UNIVERSIDADE FEDERAL DO PARÁ
NÚCLEO DE DESENVOLVIMENTO AMAZÔNICO EM ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

VICTOR SIMÕES MARTINS

APLICAÇÃO E COMPARAÇÃO DE TÉCNICAS DE CLASSIFICAÇÃO AUTOMÁTICA DE DOCUMENTOS: UM ESTUDO DE CASO COM O DATASET DO DOMÍNIO JURÍDICO "VICTOR".

Tucuruí - Pará
2024

VICTOR SIMÕES MARTINS

APLICAÇÃO E COMPARAÇÃO DE TÉCNICAS DE CLASSIFICAÇÃO AUTOMÁTICA DE DOCUMENTOS: UM ESTUDO DE CASO COM O DATASET DO DOMÍNIO JURÍDICO “VICTOR”.

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para obtenção do título de Mestre em Computação Aplicada.
Área de Concentração: Desenvolvimento de Sistemas
Orientador: CLEISON DANIEL SILVA

Tucuruí - Pará
2024

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

M379a Martins, Victor Simões.
Aplicação e comparação de técnicas de classificação automática de documentos: um estudo de caso com o dataset do domínio jurídico "VICTOR" / Victor Simões Martins. — 2024.
79 f. : il. color.

Orientador(a): Prof. Dr. Cleison Daniel Silva
Dissertação (Mestrado) - Universidade Federal do Pará, Núcleo de Desenvolvimento Amazônico em Engenharia, Mestrado Profissional em Computação Aplicada, Tucuruí, 2024.

1. Classificação de documentos. 2. Aprendizado de Máquina. 3. Processamento de Linguagem Natural. 4. Documentos Jurídico. I. Título.

CDD 004

VICTOR SIMÕES MARTINS

APLICAÇÃO E COMPARAÇÃO DE TÉCNICAS DE CLASSIFICAÇÃO AUTOMÁTICA DE DOCUMENTOS: UM ESTUDO DE CASO COM O DATASET DO DOMÍNIO JURÍDICO "VICTOR".

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada do Núcleo de Desenvolvimento Amazônico em Engenharia, da Universidade Federal do Pará, como requisito para obtenção do título de Mestre em Computação Aplicada.
Área de Concentração: Desenvolvimento de Sistemas
Orientador: CLEISON DANIEL SILVA

Aprovada em 1º de fevereiro de 2024.

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
CLEISON DANIEL SILVA
Data: 14/03/2024 10:32:08-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cleison Daniel Silva, UFPA - Orientador

gov.br Documento assinado digitalmente
Bruno Merlin
Data: 14/03/2024 18:04:12-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Bruno Merlin, UFPA

ICP Brasil Documento assinado digitalmente
ELIAS JACOB DE MENEZES NETO
Data: 15/03/2024 11:31:45-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Elias Jacob de Menezes Neto, UFRN

gov.br Documento assinado digitalmente
OTAVIO NOURA TEIXEIRA
Data: 14/03/2024 20:13:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Otávio Noura Teixeira, UFPA

AGRADECIMENTOS

Meu primeiro agradecimento vai para o meu Orientador, que acreditou na possibilidade de pesquisas na área de Processamento de Linguagem Natural dentro do PPCA, ao lado de todo o incentivo, acompanhamento, paciência e ensinamentos como acadêmico e ser humano, ao longo dessa jornada de novas descobertas e desafios em uma área de pesquisa crescente e dominada por grandes empresas do ramo de tecnologia.

Também, agradecer ao Ministério Público Federal (MPF), órgão que me dedico há 17 anos na área de Tecnologia da Informação, e pessoalmente ao Dr. Daniel Azevedo Lôbo, Secretário da Secretaria de Perícia Pesquisa e Análise (SPPEA) do MPF, e meus chefes imediatos: Rodrigo Brasil e Paulo Bitar, que possibilitaram o usufruto da Licença Capacitação para finalização dos experimentos e texto desta Dissertação, pois acreditam na qualificação e pesquisa como potencialidades que vão aprimorar os serviços prestados pela SPPEA. Além disso, incluo ainda meus colegas do Núcleo de Gestão de Dados Investigativos (NGDI): Bruno, Edgar, Luiz e William, que no período da licença abraçaram as minhas atividades dentro do Núcleo, para possibilitar plena continuidade dos projetos.

Não posso esquecer da minha paciente esposa, Ângela, que durante todo o curso do Mestrado me apoiou e foi compreensiva em momentos de férias, feriados e folgas do nosso trabalho, em que precisei dedicar-me à pesquisa, experimentos, leituras e escrita desta Dissertação.

Agradecer ainda aos meus pais, Tereza e Vitor, minha avó, irmãos e primos, que também precisaram entender que muitos momentos onde poderia estar com eles, precisou ser dedicado para esta Dissertação.

RESUMO

A aplicação do Processamento de Linguagem Natural (PLN) e Inteligência Artifical (IA) no contexto jurídico brasileiro é uma área em franco crescimento, que pode alterar o modo e rotina de trabalho dos profissionais da área, dada a quantidade de texto gerada. Dentre as possibilidades de aplicação da PLN e IA há a classificação automática de documentos, que dentre outras, pode ser empregada na automatização do processo de digitalização de Processos Judiciais que ainda estão apenas em meio físico. Assim, este trabalho aplica e compara algoritmos de IA para a classificação de documentos jurídicos. Os algoritmos são divididos em duas Abordagens diferentes, a primeira (I) separa o processo representação computacional do texto do treinamento do classificador em si aplicando SVM e Regressão Logística em conjunto com representações computacionais baseadas em: TF-IDF, Word2Vec, FastText e BERT. A segunda Abordagem (II) realiza em conjunto a representação computacional dos documentos e o treinamento do classificador, e para tal são aplicados algoritmos de *Deep Learning* baseados em redes neurais recorrentes, especificamente o ULMFiT (*Universal Language Model Fine-tuning*) e HAN (*Hierarchical Attention Networks*). O *Dataset* estudado é denominado VICTOR, composto por documentos do Supremo Tribunal Federal (STF) do Brasil. A pesquisa conclui pela possibilidade de aplicação de ambas abordagens para a classificação de documentos jurídicos do *Dataset* empregado, bem como, apesar de menos custosos computacionalmente, os *pipelines* de classificação da Abordagem I que empregam a representação computacional do documento com TF-IDF apresentam resultados equivalentes aos pipelines que empregam *Deep Learning*. Além disso, a especialização da representação computacional dos documentos com os dados do *dataset* em estudo, melhoram o desempenho dos *pipelines* que empregam Word2Vec, FastText e ULMFiT, quando comparados aos *pipelines* que aplicam as representações genéricas desses, ou seja, modelos pré-treinados com dados do contexto geral.

Palvaras-chave: Classificação de documentos. Aprendizado de Máquina. Processamento de Linguagem Natural. Documentos Jurídico.

ABSTRACT

The application of Natural Language Processing (NLP) and Artificial Intelligence (AI) in the Brazilian legal context is a rapidly growing area that can alter the way legal professionals work, given the volume of generated text. Among the possible applications of NLP and AI is the automatic classification of documents, which, among other things, can be employed in the automation of the digitization process of Judicial Proceedings that are still in physical form. Therefore, this work applies and compares AI algorithms for the classification of legal documents. The algorithms are divided into two different approaches. The first approach (I) separates the computational representation process of the text from the classifier training itself and applies SVM and Logistic Regression in conjunction with computational representations based on TF-IDF, Word2Vec, FastText, and BERT. The second approach (II) simultaneously performs the computational representation of documents and the training of the classifier, applying Deep Learning algorithms based on recurrent neural networks, specifically ULMFiT (Universal Language Model Fine-tuning), and HAN (Hierarchical Attention Networks). The studied dataset is named VICTOR, composed of documents from the Supreme Federal Court (STF) of Brazil. The research concludes that both approaches can be applied to the classification of legal documents from the employed dataset. Additionally, despite being less computationally expensive, the classification pipelines of Approach I, which use the computational representation of the document with TF-IDF, yield results equivalent to pipelines employing Deep Learning. Furthermore, embedding documents specialization with data from the *dataset* under study, improves the performance of pipelines that employ Word2Vec, FastText and ULMFiT, compared to pipelines that apply the generic representations of these, i.e., models pre-trained with data from the general context.

Keywords: Documents Classification. Machine Learning. Natural Language Processing. Legal Documents.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MPF	Ministério Público Federal
PLN	Processamento de Linguagem Natural
VC	Visão Computacional
CNJ	Conselho Nacional de Justiça
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
ML	Modelo de Linguagem
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
STF	Supremo Tribunal Federal
AILAB	Laboratório de Inteligência Artificial da Universidade de Brasília
NLTK	Natural Language Processing Toolkit
AM	Aprendizado de Máquina
ULMFiT	Universal Language Model Fine-tuning
GloVe	Global Vectors for Word Representation
MLP	Multilayer Perceptron
BoW	Bag of Words
GPU	Graphic Processing Unit
HAN	Hierarchical Attention Network
BERT	Bidirectional Encoder Representations for Transformers
GRU	Gated Recurrent Units
RRN	Recurrent Neural Network
MCC	Matthews Correlation Coefficient
CSV	Comma-separated values (Valores separados por vírgula)
PRE	Petição de Recurso Especial

ARE	Agravo em Recurso Extraordinário
LLM	Large Language Model
MLM	Masked Language Model

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e Motivação

O método tradicional de trabalho no contexto jurídico sempre envolveu um grande esforço manual na análise de documentos, tendo como consequência a necessidade do emprego de humanos em tarefas que podem ser lentas e custosas. Entretanto, a Ciência da Computação por meio de ferramentas de Processamento de Linguagem Natural (PLN) e Inteligência Artificial (IA), tem possibilitado o aprimoramento dos procedimentos de órgãos de Justiça e escritórios de advocacia, com o intuito de auxiliar na análise desses documentos para promoção da celeridade no trâmite processual, bem como na diminuição de custos públicos ou privados.

O Conselho Nacional de Justiça (CNJ) em seu relatório "Justiça Em Números de 2022"[1], indica um total de 77,3 milhões de processos em tramitação no Brasil no final do ano de 2021. Essa quantidade de processos levou o CNJ a ampliar a iniciativa de digitalização da Justiça brasileira, por meio de um conjunto de políticas denominados de Juízo 100% Digital e Justiça 4.0, normatizadas pelas Resoluções 345/2020 e 385/2021, respectivamente.

Um dos objetivos do CNJ a partir das política de digitalização da Justiça brasileira, é a implementação da plataforma CODEX [2] , que consolida o armazenamento dos processos em termos de dados estruturados e não-estruturados. O primeiro refere-se a metadados: informações de tramitação, tempo, fase atual, entre outros, e o segundo refere-se aos documentos que compõem de fato o processo judicial.

Não apenas nos órgãos de Justiça brasileiro representados pelo CNJ, mas também no âmbito do Ministério Público Federal (MPF), existem políticas de incentivo a aplicação da IA nos processos internos. Por meio da Portaria/SG/MPF nº 475 de 05/07/2018, o MPF instituiu o Comitê de Inteligência Artificial, com a finalidade de aprimorar e dar maior celeridade nas atividades internas.

O objetivo do CNJ com a plataforma CODEX, é prover dados estruturados ou não estruturados para viabilizar a aplicação de técnicas de Ciência de Dados e IA, que possibilitem a realização de diagnósticos da atuação do Justiça no Brasil, bem como promover o estudo de métodos que otimizem os procedimentos internos dos tribunais com a finalidade de dar maior celeridade no trâmite processual.

Diante da quantidade de processos judiciais em curso, o contínuo aumento da demanda aos órgãos de justiça, e a recomposição de recursos humanos no setor público em declínio, técnicas automáticas de análise e classificação de documentos jurídicos são importantes para garantia do acesso à justiça. Assim, dentre as possíveis aplicações da IA no contexto jurídico encontra-se a

[1] <https://www.cnj.jus.br/wp-content/uploads/2022/09/justica-em-numeros-2022-1.pdf>
[2] <https://www.cnj.jus.br/sistemas/plataforma-codex/>

classificação automática de documentos e processos judiciais. Publicações científicas na área sugerem algumas tarefas onde ela pode ser aplicada no Brasil:

- Migração dos processos físicos para eletrônicos (SILVA; MAIA, 2020);
- Direcionamento correto de demandas peticionadas eletronicamente, tal qual analisado em (NOGUTI; VELLASQUES; OLIVEIRA, 2020) e (MOTA et al., 2020);
- Predição da categoria de sentenças (MENEZES-NETO; CLEMENTINO, 2022);
- Classificação de Documentos e Processos do Supremo Tribunal Federal (ARAUJO et al., 2020a).

As atuais políticas de investimento para promoção da IA na Justiça brasileira, e consequente aplicações no contexto jurídico, também decorrem dos avanços recentes na área de PLN (POLO et al., 2021), metodologia responsável por representar computacionalmente a linguagem natural, tanto em texto quanto em fala (MANNING; SCHÜTZE, 1999), que por sua vez foi impulsionada pelo progresso e consolidação da IA por meio de métodos de Aprendizado de Máquina (AM) Tradicionais, e métodos que empregam modelos neurais (HORNIK; STINCHCOMBE; WHITE, 1989) denominados de Aprendizado de Máquina Profundo, ou em inglês *Deep Learning*.

Apesar das políticas de investimento, evidente possibilidade, necessidade e disponibilidade de técnicas para o emprego de IA e PLN no contexto jurídico, ela ainda é recente no Brasil quando comparada com países de língua inglesa (NOGUTI; VELLASQUES; OLIVEIRA, 2020), necessitando assim a ampliação dos estudos em diferentes tarefas de PLN, bem como o emprego e comparação de diferentes algoritmos de Aprendizado de Máquina.

Diante das oportunidades de aplicação da IA no contexto jurídico, de políticas públicas no sentido de aprimorar e incentivar o emprego de técnicas que otimizem a Justiça brasileira, bem como da disponibilização de dados para a pesquisa na área, a exemplo da plataforma CODEX do CNJ, este trabalho pretende investigar e comparar técnicas de PLN em conjunto com AM, aplicadas na classificação automática de documentos jurídicos do *dataset* VICTOR (ARAUJO et al., 2020a), a fim de possibilitar o entendimento do emprego dessas ferramentas no contexto jurídico, bem como comparar esses resultados com o intuito de subsidiar outras pesquisas ou aplicações práticas na área.

1.2 Questão de Pesquisa

Como as diferentes técnicas de representação computacional da linguagem natural, em conjunto com ferramentas de Aprendizado de Máquina, performam na classificação automática de documentos jurídicos do Brasil?

A questão de pesquisa pode ser subdividida em:

1. Como as técnicas de PLN, Aprendizado de Máquina Tradicional, e *Deep Learning* têm sido empregadas na classificação automática de documentos jurídicos em idioma português do Brasil?

2. Como as diferentes técnicas de PLN para representação computacional de documentos, em conjunto com algoritmos de AM Tradicional e *Deep Learning*, desempenham na classificação automática de documentos jurídicos?

1.3 Hipótese de Pesquisa

As técnicas de classificação de documentos jurídicos do *dataset* VICTOR que empregam *Deep Learning*, seja na fase de representação computacional dos documentos ou também no classificador, apresentam desempenho superior em relação àquelas que empregam métodos estatísticos para essa representação em conjunto com classificadores de AM Tradicional.

1.4 Objetivos

Realizar análise comparativa entre técnicas de classificação automática de documentos do contexto jurídico, especificamente para o *dataset* VICTOR, contrapondo aquelas que empregam classificadores baseados em AM Tradicional em conjunto com diferentes formas de representação computacional de documentos, denominada nesse trabalho de Abordagem I, e àquelas que empregam *Deep Learning*, mais especificamente arquiteturas embasadas em Redes Neurais Recorrentes, denominada de Abordagem II.

1.4.1 Objetivos Específicos

1. Elaborar revisão bibliográfica da literatura referente à classificação automática de documentos no contexto jurídico do Brasil.

2. Realizar análise estatística preliminar do *dataset* empregado na tarefa de classificação.

3. Comparar as técnicas de classificação automática de documentos jurídicos entre as duas Abordagens propostas;

4. Avaliar e comparar os resultados obtidos no *item* 3 em relação ao desempenho na tarefa de classificação.

1.5 Organização do Documento

No Capítulo 2 são abordados conceitos iniciais de IA, Aprendizado de Máquina e PLN. Também são revisados os trabalhos relacionados na área de classificação automática de documentos

jurídicos, com foco, mas não exclusivamente, naqueles que são aplicados no contexto jurídico brasileiro.

As ferramentas empregadas nos processos de classificação automática de documentos são apresentadas no Capítulo 3. Por sua vez, no Capítulo 4 é descrita a metodologia de coleta e tratamento dos dados do *dataset* empregado na classificação, bem como a metodologia aplicada nos experimentos.

Na sequência, o Capítulo 5 apresenta as configurações e resultados dos experimentos efetivados a partir das ferramentas indicadas no Capítulo 3, comparando e discutindo esses resultados. Ao final, no Capítulo 6, as conclusões obtidas, contribuições da pesquisa, e proposição de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 Conceitos Iniciais de PLN, AM e Classificação Automática de Documentos

O Processamento de Linguagem Natural (PLN), de acordo com (MANNING; SCHÜTZE, 1999), consiste no emprego de abordagens quantitativas e probabilísticas para a automação do processamento de textos e fala, bem como para análise e representação da linguagem natural, seja de forma escrita ou fonética (POLO et al., 2021). Assim, por definição, o PLN é ferramenta fundamental nas tarefas dentro do contexto jurídico, uma vez que a linguagem natural é forma primordial da representação de todo o trabalho dentro do Direito.

Desse modo, (SEBASTIANI, 2002) define como uma das possíveis tarefas dentro do campo de aplicação do PLN, a classificação de documentos ou textos, que consiste em agrupá-los em categorias pré-definidas, de tal modo que documentos pertencentes a uma mesma categoria possuam similaridades semânticas significativas entre si. A classificação de documentos possibilita: a consulta, o armazenamento, e a extração de informações que podem ser aplicadas de forma inteligente e automática, por meio de algoritmos de Aprendizado de Máquina (AM).

Por Aprendizado de Máquina (AM), conforme definido em (RUSSELL; NORVIG, 2010), entende-se como sendo um subcampo da Ciência da Computação e da Inteligência Artificial, que estuda a aplicação de programas de computadores capazes de "aprender" a partir da experiência de exemplos anteriores. Tal técnica baseia-se na hipótese de que, a partir de uma grande quantidade de dados rotulados, é possível inferir padrões estatísticos escondidos neles, e assim rotular novos dados não conhecidos, sendo, portanto, denominada de aprendizado de máquina supervisionado. Nesse sentido, o que se denomina nesse trabalho como Aprendizado de Máquina Tradicional é um sub-grupo que contempla algoritmos não baseados em Redes Neurais, a exemplo das Máquinas de Vetores de Suporte (Support Vectors Machines - SVM) e Regressão Logística. Para os algoritmos de AM alicerçados em modelos neurais, denomina-se *Deep Learning*.

