhor performance de aprendizado, mesmo com um número menor de dados classificados no domínio de estudo (Zhuang *et al.*, 2020).

A aplicação de transferência de aprendizagem pode ser feita seguindo mecanismos diversos. Especificamente, a aplicação de transferência de aprendizagem ao modelo BERT fica compreendida entre o treinamento supervisionado (realizado por meio de dados classificados) e não supervisionado (realizado por meio de dados não rotulados). De forma geral, a transferência de aprendizagem combina uma grande quantidade de dados não rotulados com uma quantidade limitada de dados rotulados.

As técnicas de transferência de aprendizagem tornam menor a necessidade de dados rotulados. Os cenários ideais para o uso dessas técnicas são quando o modelo está muito bem treinado para o domínio em que há uma quantidade de dados não rotulados em abundância. No entanto, o conjunto de dados rotulados é menor.

Para o propósito do projeto, adquirir dados de acórdãos rotulados é um processo bastante custoso. Dessa forma, a ideia de usar do modelo BERT já treinado para a Língua Portuguesa juntamente de um conjunto de dados pequeno de acórdãos rotulados indicava que bons resultados poderiam ser obtidos.

3.4 Conjunto de dados

Para realizar o treinamento de uma rede neural, seja por meio do aprendizado supervisionado ou da transferência de aprendizado, o uso correto do conjunto de dados é fundamental. Como parte do processo, é preciso que se tenha posse de um conjunto de dados rotulados de tamanho satisfatório, uma vez que ele precisará ser dividido em 3 subconjuntos:

- Conjunto de treinamento: dados utilizados para a construção do modelo. Esse conjunto carrega os dados de entrada e o resultado esperado do modelo. Com isso, o modelo pode ser construído a partir da comparação de suas predições e o resultado esperado;
- Conjunto de validação: dados utilizados para avaliar o modelo enquanto o treinamento ocorre. O uso do conjunto de validação garante que o modelo terá uma boa performance não apenas para o conjunto de treinamento, mas também para dados que ainda não foram vistos. Com isso, o objetivo dessa coleção de dados é prevenir sobreajuste[3] e subajuste[4]; e
- Conjunto de teste: dados utilizados para avaliar a performance do modelo quando sua construção é finalizada. Diferentemente do conjunto de validação, a coleção de dados de teste não é manipulada em tempo de treinamento do modelo.

Parte essencial do processo de divisão dos dados é garantir uma boa partição entre os três conjuntos. No passado, uma regra bastante utilizada seria particionar 70% dos dados para treinamento, 15% para validação e os 15% restantes para testes. Essa metodologia

[3] Do inglês *overfitting*, o sobreajuste denota a situação na qual o modelo se adequou exageradamente aos dados do conjunto de treinamento, resultando em uma performance fraca em dados ainda não vistos.

[4] Do inglês *underfitting*, o subajuste denota a situação na qual o modelo ainda não se adequou adequadamente aos dados do conjunto de treinamento, resultando em uma performance fraca nele.

de divisão de dados é bastante tradicional e recomendada por inúmeros autores, principalmente no passado. Em casos em que há abundância de dados, alocar uma proporção menor para o conjunto de treinamento pode gerar melhores resultados (Burkov, 2019). No entanto, durante o desenvolvimento da solução tratada nesse documento, o conjunto de dados disponível continha apenas 5524 registros.

3.5 Treinamento de uma rede neural

O treinamento de uma rede neural, de certa forma, segue um padrão que pode ser empregado na resolução de múltiplos problemas. Quando se segue o método de transferência de aprendizado, o objetivo do treinamento é relacionar os dados de entrada aos resultados esperados, assim como no aprendizado supervisionado (Rao e McMahan, 2019). Para o exemplo do classificador desenvolvido nesse projeto, os dados de entrada seriam os conteúdos da ementa de um acórdão, e os resultados corresponderiam às classificações dos acórdãos dentre os ramos do Direito.

Usualmente, para modelos que seguem esse cenário, as predições são utilizadas para ajustar os parâmetros dos modelos, por meio da otimização por gradiente. Sendo assim, a tarefa de definir os requisitos para o treinamento de um modelo torna-se mais simples. Os requisitos são: a arquitetura do modelo a ser utilizado, conjuntos de dados, uma função de perda (em inglês, *loss function*), um cronograma de taxa de aprendizado e um algoritmo de otimização.

Por enquanto, há dois requisitos que já estão definidos: arquitetura do modelo e conjuntos de dados. O primeiro, já discutido previamente, será o modelo seguindo a arquitetura BERT. Os dados de treinamento serão discutidos na seção 4.3 e representam uma relação entre a ementa de um acórdão com o ramo ao qual ele pertence.

A seguir, os demais requisitos serão apresentados, juntamente de um algoritmo de treinamento normalmente utilizado e outros conceitos importantes para essa etapa do projeto.

3.5.1 Função de perda

A função de perda (no inglês, *loss function*) fornece uma forma de medir o quão díspar uma predição está do resultado desejado. Essa função recebe a predição do modelo e o resultado esperado, retornando um valor real correspondendo à perda. A perda indica a disparidade mencionada, logo quanto menor o valor, então melhor é a perfomance do modelo.

Um exemplo de função de perda seria a função de entropia cruzada (Gordon-Rodriguez *et al.*, 2020). Ela é recomendada para problemas cuja tarefa consiste na classificação dos dados de entrada. Em outras palavras, a predição do modelo e o resultado esperado são valores discretos numéricos. Sendo assim, a função de entropia cruzada é definida como segue:

$$L_{cross\_entropy}(y, \hat{y}) = -\sum_{i}^{K} y_i * log(\hat{y}_i)$$

Acima, $\hat{y}$ define o vetor de predição do modelo, enquanto que y denota os valores esperados como saída do modelo. O valor K seria a quantidade de classes do modelo.

Dessa forma, a função de entropia cruzada resulta em um valor probabilístico para os dados discretos dentre as K classes.

3.5.2 Otimizador

Uma vez que o modelo produz as predições utilizando os dados de entrada, a função de perda as avalia para mensurar o erro entre a predição e o resultado esperado. No entanto, alguma ação precisa ser tomada no modelo para que seu desempenho se adapte ao processo de treinamento (Rao e McMahan, 2019).

Por conta disso, o otimizador desempenha um papel fundamental durante o processo de treinamento. O otimizador é responsável por atualizar os parâmetros do modelo, ou seja, vieses e pesos utilizando as métricas de erro, como a função de perda. Um otimizador elementar carregará um único hiperparâmetro[5], a taxa de aprendizado, que controlará a performance do otimizador. A taxa de aprendizado representa como as métricas de erro irão impactar na atualização dos parâmetros citados anteriormente.

De forma geral, um alto valor para taxa de aprendizado resulta em mudanças de parâmetros mais agressivas, o que resulta em má performance de convergência desses valores. Por outro lado, um valor pequeno de taxa de aprendizado resulta em pouco progresso de treinamento.

Um exemplo de otimizador é o Adam (Kingma e Ba, 2017). O grande diferencial do Adam em comparação com seus concorrentes foi a definição de taxas de aprendizado diferentes para os parâmetros do modelo. Introduzido em 2014, tornou-se um método muito importante para o treinamento de rede neurais, uma vez que oferecia uma performance superior em relação a tempo de execução e consumo de memória.

Depois disso, em 2017, uma variante do Adam foi apresentada: o AdamW (Loshchilov e Hutter, 2019). Com a popularização do Adam, muitos pesquisadores perceberam resultados inferiores com o uso do otimizador em certas tarefas, quando comparado à outros mais simples, como o gradiente descendente estocástico. Isso se dá por conta da implementação de regularização L2[6] para atualizar os parâmetros do modelo. Por sua vez, o AdamW substitui a regularização L2 pelo método de decaimento de peso[7] (Gugger e

[5] Uma variável que representa uma configuração do modelo a ser treinado. Os hiperparâmetros precisam ser ajustados por meio da execução de uma série de treinamento, a fim de prover a melhor configuração que resulta em um bom desempenho do modelo.

[6] Do inglês *L2 regularization*, consiste em um método clássico para reduzir sobreajuste por meio da adição da soma do quadrado de todos os pesos do modelo ao valor da função de perda, ainda multiplicado por um valor chamado fator de decaimento.

[7] Do inglês *weight decay*, consiste em utilizar regularização L2, porém com uma redução do fator de decaimento a cada passo de treinamento.

Howard, 2018).

3.5.3 Cronograma de taxa de aprendizado

O otimizador trabalha em conjunto com o que é chamado cronograma de taxa de aprendizado. O cronograma se preocupa em reduzir a taxa de aprendizado de acordo com o avanço no número de passos de treinamento. O cronograma parte do pressuposto de que quanto mais passos de treinamento foram executados, menos agressiva a taxa de aprendizado precisa se comportar.

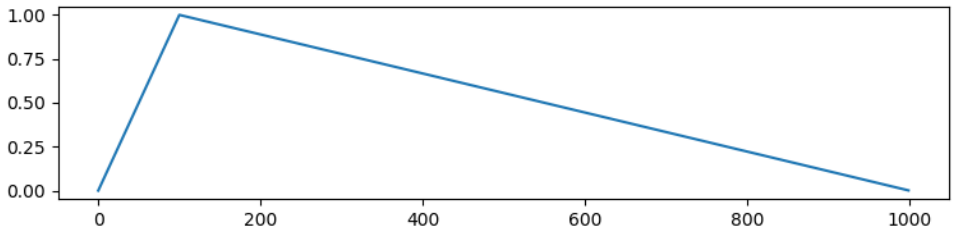


Figura 3.3: *Cronograma linear de taxa de aprendizado. A abscissa refere-se à quantidade de iteraçãoes, ou seja, a quantidade de lotes de dados consumidos, considerando todas as épocas já executadas. A ordenada refere-se ao fator de consideração da taxa de aprendizado.*
Disponível em: https://huggingface.co/transformers/v3.0.2/main_classes/optimizer_schedules.html#transformers.get_linear_schedule_with_warmup

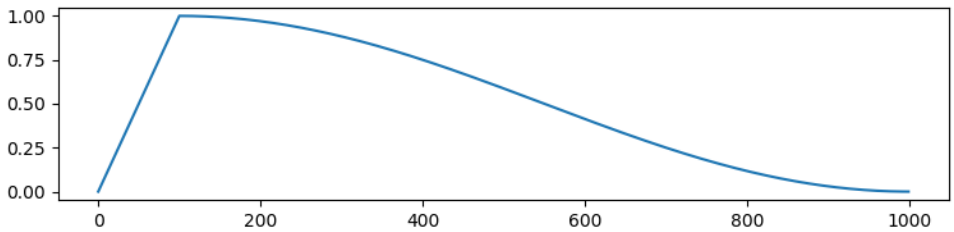


Figura 3.4: *Cronograma de cosseno de taxa de aprendizado. A abscissa refere-se à quantidade de iteraçãoes, ou seja, a quantidade de lotes de dados consumidos, considerando todas as épocas já executadas. A ordenada refere-se ao fator de consideração da taxa de aprendizado.*
Disponível em: https://huggingface.co/transformers/v3.0.2/main_classes/optimizer_schedules.html#transformers.get_cosine_schedule_with_warmup

A figura 3.3 ilustra o cronograma linear, em que a taxa de aprendizado é reduzida seguindo uma função linear. Por outro lado, a figura 3.4 ilustra o cronograma de cosseno, em que a taxa de aprendizado é reduzida seguindo a função cosseno.

Em ambos os casos, o cronograma apresenta, no início, um aumento do fator de consideração da taxa de aprendizado. Isso ocorre, pois na maioria dos casos não é recomendado iniciar o treinamento com uma taxa de aprendizado agressiva. Dessa forma, os cronogramas apresentam esse aumento inicial do fator que são chamados de passos ou iterações de aquecimento.

No uso geral, esses passos iniciais são definidos por meio da **proporção de passos de aquecimento**, com um valor real entre 0 e 1 que seleciona uma porção dos passos da

primeira época de treinamento para serem reservados para aquecimento do procedimento. Para a definição da quantidade de passos de aquecimento, a proporção é aplicada na quantidade de iterações da primeira época de treinamento.

3.5.4 Processo de treinamento

O processo de treinamento de uma rede neural utilizando o método de aprendizagem supervisionada ou de transferência de aprendizado segue uma estrutura bem definida (Stevens *et al.*, 2020). Na rotina de treinamento citada abaixo, utiliza-se dois conjuntos de dados: treinamento e validação. Além disso, alguns hiperparâmetros são definidos e alguns deles possuem relação com os lotes de dados (no inglês, *batches*)[8]:

- *batch_size_training*: tamanho de cada lote de dados utilizado para treinamento;
- *batch_size_validation*: tamanho de cada lote de dados utilizado para validação; e
- número de épocas: quantidade de vezes com que o processo de treinamento percorrerá o conjunto de treinamento.

O processo essencial para se treinar uma rede neural é mostrado abaixo.

```
1. Inicializar o modelo
2. Iterar sobre um número específico de épocas (escolhido
antecipadamente como hiperparâmetro)
    2.1. Dividir o conjunto de treinamento em lotes de tamanho
    'batch_size_training' (escolhido antecipadamente como
    hiperparâmetro)
    2.2. Iterar sobre cada lote de treinamento gerado em 2.1
        2.2.1. Usar o lote de treinamento no modelo para coleta
        da predição
        2.2.2. Calcular a diferença entre a predição e o
        resultado esperado por meio da função de perda
        2.2.3. Armazenar a diferença como métrica
        2.3.4. A partir da diferença calculada por meio da
        função de perda, atualizar os parâmetros do modelo
        (pesos e vieses) por meio de retropropagação
    2.3. Dividir o conjunto de treinamento em lotes de tamanho
    'batch_size_validation' (escolhido antecipadamente como
    hiperparâmetro)
    2.4. Iterar sobre cada lote de validação gerado em 2.3
        2.4.1. Usar o lote de validação no modelo e coletar a
        predição
        2.4.2. Calcular a diferença entre a predição e o
        resultado esperado por meio da função de perda
        2.4.3. Armazenar a diferença como métrica
    2.5. Reunir os valores adquiridos no processo de treinamento
```

[8] Para o treinamento, os conjuntos de dados são divididos em lotes do mesmo tamanho com o objetivo de reduzir o tempo de treinamento e o uso de memória necessária para carregar os dados no modelo.

```
e validação para compor um resumo geral das métricas
```

Esse foi o procedimento colocado em execução para o desenvolvimento desse projeto. Um detalhe importante de ser ressaltado é a utilização de retropropagação. A retropropagação é um método iterativo de atualização de parâmetros de um modelo. A técnica compreende duas etapas: *forward pass* e *backward pass*. A primeira passagem é frontal e consiste em utilizar o conjunto de dados para gerar as predições por meio do modelo. Depois disso, as discrepâncias são calculadas através da função de perda. Por fim, a passagem posterior é executada para atualizar os parâmetros do modelo utilizando o gradiente do resultado da função de perda.

3.5.5 Métricas

As métricas definem formas de mensurar a performance do modelo. Para extrair a máxima utilidade dos conjuntos de teste e validação, é preciso submeter o modelo ao uso das métricas. Por meio delas é possível comparar diferentes modelos, seja portando valores para hiperparâmetros distintos, ou estratégias de treinamento diversas.

A depender do problema que o modelo deve resolver e suas características, as métricas mais essenciais podem variar. No contexto da classificação de acórdãos, ou tarefas de classificação em geral, algumas métricas se destacam mais, e cada uma será discutida a seguir.

Algumas nomenclaturas devem ser estabelecidas quando métricas de classificação são discutidas. Tais nomenclaturas, no ponto de vista de uma classificação com múltiplos rótulos, devem ser sempre vistas na perspectiva de uma das categorias, por exemplo uma classe hipotética 'k':

- Verdadeiro Positivo (VP_k): o acórdão pertence à classe k e o modelo categorizou o acórdão como parte da classe k.
- Falso Positivo (FP_k): o modelo categorizou como parte da classe k, porém o acórdão pertence à outra classe;
- Verdadeiro Negativo (VN_k): o modelo categorizou o acórdão como parte de outra classe, e o acórdão não pertencia à classe k; e
- Falso Negativo (FN_k): o modelo categorizou o acórdão como parte de outra classe, no entanto o acórdão pertencia à classe k.

As definições abaixo foram baseadas em (Grandini *et al.*, 2020).

Acurácia

A acurácia é uma métrica muito popular e define de forma muito simples alguns aspectos de performance de um modelo. De forma resumida, a acurácia permite identificar a probabilidade com que o modelo acertaria a classificação de um dado. No contexto desse projeto, a acurácia mede a probabilidade com que o modelo atribui a classe correta ao acórdão, dada a sua ementa.

Sendo assim, sua definição calcula a proporção de acertos do modelo em relação a uma classe k, comparando com todos os dados classificados:

$$Acurácia_k = \frac{VP_k + VN_k}{VP_k + FP_k + VN_k + FN_k}$$

Acima, a acurácia da classe k não corresponde à acurácia do modelo, porque o problema inclui múltiplas classes. Dessa forma, dado que o conceito de acurácia sempre representa a proporção de acertos de um modelo, então tal ideia pode ser extrapolada para um problema com múltiplas classes. Sendo assim, a acurácia geral para um modelo de múltiplas classes é a proporção das classificações corretas comparando com todas as classificações:

$$Acurácia = \frac{\sum_{k=1}^{N} VP_k}{\sum_{k=1}^{N} VP_k + FN_k}$$

Para a acurácia considera-se que N é a quantidade total de classes e as classes podem variar no conjunto $\{1, 2, ..., N\}$.

$$Acurácia\_Balanceada = \frac{\sum_{k=1}^{N} \frac{VP_k}{VP_k + FN_k}}{N}$$

Acima, também é definida a acurácia balanceada do modelo. Quando o conjunto de dados é balanceado dentre as classes, então a acurácia e a acurácia balanceada assumirão valores muito similares. Sendo assim, a grande diferença entre as métricas seria que a acurácia balanceada considera que todas as classes possuem a mesma importância no modelo, mesmo aquelas em que a amostragem é menor.

Precisão

A precisão possui uma certa diferença semântica, quando comparada com a acurácia. Essa métrica, calculada com base na classe k, define a proporção de entradas que o modelo prediz serem da classe k e acerta, de fato.

Portanto, a definição da precisão para a classe k seria:

$$Precisão_k = \frac{VP_k}{VP_k + FP_k}$$

Com a definição da precisão por classe, torna-se possível definir o que seria uma precisão geral do modelo. Há diversas estratégias para condensar a precisão de todas as classes, uma delas seria pela média ponderada:

$$Precisão\_Ponderada = \frac{\sum_{k=1}^{N} Precisão_k * (VP_k + FN_k)}{\sum_{k=1}^{N} VP_k + FN_k}$$

Para isso, o fator utilizado para ponderar as precisões das diferentes classes é a ocorrência de cada classe no conjunto de dados, levando em conta a classificação real.

Recall

Bastante similar à precisão, o *recall* do modelo para a classe k, por sua vez, representa sua acurácia considerando apenas os dados que foram classificados como pertencentes à classe k:

$$Recall_k = \frac{VP_k}{VP_k + FN_k}$$

É importante notar que as definições de *recall* e acurácia para um problema com múltiplas classes são iguais.

Novamente utilizando a mesma estratégia da precisão, para agrupar o *recall* de todas as classes e adquirir uma visão geral do modelo nesse quesito, a média ponderada representa uma estratégia sólida:

$$Recall\_Ponderado = \frac{\sum_{k=1}^{N} Recall_k * (VP_k + FN_k)}{\sum_{k=1}^{N} VP_k + FN_k}$$

Pontuação F1

A pontuação F1 é uma agregação das duas métricas apresentadas acima, ou seja, precisão e *recall*. As métricas são agrupadas seguindo o conceito da média harmônica. A média harmônica é bastante indicada quando os valores manipulados representam grandezas inversamente proporcionais, e sua definição (Hayes, 2020) para uma coleção de N elementos, como $x_1, x_2, ..., x_N$, é como segue:

$$\frac{N}{\sum_{n=1}^{N} \frac{1}{x_n}}$$

À vista disso, considerando que a pontuação F1 simboliza a média harmônica entre precisão e *recall*, seu valor para uma certa classe k é:

$$F1_k = 2 * \frac{Precisão_k * Recall_k}{Precisão_k + Recall_k}$$

Para ponderar a pontuação F1 de acordo com a ocorrência de cada classe no conjunto de treinamento, deve-se seguir a definição:

$$F1\_Ponderada = \frac{\sum_{k=1}^{N} F1_k * (VP_k + FN_k)}{\sum_{k=1}^{N} VP_k + FN_k}$$

Coeficiente de Correlação Matthews

O Coeficiente de Correlação de Matthews (MCC, do inglês *Matthews Correlation Coefficient*) foi desenvolvido por Brian W. Matthews e tornou-se essencial para medir a performance de modelos que classifiquem dentre múltiplos rótulos. Diferentemente das métricas anteriores, esse coeficiente flutua no intervalo [-1, 1], e quanto maior seu valor, melhor a habilidade de predição do modelo. Para dizer mais precisamente, quando o coeficiente assume valor 1, então a predição possui uma forte correlação com o verdadeiro rótulo das entradas. Por outro lado, quando o valor assumido é 0, não há nenhuma correlação entre as duas variáveis, ou seja, o modelo está realizando suas predições de forma aleatória. Quando o valor é negativo, então há uma correlação inversa.

Para calcular o coeficiente de correlação de Matthews:

$$CCM = \frac{acertos * total - \sum_{k=1}^{N} p_k * o_k}{\sqrt{(total^2 - \sum_{k=1}^{N} p_k^2) * (total^2 - \sum_{k=1}^{N} o_k^2)}}$$

- $acertos = \sum_{k=1}^{N} VP_k$, representa a quantidade de acertos do modelo;
- $total = \sum_{k=1}^{N} VP_k + FN_k$, representa a quantidade total de predições;
- $p_k = VP_k + FP_k$, representa a quantidade total de vezes em que a predição do modelo apontou para a classe k; e
- $o_k = VP_k + FN_k$, representa a quantidade total de ocorrências reais da classe k.

Coeficiente Kappa de Cohen

O coeficiente Kappa de Cohen, foi desenvolvido por Cohen em 1960. Apesar do coeficiente representar o nível de concordância entre dois classificadores, uma outra interepretação será dada à metrica.

Índice de concordância entre classificadores é difícil de ser interpretado (Jr e Millones, 2011), então o Kappa pode ser adotado para avaliar o quão melhor o classificador é em comparação com um modelo de referência (Widmann, 2020).

Um modelo de referência[9] é utilizado como ponto de referência para ser comparado com a performance do classificador em desenvolvimento. No caso, pode ser um classificador que prediz dados considerando a distribuição das classes de predição, ou seja, a acurácia esperada que é adotada como referência padrão para o cálculo do Kappa neste trabalho, ou a acurácia obtida por uma regressão logística, por exemplo.

Em outras palavras, o Kappa de Cohen mede o quão dependente é a distribuição de predição do modelo com a verdadeira distribuição das categorias do modelo.

Para problemas com classes desbalanceadas, o coeficiente torna-se ainda mais relevante, uma vez que distribuição de dados difere muito entre as classes. Com isso, a avaliação do modelo por meio do Kappa de Cohen adquire um significado mais importante.

[9] https://developers.google.com/machine-learning/glossary#baseline

A métrica, chamada abaixo de $Kappa$, para um classificador dentre múltiplos rótulos segue como:

$$Kappa = \frac{acertos * total - \sum_{k=1}^{N} p_k * o_k}{total^2 - \sum_{k=1}^{N} p_k * o_k}$$

A definição das variáveis da fórmula acima segue a descrição provida para a métrica anterior.

Quanto mais o valor de $Kappa$ se aproxima de 1, mais próximo do valor máximo está a métrica que se deseja aprimorar e melhor é o modelo analisado em relação ao modelo de referência. Por outro lado, valores negativos para o $Kappa$ indicam o quão pior é o resultado do modelo analisado em relação ao modelo de referência.

Capítulo 4

Desenvolvimento

Durante essa seção, as etapas de desenvolvimento serão descritas. Caso seja necessário, utilize o repositório[1] como referência.

4.1 Ferramentas

A seguir, serão apresentadas as ferramentas que foram essenciais para o desenvolvimento do projeto, bem como a forma como elas foram utilizadas.

4.1.1 PyTorch

O PyTorch[2] é uma biblioteca de código livre para a linguagem Python desenvolvida pelo time de Pesquisa em Inteligência Artificial do Facebook (Paszke *et al.*, 2019). Ela facilita a construção e a elaboração de projetos de aprendizagem de máquina por ser otimizada para o trabalho com tensores[3].

No desenvolvimento do projeto, foram utilizados métodos e classes utilitários do PyTorch para alimentação dos dados ao modelo e, principalmente, para efetuar a manipulação do modelo para treinamento dentro de um ambiente de GPU[4]. Em especial, o Pytorch provê a classe chamada *Dataset* que é responsável por definir rotinas de recuperação de elementos dentro de um conjunto de dados. Dessa forma, métodos de treinamento implementados com base nas ferramentas fornecidas pelo Pytorch são programados para lidarem com Pytorch *Datasets*.

Embora seja possível, a definição do modelo não foi completamente estruturada utilizando o PyTorch. O PyTorch *Lightning* desempenhou um papel importante no processo

[1] https://github.com/yurickyh/JudgementClassifier

[2] https://pytorch.org/

[3] Um tensor é simplesmente uma representação de um vetor n-dimensional que pode ser aplicado à resolução de diversos problemas.

[4] Do inglês *Graphics Processing Unit*, corresponde a uma unidade de processamento gráfico com enorme poder de computação paralela. Inicialmente com um propósito voltado à renderização de vídeos, a GPU possui aplicação na resolução de diversos problemas, como treinamento de redes neurais.

também.

O PyTorch *Lightning*[5] é um arcabouço que fornece uma interface de alto nível para o PyTorch. Ele estrutura o código do PyTorch para que seja possível abstrair os detalhes do treinamento, o que permite uma maior escalabilidade dos modelos de aprendizagem profunda e torna os experimentos mais fáceis de entender e reproduzir. Assim, a classe do modelo de classificação de acórdãos foi definido utilizando o *LightningModule*, um módulo do PyTorch *Lightning* que organiza o código de treinamento para melhor visualização e para uso da *Trainer API*[6] do HuggingFace, que será apresentada em seguida.

4.1.2 HuggingFace

A HuggingFace[7] é uma companhia *startup* que detém uma comunidade de código livre focada em tecnologias de processamento de linguagem natural. Por meio da sua principal biblioteca, *Transformers*, ela disponibiliza uma série de implementações de modelos pré-treinados e ferramentas que permitem a modelagem e execução de tarefas de aprendizagem de máquina. No desenvolvimento do projeto foram utilizados vários desses recursos, dos quais alguns serão listados abaixo:

- *BertForSequenceClassification*: classe em Python que oferece a implementação do modelo BERT em tarefas de classificação de textos. Corresponde à base do modelo do classificador de acórdãos desenvolvido no projeto. No momento da instanciação dessa classe que ocorre a escolha da utilização do BERTimbau;
- *BertTokenizerFast*: o *tokenizador* que prepara os dados de entrada para que eles possam ser interpretados pelo modelo BERT. Todos os dados, tanto de treinamento quanto de validação e teste, são processados por ele antes de serem direcionados ao modelo;
- AdamW: otimizador que implementa a solução AdamW, descrita na seção 3.5.2; e
- *Trainer API*: recurso que provê soluções nativas de definição e customização do processo de treinamento. Alguns passos do processo comum de treinamento não precisam ser implementados manualmente, uma vez que o *Trainer API* pode ser anexado ao projeto. Utilidades como definição de *checkpoints* e escolha de estratégia de avaliação e métricas durante o treinamento são disponibilizadas por esse recurso.

4.1.3 Python

O projeto, em termos de código, foi essencialmente desenvolvido com a linguagem Python. Alguns pontos favoreceram a escolha da linguagem, principalmente a possibilidade de escrever códigos concisos, levando a um foco maior na lógica e pesquisa do problema. Ademais, o Python facilita o acesso a diversas bibliotecas e arcabouços especializados na construção de projetos de inteligência artificial.

---

[5] https://www.pytorchlightning.ai/

[6] Do inglês *Application Programming Interface*, é um conjunto de funções e procedimentos que agem como um *software* intermediário, permitindo que uma aplicação transmita dados e se comunique com outra.

[7] https://huggingface.co/

Além do Pytorch e da biblioteca *Transformers* já citados acima, o projeto também utilizou pacotes como *NumPy*[8] para manipulação de números, *pandas*[9] para análise e manuseio dos dados e *scikit-learn*[10] para avaliação de métricas e resultados.

4.1.4 Google Colab e Gooogle Drive

O Google Colab[11] é um produto da divisão de pesquisa da Google que permite o acesso pelo navegador a um ambiente de Jupyter *Notebooks*[12] para execução de códigos arbitrários em Python. Além disso, o Colab permite a conexão dos projetos ao Google Drive do usuário e a um ambiente de GPU sem custos para agilizar a realização de tarefas de aprendizagem de máquina e análise de dados.

Já o Google Drive[13] é o ambiente de armazenamento de arquivos em nuvem da Google que permite aos usuários, além de outras funcionalidades, o compartilhamento potencializado de arquivos e diretórios graças ao seu serviço de sincronização em tempo real.

Assim, o uso do Google Colab e do Google Drive foi conveniente no desenvolvimento do projeto. Com isso, foi possível executar várias rotinas de construção e treinamento do classificador, bem como manipulação e análise dos dados em um tempo menor possível, utilizando o ambiente de GPU. Portanto, a grande vantagem do uso da ferramenta é usufruir das facilidades de possuir um ambiente para execução agnóstico em termos de máquina e um sistema rápido para compartilhamento das planilhas de dados, dos Jupyter *Notebooks* e dos arquivos entre os participantes do projeto.

4.2 Conjunto de dados

Há vários componentes que são muito importantes no desenvolvimento de projetos de aprendizagem de máquina e, como já mencionado anteriormente, um dos mais fundamentais deles é o conjunto de dados. Ele é a espinha dorsal de um projeto de inteligência artificial, pois é ele que permite que os algoritmos aprendam e melhorem continuamente para resolver os diversos problemas inseridos em várias áreas. Na ausência dos dados, a tarefa de treinar e avaliar modelos torna-se mais complexa.

O conjunto de dados final utilizado no projeto consiste de uma planilha com 5524 registros e cada um deles apresenta informações referentes a um acórdão do STF. Essas informações são:

- Código do acórdão;
- Ramo do Direito ao qual o acórdão referente pertence;

[8] https://numpy.org/

[9] https://pandas.pydata.org/

[10] https://scikit-learn.org/

[11] https://colab.research.google.com/

[12] Arquivos especializados em executar código em Python de forma interativa.

[13] https://www.google.com/drive/

- Cabeçalho do acórdão;
- Ementa do acórdão;
- Texto com a decisão do relator do acórdão;
- Conjunto de palavras que foram julgadas pela jurisprudência como relacionadas ao acórdão;
- Sinalização que indica se foi possível inferir o ramo do Direito ao qual o acórdão pertence a partir apenas da ementa dele;
- Sinalização que indica se houve indicação explícita do ramo do Direito ao qual o acórdão pertence na ementa ou no voto do relator; e
- Expressões ou palavras essenciais que aparecem na ementa e que são pertinentes para a inferência do ramo do Direito a que o acórdão pertence.

Uma amostra do conjunto de dados pode ser visualizada abaixo na figura 4.1.

	cod_acordao	ramo	tipo_acordao	cabecalho	ementa	decisao	indexacao	somente_ementa	indicacao_exclusiva	expressoes_chave
2470	MS 25697	1	MS	MS 25697 / DF - DIS	EMENTA: ADMINIST	O Tribunal, por unar	['AUSÊNCIA', 'DECAI	Sim	Sim	administrativo
2106	ADI 2810	1	ADI	ADI 2810 / RS - RIO	Ementa: Processo cc	O Tribunal, por unar	['OCORRÊNCIA', 'CA	Sim	Não	servidores públicos
3255	ADI 1175	1	ADI	ADI 1175 / DF - DIST	TRIBUNAL DE CONT	Após os votos dos Se	['CONSTITUCIONAL	Sim	Sim	Tribunal de contas
3956	HC 94398	0	HC	HC 94398 / RJ - RIO	EMENTA: HABEAS (	A Turma, à unanimic	['VIDE EMENTA']	Sim	Sim	habeas corpus
4731	RHC 96093	10	RHC	RHC 96093 / PA - PA	EMENTA: RECURSC	A Turma, à unanimic	['VIDE EMENTA']	Não	Não	inquérito policial mil
4271	ADI 3035	1	ADI	ADI 3035 / PR - PAR	EMENTA: Ação Dire	O Tribunal, por unar	['DECLARAÇÃO', 'INI	Sim	Não	competência privati
1097	ADI 2536	4	ADI	ADI 2536 / DF - DIST	EMENTA: AÇÃO DIF	O Tribunal, por unar	['VIDE EMENTA']	Não	Sim	Previdência Social

Figura 4.1: *Trecho do conjunto de dados utilizado no projeto.*

Esse conjunto de dados final foi separado em outros três conjuntos de dados menores, um de treinamento, um de validação e um de teste, para que fosse possível suprir as necessidades do desenvolvimento do projeto. Logo, tanto o conjunto de teste, quanto de validação, contêm 829 registros. Enquanto isso, o conjunto de treinamento carrega o restante, resultando em 3866 registros.

A obtenção do conjunto de dados só foi possível devido à colaboração da Faculdade de Direito da Universidade de São Paulo, especialmente do Prof. Juliano Maranhão. O professor foi responsável por garantir a colaboração de alguns estudantes. Os alunos analisaram acórdãos disponíveis no portal de Jurisprudência do STF e preencheram os resultados em um formulário que armazenava respostas para uma série de questionamentos sobre os acórdãos. As perguntas tinham o foco nas informações do acórdão e sua estrutura, como a classificação dentre os ramos do Direito, a possibilidade de inferência do ramo a partir da ementa e as normas constitucionais que são citadas nele. Assim, a partir dessa planilha preenchida com a orientação do Prof. Maranhão, foi possível separar e processar os dados relevantes para o projeto e construir o conjunto de dados final que foi apresentado anteriormente na seção.

4.3 Análise dos dados

Devido à extrema importância dos dados, é fundamental que eles sejam analisados. O processo de análise dos dados auxilia para que se tenha um processamento mais efetivo, além de facilitar o entendimento do escopo do problema a ser resolvido.

4.3.1 Análise descritiva

A análise descritiva[14] foi feita com o intuito de obter informações relevantes a partir dos dados que poderiam ajudar a chegar em opções e escolhas para solucionar o problema de classificação de acórdãos. Como consequência, a análise descritiva também colaborou para interpretar os resultados do projeto e gerar determinadas conclusões.

Foram feitas quatro análises:

- A primeira análise observou a distribuição dos dados classificados dentre os ramos do Direito. Fazendo uma contagem na coluna referente ao ramo do Direito do acórdão, os resultados apresentados na figura 4.2 foram obtidos. Neles, é possível observar que mais de 69% dos dados estão classificados entre apenas dois dos treze ramos que foram considerados no projeto, o que demonstra um grande desbalanceamento no conjunto de dados. No entanto, os acórdãos do conjunto de dados foram escolhidos ao acaso, indicando que o desbalancemento pode não ser resultado de uma má amostragem, mas sim uma característica real dos acórdãos do STF;

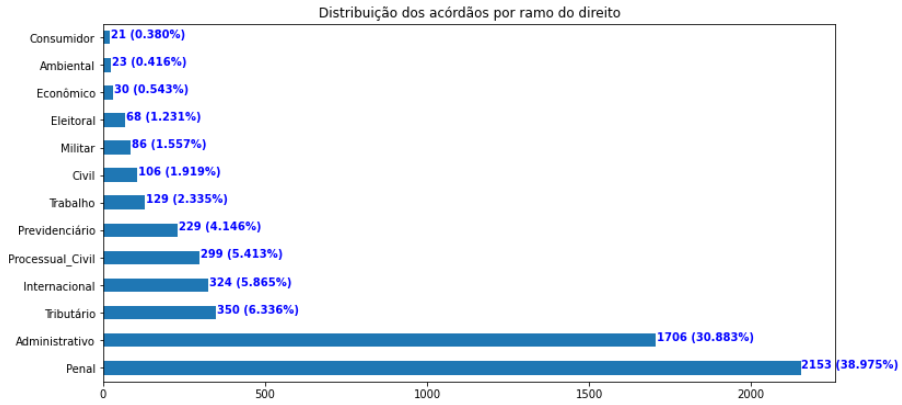


Figura 4.2: *Gráfico que mostra a distribuição dos dados classificados dentre os ramos do Direito.*

- A segunda análise calculou o nível de informação sobre o ramo do Direito presente nas ementas dos acórdãos. A figura 4.3 foi gerada por meio dos atributos que descreviam a possibilidade de inferência do ramo do Direito, a partir da ementa. Nela é possível observar que em torno em 85% dos acórdãos, um estudante de Direito poderia classificar o documento apenas pela análise da ementa. Isso indica que em grande parte dos acórdãos, existe uma relação forte entre os ramos do Direito e as ementas dos acórdãos;
- A terceira análise contabilizou os *tokens* mais frequentes nas expressões essenciais dos acórdãos. Observando a coluna do conjunto de dados que continha as expressões essenciais que ajudaram a classificar o acórdão para o seu respectivo ramo do Direito, foi feito um processamento dos registros, aplicando a *tokenização* do BERTimbau em cada expressão e agrupando os *tokens* para que fosse possível fazer uma contagem das frequências. O resultado obtido é apresentado na figura 4.4.

[14] O Jupyter *Notebook* com o código referente à analise descritiva pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Descriptive_Data_Analysis.ipynb.