No processo de aplicação dos algoritmos de AM e PLN na classificação automática de documentos, de modo geral, há três etapas bem definidas que em conjunto são designadas aqui como *pipeline*, são elas: 1) pré-processamento do texto, com o objetivo de normalizar os dados; 2) representação computacional do texto; 3) e por fim aplicação do algoritmo de AM para execução da tarefa de classificação. A Figura 1 ilustra as etapas do *pipeline* padrão da aplicação de PLN e AM, exemplificado para uma tarefa de classificação de documentos.

Tal como exposto na Figura 1, o pré-processamento é o primeiro passo, e compreende basicamente a normalização do texto, por exemplo, por meio da remoção de palavras ou *to-*

Figura 1 – Pipeline padrão de aplicação de PLN.

kens[1] que não trazem informação relevante dentro da tarefa alvo, no caso deste trabalho a classificação de documentos. Exemplos de itens normalmente removidos com essa técnica são: pontuações, palavras de ligação, números, palavras sem significância semântica, etc. Outra técnica de pré-processamento é a remoção de afixos, ou redução da palavra para um radical de mesma significância. A priori, o que precisa ficar claro nesse momento, é que essa é uma fase de "limpeza" do texto, e que dependendo da metodologia empregada na representação computacional do documento, estatística ou *Deep Learning*, ela pode ocorrer em menor ou maior grau com o intuito de aprimorar os resultados obtidos nas fases seguintes do *pipeline*.

Após o pré-processamento, ocorre a etapa que representa de fato o processamento da linguagem natural, pois é nela que o documento em linguagem natural é transformado para uma representação computacional viável de ser aplicada em algoritmos de AM. Ao longo dos anos, e com a evolução das pesquisas e aplicações no PLN, o processo de transformação do texto para a representação computacional vem sofrendo evoluções que aprimoram a aplicação da PLN nas mais diversas tarefas, a exemplo da tradução automática.

No início das pesquisas e aplicações em PLN, os algoritmos de transformações para a representação computacional de documentos não consideravam, ou consideravam muito pouco, o contexto entre as diferentes palavras e sentenças de um texto (GOLDBERG, 2016). Como exemplo dessas transformações, a mais utilizada é a técnica de *Bag-of-Words* (BoW), que representa numericamente um documento por meio de um vetor que possui o tamanho do vocabulário do *corpus* de documentos a ser analisado, e cada posição fixa nesse vetor representa uma palavra ou *token* desse vocabulário, desconsiderando a ordem das palavras dentro do documentos, sendo que o valor depende basicamente se ele consta ou não no documento representado pelo vetor.

Portanto, com a transformação do documento para representação computacional, a exemplo do BoW, é possível entregá-lo como entrada para a fase seguinte que efetiva o processamento desta representação com o objetivo de resolver a tarefa alvo, caracterizado pelo algoritmo de AM na Figura 1. Exemplos de algoritmos de AM comumente empregados em tarefas de classificação de um modo geral, e não apenas de documentos, são: SVM, Regressão Logística (MANNING; SCHÜTZE, 1999), e Redes Neurais.

A limitação em relação ao contexto das palavras no documentos, não considerada em

[1] Podem ser palavras, subpalavras, números, união de palavras, ou outra sequência qualquer de caracteres que tenham significado semântico no contexto do documento.

representações computacionais da linguagem natural a exemplo do BoW, começa a ser superada a partir da aplicação de modelos neurais nos métodos de representação que, pelo poder de aproximação de funções (HORNIK; STINCHCOMBE; WHITE, 1989), ganham destaque nas discussões científicas sobre várias tarefas de PLN, de tal modo que as arquiteturas adotadas nas resoluções de problemas da área, de certa forma, uniformizam-se em *pipelines* que empregam *Deep Learning* em pelo menos uma das fases, seja no processo de representação computacional, seja no algoritmo de classificação, ou mesmo em ambas.

As redes neurais possibilitaram o aprimoramento na representação computacional de textos, a exemplo dos algoritmos Word2Vec (MIKOLOV et al., 2013) e FastText (BOJANOWSKI et al., 2017), que consideram o contexto da palavra no idioma para a representação numérica. Porém, ainda assim, o contexto usado na representação pelo Word2Vec ou FastText, é limitado ao *corpus* de documentos empregado no processo de pré-treinamento dos algoritmos. O pré-treinamento consiste no processo de "ensinar" aos algoritmos como as palavras de um determinado idioma se relacionam em termos semânticos.

Além disso, modelo neurais têm sido empregados não apenas na fase do *pipeline* de representação computacional, mas também para a a execução da tarefa final, seja de classificação, tradução, sumarização, etc. De modo geral, quando os algoritmos de *Deep Learning* são aplicados no *pipeline*, eles já englobam as duas últimas fases, ou seja, de representação e execução da tarefa de classificação. Isto permite durante o treinamento desses algoritmos, que o processo de captura do contexto dos documentos para a representação computacional seja mais intrínseca ao *corpus* em análise, e não apenas ao contexto geral da língua.

Como exemplos de algoritmos que empregam em conjunto as fases de transformação e classificação, pode-se relacionar: Redes Neurais Convolucionais (CNN - *Convolutional Neural Network*) (KIM, 2014); Redes Neurais Recorrentes do tipo LSTM (*Long Short Term Memory*) (BRAZ et al., 2018); e Redes Neurais Recorrente Hierárquicas com Mecanismo de Atenção (HAN - *Hierarchical Attention Networks*) (YANG et al., 2016), essas últimas destacando-se por conseguirem capturar representações de forma recorrente, onde as palavras anteriores, portanto o contexto, interferem no processo de representação numérica do documento ou sentença.

As pesquisas em PLN e sua aplicação no *Deep Learning*, evoluem no sentido de aperfeiçoar a representação computacional do idioma, uma vez que essa é a matéria-prima para execução das tarefas no campo do Processamento de Linguagem Natural. Desse modo, novas formas de representação da língua estão em desenvolvimento constante, a exemplo dos Modelos de Linguagem (ML), que, de forma sucinta, são algoritmos de *Deep Learning* treinados com um *corpus* da ordem de milhões ou bilhões de documentos, das mais diversas áreas do conhecimento, com o objetivo de capturar a semântica e contexto de um determinado idioma de forma mais generalizada (HOWARD; RUDER, 2018). Esta fase de generalização denomina-se pré-treinamento, e possibilita que a representação computacional de um documento seja influenciada a partir do que o ML "aprendeu" sobre o idioma alvo.

O ML, após o pré-treinamento, pode ser empregado em diversas tarefas de processamento de linguagem natural, o que denota a transferência de aprendizado dentro do PLN (SOUZA; NOGUEIRA; LOTUFO, 2020). Como exemplos de MLs tem-se: ULMFiT e BERT, apresentados nos trabalhos (HOWARD; RUDER, 2018) e (DEVLIN et al., 2019), respectivamente. A transferência do aprendizado, de forma sucinta, viabiliza que aplicações onde a quantidade de documentos disponíveis para treinamento seja insuficiente, possam empregar o ML pré-treinado como ponto de partida na solução, diminuindo a dependência de mais exemplos de treinamentos para a obtenção de resultados satisfatórios em tarefas de PLN.

Assim, define-se para este trabalho, duas categorias básicas de algoritmos: aqueles denominados aqui de Aprendizado de Máquina Tradicional, ou seja, que não possuem no seu algoritmo de aprendizado arquiteturas de redes neurais artificias, a exemplo do SVM e Regressão Logística; e os algoritmos de *Deep Learning*, a exemplo das Redes Recorrentes Hierárquicas (HAN) e Modelos de Linguagem. Na Seção 2.2, são apresentados estudos que aplicam e comparam o emprego de PLN, AM Tradicional e *Deep Learning*, na tarefa de classificação automática de documentos jurídicos.

2.2 Trabalhos Relacionados

A partir de 2017, os trabalhos (SULEA et al., 2017a) e (SULEA et al., 2017b) surgem como referenciais na área de classificação de documentos jurídicos, uma vez que empregam técnicas de aprendizado de máquina (AM) tradicional para a classificação de textos da mais alta Corte Francesa.

Em (SULEA et al., 2017a), o trabalho aborda a classificação de acordo com a área legal a que o documento pertence, e também prediz o resultado da decisão a partir de rótulos extraídos do texto, que por sua vez é subdividida em duas tarefas, com 6 e 8 classes em cada. Para tal, emprega um *pipeline* que inclui, além do pré-processamento, a transformação do texto com *Bag-of-Words* (BOW), e em seguida a aplicação do algoritmo SVM com núcleo linear para a classificação em si.

Considerando a primeira tarefa, para definir a área legal, o índice de F1-score alcançado é de $0,903$, enquanto para as tarefas de predição do resultado das decisões, alcança índices ainda mais elevados de $0,97$ e $0,927$ para as tarefas com 6 e 8 classes respectivamente.

Em (SULEA et al., 2017b), com as mesmas tarefas alvo do trabalho anterior, os autores aplicam *ensemble*, que resumidamente significa a mescla do resultado de diversos classificadores SVM, o que resulta em melhoria no desempenho em comparação à primeira publicação, onde atinge o índice de 0,965 de F1-score para a tarefa de predição da área jurídica, e no caso das tarefas de predição da decisão, a performance é de $0,986$ e $0,958$ para F1-score, respectivamente para 6 e 8 classes.

Importante destacar, que os autores de (SULEA et al., 2017b) e (SULEA et al., 2017a) reforçam que o objetivo das pesquisas na área não é substituir os operadores do Direito pelos métodos propostos de classificação automática, mas sim possibilitar que o AM seja um aliado, por exemplo, no entendimento das possibilidades de vencer ou não uma caso. A indicação clara dos objetivos das pesquisas citadas, é importante a fim de evitar resistência dos profissionais e órgãos da área jurídica, principalmente na disponibilização de dados e conhecimento de negócio para as pesquisas.

Trazendo para a realidade brasileira, as pesquisas em PLN também evoluíram no sentido de incluir o contexto jurídico, nesse ponto pode-se citar os trabalhos (NOGUTI; VELLASQUES; OLIVEIRA, 2020), (ARAUJO et al., 2020a) e (SILVA; MAIA, 2020), como exemplos da aplicação da classificação de documentos jurídicos por meio de algoritmos de aprendizado de máquina tradicional e *Deep Learning*. Além deles, as publicações (MENEZES-NETO; CLEMENTINO, 2022) e (ARAUJO; CAMPOS; SOUSA, 2020b) empregam MLs estado da arte na classificação de documentos jurídicos, tais como o ULMFiT e BERT.

Ao se investigar (NOGUTI; VELLASQUES; OLIVEIRA, 2020), os autores realizam a classificação automática de petições eletrônicas do Ministério Público Estadual do Paraná, comparando *pipelines* que incluem no processo de representação computacional dos documentos as técnicas de TF-IDF (*Term Frequency — Inverse Data Frequency* (ZHANG; YOSHIDA; TANG, 2011)) e Word2Vec (PLN com modelo neural), bem como para o classificador os algoritmos de AM tradicionais: SVM e Regressão Logística. Além disso, experimentam também *pipelines* com Redes Convolucionais e Redes Recorrentes, nesse último caso, LSTM e GRU (*Gated Recurrente Network* (CHO et al., 2014)). Assim, além de propor uma solução para a tarefa de classificação em si, compara as combinações possíveis em termos de métodos de representação computacional dos documentos, e de algoritmos de AM empregados.

Como conclusão, (NOGUTI; VELLASQUES; OLIVEIRA, 2020) identifica para ambos algoritmos de AM tradicionais aplicados na classificação, resultados superiores a partir da transformação do texto com o método TF-IDF, com índice de $0,83$ de F1-score para SVM e Regressão Logística, contrapondo índices de $0,82$ e $0,80$, respectivamente, quando a representação computacional é realizada por meio do Word2Vec. Esse resultado é importante, uma vez que o TF-IDF é um método menos custoso computacionalmente, dado que o Word2Vec exige o treinamento pretérito de um modelo neural a partir de um *corpus* de documentos.

Quando (NOGUTI; VELLASQUES; OLIVEIRA, 2020) compara os métodos elencados no parágrafo anterior com *pipelines* que aplicam *Deep Learning*, verifica-se que o LSTM obtém $0,85$ de F1-socre, seguido pelo GRU com $0,84$ e CNN com $0,82$. Importante observar que a técnica que obteve o pior resultado dentre os citados, no caso o CNN, também teve desempenho inferior em relação ao *pipeline* que emprega TF-IDF e SVM ou Regressão Logística, conclusão interessante, uma vez que o CNN é algoritmo bem mais complexo e exige recursos computacionais mais robustos para a mesma tarefa.

Ainda em (NOGUTI; VELLASQUES; OLIVEIRA, 2020), outras duas conclusões importantes são registradas. A primeira evidencia a melhora no desempenho da tarefa de classificação quando os vetores que representam as palavras são gerados especificamente para os documentos da tarefa alvo, seja para Word2Vec, seja para a camada de *embedding* dos algoritmos de *Deep Learning*. A segunda aponta que o processo de sub-amostragem dos dados de treino para diminuir o desbalanceamento entre as classes não melhorou o desempenho.

No mesmo sentido da classificação de documentos jurídicos da mais alta corte francesa realizado em (SULEA et al., 2017b), no Brasil (ARAUJO et al., 2020a) utiliza documentos do Supremo Tribunal Federal (STF) para executar duas tarefas de classificação: a primeira referente ao tipo do documento; e a segunda referente ao tema do processo judicial, este último composto por vários documentos. Também nesse trabalho são comparados os desempenhos de *pipelines* que empregam AM tradicionais (SVM, Naïve Bayes e XGBoost (CHEN; GUESTRIN, 2016)), e de *Deep Learning* (CNN e BiLSTM).

Os resultados em (ARAUJO et al., 2020a), mostram que o *pipeline* de classificação que aplica CNN em conjunto com CRF (*Conditional Random Fields*) apresenta o melhor desempenhos para a classificação dos documentos do *dataset* VICTOR, com índice de F1-sore médio de $0,7740$, porém não muito a frente do SVM com $0,7632$. Importante destacar, que nos *pipelines* do CNN e BiLSTM sem CRF, os índices de F1-score médio estão abaixo do SVM, sendo $0,7584$ e $0,7281$, respectivamente.

Em relação à classificação temática dos processos, composto de um conjunto de documentos, portanto havendo uma quantidade maior de texto a ser processado pelos *pipelines*, o fluxo que inclui a representação computacional do documento com TF-IDF e o classificador XGBoost resultou em performance superior. Nessa tarefa, os algoritmos CNN e BiLSTM não foram testados. Além dos resultados citados, o trabalho também disponibiliza para a comunidade o *dataset* utilizado nos experimentos, denominado “VICTOR”.

Por sua vez, o estudo publicado por (SILVA; MAIA, 2020) empreende a classificação de processos do Tribunal de Justiça de Minas Gerais, e, a semelhança de (ARAUJO et al., 2020a) e (NOGUTI; VELLASQUES; OLIVEIRA, 2020), compara e analisa classificadores baseados em SVM, Naïve Bayes e AdaBoost (FREUND; SCHAPIRE, 1997), com *pipelines* que empregam *Deep Learning*, no caso em tela redes convolucionais (CNN). Ao final, (SILVA; MAIA, 2020) conclui sobre a possibilidade de realizar o processo de classificação automática dos documentos jurídicos, e que, apesar do método que aplica *Deep Learning* apresentar desempenho superior frente aos algoritmos tradicionais, a diferença relativa entre o SVM e o CNN é de $0,01\%$ para a métrica de de F1-score, com índices de $0,9340$ e $0,9350$, respectivamente. Além disso, enquanto o *pipeline* com CNN requer $1,4$ horas de treinamento, o SVM que obtém o melhor resultado entre os métodos tradicionais realizou a mesma tarefa em $14,75$ minutos.

Além de arquiteturas baseadas em redes convolucionais e LSTM, encontra-se estudos, a exemplo de (ARAUJO; CAMPOS; SOUSA, 2020b) e (MENEZES-NETO; CLEMENTINO,

2022) que empregam Modelos de Linguagem (ML) para a classificação de documentos jurídicos do Brasil. MLs são arquiteturas de Aprendizado de Máquina mais robustas do que aquelas comparadas nos trabalhos citados nos parágrafos anteriores, necessitando de mais recursos computacionais em comparação com as demais.

Dessa forma, a fim de investigar a possível superioridade do ULMFiT frente aos algoritmos de PLN e AM tradicionais, (ARAUJO; CAMPOS; SOUSA, 2020b) compara aplicação de *pipelines* com modelos estatísticos de representação computacional (TF-IDF) e classificadores SVM e Naïve Bayes, contrapondo-os ao ULMFiT, para a classificação da origem das publicações do Diário Oficial do Distrito Federal, com 25 rótulos/classes possíveis.

Em síntese, extrai-se de (ARAUJO; CAMPOS; SOUSA, 2020b) especificamente para o *dataset* estudado, que a aplicação do ULMFiT apresentou resultado inferior ao SVM para F1-score médio, com índices de $0,8469$ e $0,8755$ respectivamente. Importante destacar, que o autor indica nas suas conclusões em relação ao ULMFiT, que o tempo de treinamento é mais de 1000 vezes superior ao SVM. Além disso, é preciso considerar que o treinamento do ML necessita da utilização de GPUs (*Graphics Processing Units*), o que eleva os custos de processamento em comparação ao treinamento do SVM.

Dada a importância no estudo da aplicação de MLs no contexto jurído, por tratar-se do estado da arte em PLN, no trabalho (MENEZES-NETO; CLEMENTINO, 2022) é comparado o emprego do ULMFiT no processo de predição de decisões judiciais de documentos do Juizado Especial Federal do Tribunal Regional Federal da 5ª Região (TRF5), com os MLs BERT e Big Bird (ZAHEER et al., 2020).

Necessário destacar, que a predição da decisão representa a classificação do documento nas classes: "apelação[2] aceita"ou "apelação não aceita", portanto um classificador binário. No estudo, o ULMFiT obtém o melhor desempenho nos experimentos com índice MCC (Coeficiente de Correlação de Matthew - *Matthew's Correlation Coefficient*) de $0,3688$, o que aponta a possibilidade da aplicação desse ML como ferramenta de predição das decisões no contexto jurídico, mesmo quando comparadas a outros modelos de linguagem mais complexos, no caso o BERT e Big Bird.

Apesar de não empregar documentos do contexto jurídico do Brasil, (CHALKIDIS; ANDROUTSOPOULOS; ALETRAS, 2019) classifica documentos da Comissão Europeia de Direitos Humanos [3] com algoritmo de redes recorrentes hierárquicas em conjunto com Mecanismos de Atenção (HAN). No estudo são comparados *pipelines* que aplicam as redes hierárquicas conforme proposto em (YANG et al., 2016), o Modelo de Linguagem BERT, além de TF-IDF e SVM. Os resultados indicam que os *pipelines* que empregam mecanismo de atenção apresentam os melhores desempenhos em dois dos 6 experimentos realizados, e nos demais fica atrás apenas do HIER-BERT, que é uma arquitetura hierárquica bem mais complexa em termos da quantidade

[2] Recurso cabível contra sentenças proferidas pelo juízo ao final da lide.

[3] <https://archive.org/details/ECHR-ACL2019>

de parâmetros do modelo. Especificamente em relação à arquitetura proposta em (YANG et al., 2016), salvo melhor juízo, não foi possível identificar trabalhos no contexto jurídico do Brasil, evidenciando, portanto, a importância na análise e comparação dessa arquitetura.

Após o exame dos trabalhos relacionados, verifica-se tanto para as pesquisas com documentos jurídicos do Brasil, como aquelas exemplificadas que empregam em idioma inglês ou francês, a importância de: 1) comparação do desempenho na classificação de métodos de PLN diferentes para a representação computacional, a exemplo do TF-IDF e Word2Vec; e 2) comparação do desempenho entre classificadores que aplicam em seu *pipeline* algoritmos de aprendizado de máquina tradicionais e *Deep Learning*, incluindo neste último os Modelos de Linguagem. Em relação aos documentos jurídicos do Brasil, entende-se que há espaço para estudo comparando e analisando categorias diferentes dos *pipelines* de classificação, com a inclusão de experimentos com redes hierárquicas que empregam mecanismo de atenção e também de Modelos de Linguagem.

No Capítulo 3, são detalhadas as ferramentas empregadas nos experimentos, além das métricas de desempenho usadas na comparação dos *pipelines* experimentados.

3 REFERENCIAL TEÓRICO

3.1 Tarefa de Classificação Automática de Documentos

A tarefa de classificação em diferentes rótulos, ou entidades, é presente em diversos campos das atividades humanas, inclusive no contexto Jurídico. Essa tarefa, no campo do Processamento de Linguagem Natural (PLN), consiste em indicar o rótulo da classe a que um determinado documento pertence a partir de uma representação desse texto, que pode ser por meio de métodos simbólicos, estatísticos, de redes neurais ou híbridos (ZONG; XIA; ZHANG, 2021).

De acordo com (MANNING; RAGHAVAN; SCHüTZE, 2008), com base nessa representação, e de documentos previamente rotulados nas classes pré-definidas para a tarefa, arquiteturas de classificação de documentos automáticos são treinadas com base nesses exemplos com a finalidade de predizer, a partir de um novo documento não rotulado, a qual classe ele pertence. Neste caso, denomina-se esse processo de treinamento como supervisionado, uma vez que há exemplos de documentos previamente rotulados.

3.1.1 O problema da classificação de documentos

Em (MANNING; RAGHAVAN; SCHüTZE, 2008), o problema de classificação é descrito como a tarefa de designar um documento $d \in \mathbb{X}$, onde $\mathbb{X}$ é o espaço de documentos analisados (*córpus*), em uma classe $c \in \mathbb{C} = \{c_1, c_2, \ldots, c_j\}$. Comumente, a representação espacial dos documentos $\mathbb{X}$ possui dimensão de centenas ou até milhares de posições, e as classes, também chamadas de rótulos, são definidas por especialistas humanos de acordo com as necessidades da aplicação.

Portanto, para o ajuste de um classificador, é dado um conjunto de treino D de documentos rotulados $\langle d, c \rangle$, onde $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$. A partir de um método de aprendizado de máquina, ajusta-se um classificador, ou função de classificação γ que mapeia documentos do mesmo domínio nas suas respectivas classes:

$$\gamma : \mathbb{X} \to \mathbb{C}. \tag{3.1}$$

O tipo de aprendizado empregado, portanto, é o supervisionado, uma vez que um supervisor humano define as classes e rotula os documentos para o treinamento a partir de um algoritmo de aprendizado Γ, que gera a função de classificação γ, ou seja: $\Gamma(\mathbb{D}) = \gamma$.

A partir da Seção 3.3, são apresentados os métodos de transformação dos dados e algoritmos empregados nas arquiteturas de classificação experimentadas nesse estudo.

3.2 Pré-processamento

3.2.1 Tokenização

A definição exata do que é um *token* não é muito precisa, e está sujeito a mudanças, de acordo com (MANNING; SCHÜTZE, 1999). Diferentes conceitos são definidos de acordo com o domínio de estudo e da aplicação do *corpus*. Uma das definições mais aceitas de *token* é de uma cadeia de caracteres alfanuméricos, com espaços em ambos lados, com a possibilidade de incluir hífens ou apóstrofes, mas não sinais de pontuação (MAVERICK, 1969).

Nesse sentido, a tokenização consiste em separar o *corpus* nos *tokens*, conforme regras definidas. Para o caso em questão, considera-se apenas o espaço em branco entre as palavras. A separação em *tokens*, permite dentre outras tarefas, a definição do vocabulário do *corpus*, função de grande relevância no PLN.

3.2.2 Remoção de *Stopwords*

Stopwords são palavras, ou mesmo *tokens* que podem não apresentar significância semântica relevante na análise do documento a depender da tarefa e da técnica empregada na representação computacional. Por exemplo, normalmente a remoção de *stopwords* é empregada em processos de representação que não consideram o contexto das palavras, tal como o TF-IDF, porém não é aplicada naquelas representações onde o contexto, ou seja, a sequência das palavras importa, a exemplo do BERT e ULMFiT.

Os *stopwords* podem ser pontuações, números, palavras de ligação, preposições, links, abreviações, ou até mesmo combinações de caracteres alfanuméricos definidos por meio de expressões regulares, dentre outros. De acordo com (MANNING; SCHÜTZE, 1999) remoção de *stopwords* tende a melhorar a performance dos algoritmos de Aprendizado de Máquina.

3.2.3 Lematização

Assim como a remoção de *stopwords*, a lematização normalmente é empregada em *pipelines* que empregam BoW como método de representação computacional, e consiste em reduzir a palavra na sua base de significado. Como exemplo pode-se citar os verbos, que nesse caso tem suas flexões reduzidas para o infinitivo do verbo, ou seja, caso o texto contenha a palavra "processaram", essa será substituída por "processar". O mesmo raciocínio vale para o plural e gênero, que tem as palavras reduzidas para o singular e masculino.

3.3 Representação Computacional de Documentos

Para o treinamento dos algoritmos de aprendizagem de máquina selecionados como base de comparação para os testes das hipóteses, é necessária a transformação deles para uma

representação computacional, ou numérica. Nesse trabalho de pesquisa são utilizadas as técnicas de *Term Frequency-Inverse Document Frequency* - TF-IDF, e *word embeddings* com Word2Vec.

3.3.1 *Term Frequency-Inverse Document Frequency* (TF-IDF)

O TF-IDF (ZHANG; YOSHIDA; TANG, 2011) é um método estatístico para definir a importância relativa de um determinado termo (*token*) no documento, em relação ao *corpus* de todos os documentos do *dataset*. Para o cálculo utiliza-se a Equação 3.2

$$\text{TF-IDF}(t,d) = tf(t,d) \times idf(t), \tag{3.2}$$

onde:

$$tf(t,d) = \frac{Q_t}{Q_d} \tag{3.3}$$

Q_t *- quantidade de tokens "t" no documento "d".*

Q_d *- quantidade de tokens no documento "d".*

$$idf(t) = -\log\left[\frac{df(t)}{N}\right]. \tag{3.4}$$

N - número de documentos no dataset.

df(t) - número de documentos que contém o token "t" em todo o dataset.

Portanto, cada documento é representado por um vetor com *v* posições, onde *v* é o vocabulário definido conforme a Seção 4.4. Em cada posição desse vetor, que é fixa para cada token, o valor do TF-IDF é calculado conforme Equações 3.3, 3.4 e 3.2.

3.3.2 Word2Vec

O método de vetorização *Word2Vec*, inicialmente proposta por (MIKOLOV et al., 2013), representa as palavras por meio de vetores de acordo com o significado delas dentro do vocabulário da língua. Os vetores (*embeddings*) das palavras, são gerados a partir do treinamento de uma rede neural cujo objetivo é prever uma palavra a partir de um contexto (CBOW - Continous Bag-of-Words), ou prever o contexto de uma palavra a partir dela (Skip-gram), conforme mostrado na Figura 2

Por meio do treinamento dessas arquiteturas em grandes corpus de uma língua específica, incluindo documentos de diversas fontes, como: Wikipedia, textos científicos, notícias, blogs, livros, dentre outros, espera-se que os vetores representem no espaço vetorial as relações sintáticas e semânticas entre as palavras.

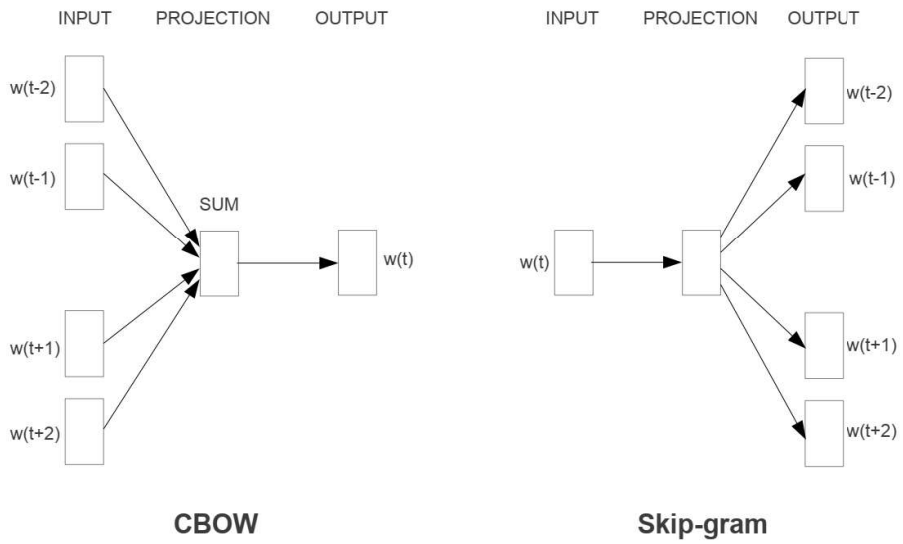


Figura 2 – Arquiteturas das redes para treinamento de vetores word2vec CBOW e Skip-gram. Fonte: (MIKOLOV et al., 2013)

O *Skip-gram* emprega um método de treinamento semi-supervisionado, para realizar a predição do entorno de uma palavra respectiva dentro do texto, assim, nos experimentos realizados em (HARTMANN et al., 2017b), esse método superou o CBOW para o português do Brasil, sendo, portanto, a técnica escolhida para a transformação Word2Vec nos experimentos desta dissertação.

3.3.3 FastText

FastText (BOJANOWSKI et al., 2017; JOULIN et al., 2017) é um método de *embedding* de palavras semelhante ao *Word2Vec*, porém os *embeddings* pré-treinados são associados a conjuntos de caracteres denominados *n-grams*, que não necessariamente formam uma palavra, mas são partes delas, ou seja, as palavras são representadas como concatenações desses *n-grams*. Dessa forma, o vetor que representa uma palavra ou *token*, é definido como a soma dos vetores de *embedding* de cada *n-gram* que forma a palavra, assim o método tenta capturar relações morfológicas mais precisas, e ainda possibilita que uma palavra inexistente no vocabulário inicial do modelo *FastText* pré-treinado, possa ser formada a partir da união dos *n-grams* pré-treinados.

3.3.4 *Document Embedding*

Ao contrário da vetorização com TF-IDF, que gera de plano um vetor que representa o documento dentro de um corpus, o *Word2Vec* e *FastText* gera um vetor para cada *token* do documento, assim, é necessário definir uma representação no espaço vetorial para o documento em si. Para tanto, utiliza-se a média aritmética simples dos vetores de cada *token* que compõem a peça processual, e que constem no vocabulário definido para os experimentos, tal como indicado em (KIM; KIM; CHO, 2017). A Equação 3.5 mostra como calcular cada elemento desse vetor.

$$docvec = \frac{1}{Q_d} \sum_{k=1}^{Q_d} wordvec_k. \tag{3.5}$$

wordvec , vetor que representa o token

docvec, vetor que representa o documento

3.3.5 BERT

BERT (DEVLIN et al., 2019) é um modelo de linguagem baseado em *Transformers* (VASWANI et al., 2017). Para efetivar a transformação do texto em vetor contínuo, ou seja, o processo de *embedding* desse texto, é utilizada uma das saída do modelo BERT denominada *C*, conforme indicado na Figura 3. Essa saída é um vetor contínuo que representa a sequência de *tokens* do texto dado como entrada para o modelo, que de acordo com os autores proponentes do modelo, o vetor *C* pode ser empregado em tarefas de classificação de documentos. Em virtude de limitação da própria arquitetura do BERT, a sequência máxima de *tokens* que ele recebe como entrada é de 512. Tal como outros modelos de linguagem, o BERT pode ter seus parâmetros ajustados para os dados do problema em análise, por meio da fase de ajuste-fino.

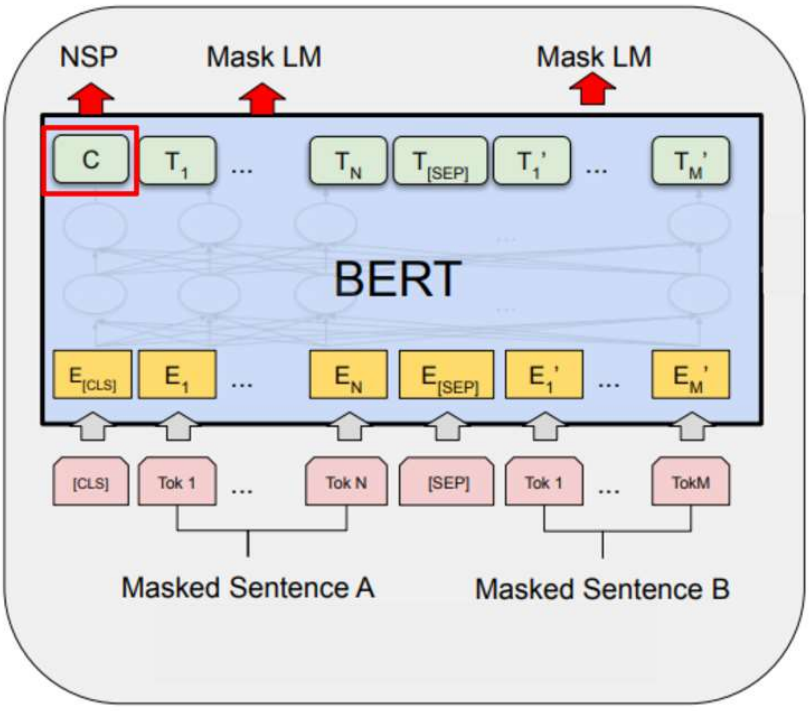


Figura 3 – Arquitetura do BERT, com indicação do vetor *C* na saída. Fonte: Adaptado de (DEVLIN et al., 2019)

3.3.6 Camada de *Embedding*

A camada de *Embedding* é uma camada neural *Multilayer Perceptron* (MLP) incluída no início de arquiteturas que recebem como entrada dados no formato de texto, e são responsáveis pela transformação de vetores de alta dimensão, a exemplo do BoW que representam palavras no formato *one-hot*[1], para vetores de baixa dimensão. Assim, as palavras que possuem representação vetorial inicial de dimensão na ordem de dezenas de milhares, após a camada de *embedding* representam a palavra em vetores de dimensões da ordem de centenas.

A camada de *embedding* é necessária no PLN, justamente por possibilitar a transformação computacional do texto, sendo empregada na entrada de arquiteturas como a LSTM e ULMFiT, e tem os parâmetros ajustados em conjunto com a arquitetura completa. No processo de treinamento e ajuste dos parâmetros da camada de *embedding*, pretende-se que o ajuste possibilite a transformação dos vetores de entrada em outros de menor dimensão que capturem com mais precisão a relação semântica das palavras. É um processo semelhante ao Word2Vec, porém totalmente dependente dos dados empregados no problema.

3.4 Algoritmos de Aprendizado de Máquina Tradicional para Classificação

Nesta Seção, são detalhados os algoritmos categorizados no trabalho como Aprendizado de Máquina Tradicional, e aplicados nos experimentos da Abordagem I em conjunto com os métodos de representação computacional indicados na Seção 3.3.

3.4.1 Regressão Logística

A regressão Logística é um dos modelos usados para a predição de variáveis categóricas, normalmente binárias. Portanto, tal como definido em (BISHOP, 2006), o algoritmo funciona como um método estatístico com o objetivo de encontrar uma equação que faça a inferência de uma variável binária, a partir de uma ou mais varáveis independentes.

Nesse sentido, o algoritmo recebe como entrada a combinação de *features*, que no caso deste trabalho pode ser o TF-IDF ou Word2Vec resultantes da transformação dos documentos, que são aplicadas em uma função sigmoide, resultando em número real entre 0 e 1, onde a partir de um valor de corte, normalmente 0,5, define-se a divisão de classes.

A Equação 3.6 mostra como é calculada a saída a partir da função linear $f(X)$, onde X é o conjunto de *features* de entrada,

[1] Representação computacional de um documento baseada em vetor composto por 0's e 1's, onde 1 representa a presença de determinada palavra do vocabulário do *corpus* no respectivo documento, e 0 a ausência dessa palavra.

$$y = \frac{1}{1 + e^{-f(X)}} \tag{3.6}$$

e

$$f(X) = \beta_0 + \beta_1 x_1 + ... + \beta_j x_j \tag{3.7}$$

onde $X = (1, x_1, ..., x_j)$, e j é o número de *features* de entrada.

No caso de problemas de classificação multiclasse, são gerados k modelos, onde k é o número de classes, e cada um deles representa uma classe versus as demais.

3.4.2 Support Vector Machines (SVM) de Núcleo Linear

O SVM de núcleo linear é um algoritmo de aprendizado de máquina tradicional bastante empregado para classificação de documentos, dada a sua facilidade de aplicação e disponibilização de bibliotecas computacionais bem eficientes. O objetivo do SVM é criar vetores de suporte que consigam separar as classes em um espaço dimensional z (sendo $z \geq n$), onde n é a dimensão dos dados analisados. Esses vetores são escolhidos de forma a obter a maior margem possível de separação entre as classes, possibilitando assim uma melhor generalização do modelo. A Figura 4 exemplifica o funcionamento do SVM com núcleo linear.