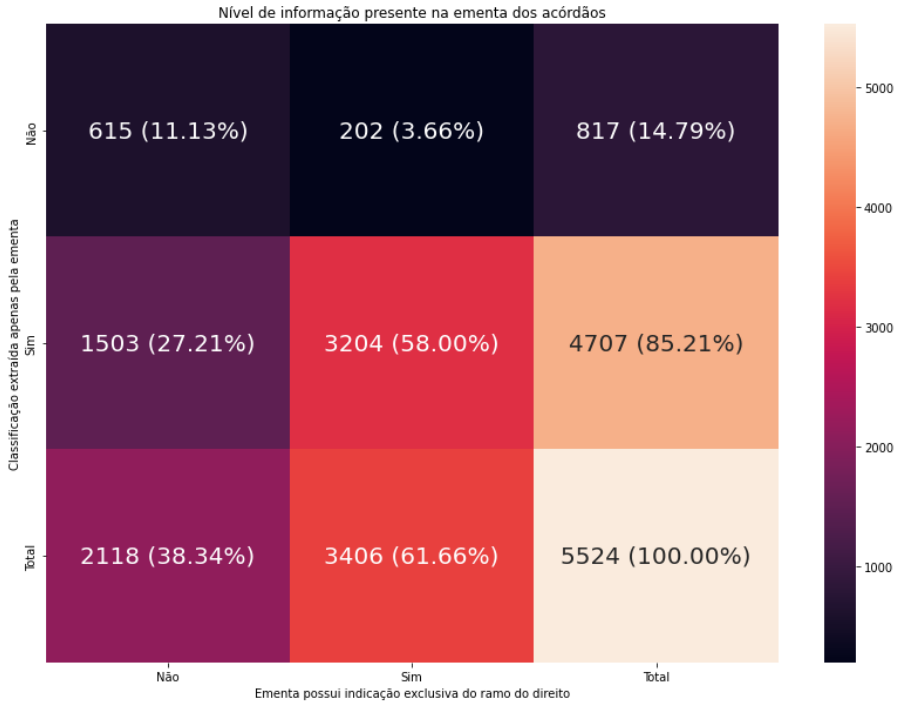


Figura 4.3: *Gráfico de calor que mostra o nível de informação do ramo do Direito presente na ementa dos acórdãos.*

Na figura é possível observar que os dois tokens com maiores frequências são os tokens “de” e “penal”. O primeiro não traz muita informação sobre os dados pois se trata em grande parte da preposição “de”, que possui a função mais focada em conectar as palavras de outras classes gramaticais. Já o segundo consegue refletir um pouco do desbalanceamento dos dados já citados anteriormente pois o *token* “penal” se encontra presente em sua maior parte nos registros classificados como do ramo do Direito penal, e tendo ele como um dos mais frequentes demonstra o quão predominante os acórdãos do Direito penal compõem o conjunto de dados; e

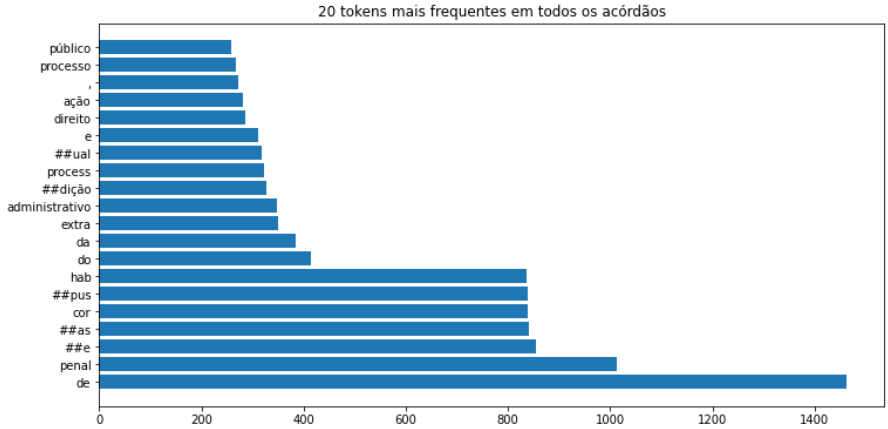


Figura 4.4: *Gráfico que mostra os vinte tokens mais frequentes na coluna de expressões essenciais dos acórdãos.*

- A última análise feita contabilizou os termos mais comuns nas ementas dos acórdãos. Da mesma forma que na análise anterior, foi observada a coluna de expressões essenciais e feita uma contagem de frequência dos termos, mas dessa vez sem a aplicação do processo de *tokenização*. Essa diferença na metodologia foi feita para que fosse possível analisar também a frequência dos valores originais dos termos, já que com a aplicação do processo de *tokenização*, algumas palavras diferentes são reduzidas aos mesmos *tokens*. O resultado obtido é apresentado na figura 4.5.

  Na figura é possível observar novamente a presença da palavra "penal" entre os dois termos mais frequentes, indicando novamente a predominância de registros do ramo do Direito penal. Além disso, também é possível observar que a expressão "habeas corpus" foi aquela mais frequente, o que reforça ainda mais a marca causada pelo desbalanceamento dos dados, já que a expressão também é predominante em acórdãos do Direito penal.

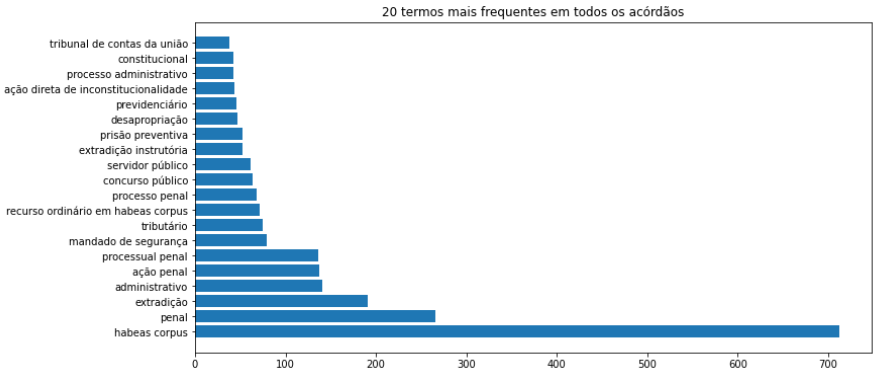


Figura 4.5: *Gráfico que mostra as vinte expressões mais frequentes na coluna de expressões essenciais dos acórdãos.*

Para tentar contornar a questão do desequilíbrio dos dados e obter informações relevantes aos outros ramos do Direito também, foi realizada a mesma análise de frequência dos termos mas separando os registros pela sua classificação dos ramos do Direito. Dois exemplos de resultados obtidos são apresentados na figura 4.6.

Observando os resultados dessa nova análise, é possível visualizar que em todos os ramos do Direito, com exceção do Direito civil, um termo específico fortemente relacionado ao ramo (como "extradição" para o ramo do Direito internacional) ou o próprio ramo (como "militar" para o ramo do Direito militar) aparecem como termos mais frequentes presentes nas ementas dos acórdãos. Esse resultado, aliado a outros obtidos anteriormente, pode dar a impressão de que nos acórdãos onde é possível deduzir o ramo do Direito pela ementa (que correspondem à maior parte dos casos), o trabalho de classificação é facilitado pois bastaria observar a presença desses termos relacionados ou da menção explícita do ramo.

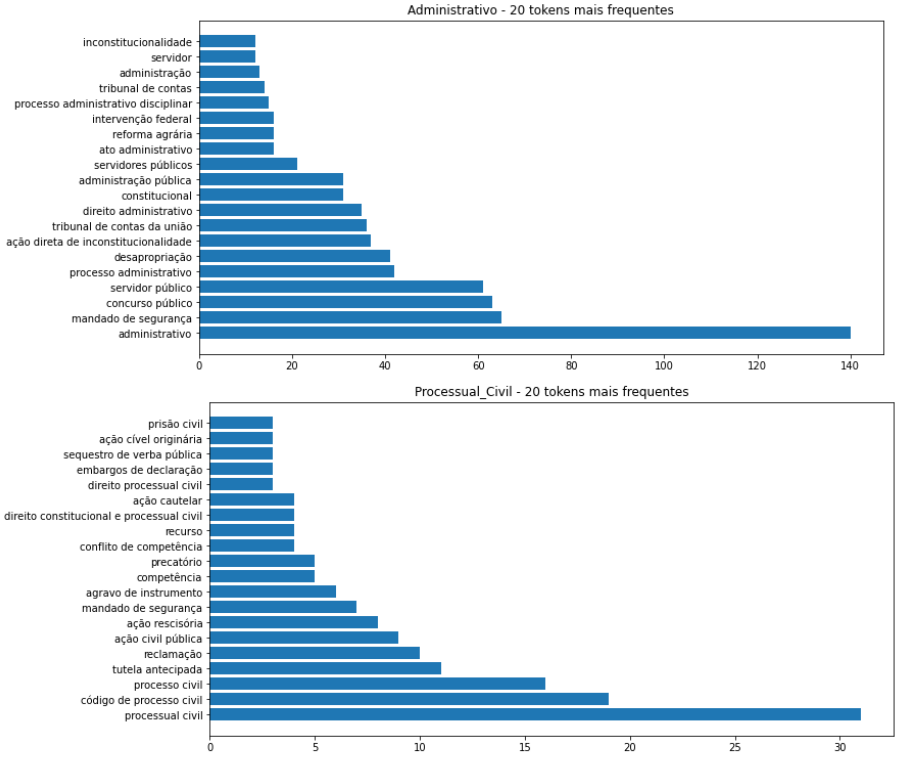


Figura 4.6: *Gráficos que mostram as vinte expressões mais frequentes para os registros dos ramos administrativo e processual civil, respectivamente.*

4.3.2 Pré-processamento dos dados

Após a análise descritiva, foi feito um simples pré-processamento[15] do conjunto de dados para que ele possa ser usado de forma universal na construção e treinamento do modelo do projeto e na avaliação das métricas.

O pré-processamento conteve duas etapas:

- A primeira etapa constituiu-se de um mapeamento. A resposta esperada pelo classificador seria o ramo do Direito ao qual o acórdão pertence. Dessa forma, padronizar essa informação resultaria em torná-la mais inteligível para o modelo BERTimbau. Portanto, foi criada uma Enum[16] para representar cada um dos ramos do Direito com quais o modelo iria trabalhar; e
- A segunda etapa consistiu de uma separação dos dados. Para que o modelo possa ser construído e treinado, são necessários conjuntos de dados de treinamento e de validação, enquanto que para que sejam feitas avaliações do modelo é necessário um conjunto de dados de teste. Assim, como já mencionado anteriormente, foi feita uma separação das informações em três conjuntos de dados: um de treinamento, um de validação e um de teste nas proporções de 70%, 15% e 15%, respectivamente.

[15] O Jupyter *Notebook* com o código referente ao pré-processamento de dados pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Data_Preprocessing.ipynb.

[16] Representação de enumerações em Python.

4.4 Treinamento do modelo

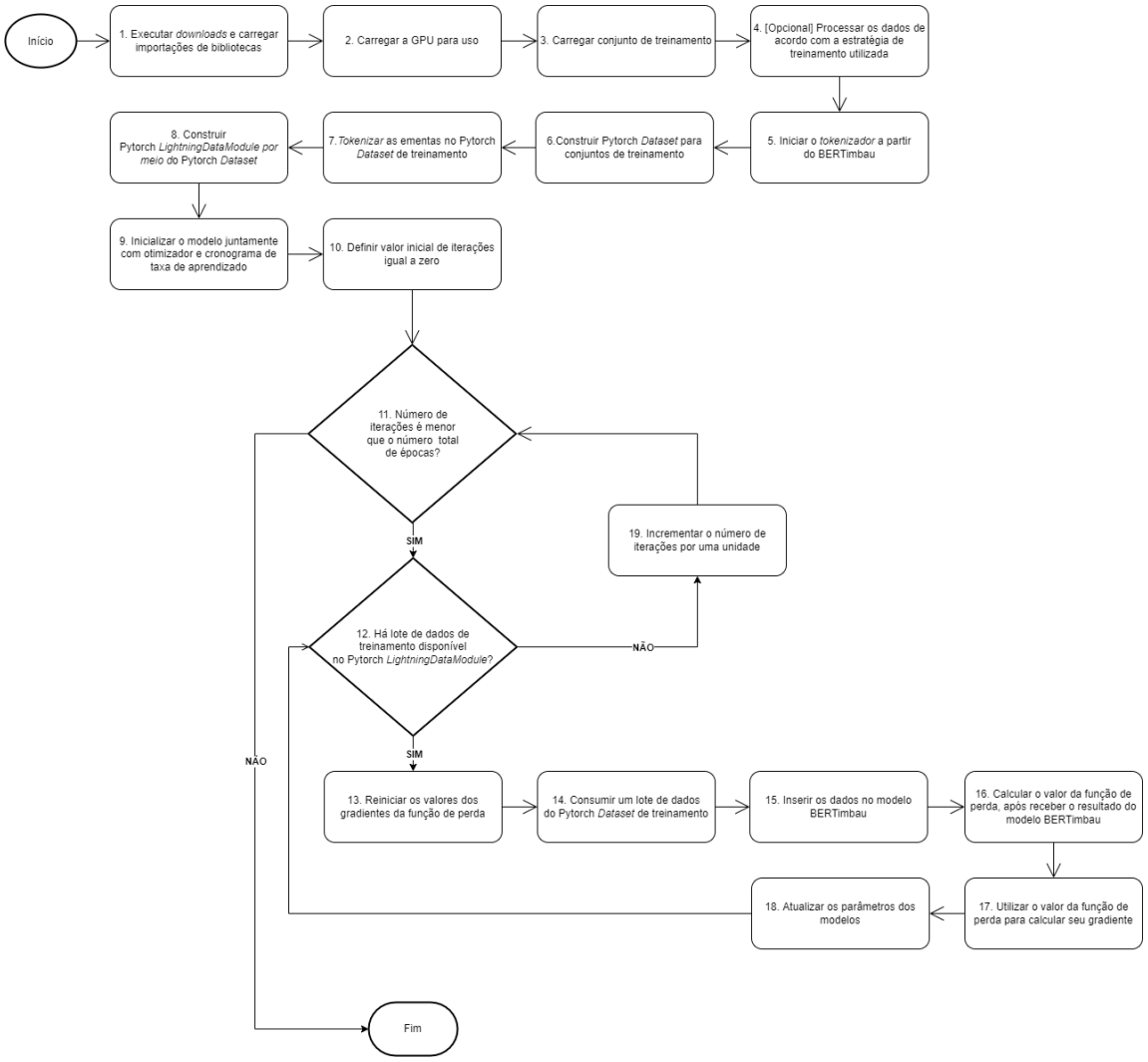


Figura 4.7: *Esquematização do processo inicial de treinamento de modelos.*

A figura 4.7 ilustra a forma como os modelos foram inicialmente treinados. A seguir, cada um dos passos serão descritos de forma mais detalhada. Os números que identificam os passos estão se referindo à imagem.

- Passo 1: Como o treinamento é executado por meio de Jupyter *Notebooks* hospedados no Google Colab, então a cada execução é preciso realizar o *download* das bibliotecas e ferramentas utilizadas, principalmente daquelas que pertencem a Pytorch e HuggingFace;
- Passo 2: Confirmação de que o *notebook* está sendo executado em um ambiente contendo a GPU. A GPU é fornecida gratuitamente pelo Google Colab;
- Passo 3: Os conjuntos de dados, já preparados para o modelo, foram armazenados de acordo com o procedimento citado na seção 4.3.2. Nesse passo, o conjunto de dados de treinamento é carregado para que o *notebook* possa utilizá-lo durante a execução;

- Passo 4: O dado deve ser processado e alterado para que a estratégia de treinamento possa ser seguida. As estratégias serão melhores definidas e explicadas na seção 4.5. Trata-se de um passo opcional, pois, a depender da estratégia escolhida, o dado pode ser colocado para treinamento sem a necessidade de processamento nesse momento;
- Passo 5: Realizar o *download* do *tokenizador* por meio das APIs do HuggingFace;
- Passo 6: Com o conjunto de dados em mãos, nesse passo as rotinas do Pytorch *Dataset* são definidas. Em específico, o Pytorch *Dataset* é definido de forma a retornar um mapeamento entre o ramo do Direito e o conteúdo da ementa, após ser *tokenizada*. Para esse passo, o Pytorch *Dataset* é definido, porém nenhuma instância ainda é construída;
- Passo 7: Enquanto o Pytorch *Dataset* é devidamente construído, o *tokenizador* é utilizado para *tokenização* das ementas contidas no conjunto de treinamento;
- Passo 8: Construção do Pytorch *LightningDataModule* a partir do Pytorch *Dataset* de treinamento com a definição dos lotes de dados, ou seja, a divisão do Pytorch *Dataset* em lotes;
- Passo 9: O modelo é habilitado para iniciar o treinamento. Da mesma forma, o cronograma de taxa de aprendizado e o otimizador são instanciados;
- Passo 10: A quantidade de iterações sobre o número de épocas é inicializada com valor zero;
- Passo 11: Verificação do número de iterações efetuadas. O limite é definido pelo número total de épocas previamente configurado;
- Passo 12: Verificação de lotes de dados. O Pytorch *LightningDataModule* divide o Pytorch *Dataset* de treinamento em lotes de dados, e o modelo irá consumi-los até que se esgotem na época corrente;
- Passo 13: Os gradientes devem ser reiniciados toda vez que um novo lote de dados irá ser consumido, uma vez que os valores da iteração anterior irão distorcer o cálculo de gradientes da iteração atual;
- Passo 14: Sabendo que há um lote de dados de treinamento disponível, ele será consumido do Pytorch *Dataset*, através do Pytorch *LightningDataModule*;
- Passo 15: Com os lotes de dados contendo as ementas já *tokenizadas* e devidamente associadas ao ramo do Direito ao qual seu acórdão pertence, eles são inseridos ao modelo BERTimbau;
- Passo 16: Depois da inserção dos dados no modelo, ele irá gerar as saídas, ou seja, as classificações para os dados de treinamento. Por meio dos resultados, a função de perda é calculada para comparar a saída do modelo com a classificação correta do acórdão em seu ramo do Direito;
- Passo 17: Com o valor da função de perda, é preciso calcular os gradientes para que os parâmetros do modelo possam ser ajustados;

- Passo 18: Depois de possuir os gradientes calculados, nesse passo o processo de treinamento executa os ajustes dos parâmetros por meio do processo de retropropagação; e
- Passo 19: O número de iterações completas é incrementado para que possa ser posteriormente comparado com o limite correspondente ao número total de épocas.

Para prosseguir com o procedimento de treinamento, o uso da *Trainer API* do Pytorch foi primordial, uma vez que os passos 9 a 19 são todos automatizados por meio da ferramenta.

Além disso, para essas etapas, o conjunto de validação não foi utilizado. Ele terá um melhor uso, no processo de *fine tuning*. Durante o processo de experimentação inicial, o modelo foi avaliado mediante o conjunto de testes.

4.5 Experimentação de modelo

Durante a construção de uma rede neural, diversas decisões devem ser tomadas. Em se tratando de modelos BERT, uma escolha essencial seria a estratégia para determinar a porção de texto que será utilizada como entrada do modelo. A importância da decisão existe em decorrência de uma limitação dos modelos BERT. De acordo com os próprios criadores da arquitetura BERT (Devlin *et al.*, 2019), a porção de texto utilizada como entrada dos modelos deve conter no máximo 512 *tokens*. Dessa forma, impõe-se um limite ponderado por performance para que o modelo BERT não se torne custoso.

Algumas estratégias são recomendadas por um estudo desenvolvido na Universidade de Fudan (Sun *et al.*, 2020). Quando os textos a serem introduzidos no modelo BERT possuem mais de 512 *tokens*, algumas opções são sugeridas para contornar a situação:

- Truncar o início do texto: considerar apenas o início da ementa do acórdão, ou seja, os 510 *tokens* iniciais;
- Truncar o fim do texto: considerar apenas o fim da ementa do acórdão, ou seja, os 510 *tokens* finais; e
- Mesclar início e fim do texto: considerar parcialmente o início da ementa do acórdão, assim como seu final.

As estratégias acima consideram apenas 510 *tokens*, pois os *tokens* especiais não devem ser contabilizados como parte do texto.

Para esse projeto, a segunda opção mostrada acima foi descartada. Essa decisão se deu por conta da pouca informação que a ementa de um acórdão carrega ao seu final. Por outro lado, o início da ementa é bastante rico e definitivamente agrega informação em abundância.

Com isso, para substituir o segundo método, foi utilizada uma abordagem em que a ementa do acórdão, quando excede o limite esperado, foi dividida em trechos contendo aproximadamente 512 *tokens*, considerando um trecho comum para interligar cada uma das porções.

Para que a comparação pudesse ser justa entre as estratégias analizadas, alguns parâmetros ficaram fixos ao longo dos modelos:

- Otimizador: AdamW com taxa de aprendizado igual a 3e−5 e fator de decaimento igual a 0.01;
- Número total de épocas: 3;
- Tamanho de cada lote de dados: 2 acórdãos, ou seja, cada lote de dados carrega a informação de 2 acórdãos, em especial a ementa e a classificação do ramo do Direito;
- Número de *tokens* considerados por ementa: 512 *tokens*, com a ressalva de que 2 *tokens* são reservados pelo BERT e ditos como especiais;
- Função de perda: Função de entropia cruzada, uma vez que é recomendada para tarefas de classificação;
- Cronograma de taxa de aprendizado: linear com um período de aquecimento; e
- Proporção de passos de aquecimento: 1%.

Sendo assim, para identificar a melhor estratégia, o procedimento utilizado foi definir um modelo para cada uma daquelas apresentadas anteriormente. O principal detalhe é a fixação dos parâmetros acima dentre os modelos. Dessa forma, cria-se uma forma justa de reconhecer a melhor estratégia.

4.5.1 Primeira opção - truncar início da ementa

O próprio *tokenizador*[17] provido pelo HuggingFace possui uma opção nativa, controlado por um atributo chamado *truncation* que define se o texto usado como entrada do *tokenizador* será automaticamente truncado de acordo com o limite de 512 *tokens*.

Com isso, para esse primeiro modelo, o passo 4, citado na seção 4.4, foi descartado. Isso significa que nenhum trabalho adicional foi desempenhado nesse passo, dado que o próprio *tokenizador* encarregou-se de truncar a ementa.

Um exemplo dessa estratégia[18] está ilustrado na figura 4.8.

4.5.2 Segunda opção - divisão da ementa em porções de textos

Como já explicado anteriormente, essa estratégia[19] divide a ementa em porções de textos contendo 512 *tokens*, quando ela excede esse limite. Para gerar essas porções de texto, a estratégia considera uma intersecção para garantir que todas as porções de texto tenham parte do contexto da porção anterior. Por exemplo, em uma ementa contendo 700

[17] https://huggingface.co/transformers/internal/tokenization_utils.html?highlight=encode_plus#transformers.tokenization_utils_base.PreTrainedTokenizerBase.encode_plus

[18] O Jupyter *Notebook* com o código referente à estratégia pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Model-Word_Truncated.ipynb.

[19] O Jupyter *Notebook* com o código referente à estratégia pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Model-Word_Batches.ipynb.

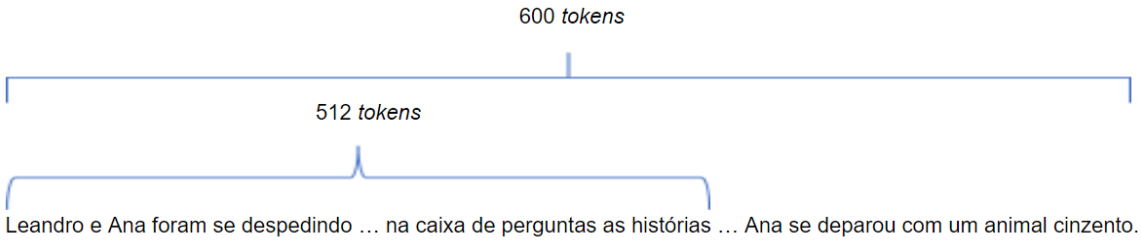


Figura 4.8: *Ilustração do funcionamento da estratégia de truncamento do início da ementa. Nesse exemplo, o texto possui 600 tokens, mas apenas os 512 tokens iniciais são selecionados.*

tokens, os primeiros 512 *tokens* são separados para formar a primeira porção de texto. A segunda porção de texto será construída a partir da concatenação de alguns *tokens* finais da primeira porção, com os 188 *tokens* restantes que não constam na primeira porção. O tamanho da intersecção pode ser de 100 *tokens*, por exemplo.

Dessa forma, a interseção de tokens entre uma porção e sua anterior garante que cada uma tenha certo contexto em relação ao trecho da ementa. Nesse caso, o tamanho da intersecção escolhido foi de 100 *tokens*, um valor considerado razoável durante o desenvolvimento da estratégia.

Portanto, no passo 4, citado na seção 4.4, foi implementada essa rotina de dividir a ementa em porções de textos de 512 *tokens*. É importante ressaltar que para a implementação, a divisão não foi realizada em termos de *tokens*, mas em palavras. Apesar da descrição do método acima ter tido foco nos *tokens*, a implementação teve foco nas palavras, por conta de dois motivos:

- As palavras abstraem de forma satisfatória o que o *tokenizador* irá considerar como *token*. Nem sempre uma palavra irá representar um *token*, mas essa é a situação mais comum; e
- Em uma outra abordagem, a implementação consistiria em inicialmente *tokenizar* a ementa, com tamanho maior que 512 *tokens*, para então dividi-la em porções menores. Porém, isso não é possível, dado que o *tokenizador* BERT não suporta processamento de textos mais extensos que o limite de 512 *tokens*.

Veja a figura 4.9 para uma ilustração de um exemplo.

4.5.3 Terceira opção - concatenação do início e fim da ementa

Nessa terceira opção, a estratégia[20] é considerar parte do início e do fim da ementa. Dessa forma, quando a ementa possui comprimento maior que 512 *tokens*, então metade dos *tokens* disponíveis são retirados da introdução da ementa e o restante do final dela.

O objetivo é garantir que o modelo adquira contexto de dois trechos fundamentais da ementa: introdução e conclusão. Citado na seção 4.4, o passo 4 abriga a implementação

[20] O Jupyter *Notebook* com o código referente à estratégia pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Model-Word_Front_Back.ipynb.

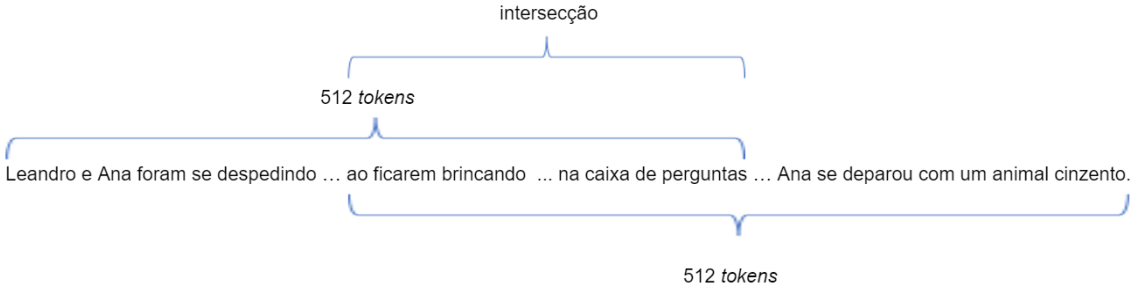


Figura 4.9: *Ilustração do funcionamento da estratégia de divisão da ementa em porções de texto. Nesse exemplo, o texto foi dividido em duas porções considerando uma intersecção que poderia ser de 100 tokens, no caso da estratégia descrita acima.*

de selecionar dentre os 512 *tokens* disponíveis, metade da introdução e o restante da conclusão. Novamente, a implementação se deu com base em palavras, ao invés de *tokens*, pelos mesmos motivos citados na estratégia anterior.

A figura 4.10 mostra um exemplo.

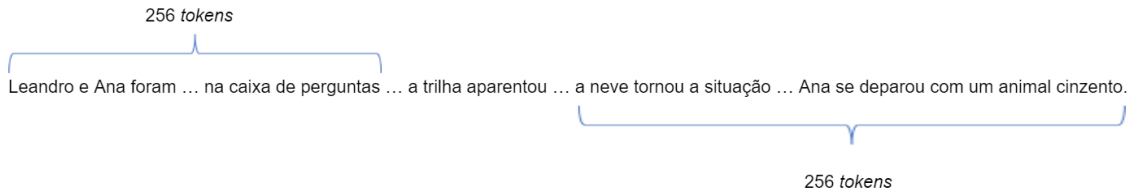


Figura 4.10: *Ilustração do funcionamento da estratégia de concatenação do início e fim da ementa. Nesse exemplo, foram extraídos 256 tokens do início e do fim do texto.*

4.5.4 Resultados e escolha da estratégia

Após a criação e treinamento dos modelos com as três diferentes estratégias, todos foram avaliados no conjunto de dados de teste[21]. Algumas métricas foram obtidas a partir dos resultados da avaliação para que uma escolha entre as opções de estratégia pudesse ser feita baseada nelas. Os valores das métricas obtidos podem ser visualizados na figura 4.11 e na tabela 4.1.

Examinando os valores das métricas é possível observar que os três modelos obtiveram resultados bem próximos, sendo que a maior diferença entre valores de uma mesma métrica não passou de 0.04. É possível observar também que a estratégia de divisão da ementa em porções de textos foi inferior às outras duas em todas as métricas, podendo assim ser descartada logo de início. Uma potencial justificativa para a inferioridade dessa estratégia pode ser em razão de que ao fazer a divisão da ementa, as porções referentes ao meio do texto podem ficar sem sentido e não contendo informações relevantes para distinguir entre os ramos do Direito, o que pode confundir o classificador e afetar a sua performance.

[21] O Jupyter *Notebook* com o código referente à avaliação pode ser encontrado em https://github.com/yurickyh/JudgementClassifier/blob/main/model/Metrics.ipynb.

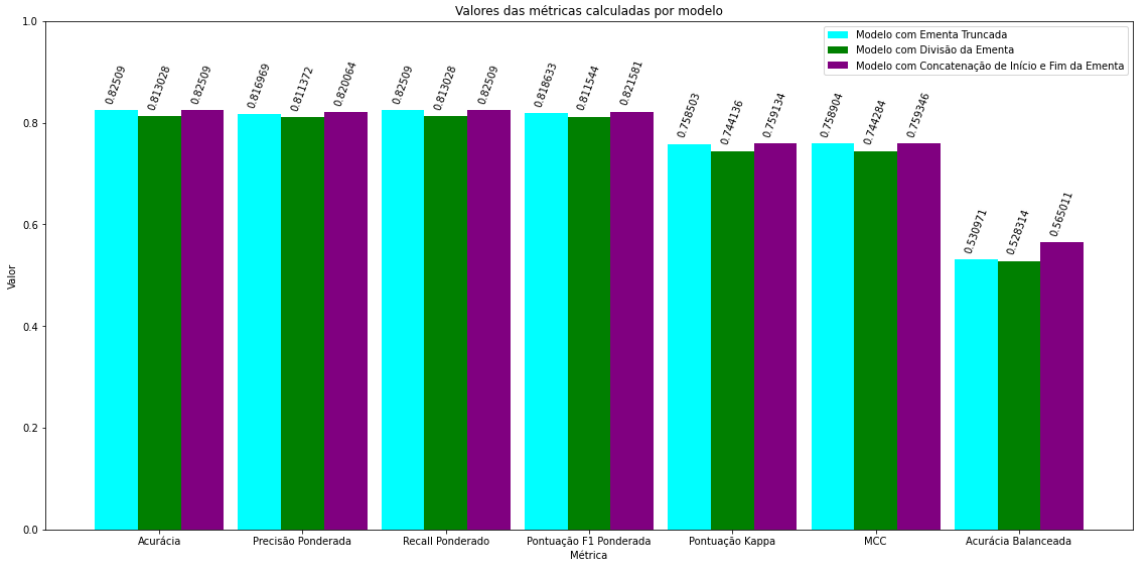


Figura 4.11: *Gráfico que mostra a relação dos valores obtidos das métricas para cada estratégia de construção de modelo.*

	Primeira estratégia	Segunda estratégia	Terceira estratégia
Acurácia	0.8250	0.8130	0.8250
Precisão ponderada	0.8169	0.8113	0.8200
Recall ponderado	0.8250	0.8130	0.8250
Pontuação F1 ponderada	0.8186	0.8115	0.8215
Pontuação Kappa	0.7585	0.7441	0.7591
MCC	0.7589	0.7442	0.7593
Acurácia balanceada	0.5309	0.5283	0.5650

Tabela 4.1: *Resultados da avaliação das três estratégias no conjunto de teste.*

Já comparando o modelo construído utilizando a estratégia de truncação da ementa com o modelo construído utilizando a estratégia de concatenação do início e fim da ementa, é possível visualizar que os resultados ficaram mais semelhantes, com o último tendo valores nas métricas minimamente melhores que o primeiro. Isso pode significar uma equivalência entre a qualidade dos modelos, mas ainda assim, uma escolha entre os dois modelos era necessária e foi feita baseando-se em algumas métricas principais. A razão pela escolha da análise dessas métricas específicas se deu devido à natureza do problema abordado no desenvolvimento do projeto. Como tratado na seção 4.3.1, os conjuntos de dados utilizados durante todas as fases do projeto possuem um grande desbalanceamento entre as classes. Assim, torna-se necessário levar em consideração essa circunstância na hora de avaliar e comparar os modelos construídos.

Nessa linha, um experimento feito por César Ferri, da Universidade Politécnica de Valencia (Ferri *et al.*, 2009), mostra que métricas como a pontuação F1 e a pontuação Kappa demonstram-se robustas ao avaliar a qualidade de modelos que possuem classes com uma porcentagem de frequência de elementos muito baixa em relação às outras. Além

disso, as implementações adaptadas de acurácia balanceada e do coeficiente de correlação de Matthews para atender a modelos de múltiplas classes também se revelam como boas métricas para avaliar modelos com classes de dados desbalanceadas (Grandini *et al.*, 2020).

Dado isso, observando os valores da acurácia balanceada, da pontuação F1 ponderada, da pontuação Kappa e do coeficiente de correlação de Matthews, foi possível visualizar que o modelo de concatenação de início e fim da ementa apresentou valores melhores que o modelo de truncação da ementa em todas as métricas específicas, com diferenças que são mostradas na figura 4.12 e resumidas na tabela 4.2, e portanto, ele foi escolhido como a estratégia mais adequada para dar continuidade ao desenvolvimento do projeto.

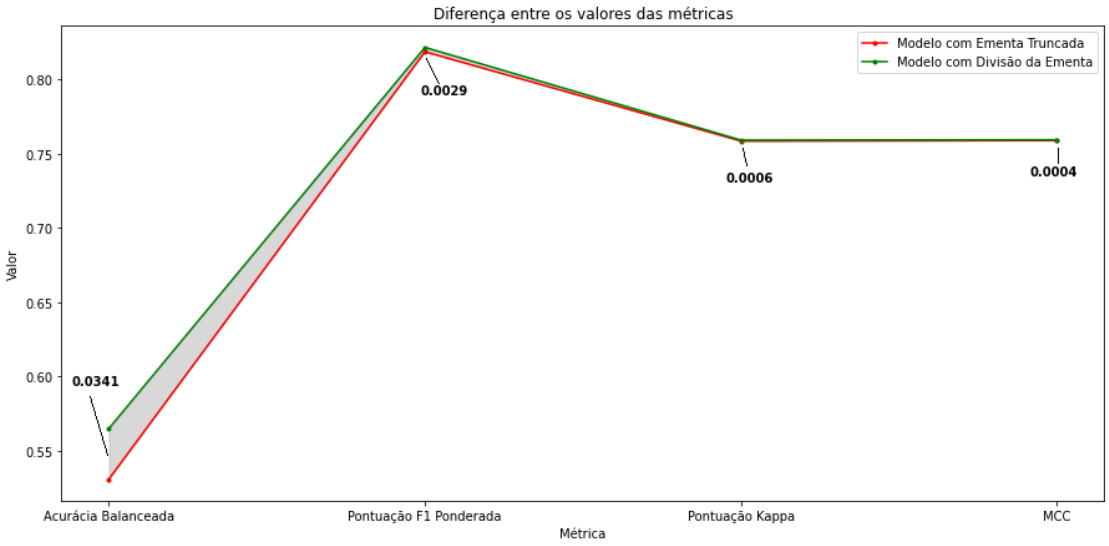


Figura 4.12: *Gráfico que mostra a diferença absoluta entre os valores obtidos das métricas específicas para o modelo com truncamento de ementa e o modelo com concatenação de início e fim da ementa.*

	Diferença absoluta entre os valores
Acurácia balanceada	0.0341
Pontuação F1 ponderada	0.0029
Pontuação Kappa	0.0006
MCC	0.0004

Tabela 4.2: *Diferenças dos valores das métricas do modelo construído com concatenação da ementa para o modelo construído com truncação da ementa.*

4.6 *Fine tuning*

O modelo a ser escolhido para investir no processo de ajustes finos é aquele que concatena o início e fim da ementa, discutido na seção 4.5.3.

4.6.1 Execução de experimentos

O objetivo do processo de ajustes finos (no inglês, *fine tuning*) é definir um conjunto de valores de hiperparâmetros que sejam ideais para extrair uma boa performance do modelo. O processo de *fine tuning* iniciou-se com a definição de um conjunto de valores para cada um dos hiperparâmetros, assim todas as combinações de valores foram construídas com base nesses conjuntos.

Os hiperparâmetros trabalhados foram:

- Número de palavras da introdução;
- Épocas;
- Taxa de aprendizado;
- Fator de decaimento; e
- Proporção de passos de aquecimento.

A quantidade de palavras a serem retiradas da introdução da ementa também define a quantidade de palavras que serão consideradas para a conclusão do texto. A todo momento que a ementa ultrapassar o limite de 512 palavras, então serão extraídas *n_introdução* palavras da introdução e (512 - *n_introdução*) palavras da conclusão. Para essa rotina de extração de palavras, assume-se que uma palavra representa satisfatoriamente o que seria um *token* para o modelo.

O número de épocas define a quantidade de iterações com que o processo de treinamento irá percorrer os conjuntos de dados de treinamento e validação. Para esse hiperparêmetro em específico, não é preciso definir um conjunto de valores, isso porque o conjunto de validação auxilia a definir qual a melhor quantidade de épocas para cada experimento.

A taxa de aprendizado define a quantidade de conhecimento que será extraída a cada iteração do processo de treinamento. Os valores escolhidos para serem testados são: 2e-5, 3e-5 e 5e-5. Os estudos desenvolvidos pela divisão de linguagens da Google apontaram que esses valores seriam ideais para os modelos BERT (Devlin *et al.*, 2019).

O fator de decaimento define o coeficiente com que ocorrerá a regularização dos pesos, seguindo o método de decaimento de pesos. Os valores a serem testados foram sugeridos por um estudo de treinamento do BERT: 0.001 e 0.01 (You *et al.*, 2020).

A proporção de aquecimento define quantas iterações da primeira época serão de aquecimento, do ponto de vista do otimizador. Os valores a serem testados são: 0.1 e 0.3.

Novamente, o processo de ajustes finos se deu muito similar ao esquema mostrado na seção 4.4, porém com algumas modificações, de acordo com a figura 4.13.