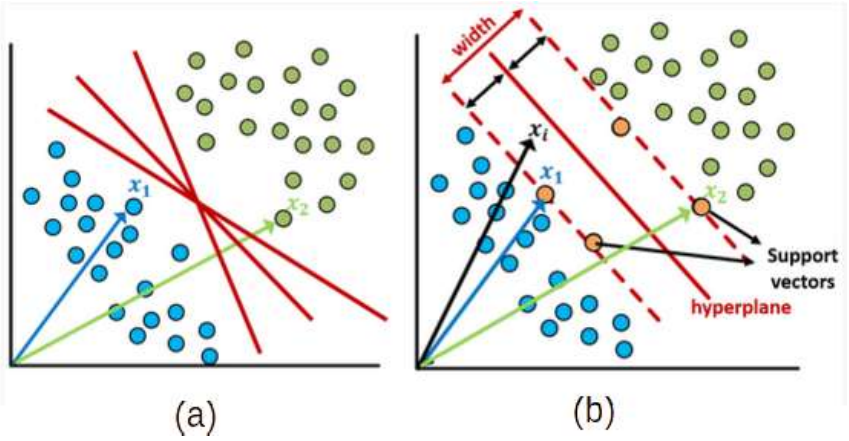


Figura 4 – Possíveis Hiperplanos de separação (a). Hiperplano com a melhor margem de separação (b). Fonte: (SAVAS; DOVIS, 2019)

Na Figura 4, os vetores de suporte são os pontos de dados (x_1 e x_2) mais próximos da função de separação das classes, que é representada pela linha preenchida entre eles, e denominada de hiperplano de separação. O algoritmo SVM, por meio dos exemplos, minimiza o erro de classificação, buscando o melhor hiperplano que separa as classes (pontos verdes e azuis na Figura 4), mantendo a generalização do algoritmo no processo de classificação.

3.5 Algoritmos de *Deep Learning*

Nesta seção, são apresentados os algoritmos de *Deep Learning* empregados nos experimentos da Abordagem II, que ao contrário dos algoritmos de AM Tradicional aplicados na fase de

classificação dos experimentos da Abordagem I, caracterizam por arquiteturas embasadas em redes neurais artificiais, onde a camada de classificação tem seus parâmetros ajustados em conjunto com a camada de representação computacional do documento.

3.5.1 Redes Hierárquicas com Mecanismo de Atenção (HAN)

A HAN (YANG et al., 2016) é composta de uma arquitetura hierárquica de redes recorrentes do tipo GRU (Gated Recurrent Unit), que por sua vez são baseadas em redes LSTM (Long-Short Term Memory), e de forma geral são denominadas de redes neurais neurais recorrentes (Recurrent Neural Network - RNN), sendo pensada com o objetivo de evitar o problema do desaparecimento (*vanishing*) do gradiente, comum em RNNs. A Figura 5 exemplifica o funcionamento de uma RNN de modo geral, e a Figura 6 evidencia a GRU.

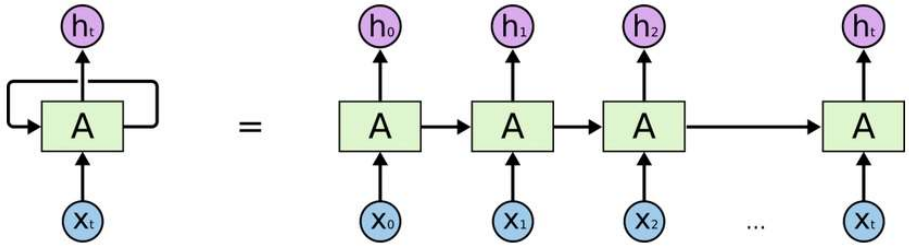


Figura 5 – Funcionamento geral de uma rede recorrente. Fonte: <https://www.deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/>

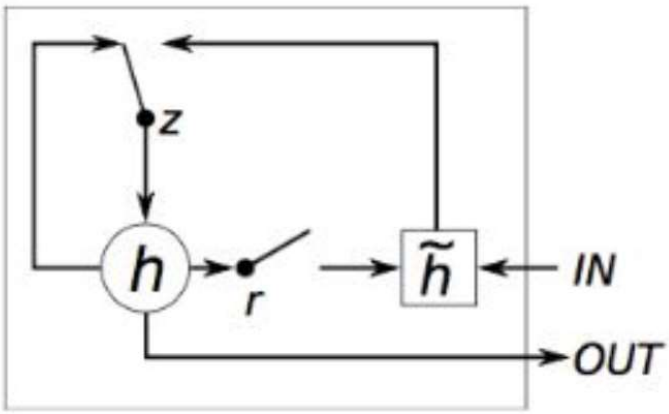


Figura 6 – Ilustração de GRU. Fonte: (CHUNG et al., 2014)

Na Figura 6, r é o *reset gate*, z é o *update gate*, h é a ativação atual, e $\tilde{h}$ é a ativação candidata. O *reset gate* tem a função de medir a ativação anterior em comparação a candidata, com o objetivo de esquecer o estado anterior da célula, enquanto o *update gate* decide se usa ou não o candidato a ativação para atualizar o estado da célula. Essa arquitetura é mais simples em comparação ao LSTM (CHUNG et al., 2014), consequentemente mais rápida para treinamento.

Definido o componente básico da HAN, no caso a GRU, a Figura 7 mostra a arquitetura da rede hierárquica com mecanismo de atenção, que normalmente é composta de 6 camadas:

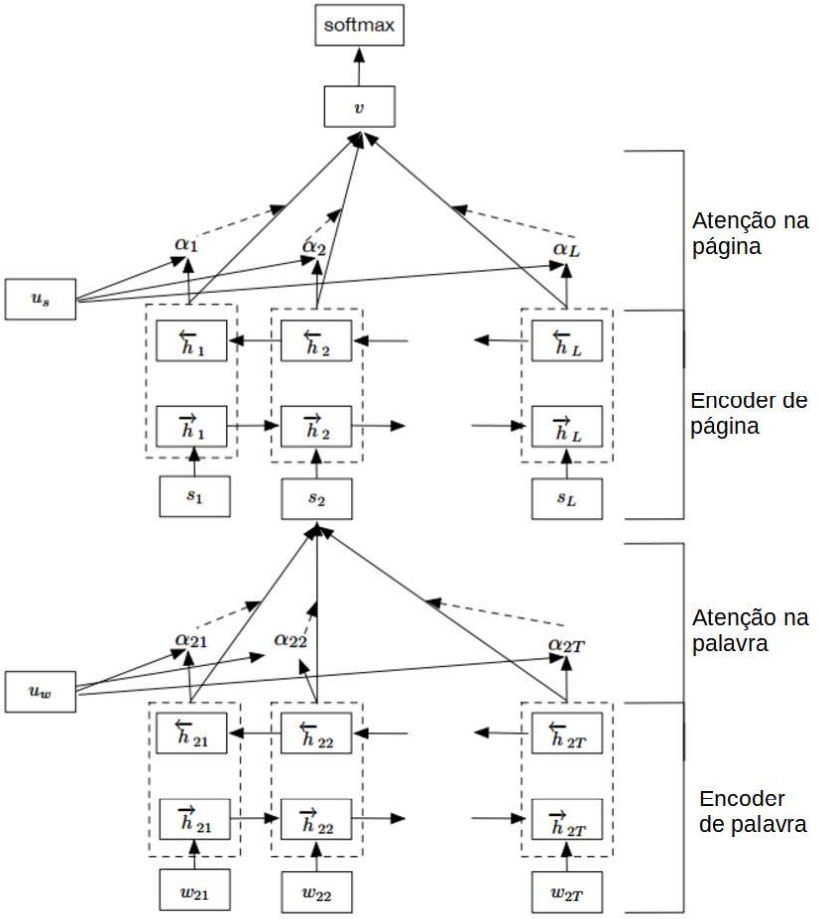


Figura 7 – Arquitetura de uma rede hierárquica com mecanismo de atenção. Fonte:(YANG et al., 2016)

- Camada de *Embedding*: responsável por realizar a transformação computacional das palavras (*tokens*) do documento de entrada;
- Camada de codificação das palavras (*word encoder*): com o objetivo de obter uma representação das palavras por meio da rede recorrente, onde palavras anteriores e posteriores influenciam nesse processo, portanto é uma rede recorrente bi-direcional;
- Camada de atenção para as palavras (*word attention layer*): para capturar a importância da palavra na sentença;
- Camada de codificação das sentenças (*sentence encoder*): com o mesmo objetivo da *word encoder*, porém ao nível de sentenças, e garantindo também a bi-direcionalidade.
- Camada de atenção no nível das sentenças (*sentence attention layer*): para capturar as sentenças mais importantes para o processo de decisão;

- A camada final, que aplica a função de ativação *softmax*, representada na Equação 3.8, que recebe como entrada a saída da camada de atenção ao nível de sentença, caracterizado pelo vetor v com a representação computacional do documento de entrada.

O mecanismo de atenção, empregado ao nível de palavras e sentenças, consiste em capturar as partes mais importantes do texto de entrada, e que influenciem com maior intensidade no processo de decisão do algoritmo de classificação. Matematicamente, o mecanismo de atenção é uma rede neural de múltiplas camadas (*Multilayer Perceptron* - MLP), com o objetivo de aprender a importância relativa de cada palavra ou sentença, em relação às demais. O ajuste dos parâmetros da rede MLP é realizado durante o treinamento da arquitetura completa por meio do *Backpropagation*.

Em virtude do *dataset* empregado nos experimentos, não conter as separações de sentenças por meio de pontuações, o conceito hierárquico do documento será dividido em palavras e páginas, ou seja, ao invés de se ter o *encoder* de sentença, este será substituído pelo *encoder* de página, assim a hierarquia do documento começa nas palavras, em seguida para as páginas, e por fim o documento em si.

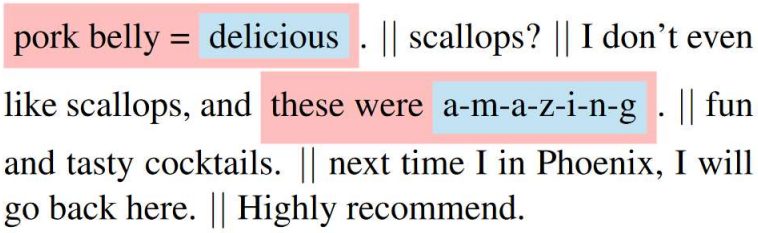


Figura 8 – Exemplo do funcionamento do Mecanismo de Atenção. As sentenças e palavras marcadas, influenciam com maior intensidade no processo de classificação. Fonte:(YANG et al., 2016)

3.5.1.1 Camada densa para classificação dos documentos

A camada de classificação dos documentos é composta por uma rede neural MLP (*multilayer perceptron*), com neurônios *softmax* na saída:

$$p = softmax(W \times v + b), \tag{3.8}$$

onde v é a representação vetorial do documento a ser classificado, W é a matriz de pesos da rede MLP, e b é o bias.

3.5.1.2 Função de perda

Para a atualização dos parâmetros da rede HAN, é empregada a função de perda entropia cruzada (*Cross-Entropy*):

$$L = -\frac{1}{N}\sum_{d=1}^{N}\sum_{i=1}^{n}\hat{y}_{id}\log(p_{id}), \tag{3.9}$$

onde N é a quantidade de documentos dados como entrada na rede antes do ajuste dos parâmetros, n é a quantidade de classes, $\hat{y}_{id}$ é o rótulo correto para o documento d, e p_{id} é a saída da camada *softmax*, que representa a probabilidade do documento d pertencer à classe i.

3.5.2 Metodologia ULMFiT

A metodologia ULMFiT foi proposta como objetivo de obter uma generalização para a transferência indutiva de conhecimento dentro do PLN. Portanto, a hipótese consiste em: dado uma tarefa fonte (*source task*) τ_S, e uma tarefa final qualquer τ_T, com $\tau_S \neq \tau_T$, o modelo de linguagem é visto como a tarefa fonte ideal a fim de aprimorar a performance em τ_T, por exemplo, de classificação de documentos.

Assim, por meio da tarefa fonte, o ML capta características da linguagem que são importantes para as tarefas finais, tais como: dependências de longo termo, relações hierárquicas e sentimento (HOWARD; RUDER, 2018). Além disso, o ML pode ser adaptado para as características específicas da tarefa alvo, por meio do ajuste fino dele, que se mostra relevante para o aprimoramento da performance.

Nesse contexto, o ULMFiT é proposto por (HOWARD; RUDER, 2018), a partir do pré-treinamento dele em um grande *corpus* de textos do domínio geral do conhecimento. O método é dito *universal* no sentido em que consegue: 1) funcionar em tarefas com que variam em termos da quantidade de documentos, do tamanho deles e do tipo de rótulo; 2) utilizar uma única arquitetura e processo de treinamento em todas as fases; 3) não requerer uma engenharia de características (*feature engineering*) ou pré-processamentos elaborados; 4) não necessitar de documentos adicionais do mesmo domínio, além daqueles da tarefa final, tal como nos modelos de linguagem pretéritos ao ULMFiT.

O ULMFiT utiliza a arquitetura AWD-LSTM (MERITY; KESKAR; SOCHER, 2017), que é composta por camadas LSTM, sem mecanismo de atenção, ou outras sofisticações adicionais. Ele consiste dos seguintes passos, conforme exposto na Figura 10, e discutidos com mais detalhes nos itens a seguir.

3.5.2.1 Pré-treinamento

Para o pré-treinamento do modelo, é necessário um corpus de textos com uma quantidade relevante de documentos, como exemplo, no trabalho que propôs a metodologia do ULMFiT, é empregado o Wikitext-103 (MERITY et al., 2016), com 28.595 artigos da Wikipédia em inglês, totalizando mais de 103 milhões de palavras. A tarefa no pré-treinamento consiste no modelo predizer a próxima palavra a partir de uma sequência anterior, conforme exemplificado na Figura 10.

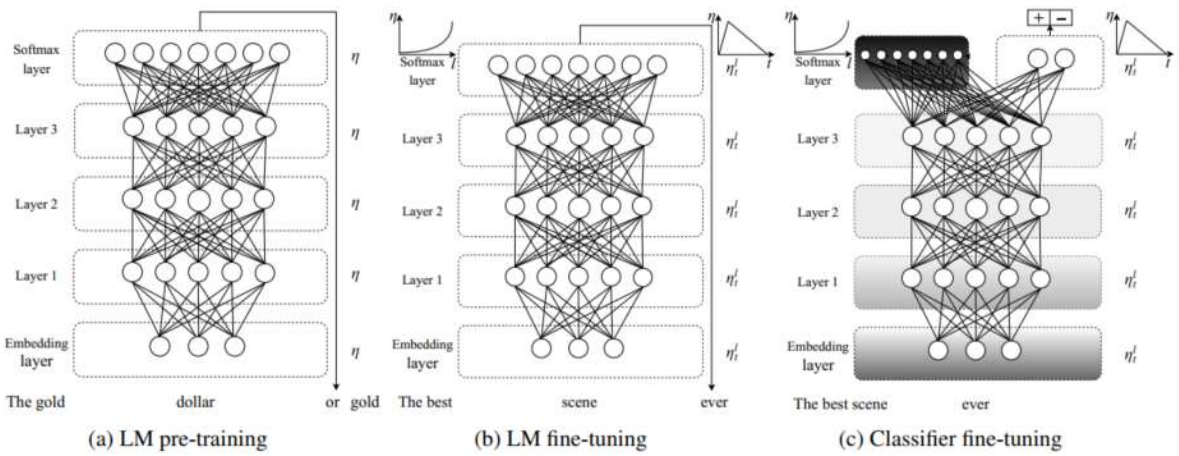


Figura 9 – Estágios do ULMFiT. (a) Pré-treinamento no domínio geral. (b) Ajuste fino nos dados da tarefa final. (c) Ajuste fino do classificador. Fonte:(HOWARD; RUDER, 2018)

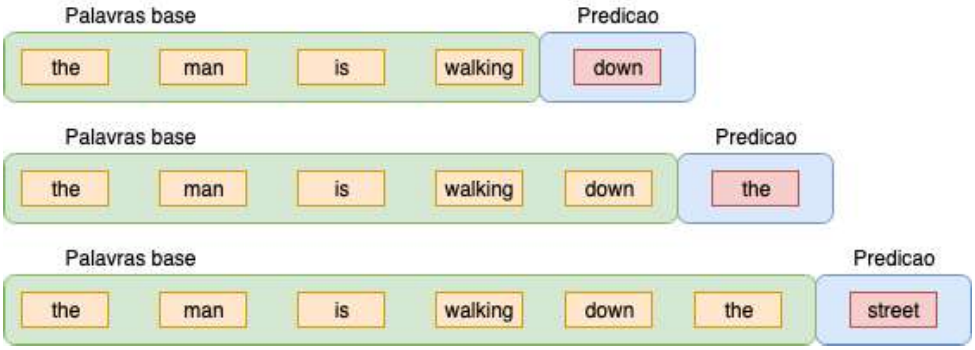


Figura 10 – Exemplo da tarefa empregada no pré-treinamento de um ML. Fonte:<https://medium.com/turing-talks/turing-talks-27-modelos-de-prediÃğÃčo-lstm-df85d87ad210>

A partir da tarefa de predição da próxima palavra, considerando o contexto anterior, os parâmetros da arquitetura do ULMFiT são ajustados com o objetivo de diminuir o erro de predição, e com isso capturar o contexto da língua do corpus que é utilizado no pré-treinamento.

3.5.2.2 Ajuste fino do ULMFiT para os dados do problema

Dado o ML pré-treinado, os parâmetros dele são ajustados a partir da mesma tarefa de prever a palavra seguinte a partir da sequência anterior, porém empregando os dados do problema específico, ou seja, já no contexto próprio da tarefa final, por exemplo de classificação de documentos jurídicos. Para uma melhor adaptação do modelo, e prevenção do problema de "esquecimento catastrófico" (MCCLOSKEY; COHEN, 1989), os autores do ULMFiT propõe o ajuste fino discriminativo e a taxa de aprendizado triangular inclinada.

3.5.2.3 Ajuste do ULMFiT para a Tarefa de Classificação

Após o processo de pré-treinamento e ajuste fino do ML, são incluídas duas camadas adicionais ao modelo com parâmetros inicializados aleatoriamente, *batch normalization* (IOFFE; SZEGEDY, 2015), e *dropout*, com funções de ativação RELU na camada intermediária e *softmax*

na saída, para indicar a probabilidade do documento pertencer a cada classe ou rótulo do conjunto de dados. A primeira camada das duas inseridas nesta fase, recebe como entrada os estados internos agrupados (*pooled*) da última camada da arquitetura LSTM, e que foi pré-treinada e ajustada nas fases anteriores.

Assim, a partir do documento dado como entrada para o Modelo de Linguagem, é gerada uma representação computacional dele, que servirá como entrada para as camadas adicionais, e que, nesta fase, são responsáveis pela classificação em si do documento. Pelo fato da adição dessas novas camadas na arquitetura do ULMFiT, os autores que propuseram o modelo, também recomendam que o ajuste dos parâmetros nessa fase seja por meio da técnica de Descongelamento Gradual de Camadas (*Gradual Unfreezing*).