A maior modificação prática foi a utilização da *Trainer API* do próprio HuggingFace que permite um uso mais aprimorado de soluções providas nativamente. Um dos benefícios é deixar de usar o Pytorch *LightningDataModule* para tornar a rotina de treinamento mais simples, uma vez que o HuggingFace define um *DataCollator* automaticamente. No caso,

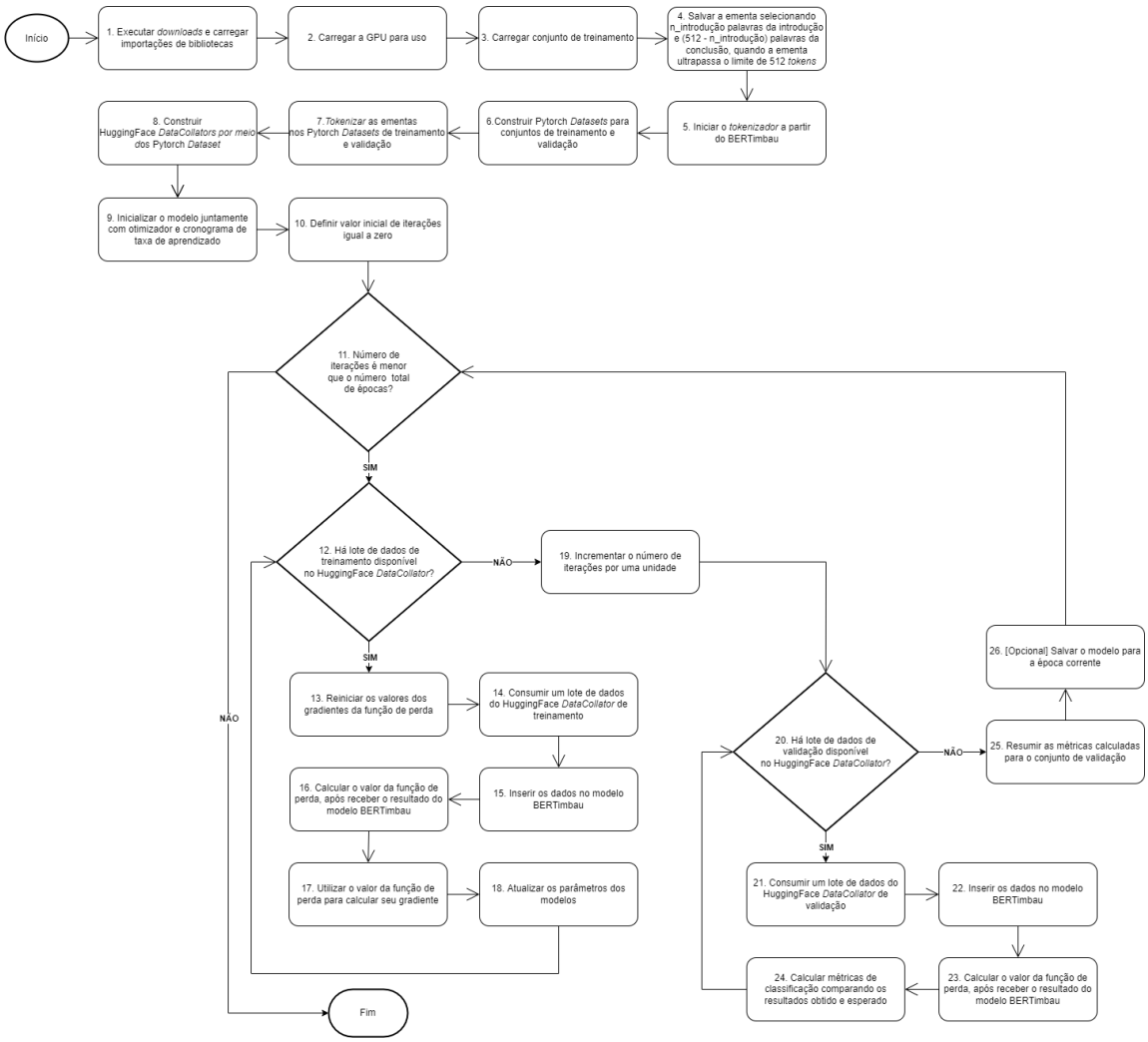


Figura 4.13: *Esquematização do processo de treinamento do modelo durante a execução do fine tuning.*

o *DataCollator* é análogo ao Pytorch *Dataset*, mas é uma classe específica que bibliotecas do HuggingFace manipulam.

Outro ponto, e primordial para a etapa de ajustes finos, foi a introdução de alguns passos responsáveis por efetivamente processar o conjunto de dados de validação. Como já dito previamente, esse conjunto será um grande indicativo da influência que os valores designados aos hiperparâmetros impõem na performance do modelo.

Por fim, o quarto passo demonstrado na imagem, mostra a rotina que manipula a ementa de acordo com a estratégia do modelo, quando seu tamanho ultrapassa o limite de 512 palavras. Esse passo depende do valor do hiperparâmetro *n_introdução*, equivalente ao primeiro item da lista acima.

Capítulo 5

Resultados

5.1 Resultados do processo de *fine tuning*

Os resultados podem ser vistos na íntegra por meio do Apêndice A. A primeira coluna da tabela identifica um cenário, ou seja, uma seleção de valores de hiperparâmetros. Essa coluna, aliada à coluna referente às épocas de treinamento, identifica um modelo que pode ser utilizado para a classificação de um acórdão dentre as classes do Direito.

Ainda pelo Apêndice A, é visível que escolher uma taxa de aprendizado correspondente a 5e−5 gera resultados insatisfatórios. O mesmo ocorre com os modelos do cenário 9, que também obtiveram métricas ruins em comparação ao restante.

Para visualizar as possíveis diferenças entre os modelos resultantes de cada cenário, foram construídos gráficos que ilustram cada uma das métricas:

- Valores da função de perda de treinamento na figura 5.1;
- Valores da função de perda de validação na figura 5.2;
- Acurácia de validação na figura 5.3;
- Acurácia balanceada de validação na figura 5.4;
- Precisão ponderada de validação na figura 5.5;
- *Recall* ponderado de validação na figura 5.6;
- Pontuação F1 ponderada de validação na figura 5.7;
- Coeficiente Kappa de validação na figura 5.8; e
- Coeficiente de correlação de Matthews de validação na figura 5.9.

Nos gráficos, apenas dois modelos são referenciados por cenário. O primeiro é o modelo do cenário gerado na época de treinamento que apresentou o menor resultado da função de perda no conjunto de validação e o segundo modelo representa aquele que obteve a melhor performance geral quanto às métricas de classificação analisadas, dentre todas as épocas de treinamento destacadas.

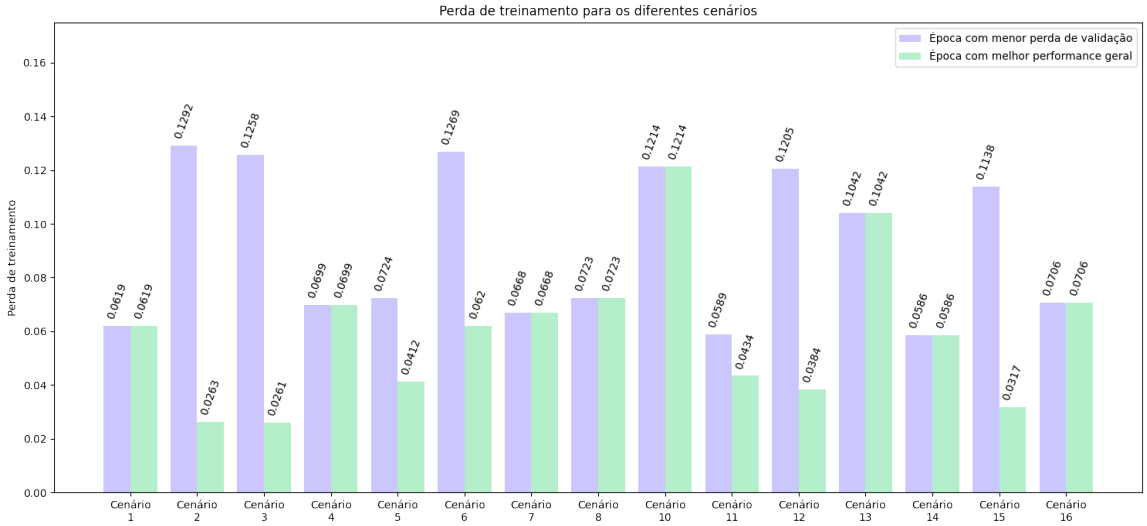


Figura 5.1: *Valores da função de perda de treinamento para os cenários selecionados.*

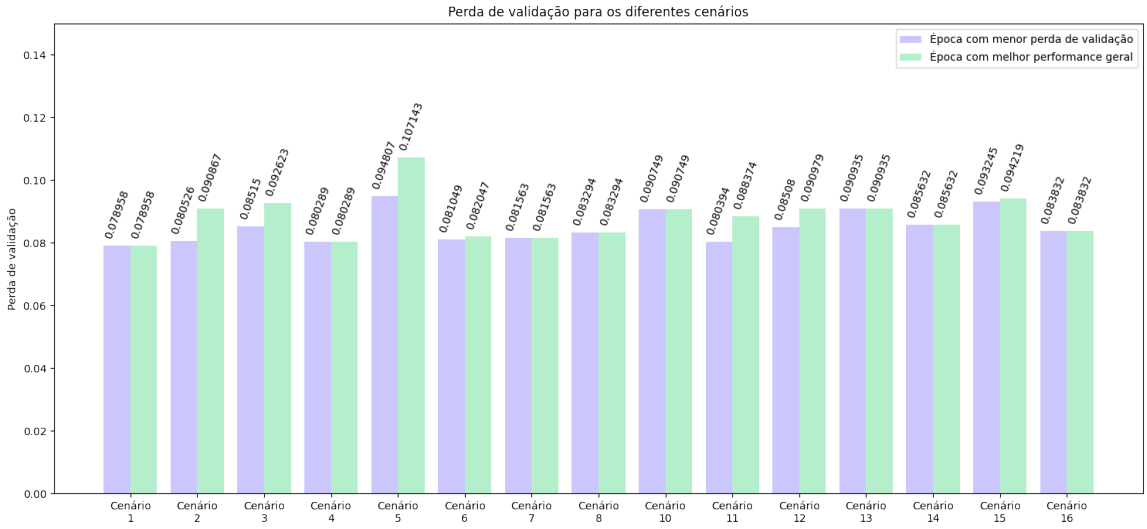


Figura 5.2: *Valores da função de perda de validação para os cenários selecionados.*

Pela análise dos gráficos, é possível perceber que os modelos dos diferentes cenários possuem uma performance bastante semelhante com a diferença de alguns centésimos nos valores das métricas, de forma geral. Ademais, a adoção de uma taxa de aprendizado menor aparentou ser mais benéfica ao classificador.

Para obter um critério que pudesse uniformizar a escolha do segundo modelo, referido acima, cada um dos cinco modelos de um cenário obteve uma pontuação associada. A pontuação de um modelo é calculada por meio da contagem da quantidade de métricas em que aquele modelo se destacou em relação aos restantes do cenário. As métricas utilizadas para esse critério foram as mesmas já citadas anteriormente na seção: acurácia de validação, acurácia balanceada de validação, precisão ponderada de validação, *recall* ponderado de validação, pontuação F1 ponderada de validação, coeficiente de Cohen Kappa de validação e coeficiente de correlação de Matthews de validação.

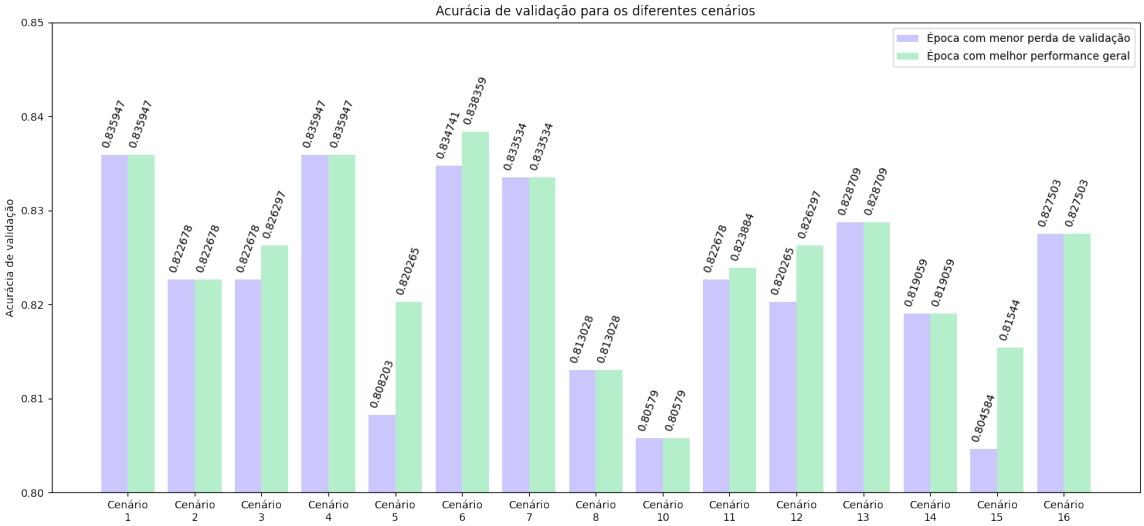


Figura 5.3: *Valores de acurácia de validação para os cenários selecionados.*

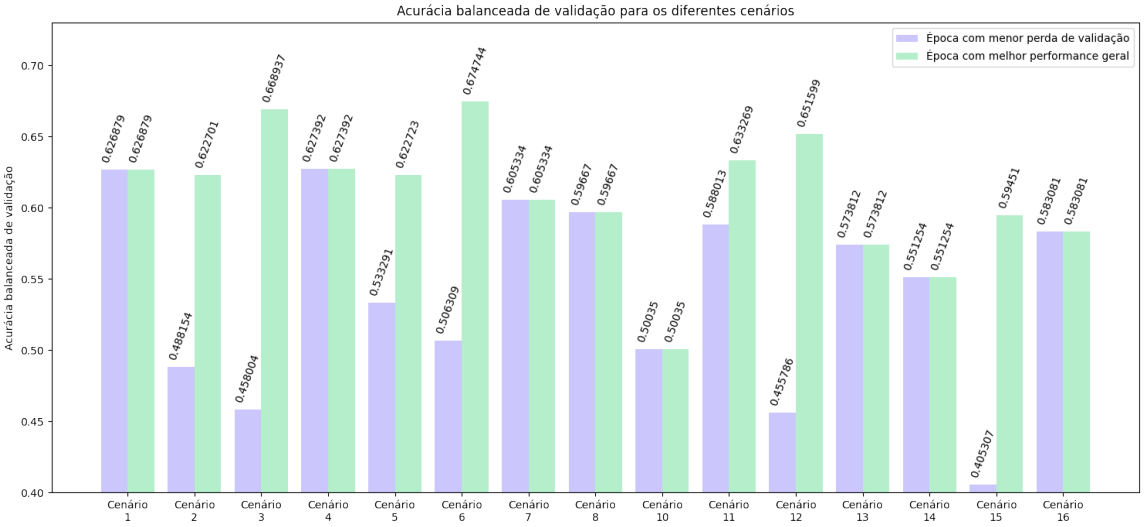


Figura 5.4: *Valores de acurácia balanceada de validação para os cenários selecionados.*

Quando dois modelos (ou mais) possuem a mesma pontuação, então prioriza-se aquele que foi selecionado como o modelo com menor perda de validação. Por exemplo, considere dois modelos A e B, em que o primeiro possui menor perda de validação, mas ambos possuem os mesmos valores de métricas de classificação. Então, o modelo A é escolhido como aquele que apresenta melhor performance geral também.

Em diversos casos, o modelo de menor perda de validação e o modelo de melhor performance geral nas métricas são os mesmos.

No Apêndice A, o modelo com menor perda de validação é apresentado pela linha roxa. Diferentemente, o modelo com melhor performance geral corresponde à linha verde. Quando ambos correspondem a um único modelo, então a linha da tabela é colorida com amarelo.

Sendo assim, de todas as linhas coloridas da tabela, apenas dois modelo foram selecio-

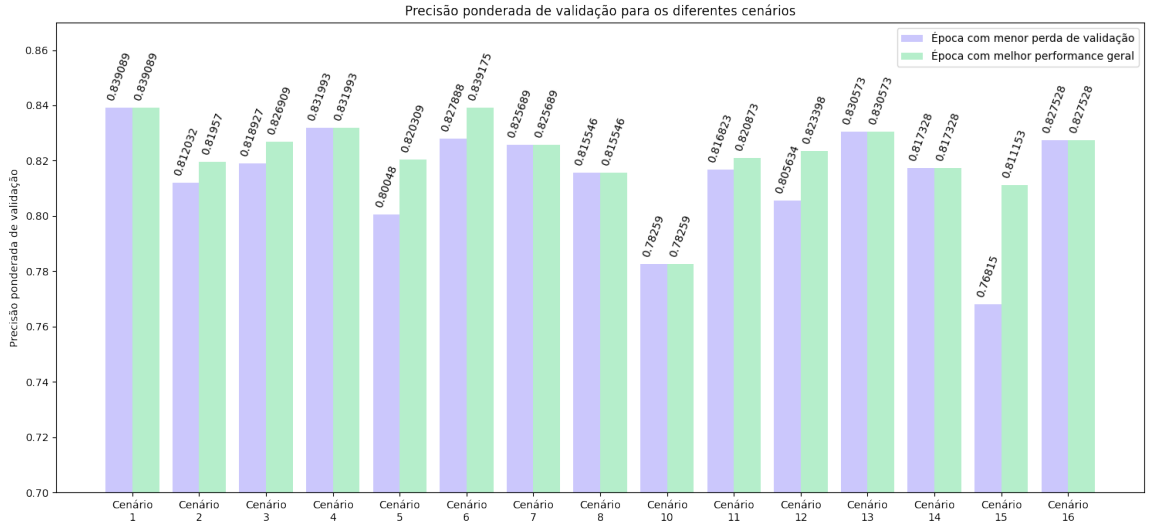


Figura 5.5: *Valores de precisão ponderada de validação para os cenários selecionados.*

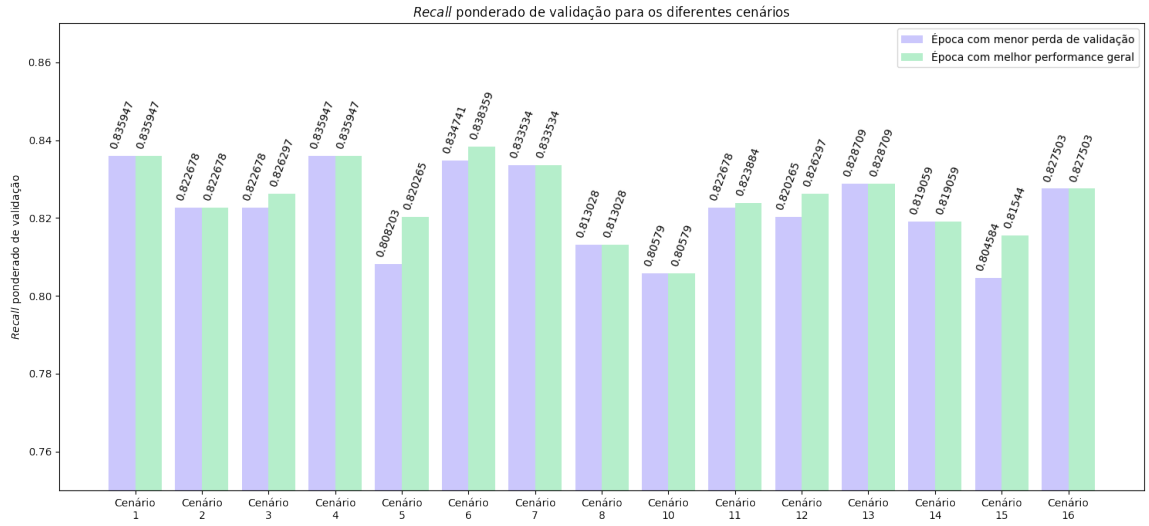


Figura 5.6: *Valores de recall ponderado de validação para os cenários selecionados.*

nados: aquele que apresentou a menor perda de validação dentre todos os cenários e aquele que apresentou melhor performance geral dentre todos os cenários. Os critérios foram os mesmos descritos acima, porém dessa vez considerando somente as linhas coloridas.

Como resultado, o modelo que apresentou a menor perda de validação foi aquele presente no cenário 1 gerado durante a terceira época de treinamento e o modelo com a melhor performance geral foi aquele presente no cenário 6 gerado durante a terceira época de treinamento. Ambos constam como linhas em negrito da tabela.

Para fazer uma escolha entre os dois modelos apontados acima, foi feita uma avaliação onde os dois modelos foram usados para classificar os acórdãos presentes no conjunto de dados de teste. Assim, o modelo que obtivesse a melhor performance seria escolhido.

Visualizando a figura 5.10 é possível observar que o modelo do cenário 1 demonstrou valores melhores ou iguais que o modelo do cenário 6 em todas as métricas gerais analisadas

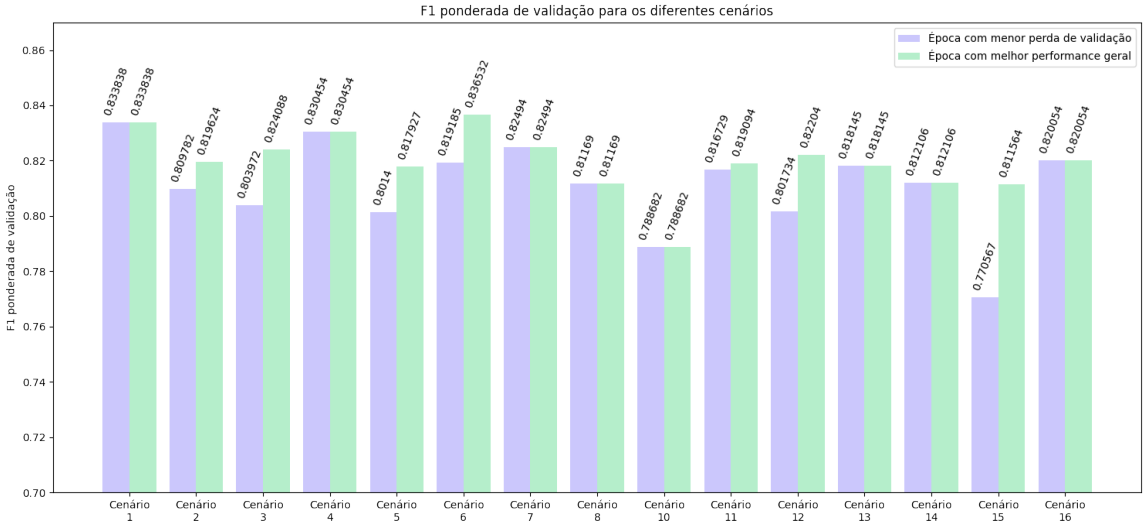


Figura 5.7: *Valores de F1 ponderada de validação para os cenários selecionados.*

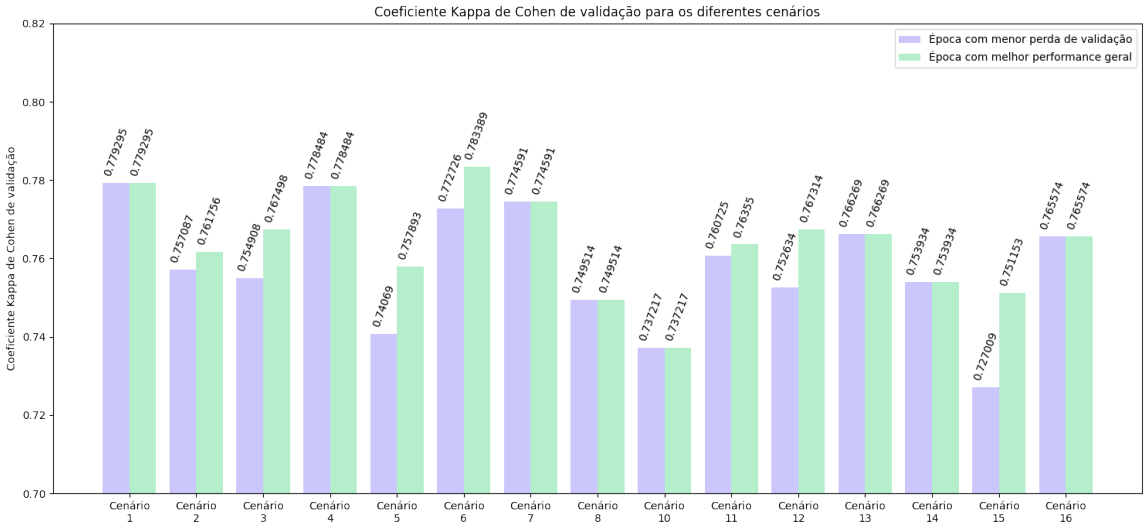


Figura 5.8: *Valores do coeficiente Kappa de Cohen de validação para os cenários selecionados.*

e portanto, foi escolhido como o modelo final desenvolvido no trabalho.

Focando apenas nos dados da tabela 5.1, a performance do classificador resultante encontra-se em torno de 82% de acurácia, significando que ele tende a acertar o ramo de 82 acórdãos em um total de 100. No entanto, a acurácia balanceada apresenta um cenário um pouco diferente, pois ela penaliza mais um desbalanceamento da distribuição de acórdãos na classe. Isso ilustra um cenário mais realista mostrando que em ramos do Direito nos quais o classificador não está muito habituado, ele terá uma performance menor. Já as demais métricas acompanham a tendência dos valores da acurácia geral.

Visualizando a figura 5.13 é possível identificar que nas classes com quantidades menores de registros, o classificador obteve resultados inferiores de acurácia ao comparar com as classes que possuem números mais abundantes.

A acurácia balanceada penaliza bastante o modelo, pois dá a mesma importância a

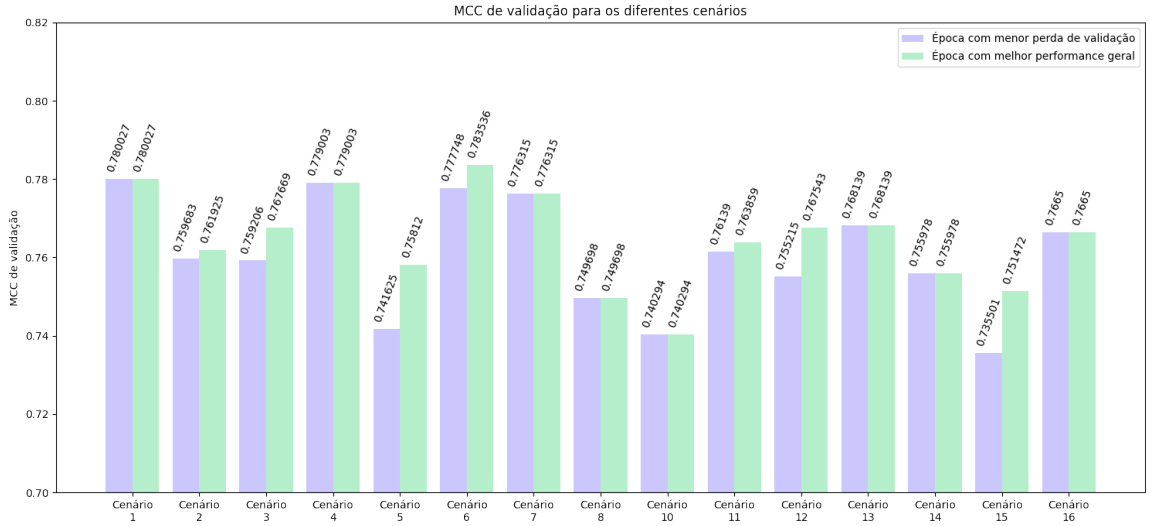


Figura 5.9: *Valores do coeficiente de correlação de Matthews de validação para os cenários selecionados.*

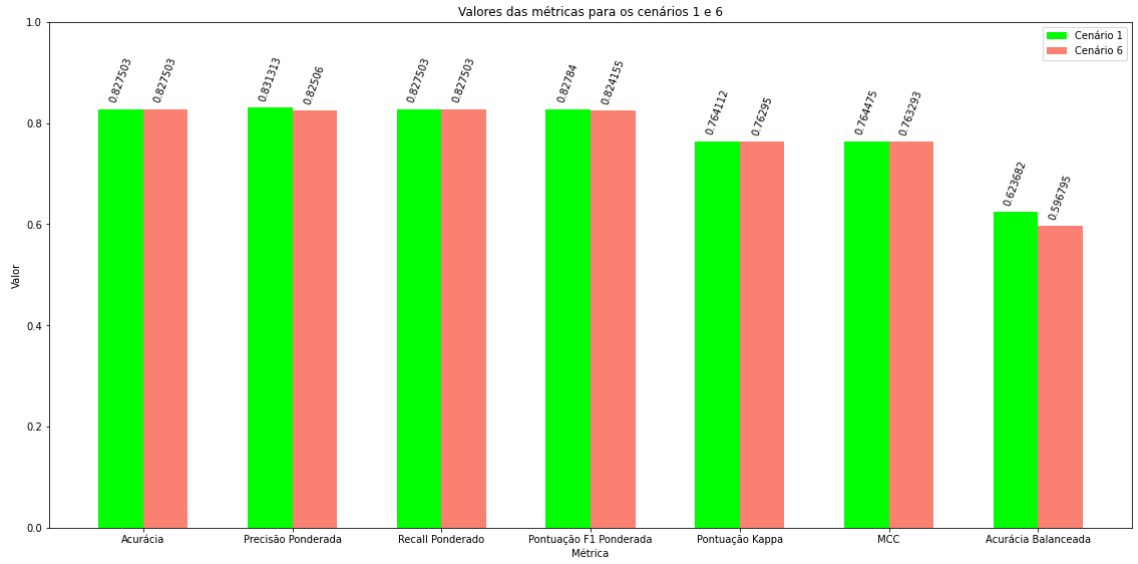


Figura 5.10: *Valores das métricas para os cenários 1 e 6.*

todas as classes. No entanto, o valor do coeficiente de correlação de Matthews mostra que o modelo resultante possui uma semelhança satisfatória com um classificador ideal. Da mesma forma, o Coeficiente Kappa de Cohen também indica que o modelo tem uma capacidade bastante relevante para entender a distribuição dos acórdãos nos ramos do Direito.

5.2 Experimentações para melhorias

Após o processo de *fine tuning* e a descoberta dos hiperparâmetros mais adequados para a construção do classificador, foram feitos alguns outros experimentos. Eles tinham o objetivo de verificar como o modelo reagiria ao conhecer atributos novos. Dessa forma, o

Modelo com melhor performance

Identificador do Cenário	1	6
Taxa de Aprendizado	2e−5	2e−5
Número de Palavras da Introdução	256	384
Fator de Decaimento	0.001	0.001
Proporção de Aquecimento	0.1	0.3
Época	3	3
Acurácia de Teste	0.82750301	0.82750301
Acurácia Balanceada de Teste	0.62368161	0.59679474
Precisão Ponderada de Teste	0.83131319	0.82505962
Recall Ponderado de Teste	0.82750301	0.82750301
F1 Ponderada de Teste	0.82784046	0.82415505
Kappa de Teste	0.76411186	0.76294961
MCC de Teste	0.76447489	0.76329306

Tabela 5.1: *Resultado da avaliação dos modelos do cenário 1 e do cenário 6 no conjunto de dados de teste.*

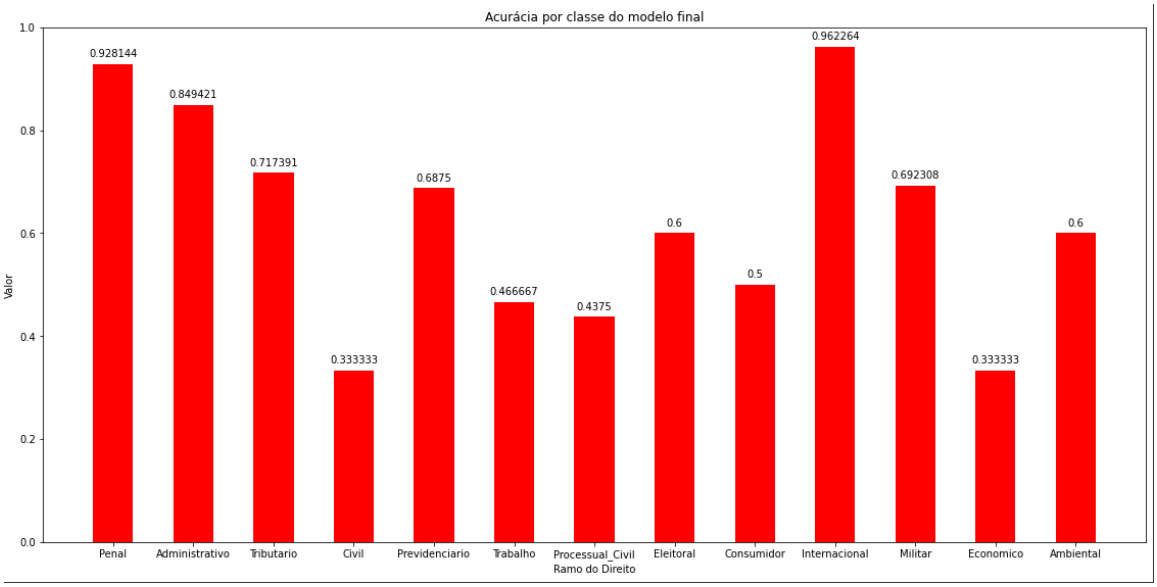


Figura 5.13: *Valores de acurácia por ramo do Direito para o modelo do cenário 1.*

modelo foi alimentado com algumas colunas novas do conjunto de dados juntamente com a ementa do modelo.

5.2.1 Adição do campo de indexação

O primeiro experimento consistiu na tentativa de adição da coluna referente ao campo de indexação do acórdão junto com a ementa como entrada para o modelo. O campo de indexação é um conjunto de palavras que foram obtidas a partir da interpretação do conteúdo do acórdão com o intuito de facilitar a recuperação de informações. Esse conjunto de palavras pode conter o nome dos ministros envolvidos, expressões presentes no texto

da ementa ou no texto da decisão e até palavras que resumem o contexto do acórdão.

O desenvolvimento do experimento seguiu os mesmos passos usados na construção do modelo com a estratégia de concatenação do início e do fim da ementa com exceção da definição dos hiperparâmetros e do processamento dos dados. Os hiperparâmetros usados no experimento foram os valores otimizados obtidos no processo de *fine-tuning* enquanto o processamento dos dados transcorreu da seguinte forma:

- Os conjuntos de dados de treinamento e de validação foram carregados e foi feita uma leitura da coluna de indexação;
- Alguns caracteres especiais como "[" e ":" foram removidos;
- Foi feita a separação das expressões por meio de vírgulas. As expressões foram posteriormente agregadas como uma lista;
- As frequências de todas as expressões foram contadas e os valores foram ordenados em ordem decrescente;
- Observando os valores e as frequências, foram selecionados e apagados dos registros de ambos os conjuntos de dados os termos que não agregariam na inferência do ramo do Direito do acórdão, como nomes de ministros, "AGUARDANDO INDEXAÇÃO" e "VIDE EMENTA"; e
- Foram criados novas tabelas de dados para serem usadas no experimento com apenas duas colunas, sendo uma com as classificações dos acórdãos e outra contendo os textos das ementas e das expressões de indexação.

Após os conjuntos de dados serem ajustados, o processo de treinamento do modelo seguiu de forma análoga aquele usado para o *fine tuning*, utilizando o módulo do Pytorch *Lightning* e a *Trainer API* do HuggingFace.

Depois da fase de treinamento, o modelo construído foi avaliado utilizando o conjunto de dados de teste para que fosse possível visualizar e comparar os resultados com os obtidos no modelo otimizado no processo de *fine tuning*. A avaliação consistiu em utilizar os modelos para classificar os acórdãos do conjunto de dados de teste, anotar os resultados das predições e compará-los com as classificações reais, com o auxílio de algumas métricas. Os resultados obtidos podem ser visualizados na figura 5.14 e na tabela 5.2.

	Modelo otimizado	Modelo com campo indexação
Acurácia	0.8275	0.8335
Precisão ponderada	0.8313	0.7926
Recall ponderado	0.8275	0.8335
Pontuação F1 ponderada	0.8278	0.8085
Pontuação Kappa	0.7641	0.7637
MCC	0.7644	0.7667
Acurácia balanceada	0.6236	0.4179

Tabela 5.2: *Resultados das métricas comparando o modelo otimizado com o modelo incrementado pelo campo de indexação.*

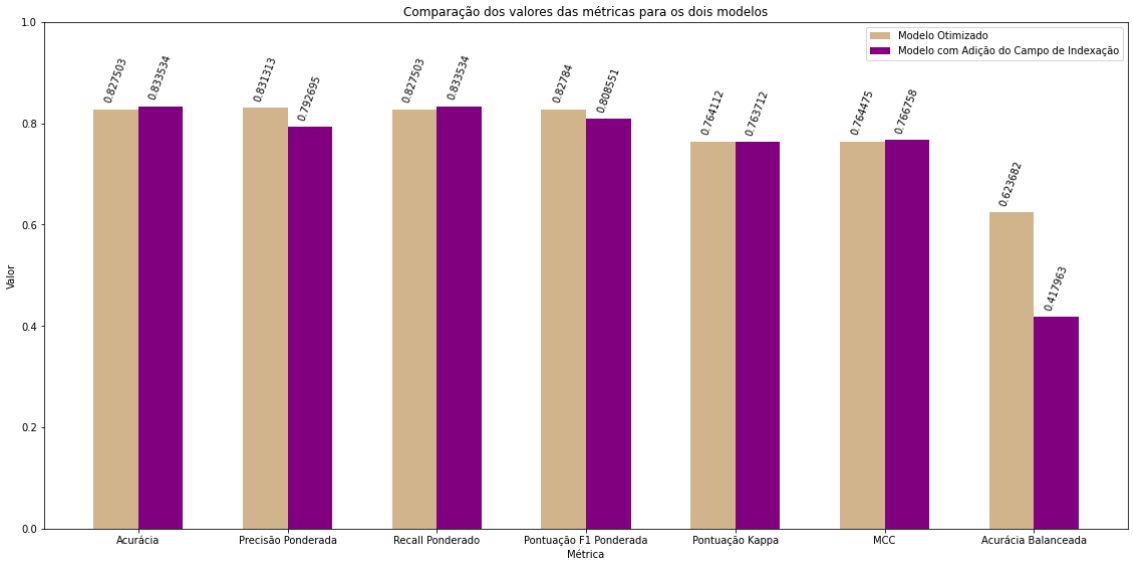


Figura 5.14: *Gráfico que mostra a relação dos valores obtidos das métricas para o modelo otimizado e para o modelo com adição do campo de indexação dos acórdãos.*

Observando os resultados do experimento é possível visualizar que o modelo treinado com a adição do campo de indexação do acórdão obteve resultados consideravelmente piores que o modelo otimizado apenas com a ementa do acórdão em todas as métricas, com exceção da acurácia, do *recall* ponderado e do MCC que se mostraram levemente melhores. Em especial, é possível destacar negativamente o valor da acurácia balanceada do modelo com adição do campo de indexação, que apresentou uma diferença de mais de 20% em relação ao modelo otimizado. Esses resultados demonstram que as expressões presentes no campo de indexação dos acórdãos não chegam a influenciar positivamente na tarefa de classificação dos acórdãos dentre os ramos do Direito.

Esse impacto negativo na classificação pode estar relacionado ao fato de que o campo de indexação do acórdão é um campo que não é preenchido na maioria dos casos e que possui muitas informações irrelevantes para o processo de classificação. Assim, as informações desse campo não chegam a agregar utilmente ao classificador, principalmente nos casos em que elas são preenchidas de maneira subótima, podendo até prejudicá-lo no seu processo de predição.