A técnica do Descongelamento Gradual de Camadas é importante para evitar o problema do esquecimento Catastrófico (MCCLOSKEY; COHEN, 1989) da rede, uma vez que as camadas responsáveis pela classificação, estão sendo sobrepostas a outras que já estão com os parâmetros ajustados pelas fases anteriores. Nesse sentido, o descongelamento gradual de camadas consiste em ajustar os parâmetros de uma camada LSTM por vez, a partir da última para as mais internas, ou seja, atualizando os parâmetros de uma camada por vez a cada iteração do treinamento, até que todas sejam ajustadas ao mesmo tempo e ocorra a consequente convergência do modelo para solução da tarefa final.

3.5.2.4 Modelos de Linguagem Bidirecionais

O conceito de bidirecionalidade, já exemplificado na Seção 3.5.1, pode também ser aplicado em ML, nesse caso, são pré-treinados dois modelos de linguagem, um em cada sentido da sequência de palavras, *forward* (FW), e *backward* (BW). Após, o mesmo processo é realizado para ajuste fino do ML nos dados do problema final, bem como para o ajuste do classificador. Assim, tem-se duas arquiteturas, cada uma recebendo os documentos em um sentido da sequência de palavras, que após a classificação de cada um deles, a predição final é obtida a partir da média das predições em cada direção.

3.6 Métricas de Desempenho

As métricas de avaliação utilizadas para a seleção e comparação dos pipelines são: F1-score, Precisão, *Recall* e Correlação de Matthews (*Matthews Correlation Coefficient* - MCC). Entretanto, antes é necessária as definição de Matriz de Confusão, que dá suporte ao cálculo das métricas de desempenho.

3.6.1 Matriz de Confusão

Para melhor entendimento do conceito, considera-se um problema de classificação binária, ou seja, duas classes, onde uma representa a ocorrência de uma condição e a outra indica a

não ocorrência. Assim, a Tabela 1 representa a tabela de contingência ou Matriz de Confusão (NEGRI, 2021), evidenciando em cada quadrante o total de exemplos que foram inferidos correta ou incorretamente em cada condição.

		Valor Real	
		Condição Positiva	**Condição Negativa**
Predição	**Condição Positiva**	Verdadeiro Positivo (VP)	Falso Positivo (FP)
	Condição Negativa	Falso Negativo (FN)	Verdadeiro Negativo (VN)

Tabela 1 – Tabela de Contingência ou Matriz de Confusão.

Com base nos valores de VP, FP, FN e VN, são calculadas as demais métricas.

3.6.2 Precisão (*Precision*)

A métrica de Precisão indica a proporção de acerto entre todas as classificações positivas que o modelo fez, tal como Equação 3.10.

$$P = \frac{VP}{VP+FP} \tag{3.10}$$

3.6.3 Revocação (*Recall*)

A métrica de Revocação indica a proporção de previsões corretas, dentre todas os exemplos positivos, conforme Equação 3.11.

$$R = \frac{VP}{VP+FN} \tag{3.11}$$

3.6.4 F1-score

A métrica de F1 indica o nível de significância da acurácia e representa uma média harmônica entre Precisão e Revocação, conforme a Equação 3.12, tornando-se, portanto, uma medida que agrega as informações de precisão e revocação em um único índice.

$$F1 = 2 \times \left(\frac{P \times R}{P+R} \right). \tag{3.12}$$

No caso de problemas de classificação com mais de dois rótulos, o F1-score médio pode ser calculado por meio da média simples dos F1-scores de cada classe, ou ainda a partir da média ponderada deles em relação à proporção de distribuição de cada classe dentro do *dataset*. Assim, os F1-scores de classes que apresentam mais exemplos têm maior peso no valor final, do que aquelas que apresentam menos exemplos.

3.6.5 Matthews Correlation Coefficient (MCC)

Apesar de bastante utilizado, o F1-score não leva em consideração os verdadeiros negativos (VN), podendo a induzir interpretações equivocadas quando os exemplos empregados para a avaliação dos resultados não encontram-se distribuídos de forma homogênea entre as duas classes do problema (NEGRI, 2021). Para esses casos, e aprimorar o processo de avaliação, será considerado também o coeficiente de correlação de Matthews (MCC):

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP+FP)(VP+FN)(VN+FP)(VN+FN)}}. \tag{3.13}$$

Os valores MCC estão limitados entre -1 e 1, que indicam predições totalmente discordantes e concordantes, respectivamente, e valores próximos de 0 denotam uma concordância aleatória.

No Capítulo 4, são apresentados o *dataset* e suas variações empregadas nos experimentos, detalhes do pré-processamento e representação computacional dos documentos, além dos *pipelines* de classificação utilizados na comparação, e métodos de seleção de modelos.

4 METODOLOGIA

Neste capítulo é caracterizado o domínio dos dados utilizados nos experimentos, a forma de coleta desses dados, bem como a forma como os experimentos são executados. É descrito também as medidas de avaliação para comparação da arquitetura proposta com outras já sedimentadas na literatura como *baselines*.

4.1 Coleta dos Dados

Conforme disponibilizado pelo trabalho (ARAUJO et al., 2020a), da equipe do Laboratório de Inteligência Artificial da Universidade de Brasília (AILAB)[1], os dados foram solicitados por meio de formulário[2] próprio, o qual indica a forma como eles podem ser utilizados. Após o preenchimento dos dados do formulário e aceite das condicionantes, foi recebido um *link* no e-mail indicado, para *download* do *dataset*.

4.1.1 Caracterização do Domínio

Nos experimentos são utilizados dados relativos ao domínio jurídico da língua portuguesa, tratando-se de peças processuais impetradas ou produzidas no Supremo Tribunal Federal (STF), e coletadas de acordo com o descrito em (ARAUJO et al., 2020a). As peças rotuladas consistem de: Acórdãos, Petição de Recursos Extraordinários (PRE), Agravos de Recurso Extraordinário (ARE), Despacho, Sentença, e Outros, estes últimos não rotulados.

4.1.2 Formato dos dados

Os dados coletados já estão em formato CSV, e dispostos em colunas e linhas, tal como exemplificado na Tabela 2. Em cada linha do *dataset* consta o conteúdo de cada página de um documento ("body"), que pode ser associada com as demais por meio de um código único ("file_name"). Há também a indicação do tipo do documento ("document_type"), conforme definido na Seção 4.1.1, o tipo do tema processual ("theme"), o número do processo ("process_id"), e o número da página ("pages") do documento a que pertence.

[1] <https://ailab.unb.br/>
[2] <https://ailab.unb.br/victor/lrec2020>

file_name	document_type	pages	body
AI_850120_789456_93_11052013.pdf	outros	1	{"processo termo remessa tendo vista disposto ...
AI_850127_273628856_1280_28102014.pdf	outros	1	{"supremo tribunal federal ofício brasília de ...

Tabela 2 – Exemplo dos dados no *dataset* VICTOR.

O texto de cada página já está com a tokenização previamente realizada pela equipe do AILAB, incluindo as citações de Leis e artigos que são substituídos por "LEI_X"e "ARTIGO_X", onde "X"representa o respectivo número da lei ou artigo (SILVA; BRAZ; CAMPOS, 2018).

4.2 Pré-processamento

São aplicados pré-processamentos adicionais aos dos dados, em 3 níveis diferentes: N1, N2 e N3. No N1, são removidos *links*, *e-mails* e expressões comuns em processos judiciais que referenciam folhas dos processos mas que não trazem significados para o processo de classificação, sendo consideradas as seguintes: ‘’fl.”, ‘’fls.”, ‘’fl”, ‘’fls”, ‘’pg.” e ‘’pg”.

No N2, são removidas *stopwords*, selecionadas por meio da biblioteca de Processamento de Linguagem Natural do Python (NLTK) (BIRD; LOPER; KLEIN, 2009), que possui configurações para diversos idiomas, dentre eles o Português do Brasil. No N3, último nível de pré-processamento, os documentos após passarem pelos níveis 1 e 2, recebem o processo de lematização, que consiste na redução das inflexões das palavras, ou seja, traz os verbos para os infinitivos, plural para singular, por exemplo, e também a formação de *bigrams*, que traduz-se pela formação de um vocábulo a partir da junção de duas palavras que comumente são encontradas juntas no texto, por exemplo, nomes compostos de cidades. A Figura 11 indica todos os 3 níveis de pré-processamento.

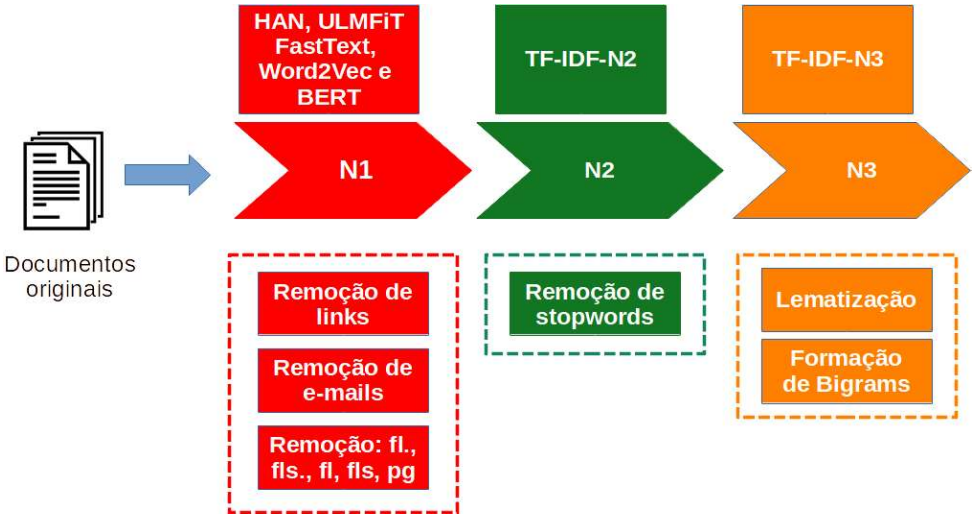
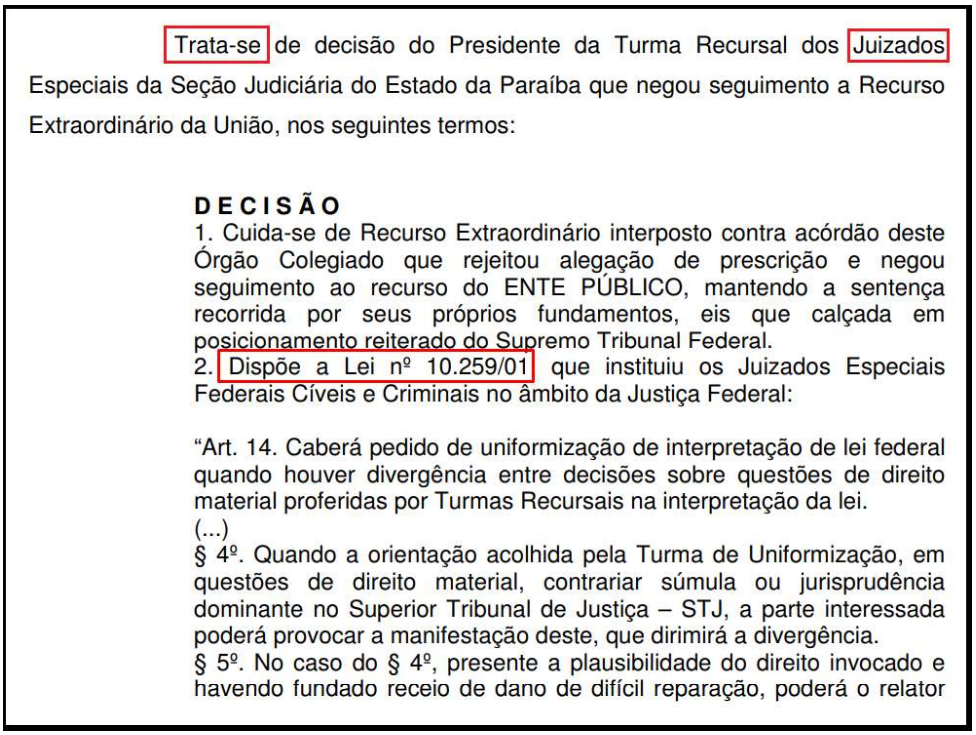


Figura 11 – Níveis de pré-processamento dos documentos, com a indicação de quais algoritmos onde cada um deles é aplicado.

Na Figura 11, observa-se que o N1 é aplicado aos *pipelines* de classificação que posseum processos de transformação diferentes do TF-IDF, enquanto os N1 e N2 são aplicados apenas no TF-IDF, com a distinção entre TF-IDF-N2 e TF-IDF-N3, para observar o impacto da inclusão da lematização e formação de bigrams nos resultados. Na Figura 12 é exibido o exemplo de um trecho do documento em PDF (original), e no parágrafo seguinte o respectivo texto extraído

após o pré-processamento com os 3 níveis, a exceção da formação de bigrams, pois é uma etapa executada pelo próprio algoritmos da transformação TF-IDF.



Trata-se de decisão do Presidente da Turma Recursal dos Juizados Especiais da Seção Judiciária do Estado da Paraíba que negou seguimento a Recurso Extraordinário da União, nos seguintes termos:

DECISÃO

1. Cuida-se de Recurso Extraordinário interposto contra acórdão deste Órgão Colegiado que rejeitou alegação de prescrição e negou seguimento ao recurso do ENTE PÚBLICO, mantendo a sentença recorrida por seus próprios fundamentos, eis que calçada em posicionamento reiterado do Supremo Tribunal Federal.
2. Dispõe a Lei nº 10.259/01 que instituiu os Juizados Especiais Federais Cíveis e Criminais no âmbito da Justiça Federal:

"Art. 14. Caberá pedido de uniformização de interpretação de lei federal quando houver divergência entre decisões sobre questões de direito material proferidas por Turmas Recursais na interpretação da lei.
(...)
§ 4º. Quando a orientação acolhida pela Turma de Uniformização, em questões de direito material, contrariar súmula ou jurisprudência dominante no Superior Tribunal de Justiça – STJ, a parte interessada poderá provocar a manifestação deste, que dirimirá a divergência.
§ 5º. No caso do § 4º, presente a plausibilidade do direito invocado e havendo fundado receio de dano de difícil reparação, poderá o relator

Figura 12 – Exemplo de documento PDF antes da extração do texto e pré-processamento.

O mesmo trecho da Figura 12 após o pré-processamento, quando aplicada a remoção de *stopwords* e lematização:

"... tratar decisão presidente turma recursal juizado especial seção judiciário estado paraíba negar seguimento recurso extraordinário união seguinte termo decisão cuidar recurso extraordinário interposto contra acórdão órgão colegiado rejeitar alegação prescrição negar seguimento recurso ente público manter sentença recorrir próprio fundamento eis calçada posicionamento reiterar supremo tribunal federal dispor LEI_10259 instituir juizado especial federal cível criminal âmbitor justiça federal ARTIGO_14 caber pedir uniformização interpretação lei federal divergência decisão sobre questão direito material proferir turma recursal interpretação LEI_4º orientação acolhir turma uniformização questão direito material contrariar súmula jurisprudência dominante alto tribunal justiça stj parte interessado poder provocar manifestação dirimir divergência caso presente plausibilidade direito invocar fundar receio dano difícil reparação poder relator..."

Observar os termos marcados em vermelhos na Figura 12 e no trecho acima, que exemplifica a transformação efetivada na expressão "Lei nº 10.259/01", bem como a lematização das palavras "trata-se" para o radical "tratar", e "juizados" para "juizado".

4.3 Divisão do *Dataset* para os Experimentos

A coleção de documentos do *dataset* VICTOR é composta por dois conjuntos: *Medium* e *Small*, que representam subconjuntos da coleção completa de documentos coletados junto ao STF pela equipe do AILAB. Em cada subconjunto desses, os dados são separados também em conjuntos de treino, validação e teste, com respectivamente: 70%, 15% e 15%. Para os experimentos, foi empregado o *dataset* VICTOR *Small*.

Para a divisão dos dados em conjuntos de treino, validação e teste, em virtude do próprio *dataset* VICTOR já possuir essa distinção, os dados de teste foram mantidos conforme disponibilizados originalmente. Apenas para o caso da seleção de modelos de classificadores lineares, os dados de treino e validação foram unidos em um só conjunto, a fim de possibilitar a aplicação da validação cruzada, conforme indicado na Seção 4.5.1.

Para efetivação do treino e teste dos modelos, foram considerados 3 subconjuntos a partir dos dados originais, conforme os itens abaixo:

a) **Dataset sem documentos da classe "OUTROS" - (NO)**: *Dataset* composto apenas por documentos rotulados a exceção daqueles classificados como "OUTROS";

b) **Dataset que inclui todos os documentos da classe "OUTROS"- (O100)**: composto por 100% dos documentos classificados como "OUTROS", e todos os demais;

c) **Dataset que inclui 10% dos documentos da classe "OUTROS" - (O10)**: composto por 10% dos documentos classificados como "OUTROS", e todos das demais classes. Nesse caso, é utilizada uma amostra aleatória com distribuição uniforme, onde os documentos são selecionados a partir dos índices, e sem repetição.

4.4 Representação Computacional dos Documentos

Nesta seção são definidos os parâmetros para a representação computacional dos documentos, para os algoritmos TF-IDF, Word2Vec, FastText e BERT. O tamanho do vocabulário definido para os experimentos é restrito aos 70.000 *tokens* mais frequentes, excluídos aqueles que aparecem em apenas um documento, bem como os que constam em mais de 50% deles, conforme proposto em (ARAUJO et al., 2020a).

Importante destacar também, que a construção dos modelos TF-IDF, ou ainda a geração de vetores especializados a partir do Word2Vec e FastText, bem como o ajuste fino do BERT, são realizados apenas com os dados de treino.

4.4.1 TF-IDF

Na aplicação do TF-IDF, além do limite de 70.000 *tokens*, são ignorados os termos que aparecem em apenas 2 dos documentos, bem como aqueles que aparecem em mais de 50%, conforme indicado em (ARAUJO et al., 2020a). Além disso, no TF-IDF-N3, que usa o pré-processamento nível 3, é aplicada a formação de *bigrams*. Assim, temos dois modelos de TF-IDF empregados nos experimentos, TF-IDF-N2, e TF-IDF-N3, com níveis de pré-processamento distintos.

4.4.2 Word2Vec e FastText

Para as transformações a partir de vetores que representam o *embedding* de palavras, são também considerados dois tipos, o genérico, indicado pela sigla "GEN" nos nomes dos experimentos, e o especializado, indicado por sua vez pela sigla "ESP".

Os vetores genéricos são aqueles pré-treinados a partir de documentos estranhos ao *dataset* VICTOR, porém com uma ampla gama de textos das mais diversas fontes. Esses vetores são disponibilizados no estudo (HARTMANN et al., 2017a), da equipe do Núcleo Interinstitucional de Linguística Computacional (NILC) da Universidade do Estado de São Paulo (USP). Nesse caso, optou-se pelo uso dos vetores gerados com a arquitetura *Skip-gram*, com tamanho de 600 posições, tanto para o Word2Vec quanto para o FastText. Essa escolha parte dos resultados indicados em (NOGUTI; VELLASQUES; OLIVEIRA, 2020).

No caso dos vetores genéricos, é importante destacar que o vocabulário é pré-definido, ou seja, *tokens* que constem nos documentos do *dataset* VICTOR, porém não constem nesse vocabulário, são ignorados no processo de formação do vetor que representa o documento, conforme método indicado na Seção 3.3.4.

Por outro lado, os vetores especializados são gerados unicamente a partir dos documentos do conjunto de Treino do *dataset* VICTOR, sendo empregados como parâmetros de treinamento para *"Window Size"* e contagem mínima de palavra para integrar o vocabulário, os valores respectivamente de 10 e 5, assim como a arquitetura de treinamento *Skip-gram*. Além disso, o treinamento persistiu por 6 épocas, com taxa de aprendizado 0,025. Esses parâmetros são empregados tanto na geração de vetores com Word2Vec quanto para o FastText.

Para o carregamento e utilização dos vetores genéricos, bem como para a execução dos treinamentos para geração dos vetores especializados, é empregada a biblioteca Gensim (ŘEHŮŘEK; SOJKA, 2010)[3] do Python, por meio das classes específicas para Word2Vec e FastText.

[3] <https://pypi.org/project/gensim/>

4.4.3 BERT

Para a representação de documentos com o modelo de linguagem BERT é empregado o BERTimbau-Base, modelo pré-treinado com documentos em idioma português do Brasil, e disponibilizado por meio do estudo (SOUZA; NOGUEIRA; LOTUFO, 2020).

A execução do ajuste fino do BERTimbau para os documentos do *dataset* VICTOR, e posterior geração dos vetores que representam cada documento, são executadas por meio da biblioteca Python *Transformers* (WOLF et al., 2020)[4] da *Hugging Face*. Os parâmetros para ajuste-fino do BERTimbau são: 3 épocas, *batch size* de 8, e taxa de aprendizado de 2.10^{-5}, obtidos em (CHALKIDIS et al., 2020), aplicados na tarefa de *masked-language modeling* (MLM).

Em virtude da limitação imposta pela arquitetura do BERT, são usados apenas os 512 primeiros *tokens* dos documentos para a geração do vetor que o representa, extraído a partir da saída *C* do modelo conforme a Figura 3, bem como essa representação é um vetor com 768 posições, uma vez que trata-se da versão *base* do BERTimbau.

4.5 Seleção de Modelos e Definição de Hiperparâmetros

Nesta seção são indicadas as técnicas utilizadas para a seleção de modelos de Aprendizado de Máquina Tradicional, bem como os requisitos empregados na seleção dos hiperparâmetros dos algortimos de *Deep Learning*. Para ambas as classes de algoritmos, a métrica do F1-score médio é empregada para a comparação e seleção de modelos.

4.5.1 Algoritmos de Aprendizado de Máquina Tradicional

Para a seleção de modelos dos algoritmos de Aprendizado de Máquina Tradicional, são empregados os métodos de Validação Cruzada (*k-fold*) e Busca em Grade (*Grid-Search*). A validação cruzada é aplicada com valor de $k = 5$, portanto, são realizados 5 treinamentos com a mesma configuração de algoritmo e *dataset*, onde em cada treinamento 20% dos dados são usados para validação e os demais para o treinamento. A Figura 13 mostra como funciona na prática a validação cruzada.

O valor do hiperparâmetro de regularização *C* para o SVM e Regressão Logística é selecionado por meio da busca em grade, onde para cada valor de *C* são treinados modelos a partir da validação cruzada exemplificada na Figura 13. O hiperparâmetro de regularização *C* é escolhido a partir daquela configuração que obtém o melhor desempenho médio para a validação cruzada, de acordo com o índice de F1-score médio. Para a execução da busca em grade, os conjuntos de Treino e Validação são unidos, conforme também indicado na Figura 13. Após a seleção do parâmetro *C*, o experimento é realizado a partir do ajuste do algoritmo SVM ou Regressão

[4] <https://huggingface.co/docs/transformers>

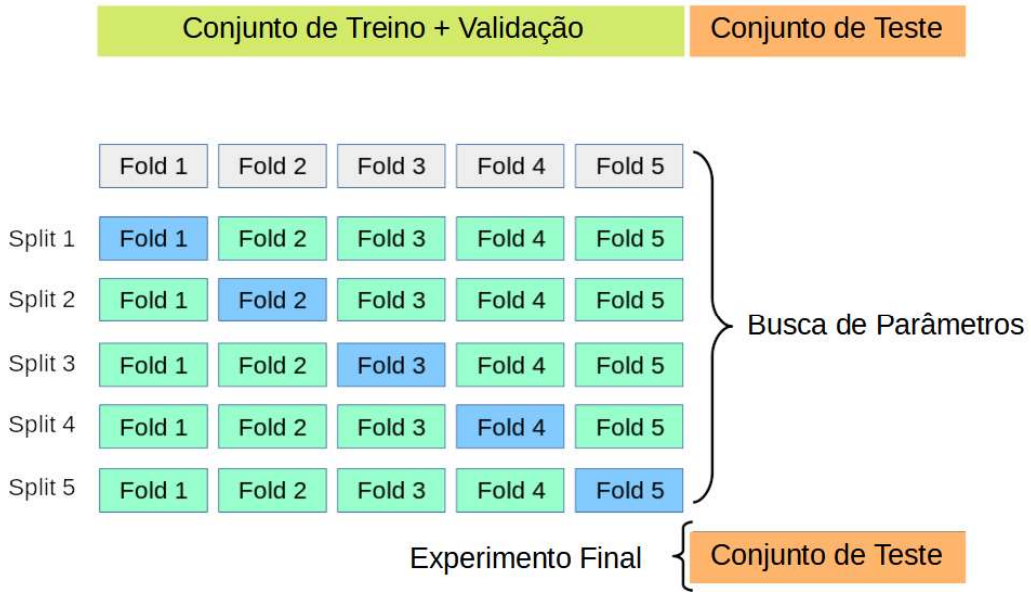


Figura 13 – Exemplo de treinamento com o *k-fold*. Fonte: Adaptado de <https://scikit-learn.org/stable/modules/cross_validation.html>

Logística com os dados do conjunto de Treino, e ao final aplicados ao conjunto de Teste para obtenção dos índices de F1-score e MCC.

4.5.2 Definição de Hiperparâmetros para *Deep Learning*

Para os algoritmos de *Deep Learning*, no caso ULMFiT e HAN, os hiperparâmetros, a exceção do número de épocas, são definidos a partir das publicações que propuseram os métodos ou aplicam os algoritmos em *datasets* do domínio jurídico, neste caso, (MENEZES-NETO; CLEMENTINO, 2022) e (ARAUJO; CAMPOS; SOUSA, 2020b) para ULMFiT, e (YANG et al., 2016) para o HAN.

Em relação ao número de épocas de treinamento, é empregado o conceito de *Early Stopping*, ou "Parada Precoce". O *Early Stopping* consiste ao final de cada época calcular a métrica de desempenho alvo para o conjunto de Validação, e a partir do momento em que esse valor parar de melhorar segundo critérios pré-estabelecidos, o treinamento é finalizado evitando assim o *overfitting* do modelo.

A semelhança do que é aplicado com os classificadores lineares, após o treinamento e ajuste dos parâmetros do modelo HAN ou ULMFiT, o mesmo é testado com o conjunto de Teste para obtenção dos índices Precisão, *Recall*, F1-score médio e MCC, utilizados na comparação dos algoritmos.

No Capítulo 5 são mostradas medidas estatísticas do *dataset* VICTOR em termos de quantidades de páginas e *tokens*, bem como as configurações dos experimentos, com os respectivos resultados.

5 EXPERIMENTOS E RESULTADOS

Nesse Capítulo, são indicadas estatísticas básicas do *dataset* empregado nos experimentos, bem como os resultados de cada uma das categorias de *pipeline* de classificação propostas, ou seja, Abordagens I e II, conforme indicado na Seção 1.4 do Capítulo de Introdução.

5.1 Estatísticas básicas do *Dataset*

Nessa Seção, são apresentadas estatísticas básicas do *dataset* VICTOR (*Small*) trabalhado nos experimentos, que auxiliam no entendimento do desbalanceamento entre as classes, bem como da quantidade de páginas e *tokens*, que auxiliam no processo de definição de parâmetros de número de páginas e quantidades de *tokens* empregados nos treinamentos dos algoritmos. Inicialmente é mostrada a distribuição das classes para cada subdivisão no item 5.1.1, a seguir, no item 5.1.3, os dados estatísticos para quantidade de páginas por documento, bem como de *tokens* por página e por documento.

5.1.1 Distribuição das Classes

Considerando a subdivisão do *dataset* em O100, O10 e NO, conforme indicado no item 4.3, a distribuição das classes é disposta nas Tabelas 3 a 5 separadas para cada um deles conforme os itens a seguir. As Tabelas indicam as quantidades para cada subconjunto do *dataset*, ou seja, Treino, Validação e Teste, bem como a proporção que cada classe representa no total de cada subconjunto:

a) **O100**: No *dataset* que inclui todos os documentos, é importante observar na Tabela 3, o desbalanceamento desse conjunto de dados, onde cerca de 96% dos documentos pertencem à classe 'Outros'. Observar que a mesma proporção se mantém para todos os subconjuntos.

Classe	Treino		Validação		Teste	
Outros	37.113	95,62%	24.286	95,43%	24.191	95,62%
Acórdão	301	0,77%	227	0,90%	203	0,80%
ARE	266	0,68%	198	0,78%	197	0,78%
Despacho	265	0,68%	143	0,57%	146	0,58%
PRE	450	1,17%	317	1,24%	301	1,19%
Sentença	420	1,08%	277	1,08%	262	1,03%
Total	**38.815**	**100%**	**25.448**	**100%**	**25.300**	**100%**

Tabela 3 – Quantidades de classe de documentos no *dataset* VICTOR-O100 em cada conjunto.

b) **NO**: No *dataset* que exclui todos os documentos da classe "Outros", observa-se na Tabela 4 que há um relativo balanceamento entre as 5 classes, onde aquela que apresenta maior

número de documentos, ou seja, “Petição do Recurso Especial” (PRE), não chega ao dobro da quantidade daquela com menor proporção, no caso “Despacho de Admissibilidade” (Despacho).

Classe	**Treino**		**Validação**		**Teste**	
Acórdão	301	17,69%	227	19,54%	203	18,30%
ARE	266	15,63%	198	17,04%	197	17,76%
Despacho	265	15,57%	143	12,31%	146	13,17%
PRE	450	26,43%	317	27,28%	301	27,15%
Sentença	420	24,68%	277	23,83%	262	23,62%
Total	**1.702**	**100%**	**1.162**	**100%**	**1.109**	**100%**

Tabela 4 – Quantidades de classe de documentos no *dataset* VICTOR-NO em cada conjunto.

c) **O10**: No *dataset* que seleciona aleatoriamente 10% dos documentos da classe ’Outros’, observa-se na Tabela 5, que há ainda um desbalanceamento entre a classe “Outros” e as demais, sendo a primeira com mais de 68% dos documentos. Importante notar, que no caso do subconjunto de Teste todos os documentos da classe “Outros” foram mantidos, uma vez que a estratégia de usar o *dataset* O10, consiste na seleção de hiperparâmetros para o *dataset* O100, bem como experimentos para avaliar se um subconjunto de Treino com menos desbalanceamento entre as classes aprimora o resultado da tarefa de classificação, e nesse ponto, o algoritmo treinado é testado empregando-se o subconjunto de Teste O100 para melhor comparação.

Classe	**Treino**		**Validação**		**Teste**	
Outros	3.711	68,56%	2.428	67,63%	24.191	95,62%
Acórdão	301	5,56%	227	6,32%	203	0,80%
ARE	266	4,91%	198	5,52%	197	0,78%
Despacho	265	4,90%	143	3,98%	146	0,58%
PRE	450	8,31%	317	8,83%	301	1,19%
Sentença	420	7,76%	277	7,72%	262	1,03%
Total	**38.815**	**100%**	**25.448**	**100%**	**25.300**	**100%**

Tabela 5 – Quantidades de classe de documentos no *dataset* VICTOR-O10 em cada conjunto.

5.1.2 Quantidade de páginas por documento

Na Figura 14 (a), que representa o *boxplot* da distribuição da quantidade de páginas por documento, é possível identificar que 85% dos documentos têm até 6 páginas. Já na Figura 14 (b), que representa a curva de densidade de probabilidade estimada por meio da técnica KDE (*Kernel Densisty Estimate*) (CHEN, 2017), verifica-se que a maioria dos documentos tem menos de 20 páginas, dada a baixa densidade a partir desse ponto no gráfico, com uma média em 3,74, havendo maior concentração de documentos entre 1 e 3 páginas.

O gráfico de *boxplot* é interessante por representar os quartis da série de dados, bem como os limites inferiores e superiores que auxiliam na identificação de *outliers*. Os limites são calculados a partir do intervalo interquartil para os quantis 25% e 75% multiplicado por 1,5, ou seja, para o limite inferior esse valor é diminuído do quantil 25%, e para o limite superior esse valor é adicionado ao quantil de 75%. A linha vermelha no *boxplot* representa o quantil indicado no texto próximo a ela. As considerações sobre o *boxplot* aqui, devem ser consideradas para os demais itens desta seção.

Relativamente ao gráfico de densidade de probabilidade, este é estimados por meio da técnica KDE, que consiste no método de visualização da distribuição dos dados análogo a um histograma, porém representado por meio de uma curva de densidade de probabilidade contínua. Esse formato de visualização permite melhor inferência sobre os dados, uma vez que é possível estimar medidas como moda, e valores máximos e mínimos.

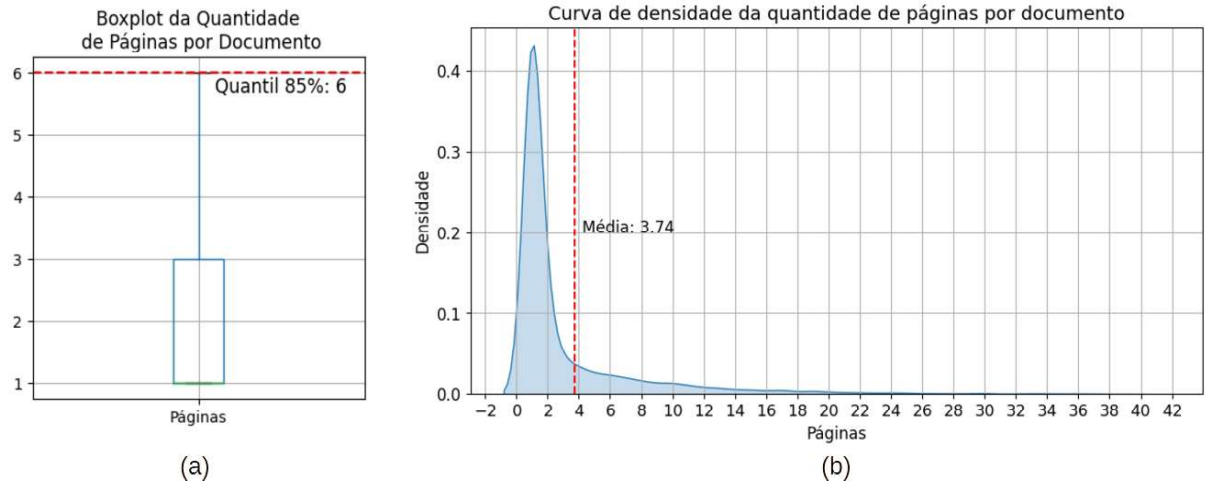


Figura 14 – Quantidade de páginas por documento para o *Dataset* VICTOR-O100 (Treino). (a) *Boxplot* indicando o Quantil 85%. (b) Curva de densidade.

5.1.3 Quantidade de *Tokens*

Inicialmente, convém destacar que o subconjunto de Treino possui 22.105.103 *tokens* no total, os quais são definidos como um conjuntos de caracteres limitados por espaço em branco. Nos itens a seguir, são indicadas as medidas de média e quantis para o conjunto de Treino "O100" em relação às quantidades de *tokens* por página, bem como *tokens* por documento.

Importante destacar, que o *dataset* VICTOR na forma como disponibilizado pela equipe do AILAB, vem com os documentos separados por página, ou seja, cada linha da tabela é uma página separada de um determinado documento, que podem ser concatenadas para formar o documento original a partir do nome do arquivo indicado na coluna "file_name".

Dessa forma, há a necessidade de estudar as medidas estatísticas tanto por página, quanto por documento, uma vez que o algoritmo HAN empregado nos experimentos possui essa divisão lógica para implementação do modelo hierárquico.

a) **Quantidade de *tokens* por página**: na Figura 15 (a), é possível identificar que 90% das páginas possuem até 255 *tokens*. Bem como na Figura 15 (b), a média de *tokens* por página é em torno de 148, e a maioria tem até 500 *tokens*, região onde a densidade tende a zero.

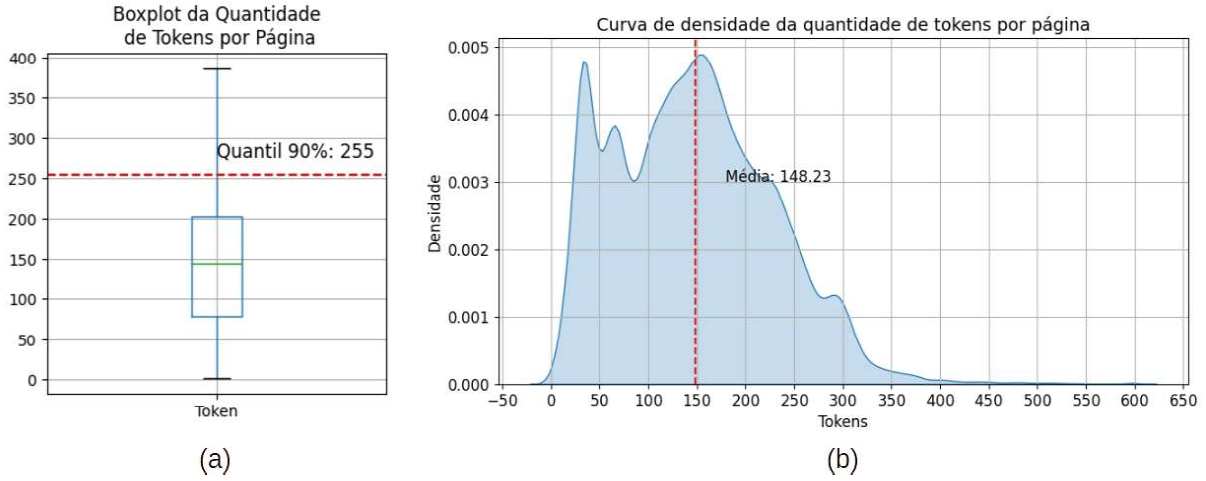


Figura 15 – Quantidade de *tokens* por páginas para o *Dataset* VICTOR-O100 (Treino).(a) *Boxplot* indicando o Quantil 90%. (b) Curva de densidade.

b) **Quantidade de *tokens* por documento**: na Figura 16 (a), observa-se que que 90% dos documentos têm até 1.486 *tokens*, e essa quantia inclusive ultrapassa o limite superior do *boxplot*. A Figura 16 (b), indica a média de aproximadamente 570 *tokens*, onde é possível perceber também que em torno de 1600 *tokens*, a densidade tende a zero.