5.2.2 Adição do campo de classe processual

O segundo experimento feito consistiu da tentativa de adição da coluna referente à classe processual do acórdão junto com a ementa como entrada para o modelo. A classe processual é uma classificação de nível nacional do tipo do procedimento abordado no acórdão em questão. Ela é utilizada para classificar não apenas acórdãos mas também todos os tipos de documentos judiciais.

O desenvolvimento desse experimento foi análogo ao experimento anterior com o campo de indexação, com exceção do processamento dos dados. O processamento de dados do segundo experimento seguiu da seguinte forma:

- Um Enum foi definido contendo um mapeamento entre todas as siglas das classes processuais presentes no conjunto de dados e seus significados;
- Os conjuntos de dados de treinamento e validação foram carregados e foi realizada uma leitura da coluna das classes processuais;
- Os valores de classes processuais foram filtrados de forma a eliminar conteúdos nulos ou vazios. Depois, cada acórdão foi relacionado com a classe processual escrita por extenso; e
- Foram criadas novas tabelas de dados para serem usadas no experimento com apenas duas colunas, sendo uma com as classificações dos acórdãos e outra contendo os textos das ementas e das classes processuais por extenso. As classes processuais foram processadas da mesma forma que a ementa, ou seja, *tokenizadas* apenas durante a rotina de treinamento.

Preparados os conjuntos de dados, o processo de treinamento também utilizou o módulo do Pytorch *Lightning* e a *Trainer API* do HuggingFace.

Após a fase de treinamento, foi feita a avaliação e comparação dos modelos analogamente ao experimento anterior. Os resultados obtidos deste experimento podem ser visualizados na figura 5.15 e na tabela 5.3.

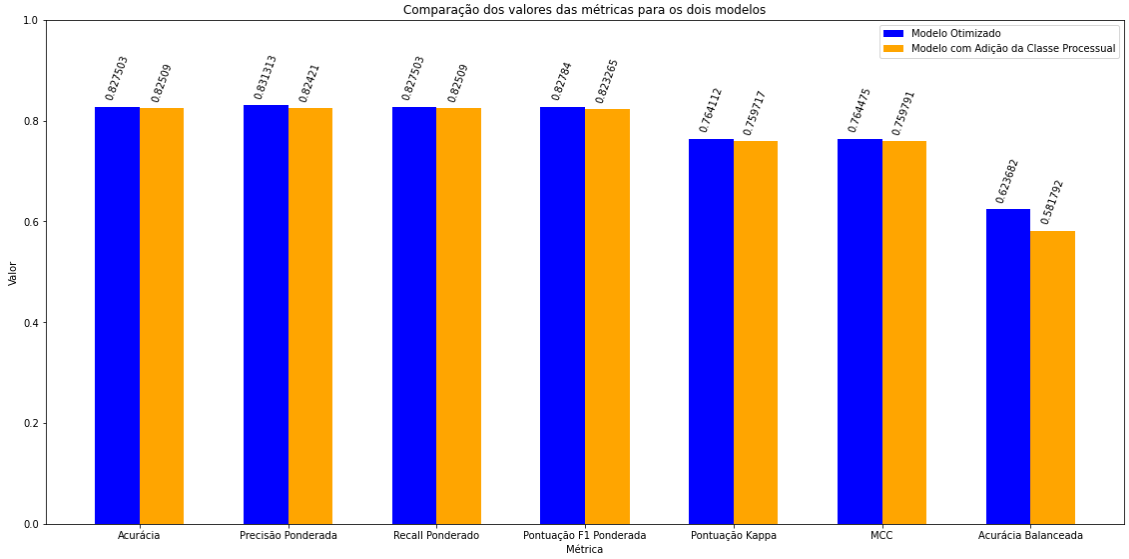


Figura 5.15: *Gráfico que mostra a relação dos valores obtidos das métricas para o modelo otimizado e para o modelo com adição da classe processual.*

Observando os resultados do experimento é possível visualizar que o modelo treinado com a adição da classe processual do acórdão obteve resultados ligeiramente piores ou iguais que o modelo otimizado com apenas a ementa do acórdão. Isso indica que o uso das classes processuais no treinamento do modelo não influencia de maneira positiva na tarefa de classificação dos acórdãos dentre os ramos do Direito, causando até uma piora, mesmo que mínima, nas predições.

Esse efeito negativo na classificação pode estar relacionado ao próprio conteúdo da informação que é a classe processual do acórdão. Em alguns contextos a classe processual

	Modelo otimizado	Modelo com classe processual
Acurácia	0.8275	0.8250
Precisão ponderada	0.8313	0.8242
Recall ponderado	0.8275	0.8250
Pontuação F1 ponderada	0.8278	0.8232
Pontuação Kappa	0.7641	0.7597
MCC	0.7644	0.7597
Acurácia balanceada	0.6236	0.5817

Tabela 5.3: *Resultados das métricas comparando o modelo otimizado com o modelo com adição da classe processual do acórdão.*

está fortemente conectada a um ramo do Direito, como é o caso da classe "Habeas Corpus" que é relacionada quase que exclusivamente ao ramo do Direito penal, enquanto que em outros não há essa ligação. Isso faz com que para esses casos, devido à amplitude de opções e falta de uma linha lógica mais definida entre os dados, o classificador tenha uma maior dificuldade em fazer a sua predição, afetando os seus resultados.

Capítulo 6

Considerações finais

Os acórdãos são documentos do meio jurídico que estão se tornando digitalizados em um ritmo lento. Aos poucos, informações são extraídas dos acórdãos e levadas ao meio digital, a exemplo de todos os dados que constam na Pesquisa de Jurisprudência do STF. Muitas informações dos acórdãos são difíceis de serem digitalizadas, e por muitas vezes, precisam ser manualmente retiradas utilizando digitação, por exemplo. Outros dados dos acórdãos são adquiridos apenas por meio da análise de um profissional com conhecimento da área do Direito, como o ramo do Direito ao qual um acórdão está inserido.

Em diversas situações, um cidadão, sem experiência ou aprendizado prévio no campo de estudos do Direito, pode ter a necessidade de analisar um acórdão e identificar o ramo do Direito. Isso demanda que ele tenha alguém à disposição que possa auxiliá-lo, ou custeie um profissional que possa desempenhar tal função. Em um outro cenário mais específico, alunos de Direito, principalmente aqueles que ainda não possuem um entendimento consistente dos ramos do Direito, podem precisar de uma ferramenta que os ajudem de forma simples e prática na identificação do ramo do Direito, dado um acórdão.

O modelo criado pode resolver os problemas citados, possuindo apenas a ementa de um acórdão. O conteúdo textual da ementa do acórdão oferecerá a maior parte da informação que o modelo precisa para predizer o ramo do Direito ao qual um acórdão está inserido. Alguns resultados como os valores de Kappa de Cohen indicam que o classificador desenvolvido tem uma performance melhor do que modelos mais simples. No entanto, como visto durante a análise dos dados, alguns acórdãos não apontam claramente o ramo do Direito pela ementa. Sendo assim, o classificador ainda pode ter um desempenho melhor, caso consiga acessar outros atributos do acórdão que possam agregar na sua predição.

Uma das grandes dificuldades para extrair um melhor rendimento do classificador foi a deficiência na quantidade de dados. Algumas tentativas mais simples de contornar essa situação não obtiveram sucesso, como as citadas ao longo das seções 5.2.1 e 5.2.2. Ainda assim, todo o processo de elaboração do modelo final desenvolvido neste trabalho gerou bons resultados, principalmente ao se comparar com modelos construídos a partir de técnicas menos complexas, como é possível visualizar no Apêndice B.

6.1 Próximos passos

Apesar de atingir resultados satisfatórios, ainda há diversas oportunidades de aprimoramentos do modelo. A principal alteração que pode levar a um avanço notório na performance do modelo seria um aumento na quantidade de dados rotulados. O projeto pode também receber aplicações adjacentes com o objetivo de facilitar a ingestão de dados por parte do modelo. Essas aplicações podem prover uma boa experiência para que o usuário consiga anexar acórdãos classificados. Nesse caso, alunos ou especialistas do Direito, por exemplo, podem fornecer acórdãos já rotulados a essa aplicação que auxiliará o modelo a ingerir os novos dados.

O modelo também pode desfrutar de outros dados além da ementa do acórdão. Dessa forma, o classificador poderia considerar o inteiro teor do acórdão ou outros atributos que possam ser úteis para uma classificação. Em certas ocasiões, o modelo poderia continuar a usar a ementa, porém analizar algumas expressões ou termos que são essenciais para identificar cada ramo. Nesse sentido, o modelo passaria a tratar as especificidades de cada ramo a fim de melhorar sua performance.

Ademais, com um foco diferente da performance do modelo, um próximo passo seria também tornar o modelo disponível em alguma ferramenta em que os usuários possam desfrutar de uma grande experiência. Para entender como o classificador poderia ser anexado a uma aplicação, veja o Apêndice C. Uma boa oportunidade de incorporação do classificador seria a aplicação desenvolvida por Oliveira (Oliveira, 2017). Uma ferramenta com um alcance ainda maior seria a Pesquisa de Jurisprudência do STF.

Apêndice A

Resultados dos experimentos de *fine tuning*

	Época com menor valor da função de perda
	Época com melhores valores de métricas em geral
	Época em que ambos ocorrem

Identificador do cenário	Taxa de Aprendizado	Número de palavras da introdução	Fator de decaimento	Proporção de Aquecimento	Época	Perda de Treinamento	Perda de Validação	Acurácia de Validação	Acurácia Balanceada de Validação	Precisão Ponderada de Validação	*Recall* Ponderado de Validação	F1 Ponderada de Validação	Cohen Kappa de Validação	MCC de Validação
1	2e−5	256	0.001	0.1	1	0.111300	0.084331	0.820265	0.433744	0.795340	0.820265	0.794642	0.751233	0.755923
					2	0.124600	0.083498	0.816647	0.483429	0.793683	0.816647	0.800072	0.748517	0.750557
					3	**0.061900**	**0.078958**	**0.835947**	**0.626879**	**0.839089**	**0.835947**	**0.833838**	**0.779295**	**0.780027**
					4	0.035400	0.091639	0.821472	0.613385	0.820597	0.821472	0.817525	0.760106	0.760450
					5	0.043500	0.093160	0.831122	0.638819	0.833306	0.831122	0.829227	0.773200	0.773515
2	2e−5	256	0.001	0.3	1	0.111800	0.090903	0.804584	0.410584	0.771140	0.804584	0.776280	0.729080	0.734503
					2	0.129200	0.080526	0.822678	0.488154	0.812032	0.822678	0.809782	0.757087	0.759683
					3	0.115200	0.092346	0.820265	0.579598	0.801543	0.820265	0.808053	0.755582	0.756518
					4	0.026300	0.090867	0.822678	0.622701	0.819570	0.822678	0.819624	0.761756	0.761925
					5	0.049700	0.094292	0.821472	0.606667	0.820114	0.821472	0.817575	0.759281	0.759703
3	2e−5	256	0.01	0.1	1	0.112200	0.085410	0.822678	0.496702	0.798607	0.822678	0.804067	0.756977	0.759941
					2	0.125800	0.085150	0.822678	0.458004	0.818927	0.822678	0.803972	0.754908	0.759206

					3	0.063200	0.090066	0.814234	0.564761	0.818285	0.814234	0.807708	0.748018	0.749957
					4	0.026100	0.092623	0.826297	0.668937	0.826909	0.826297	0.824088	0.767498	0.767669
					5	0.055400	0.097231	0.821472	0.641641	0.822184	0.821472	0.819238	0.760069	0.760345
					1	0.120400	0.088210	0.816647	0.435207	0.786454	0.816647	0.790627	0.746201	0.751790
					2	0.131900	0.083910	0.810615	0.441199	0.794418	0.810615	0.790888	0.739349	0.742856
4	2e−5	256	0.01	0.3	3	0.069900	0.080289	0.835947	0.627392	0.831993	0.835947	0.830454	0.778484	0.779003
					4	0.020400	0.088767	0.829916	0.628624	0.827156	0.829916	0.825995	0.771393	0.771611
					5	0.069900	0.091039	0.823884	0.617909	0.821695	0.823884	0.820564	0.763185	0.763432
					1	0.072400	0.094807	0.808203	0.533291	0.800480	0.808203	0.801400	0.740690	0.741625
					2	0.089600	0.098364	0.821472	0.518264	0.810263	0.821472	0.803797	0.754487	0.758530
5	2e−5	384	0.001	0.1	3	0.048700	0.107149	0.814234	0.584028	0.808729	0.814234	0.807537	0.748088	0.748696
					4	0.019100	0.107460	0.813028	0.618038	0.811892	0.813028	0.809661	0.748731	0.748917
					5	0.041200	0.107143	0.820265	0.622723	0.820309	0.820265	0.817927	0.757893	0.758120
					1	0.121600	0.085153	0.805790	0.454687	0.765932	0.805790	0.784325	0.735670	0.737319
					2	0.126900	0.081049	0.834741	0.506309	0.827888	0.834741	0.819185	0.772726	0.777748
6	2e−5	384	0.001	0.3	3	**0.062000**	**0.082047**	**0.838359**	**0.674744**	**0.839175**	**0.838359**	**0.836532**	**0.783389**	**0.783536**
					4	0.024900	0.088094	0.826297	0.665241	0.827264	0.826297	0.824557	0.767108	0.767286
					5	0.053000	0.091987	0.825090	0.659191	0.825595	0.825090	0.823051	0.765255	0.765430
					1	0.117100	0.088482	0.815440	0.446094	0.789309	0.815440	0.790498	0.744429	0.748333
					2	0.126300	0.082568	0.827503	0.466331	0.817663	0.827503	0.808718	0.762191	0.765224
7	2e−5	384	0.01	0.1	3	0.066800	0.081563	0.833534	0.605334	0.825689	0.833534	0.824940	0.774591	0.776315
					4	0.023800	0.089222	0.823884	0.609779	0.821223	0.823884	0.818133	0.761965	0.762474
					5	0.042100	0.091618	0.821472	0.611621	0.819621	0.821472	0.816024	0.758825	0.759370
					1	0.121500	0.087015	0.814234	0.487834	0.791065	0.814234	0.795623	0.746335	0.750585
					2	0.157000	0.083658	0.817853	0.459679	0.802639	0.817853	0.801648	0.749760	0.753550
8	2e−5	384	0.01	0.3	3	0.072300	0.083294	0.813028	0.596670	0.815546	0.813028	0.811690	0.749514	0.749698
					4	0.037600	0.091601	0.814234	0.588415	0.813743	0.814234	0.811444	0.750547	0.750892
					5	0.067500	0.095229	0.809409	0.558301	0.808444	0.809409	0.805752	0.743206	0.743565
					1	0.219900	0.203064	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.203100	0.204526	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
9	3e−5	256	0.001	0.1	3	0.230000	0.202134	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.211700	0.202398	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194000	0.202601	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					1	0.121400	0.090749	0.805790	0.500350	0.782590	0.805790	0.788682	0.737217	0.740294
					2	0.207500	0.225625	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
10	3e−5	256	0.001	0.3	3	0.228500	0.206538	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					4	0.210400	0.204913	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.193100	0.205375	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					1	0.099300	0.096866	0.784077	0.441114	0.730139	0.784077	0.747113	0.701368	0.711393
					2	0.117100	0.091209	0.811821	0.437699	0.787772	0.811821	0.792379	0.742625	0.745665
11	3e−5	256	0.01	0.1	3	0.058900	0.080394	0.822678	0.588013	0.816823	0.822678	0.816729	0.760725	0.761390
					4	0.043400	0.088374	0.823884	0.633269	0.820873	0.823884	0.819094	0.763550	0.763859
					5	0.042200	0.093037	0.819059	0.631969	0.817720	0.819059	0.815021	0.756675	0.757016
					1	0.125000	0.090479	0.800965	0.404290	0.763167	0.800965	0.772614	0.725243	0.730701
					2	0.120500	0.085080	0.820265	0.455786	0.805634	0.820265	0.801734	0.752634	0.755215
12	3e−5	256	0.01	0.3										

					3	0.063400	0.087740	0.822678	0.607547	0.820861	0.822678	0.819215	0.761381	0.761670
					4	0.038400	0.090979	0.826297	0.651599	0.823398	0.826297	0.822040	0.767314	0.767543
					5	0.056100	0.094706	0.819059	0.615423	0.815909	0.819059	0.814069	0.756553	0.756911
13	3e−5	384	0.001	0.1	1	0.141800	0.108946	0.800965	0.438121	0.769877	0.800965	0.776515	0.724572	0.728808
					2	0.121000	0.096353	0.805790	0.484106	0.776511	0.805790	0.785790	0.732950	0.735991
					3	0.104200	0.090935	0.828709	0.573812	0.830573	0.828709	0.818145	0.766269	0.768139
					4	0.021800	0.093712	0.814234	0.597908	0.812374	0.814234	0.811447	0.750074	0.750268
					5	0.051700	0.098204	0.815440	0.584254	0.813171	0.815440	0.810917	0.750015	0.750700
14	3e−5	384	0.001	0.3	1	0.115600	0.093441	0.806996	0.423749	0.783674	0.806996	0.783383	0.733321	0.739325
					2	0.127300	0.093221	0.805790	0.418761	0.793681	0.805790	0.774429	0.728241	0.739971
					3	0.058600	0.085632	0.819059	0.551254	0.817328	0.819059	0.812106	0.753934	0.755978
					4	0.036700	0.097949	0.809409	0.603548	0.806155	0.809409	0.804000	0.741990	0.742613
					5	0.066600	0.099372	0.813028	0.610817	0.811463	0.813028	0.808656	0.747179	0.747747
15	3e−5	384	0.01	0.1	1	0.113800	0.093245	0.804584	0.405307	0.768150	0.804584	0.770567	0.727009	0.735501
					2	0.129200	0.279335	0.798552	0.436349	0.805675	0.798552	0.791070	0.725159	0.728661
					3	0.099600	0.284189	0.787696	0.525350	0.812084	0.787696	0.789147	0.713868	0.715614
					4	0.031700	0.094219	0.815440	0.594510	0.811153	0.815440	0.811564	0.751153	0.751472
					5	0.084500	0.097850	0.811821	0.582301	0.806565	0.811821	0.807508	0.745855	0.746216
16	3e−5	384	0.01	0.3	1	0.136800	0.093545	0.776840	0.379079	0.748381	0.776840	0.742626	0.689453	0.702296
					2	0.124400	0.087691	0.815440	0.482998	0.804529	0.815440	0.792597	0.744708	0.750208
					3	0.070600	0.083832	0.827503	0.583081	0.827528	0.827503	0.820054	0.765574	0.766500
					4	0.039300	0.089284	0.820265	0.636221	0.819752	0.820265	0.817492	0.757970	0.758299
					5	0.052500	0.091153	0.822678	0.633867	0.820067	0.822678	0.818706	0.760334	0.760889
17	5e−5	256	0.001	0.1	1	0.223200	0.204239	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.202500	0.205141	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.228000	0.202181	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.212200	0.202663	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194200	0.202861	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
18	5e−5	256	0.001	0.3	1	0.220300	0.203675	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.204000	0.204905	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.231000	0.202156	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.210400	0.202663	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194000	0.202781	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
19	5e−5	256	0.01	0.1	1	0.223200	0.204239	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.205300	0.205286	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.229600	0.202442	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.211700	0.202579	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194200	0.202760	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
20	5e−5	256	0.01	0.3	1	0.222400	0.204456	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.204400	0.205198	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.229100	0.202334	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.211600	0.202655	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194900	0.202736	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
21	5e−5	384	0.001	0.1	1	0.220300	0.203563	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000

					2	0.204200	0.205190	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.229100	0.202254	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.211800	0.202648	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.193900	0.202814	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
22	5e−5	384	0.001	0.3	1	0.142200	0.108737	0.781665	0.354544	0.743071	0.781665	0.741689	0.694140	0.705808
					2	0.205400	0.205154	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.230200	0.202221	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.210100	0.202623	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.194300	0.202732	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
23	5e−5	384	0.01	0.1	1	0.219100	0.203425	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.206100	0.204893	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.232000	0.202339	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.210700	0.202548	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.195000	0.202730	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
24	5e−5	384	0.01	0.3	1	0.224600	0.203829	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					2	0.204000	0.205406	0.300362	0.076923	0.090217	0.300362	0.138757	0.000000	0.000000
					3	0.229500	0.202340	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					4	0.210600	0.202579	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000
					5	0.193600	0.202676	0.382388	0.076923	0.146221	0.382388	0.211548	0.000000	0.000000

Tabela A.1: *Resultados dos experimentos gerados como parte do processo de fine tuning*

Apêndice B

Modelos para comparação

Além do modelo final desenvolvido no trabalho, também foram construídos outros dois modelos mais simplificados para fins de comparação.