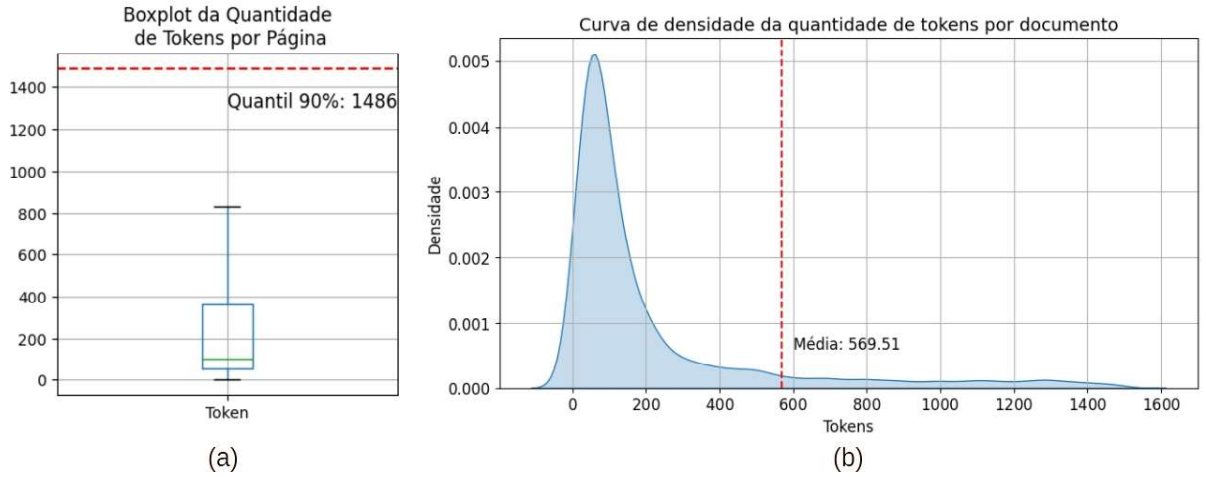


Figura 16 – Quantidade de *tokens* por documento para o *Dataset* VICTOR-O100 (Treino).(a) *Boxplot* indicando o Quantil 90%. (b) Curva de densidade.

Após a exposição dos indicadores de quantidade de páginas e quantidade de *tokens*, considerando que de acordo com a Figura 14 (a) 85% dos documentos tem até 6 páginas, essa seria uma escolha interessante para o parâmetro de número de páginas máximo para o treinamento do algoritmo HAN. Além disso, nas Figuras 15 (a) e 16 (a) é evidenciado que 90% das páginas tem até 255 *tokens*, e 90% dos documentos tem até 1486 *tokens*, respectivamente, o que indica que ao

se considerar como parâmetros para o HAN o máximo de 6 páginas, com o máximo 255 *tokens* em cada página, e para os demais algoritmos a quantidade máxima de 1530 (6x255) *tokens*, são valores que permitem comparações justas entre eles. A exceção fica para a trasformação TF-IDF que por essência emprega todos os *tokens* do documento.

Essas considerações permitem a limitação na quantidade de *tokens* por documento empregada para o treinamento, ao mesmo tempo que garante que pelo menos 85% dos documentos estão sendo entregues de forma completa para os algoritmos de treinamento.

5.2 Abordagem I - Aprendizado de Máquina Tradicional

Nesta seção são apresentados os resultados e configuração dos experimentos para os classificadores SVM e Regressão Logística em conjunto com representações computacionais dos documentos baseadas em: TF-IDF, Word2Vec, FastText e BERT. Cabe ressaltar que em ambos foi empregada a busca em grade para a definição do parâmetro de regularização, a partir do conjunto de possibilidades $10, 100, 1000$, com $k\text{-}fold = 5$. Portanto, para cada experimento, a busca consiste em 15 execuções do algoritmo, onde o valor de C é escolhido a partir da configuração que obtém a melhor média para o F1-score médio. No caso do *dataset*, em virtude da quantidade de documentos, é empregado o *dataset* O10 na busca desse parâmetro.

Todos os *pipelines* indicados na Figura 17 são experimentados para os *datasets* VICTOR-NO e VICTOR-O100. A Figura 17 auxilia no entendimento da configuração desses *pipelines* e seus respectivos nomes usados para distinção dos experimentos nas tabelas de resultados, além de caracterizar de forma mais precisa o padrão dos *pipelines* da Abordagem I, onde fica claro que há uma separação nos processos de treinamento da representação computacional do documento (tracejado em vermelho) e do algoritmo de classificação (tracejado em azul).

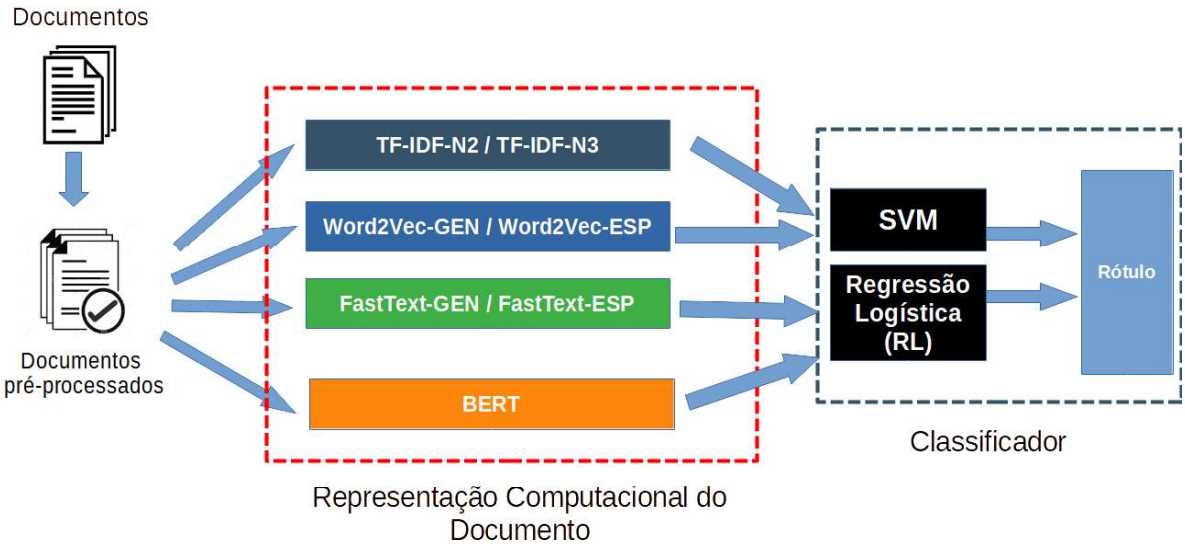


Figura 17 – Fluxo e denominação dos experimentos referente à Abordagem I.

Nesse caso, o experimento que emprega o TF-IDF com o segundo nível de pré-processamento,

e o SVM como classificador para o *dataset* VICTOR-NO, denomina-se TF-IDF-N2_SVM-NO, da mesma forma, o que o usa o Word2Vec especializado para representação computacional do documento, e regressão logística como classificador para o *dataset* VICTOR-O100, é indicado por Word2Vec-ESP_RL-O100.

5.2.1 Experimentos com *dataset* VICTOR-NO

A Tabela 6 indica os índices de F1-score, Precisão, e *Recall* médios, além do MCC para o *dataset* VICTOR-NO. Os índices dos experimentos para TF-IDF, Word2Vec e FastText, pelo fato de possuírem variações no processo de transformação empregado, são indicados para cada tipo na mesma linha, respectivamente a especificação entre parênteses.

Experimento	F1-score		MCC		Precisão		Recall	
TF-IDF-(**N2/N3**)_SVM-NO	0,9701	**0,9715**	0,9600	**0,9623**	0,9705	**0,9715**	0,9696	**0,9715**
TF-IDF-(**N2/N3**)_RL-NO	0,9696	0,9670	0,9600	0,9578	0,9696	0,9665	0,9697	0,9679
Word2Vec-(**ESP/GEN**)_SVM-NO	0,9582	0,9461	0,9462	0,9338	0,9592	0,9476	0,9574	0,9454
Word2Vec-(**ESP/GEN**)_RL-NO	0,9576	0,9448	0,9440	0,9302	0,9596	0,9467	0,9558	0,9431
FastText-(**ESP/GEN**)_SVM-NO	0,9627	0,9461	0,9520	0,9338	0,9636	0,9476	0,9619	0,9454
FastText-(**ESP/GEN**)_RL-NO	0,9624	0,9448	0,9521	0,9302	0,9657	0,9467	0,9598	0,9431
BERT-SVM-NO	0,9401		0,9246		0,9417		0,9389	
BERT-LR-NO	0,9444		0,9281		0,9454		0,9441	

Tabela 6 – Índices F1-score, Precisão, e *Recall* médios, e MCC para cada experimento da Abordagem I no *dataset* VICTOR-NO. Os valores entre parênteses indicam as variações no processo de transformação.

Observa-se na Tabela 6 que os índices de F1-scores estão equivalentes em todos os experimentos, com ligeira vantagem para o experimento com *pipeline* composto de TF-IDF nível 3 e SVM. Depreende-se também da mesma Tabela, que as transformações Word2Vec e FastText apresentam os melhores resultados para as versões que empregam os vetores especializados, tanto para SVM quanto para a regressão logística. No caso do TF-IDF, o melhor resultado para regressão logística foi com o nível 2 de pré-processamento, divergindo do SVM que apresenta resultado superior com o nível 3.

Quando examina-se apenas para o classificador, o SVM possui ligeira vantagem em relação à regressão logística, porém para a transformação com BERT essa situação se inverte. Nesse ponto, importante destacar que o BERT, apesar de possuir os menores índices, utiliza apenas 512 *tokens* como entrada para o processo de transformação, enquanto Word2Vec e FastText usam 1530, e o TF-IDF todos. Para os demais índices, ou seja, MCC, Precisão e *Recall*, verifica-se o mesmo comportamento do F1-score, com vantagem centesimal para o TF-IDF-(**N2/N3**)_SVM-NO.

A Tabela 7 exibe os índices separados por classe para o experimento TF-IDF-(**N2/N3**)_SVM-NO. Nela fica evidente que, na ausência da classe ‘’Outros”, os resultados de Precisão e *Recall* são semelhantes para cada classe. Já a Figura 18, mostra a matriz de confusão proporcional para *Recall* relacionada ao mesmo experimento, onde percebe-se que documentos da classe ARE são

classificados como do ‘’Petição de Recurso Extraordinário”, e vice-versa, sendo as duas classes com menores índices.

Classe	F1-score	Precisão	*Recall*
Acórdão	**0,9949**	**0,9949**	**0,9949**
ARE	0,9435	0,9412	0,9458
Despacho	0,9794	0,9794	0,9795
PRE	0,9569	0,9538	0,9601
Sentença	0,9827	0,9884	0,9771
Média	0,9715	0,9715	0,9715

Tabela 7 – Índices separados por classe de F1-score, Precisão e *Recall*, para o experimento do *pipeline* TF-IDF-N3_SVM-NO para o *dataset* VICTOR-NO.

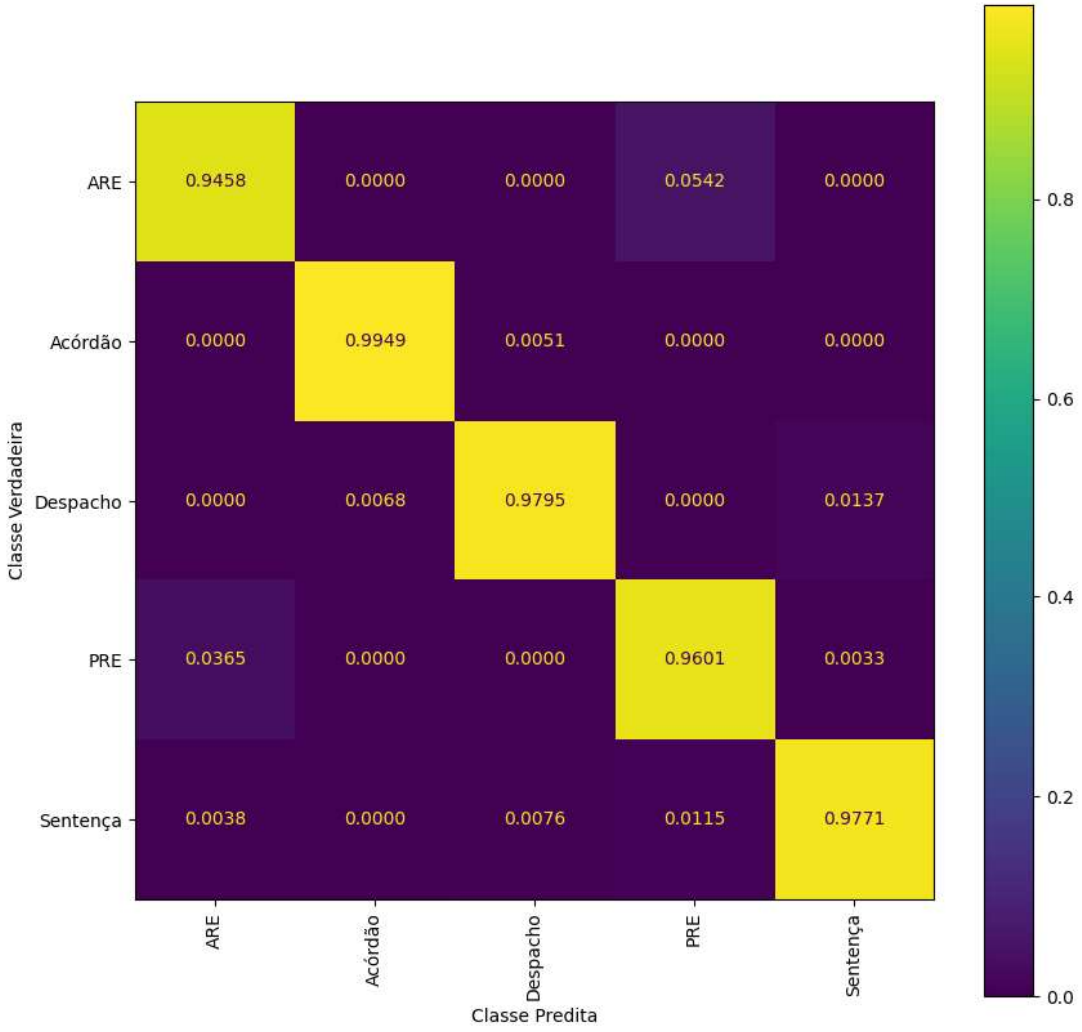


Figura 18 – Matriz de confusão para *Recall*, relacionado ao experimento TF-IDF-N3_SVM-NO

5.2.2 Experimentos com *dataset* VICTOR-O100

Em relação ao *dataset* VICTOR-O100, a Tabela 8 indica que a tarefa de classificação para esse *dataset* é relativamente mais complicada, uma vez que atinge valores de F1-score médio e

MCC menores em todos os casos quando comparados com o VICTOR-NO. Essa dificuldade maior possivelmente é explicada em virtude do desbalanceamento entre as classes, onde a classe ‘’Outros” possui mais de 95% dos documentos em todos os conjuntos, conforme apresentado na Tabela 3.

Experimento	F1-score		MCC		Precisão		*Recall*	
TF-IDF-(**N2/N3**)_SVM-O100	0,8289	0,8352	0,8096	0,8096	0,8289	0,8762	0,7792	**0,8018**
TF-IDF-(**N2/N3**)_RL-O100	0,8250	**0,8410**	0,8051	**0,8206**	0,8998	**0,9107**	0,7699	0,7882
Word2Vec-(**ESP/GEN**)_SVM-O100	0,7538	0,7178	0,7355	0,6834	0,8595	0,7515	0,6878	0,6966
Word2Vec-(**ESP/GEN**)_RL-O100	0,7532	0,6854	0,7160	0,6475	0,8448	0,7726	0,6921	0,6326
FastText-(**ESP/GEN**)_SVM-O100	0,7418	0,7178	0,7323	0,6834	0,8668	0,7515	0,6765	0,6966
FastText-(**ESP/GEN**)_RL-O100	0,7406	0,6788	0,7151	0,6404	0,8381	0,7627	0,6802	0,6252
BERT-SVM-O100	0,7348		0,6973		0,7293		0,7424	
BERT-LR-O100	0,7268		0,6944		0,7897		0,6862	

Tabela 8 – Índices F1-score, Precisão, e *Recall* médios, e MCC para cada experimento da Abordagem I no *dataset* VICTOR-O100. Os valores entre parênteses indicam as variações no processo de transformação.

A Tabela 8 evidencia que o experimento com TF-IDF-N3_RL-O100 obtém vantagem em relação aos demais para F1-score, MCC e Precisão, perdendo apenas em termos de *Recall* para o TF-IDF-**(N2/N3)**_RL-O100. No caso do *dataset* VICTOR-O100, fica evidente também que a geração dos vetores especializados para Word2Vec e FastText, bem como o maior nível de pré-processamento para o TF-IDF, trazem ganhos para a tarefa de classificação, o que corrobora com resultados em comparações semelhantes efetivadas por (NOGUTI; VELLASQUES; OLIVEIRA, 2020).

Quando compara-se o desempenho em termos de F1-score e MCC entre os classificadores, apesar de o melhor índice ser obtido pela Regressão Logística, em todos os demais casos o SVM leva vantagem, entretanto, convém destacar que o custo computacional dele em relação à Regressão Logística é maior, onde o primeiro precisa de 5 a 10 vezes mais tempo de treinamento, a depender da representação computacional. Para os índices de Precisão e *Recall* os resultados são variados sem clara vantagem para um ou outro tipo de classificador.

O *pipeline* com BERT, mais uma vez obteve os menores índices, superando apenas os modelos genéricos de Word2Vec e FastText, e mesmo nesses casos não superou em termos de Precisão. Necessário destacar que o modelo de linguagem BERT recebeu ajuste fino com os dados do *dataset* VICTOR-O100, o que é uma tarefa relativamente custosa em comparação às demais transformações analisadas, necessitando de GPU (*Graphics Processing Unit*) para o processamento.

Outra consideração importante a se destacar na Tabela 8, é a diferença nos resultamos médios de Precisão e *Recall*, onde o primeiro aparece com os maiores índices, o que indica que, na média, os *pipelines* de classificação da Abordagem I estão predizendo com uma quantidade maior de falsos negativos quando comparados aos falsos positivos.