O primeiro modelo faz a sua escolha dentre os ramos do Direito baseado na contagem das frequências de certas expressões presentes no texto da ementa. Essas expressões foram extraídas da coluna do conjunto de dados que contém as expressões e palavras-chave que são pertinentes à classificação do acórdão. Assim, foi gerado um mapeamento entre os ramos do Direito e todas as expressões que é utilizado por esse modelo simplificado para fazer as suas predições.

O segundo modelo também utiliza os valores da coluna de expressões e palavras chave do conjunto de dados, mas não para fazer a predição diretamente com eles e sim usá-los como vocabulário na construção de um transformador TF-IDF[1]. Com esse transformador, foi possível criar vetores a partir das ementas dos acórdãos e alimentá-los em um modelo baseado no algoritmo de Regressão Logística para realizar o treinamento.

Assim, após a construção e treinamento desses dois modelos, foi possível submetê-los a uma avaliação no conjunto de teste para comparar a performance dos dois modelos com o modelo final desenvolvido no trabalho. Os resultados obtidos podem ser visualizados na tabela B.1 e na figura B.1.

Modelo	Acurácia de Teste	Precisão Ponderada de Teste	*Recall* Ponderado de Teste	F1 Ponderada de Teste	Acurácia Balanceada de Teste	Cohen Kappa de Teste	Coeficiente Matthews de Teste
BERT	0.827503	0.831313	0.827503	0.827840	0.623682	0.764112	0.764475
Regressão Logística	0.802171	0.776433	0.802171	0.777735	0.375222	0.716301	0.723584
Simples	0.704463	0.604922	0.704463	0.607441	0.163863	0.544913	0.577128

Tabela B.1: *Resultado da avaliação dos modelos no conjunto de dados de teste.*

[1] Sigla em inglês para Frequência do Termo–Inverso da Frequência nos Documentos. É uma pontuação numérica utilizada para medir o quão relevante é uma determinada palavra para um determinado documento dentro de um conjunto de documentos específicos.

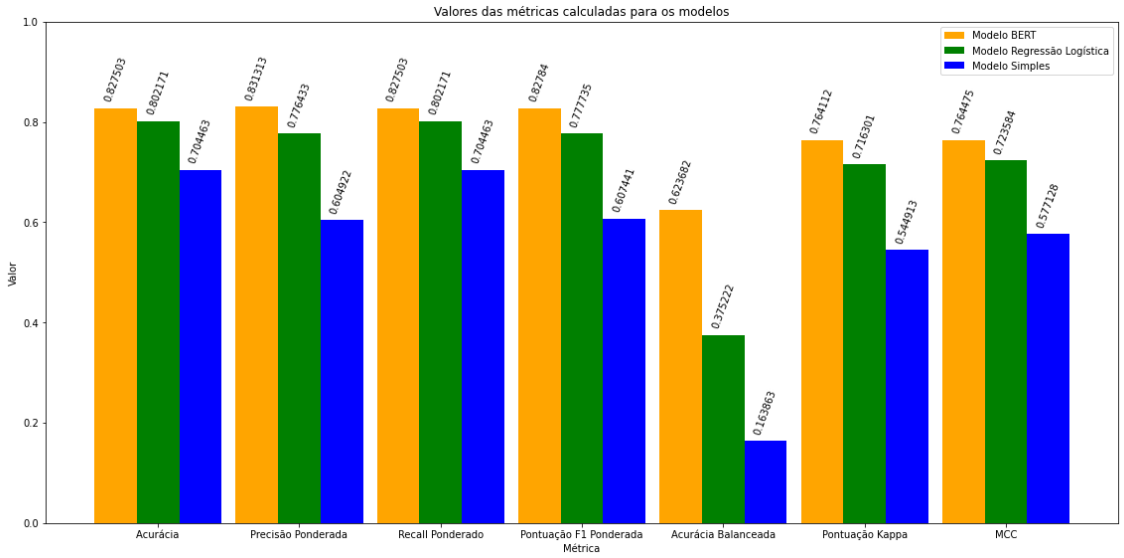


Figura B.1: *Gráfico que mostra o resultado da avaliação dos modelos no conjunto de teste.*

Observando os resultados é possível perceber que o modelo mais simples baseado apenas nas expressões chaves apresentou valores nas métricas muito abaixo que os outros dois, especialmente na acurácia balanceada onde obteve apenas 16%, o que é de se esperar devido à falta de cobertura na estrutura do modelo para lidar com o desbalanceamento dos dados. Além disso, também é possível observar que o modelo de Regressão Logística não apresentou um desempenho ruim no geral mas que ainda assim obteve resultados piores em todas as métricas que o modelo baseado no BERT, o que permite a conclusão de que este último é o melhor modelo dentre os três analisados e de que todo o processo e desenvolvimento deste trabalho para a construção do modelo final oferece uma base de aspecto positivo para a resolução do problema de classificação de documentos jurídicos.

Apêndice C

Exemplo de aplicação do classificador

No desfecho desse trabalho foi desenvolvida uma API simples com o objetivo de exibir o funcionamento do classificador final em tarefas de classificação de ementas. A estrutura da API foi construída usando as seguintes ferramentas:

- Para o servidor foi utilizado o FastAPI[1] que é um arcabouço que fornece soluções rápidas e de alta performance para construção de servidores na linguagem Python;
- Para a interface gráfica foi utilizado o React[2] que é uma biblioteca muito eficiente e flexível na linguagem Javascript para construção de interfaces de usuário; e
- Para o empacotamento do código foi utilizado o Docker[3] que é uma tecnologia de virtualização de *containers* que permite criar ambientes isolados e portáveis para as aplicações, facilitando o desenvolvimento e a implantação delas.

A aplicação utiliza o modelo desenvolvido neste trabalho para classificar textos de ementas que são recebidos via *endpoint* REST. A API expõe a rota *'/classify'* e espera por requisições do tipo POST com o seguinte formato:

```
{
   "annotation": "<Texto da ementa a ser classificada>"
}
```

O resultado da classificação da ementa fornecida dentre os ramos do Direito assim como valores numéricos de confiança e probabilidades são retornados ao usuário dentro de um objeto que possui o seguinte formato:

```
{
   "branch": "<Nome do ramo do Direito no qual a ementa foi classificada>",
   "confidence": "<Valor da confiança do classificador no ramo escolhido>",
   "probabilities": "<Dicionário com as probabilidades retornadas pelo
       classificador de que a ementa pertença a cada ramo do Direito>"
```

[1] https://fastapi.tiangolo.com/

[2] https://reactjs.org/

[3] https://www.docker.com/

```
5     }
```

A interface de usuário da aplicação conta com apenas uma página *web* onde são feitas tanto as requisições de classificação quanto a visualização dos resultados.

Em relação às requisições, elas se apresentam e são executas no topo do formulário principal da página, como se pode observar na figura C.1. Esse formulário contém um campo de texto livre, onde é esperado o texto da ementa do acórdão a ser classificado e um botão para submeter a requisição. Além disso, nele também está contida uma lista suspensa com uma série de códigos de acórdãos que são extraídos do conjunto de dados utilizado neste trabalho. Ao selecionar um item da lista, a aplicação preenche campo de texto livre com a sua respectiva ementa automaticamente.

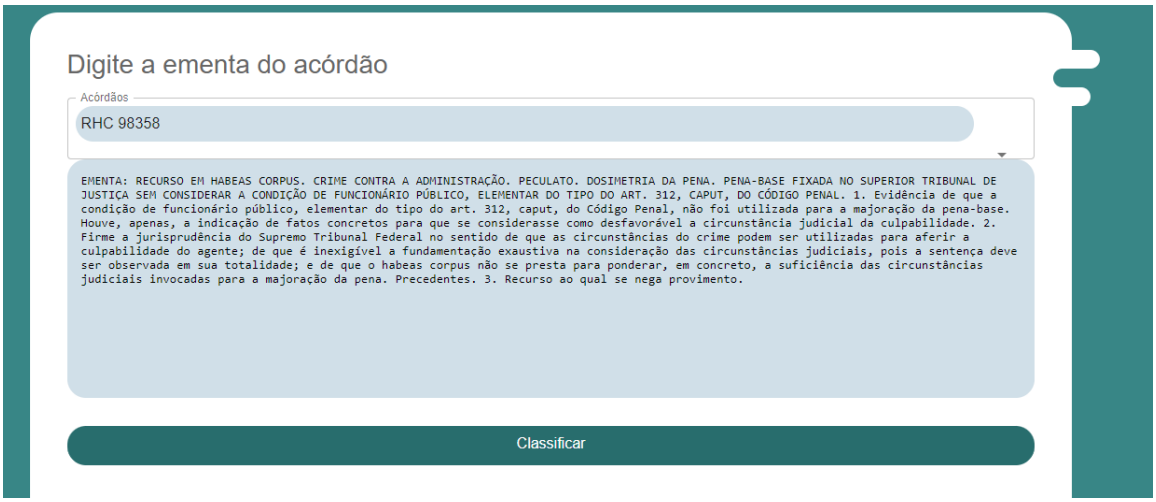


Figura C.1: *Visualização do formulário de requisição para classificação de ementas na página web.*

Quanto à visualização dos resultados, a interface da aplicação conta com campos de texto com escrita restringida para apresentar cada uma das informações retornadas. Além disso, a interface também exibe um gráfico que sintetiza as probabilidades de classificação do acórdão para cada um dos ramos do Direito. Um exemplo de visualização de resultados pela interface pode ser observado na figura C.2.

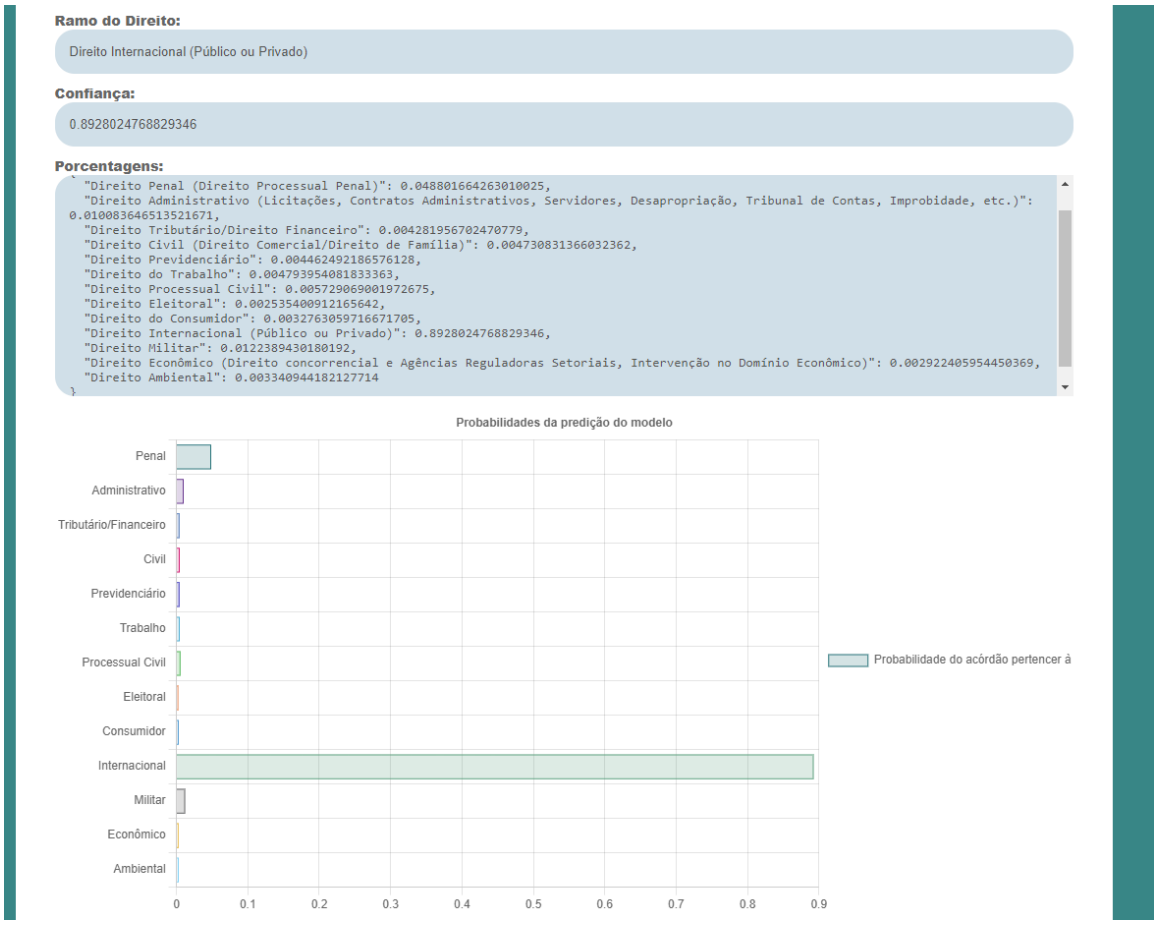


Figura C.2: *Visualização dos resultados da classificação de uma ementa mostrados na página web.*

Referências

[Betioli 2015] Antonio Bento Betioli. *Introdução ao direito*. 14ª ed. São Paulo: Saraiva, 2015 (citado na pg. 3).

[Burkov 2019] Andriy Burkov. *The Hundred-Page Machine Learning Book*. 1ª ed. 2019 (citado nas pgs. 12, 16).

[Chakravarthy 2020] Srinivas Chakravarthy. *Tokenization for Natural Language Processing*. 2020. url: https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4 (acesso em 14/09/2021) (citado na pg. 13).

[Devlin *et al.* 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. "Bert: pre-training of deep bidirectional transformers for language understanding". Em: *arXiv preprint arXiv:1810.04805* (2019). url: https://arxiv.org/abs/1810.04805 (citado nas pgs. 12, 35, 41).

[Ferri *et al.* 2009] César Ferri, José Hernández-Orallo e R. Modroiu. "An experimental comparison of performance measures for classification". Em: *Pattern Recognition Letters* 30.1 (2009), pgs. 27–38. doi: https://doi.org/10.1016/j.patrec.2008.08.010 (citado na pg. 39).

[Gordon-Rodriguez *et al.* 2020] Elliott Gordon-Rodriguez, Gabriel Loaiza-Ganem, Geoff Pleiss e John P. Cunningham. *Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning*. 2020. arXiv: 2011.05231 [stat.ML] (citado na pg. 16).

[Grandini *et al.* 2020] Margherita Grandini, Enrico Bagli e Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. arXiv: 2008.05756 [stat.ML] (citado nas pgs. 20, 40).

[Gugger e Howard 2018] Sylvain Gugger e Jeremy Howard. *AdamW and Super-convergence is now the fastest way to train neural nets*. 2018. url: https://www.fast.ai/2018/07/02/adam-weight-decay/ (acesso em 22/09/2021) (citado na pg. 17).

[Gurney 1997] Kevin Gurney. *An introduction to neural networks*. 1ª ed. Londres e Nova Iorque: UCL Press, 1997 (citado na pg. 11).

[Hayes 2020] Adam Hayes. *Harmonic Mean*. 2020. URL: https://www.investopedia.com/terms/h/harmonicaverage.asp (acesso em 20/12/2021) (citado na pg. 22).

[Jr e Millones 2011] Robert Gilmore Pontius Jr e Marco Millones. "Death to kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment". Em: *International Journal of Remote Sensing* 32.15 (2011), pgs. 4407–4429. DOI: 10.1080/01431161.2011.552923. eprint: https://doi.org/10.1080/01431161.2011.552923. URL: https://doi.org/10.1080/01431161.2011.552923 (citado na pg. 23).

[Kingma e Ba 2017] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] (citado na pg. 17).

[Loshchilov e Hutter 2019] Ilya Loshchilov e Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG] (citado na pg. 17).

[Nader 2014] Paulo Nader. *Introdução ao estudo do direito*. 36ª ed. Rio de Janeiro: Forense, 2014 (citado na pg. 3).

[Oliveira 2017] Rafael Brito de Oliveira. "Utilização de Ontologias para Busca em Base de Dados de Acórdãos do STF". Diss. de mestr. São Paulo, Brasil: Instituto de Matemática e Estatística, Universidade de São Paulo, 2017. URL: https://teses.usp.br/teses/disponiveis/45/45134/tde-24012018-110738/pt-br.php (citado nas pgs. 7, 56).

[Paszke *et al.* 2019] Adam Paszke *et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG] (citado na pg. 25).

[Rao e McMahan 2019] Delip Rao e Brian McMahan. *Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning*. 1ª ed. O'Reilly Media, 2019 (citado nas pgs. 16, 17).

[Salvo Venosa 2019] Sílvio de Salvo Venosa. *Introdução ao estudo do direito*. 6ª ed. São Paulo: Atlas, 2019 (citado na pg. 4).

[Souza *et al.* 2019] Fábio Souza, Rodrigo Nogueira e Roberto Lotufo. "Portuguese named entity recognition using bert-crf". Em: *arXiv preprint arXiv:1909.10649* (2019). URL: http://arxiv.org/abs/1909.10649 (citado na pg. 14).

[Stevens *et al.* 2020] Eli Stevens, Luca Antiga e Thomas Viehmann. *Deep Learning with Pytorch*. 1ª ed. Manning Publications, 2020 (citado na pg. 19).

[Sun *et al.* 2020] Chi Sun, Xipeng Qiu, Yige Xu e Xuanjing Huang. *How to Fine-Tune BERT for Text Classification?* 2020. arXiv: 1905.05583 [cs.CL] (citado na pg. 35).

[Tardi 2021] Carla Tardi. *Moore's Law*. 2021. URL: https://www.investopedia.com/terms/m/mooreslaw.asp (acesso em 27/08/2021) (citado na pg. 1).

[Trehan 2020] Daksh Trehan. *Gradient Descent Explained*. 2020. URL: https://towardsdatascience.com/gradient-descent-explained-9b953fc0d2c (acesso em 08/09/2021) (citado na pg. 12).

[Widmann 2020] Maarit Widmann. *Cohen's Kappa: what it is, when to use it, how to avoid pitfalls*. 2020. URL: https://www.knime.com/blog/cohens-kappa-an-overview (acesso em 18/12/2021) (citado na pg. 23).

[You *et al.* 2020] Yang You *et al. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes*. 2020. arXiv: 1904.00962 [cs.LG] (citado na pg. 41).

[Zhuang *et al.* 2020] Fuzhen Zhuang *et al. A Comprehensive Survey on Transfer Learning*. 2020. arXiv: 1911.02685 [cs.LG] (citado na pg. 14).