A diferença nos valores médios de Precisão e *Recall* fica mais explícita quando analisa-se o resultado por classe para o experimento TF-IDF-**(N2/N3)**_RL-O100, a Tabela 9 mostra os índices de F1-score, Precisão e *Recall* de cada classe para esse caso, onde fica evidente que o último influencia de forma a diminuir o valor de F1-score para cada classe, uma vez que possui os menores índices.

Fica evidente também na Tabela 9, que a classe ARE (Agravo em Recurso Extraordinário), possui o menor valor de F1-score, muito em virtude do índice dela para *Recall*. Na Figura 19, que mostra os índices de *Recall* para cada classe no formato de Matriz de Confusão, infere-se que os falsos negativos que diminuem o índice *Recall* da classe ARE decorrem do fato desta classe estar sendo classificada de forma errada na classe ‘’Outros”.

Classe	**F1-score**	**Precisão**	***Recall***
Acórdão	0,9587	0,9738	0,9442
ARE	0,6704	0,7742	0,5911
Despacho	0,7459	0,9286	0,6233
Outros	**0,9926**	**0,9890**	**0,9962**
PRE	0,7928	0,8661	0,7309
Sentença	0,8857	0,9325	0,8435
Média	0,8410	0,9107	0,7882

Tabela 9 – Índices separados por classe de F1-score, Precisão e*Recall*, para o experimento do *pipeline* TF-IDF-N3_RL-O100.

É possível identificar também na Figura 9, que para outras classes, em menor grau, ocorre o mesmo problema, sendo que os documentos da classe ‘’Acórdão” são os que possuem o menor índice de documentos falsos negativos classificados como da classe ‘’Outros”. Importante notar também, que esse cenário ocorre justamente em relação à classe ‘’Outros” que possui mais de 95% dos exemplos empregados no treinamento e teste dos modelos, ou seja, o algoritmo, em maior ou menor grau, tende a classificar documentos das demais classes como sendo da classe com maior número de exemplos.

5.3 Abordagem II - *Deep Learning*

A Figura 20 indica os *pipelines* para classificação dos documentos do *dataset* VICTOR que empregam algoritmos definidos para a Abordagem II (*Deep Learning*), indicados nos Itens 3.5.1 e 3.5.2, HAN e ULMFiT, respectivamente, ambos embasados em redes recorrentes.

Observa-se na Figura 20, que há necessidade inicial de se gerar uma representação computacional do documento, representada pela linha tracejada em vermelho, e em seguida realizar o processo de classificação por meio de uma camada densa, representada pela linha tracejada em azul, a semelhança do que ocorre com os *pipelines* da Abordagem I, conforme a Figura 17, com a diferença de que o ajuste dos parâmetros dos modelos de *Deep Learning* ocorre simulta-

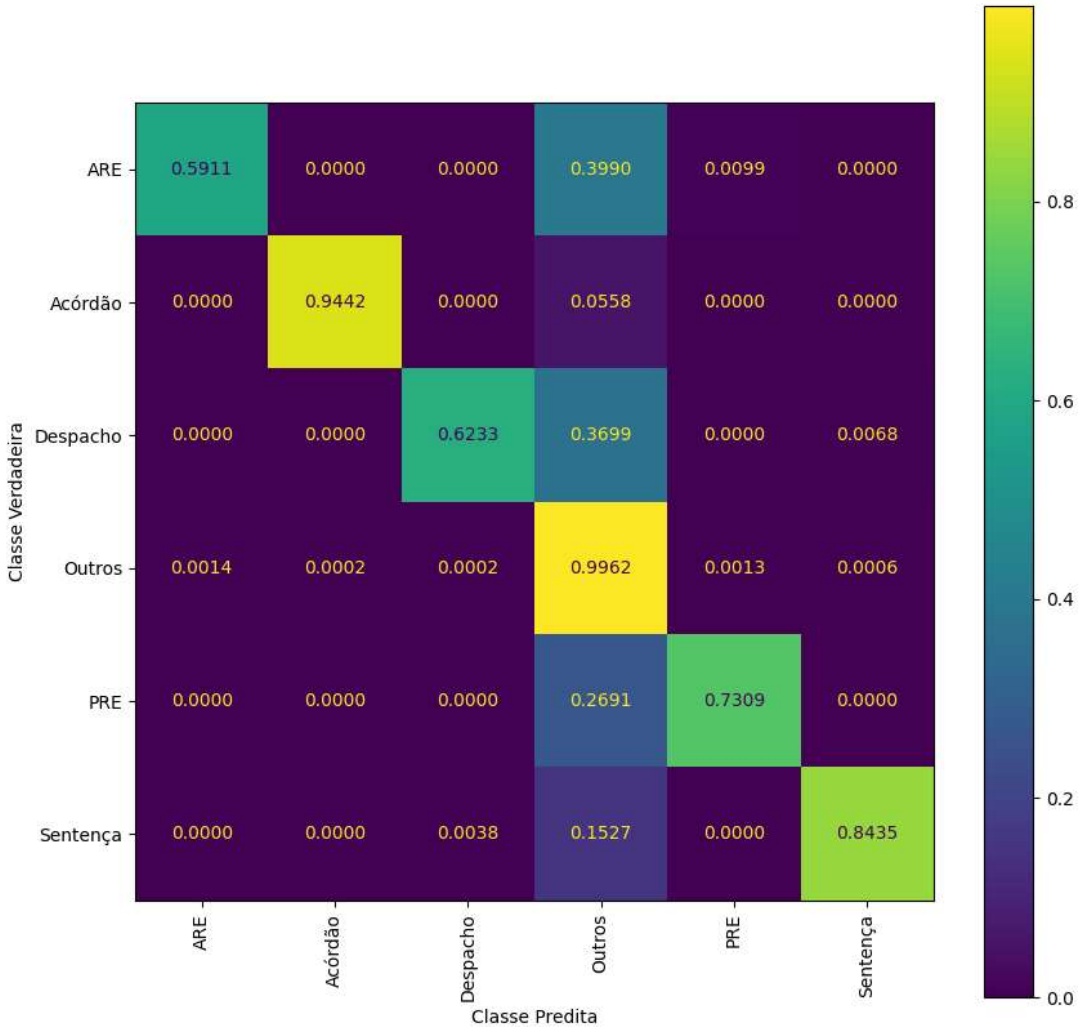


Figura 19 – Matriz de confusão para *Recall*, relacionado ao experimento TF-IDF-N3_RL para VICTORO100.

neamente com o ajuste dos parâmetros do classificador, representado pela linha tracejada verde, denominada de "Adequação ao domínio".

A respeito do *pipeline* da arquitetura HAN (fluxos 1 e 2) na Figura 20, nota-se que há a possibilidade de 2 tipos de camadas de *embedding*, a do fluxo 1 é a forma especializada (ESP), e a do fluxo 2 usa os parâmetros pré-treinados com a arquitetura GloVe e disponibilizados por (HARTMANN et al., 2017a). Em cada um dos fluxos 1 e 2, os parâmetros da camada de *embedding* são ajustados durante o treinamento, mesmo para aqueles pré-treinados. Além disso, as palavras que por ventura não constem no vocabulário do *embedding* GloVE pré-treinado, para possibilitar a inclusão delas na camada de *embedding*, são iniciadas com valores aleatórios de acordo com uma distribuição Normal de média 0 e desvio padrão 1.

Fazendo um paralelo com os *pipelines* da Abordagem I, a camada de *embedding* treinada do zero, representada pelo fluxo 1, assemelha-se com os vetores especializados para Word2Vec e FastText. Já a camada pré-treinada GloVe, representada pelo fluxo 2, relaciona-se com os vetores genéricos, com a diferença de, no caso da arquitetura HAN, os vetores GloVe pré-treinados são

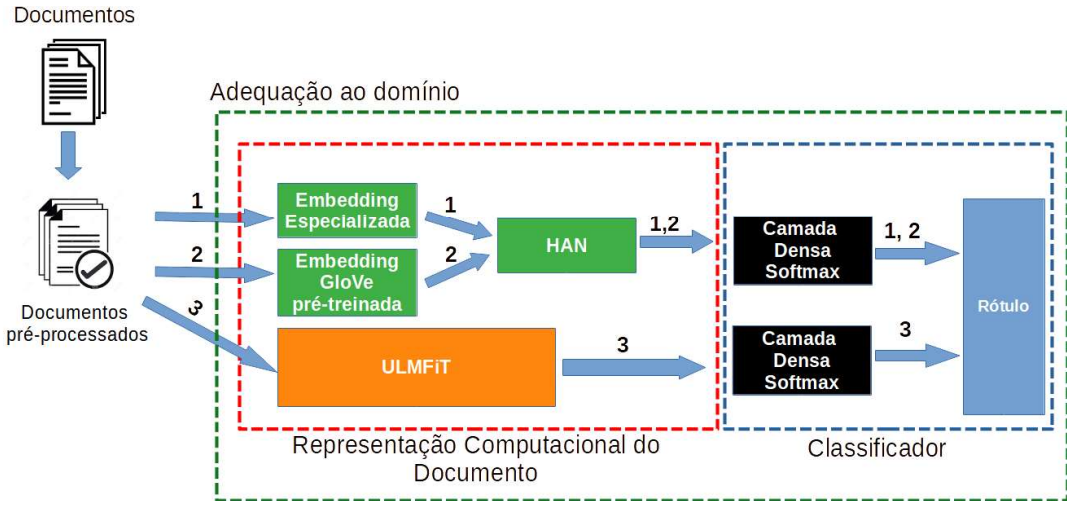


Figura 20 – Pipelines da Abordagem II para a arquitetura HAN (fluxos 1 e 2) e algoritmo ULMFiT (fluxo 3).

ajustados também durante o treinamento, o que não ocorre no caso da Abordagem I que aplicam os vetores genéricos Word2Vec e FastText.

5.3.1 Redes Recorrentes Hierárquicas com Mecanismo de Atenção - HAN

Os experimentos com a arquitetura HAN, embasaram-se nos hiperparâmetros de (CHALKIDIS et al., 2020), com camada de *embedding* de tamanho 200, taxa de aprendizado 10^{-3} com técnica de *early stopping* para a seleção do número de épocas, com paciência de 5 épocas, ou seja, após alcançar o melhor índice de perda (*loss*) para validação, o treinamento perdura por mais cinco épocas, em caso de não melhora, o treinamento é interrompido. Em virtude de vetores de tamanho 200 não estarem disponíveis para o GloVe no repositório de *Word Embeddings* do NILC-USP, nos experimentos que empregam o *embedding* pré-treinado são usados os vetores de tamanho 300.

Em relação ao *batch size* e tamanho do vetor de representação do estado interno da rede recorrente em ambos sentidos, conforme indicado na Figura 7, é empregado a busca em grade para os seguintes parâmetros: *batch size*: {32, 64} e vetor que representa o estado interno:{100, 150, 200}. A busca é realizada nos *datasets* VICTOR-O10 e VICTOR-NO, aplicando-se os dados de Treino e Validação, e optando-se pela combinação de parâmetros com menor valor de perda para o conjunto de Validação. Os valores obtidos para o VICTOR-O10, e consequentemente aplicados aos experimentos com o VICTOR-O100 são 32 e 200, bem como para o VICTOR-NO são obtidos 32 e 100, para *batch size* e representação interna, respectivamente.

Além desses hiperparâmetros, pelo fato da arquitetura HAN proposta separar o documento em páginas, são experimentadas 2 possibilidades de configuração de quantidade máxima de *tokens* por páginas e da quantidade máxima de páginas. Assim, considerando a análise estatística apresentada no Item 5.1.3, são experimentadas combinações com limitações de 6 páginas e 255 *tokens* por página, e também de 3 páginas com 500 *tokens* por página.

Exclusivamente em relação ao VICTOR-O100, em virtude da maior dificuldade do processo de classificação desse conjunto de dados em comparação ao VICTOR-NO, é realizado um experimento com limitações de 6 páginas e 500 *tokens* por página, o que totalizaria no máximo 3000 *tokens* por documento sendo processados, a fim de comparar o efeito no aumento do número de *tokens* nos índices de F1-score médio e MCC para a classificação. Para facilitar a identificação dos *pipelines* nas Tabelas 10 e 12, com resultados para VICTOR-NO e VICTOR-O100, respectivamente, os mesmos são representados da seguinte forma: HAN-<tipo_embedding>-<paginas>-<tokens>-<dataset>, ou seja, o experimento com camada de *embedding* genérica, 6 páginas e 255 *tokens* para o *dataset* VICTOR-O100, é identificado por HAN-ESP-6-255-O100. Para o mesmo experimento com a camada GloVe pré-treinada e *dataset* VICTOR-NO, a identificação é HAN-GloVe-6-255-NO.

5.3.1.1 Experimentos com *dataset* VICTOR-NO

Na Tabela 12, que mostra os resultados em relação ao *dataset* VICTOR-NO, o experimento que emprega camada de *embedding* especializada e no máximo 3 páginas com 500 *tokens* de limite por página (HAN-ESP-3-500-NO), apresenta os melhores índices de F1-score, MCC, e Precisão, entretanto, ainda com uma diferença aproximada de 0,02 para o pior caso em relação ao F1-score, representado pelo experimento HAN-ESP-6-255-NO. Portanto, a estratégia de diminuir o número máximo de páginas de 6 para 3, e aumentar a quantidade limite de *tokens* por página de 255 para 500, aprimora o resultado.

Experimento	**F-1 score**	**MCC**	**Precisão**	***Recall***
HAN-ESP-6-255-NO	0,9420	0,9285	0,9467	0,9404
HAN-ESP-3-500-NO	**0,9604**	**0,9523**	**0,9619**	0,9605
HAN-GloVe-6-255-NO	0,9580	0,9477	0,9560	**0,9608**
HAN-GloVe-3-500-NO	0,9555	0,9488	0,9563	0,9557

Tabela 10 – Índices F1-score, Precisão, e *Recall* médios, e MCC para experimentos com a arquitetura HAN no *dataset* VICTOR-NO.

As estratégias que adotam camada pré-treinada com GloVe possuem índices com diferenças na terceira casa decimal apenas, e superam a estratégia que emprega camada de *embedding* especializada, 6 páginas e 255 *tokens* como limites. No caso da HAN-GloVe-6-255-NO, é o experimento com melhor valor de *Recall*.

Na análise do desempenho em termos de F1-score, Precisão e *Recall* por classe em relação ao *pipeline* HAN-ESP-3-500, a Tabela 11 indica que o menor F1-score é obtido pela classe ‘’Despacho”, e que também tem o menor *Recall*, ou seja, é a classe que, proporcionalmente, tem a maior quantidade de documentos classificado de forma equivocada em relação às demais.

A Figura 21, que representa a matriz de confusão para *Recall* em relação ao experimento HAN-ESP-3-500, exemplifica bem a situação da classe “Despacho”, pois evidencia que documentos pertencentes a ela estão sendo classificados como ARE ou ‘’Acórdão”, o que reflete no

Classe	F1-score	Precisão	*Recall*
Acórdão	**0,9752**	0,9517	**1,0000**
ARE	0,9478	0,9134	0,9852
Despacho	0,9433	0,9779	0,9110
PRE	0,9651	0,9667	0,9635
Sentença	0,9705	**1,0000**	0,9427
Média	0,9604	0,9619	0,9605

Tabela 11 – Índices separados por classe de F1-score, Precisão e*Recall*, para o experimento de *pipeline* HAN-ESP-3-500-NO.

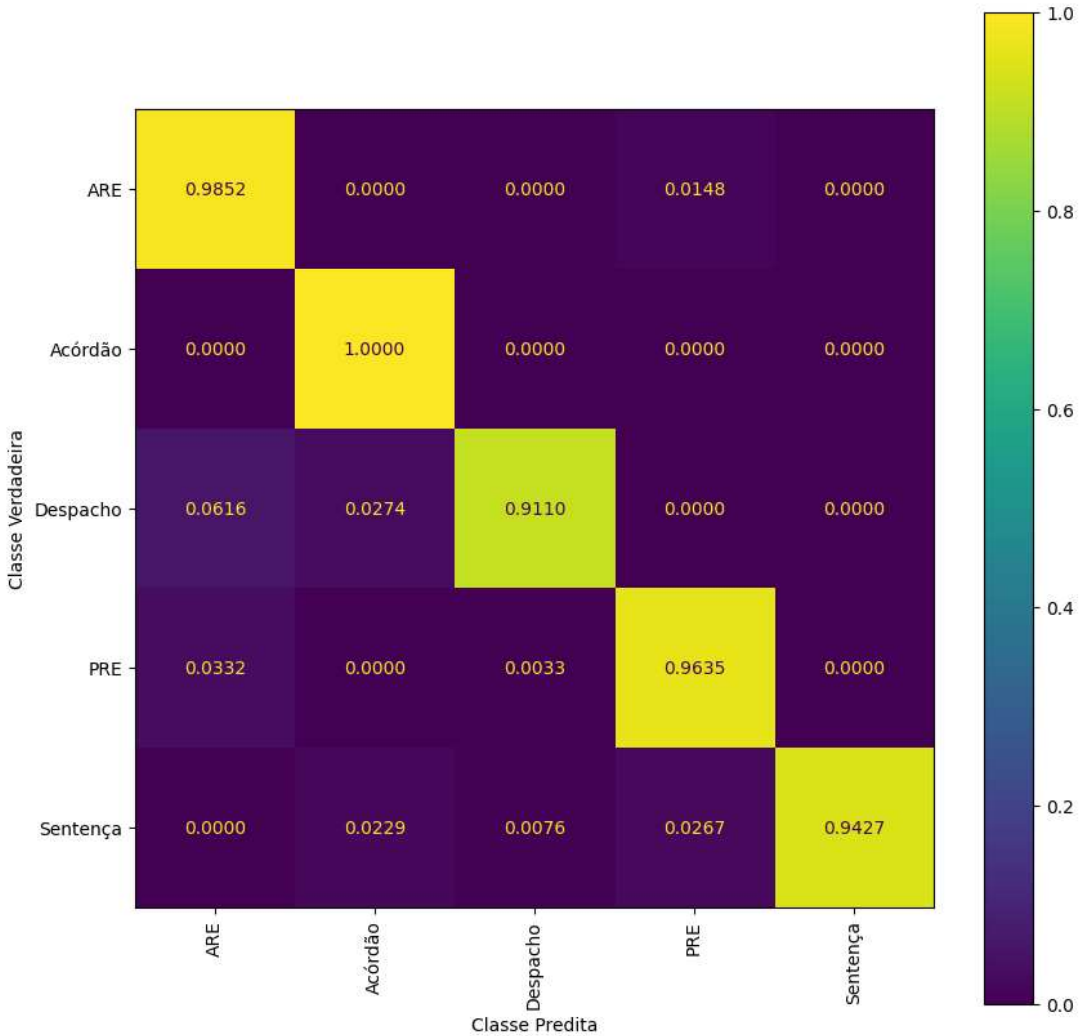


Figura 21 – Matriz de confusão para *Recall*, relacionado ao experimento HAN-ESP-3-500 para VICTOR-NO.

índice de Precisão da classe “Acórdão” que mesmo tendo o maior F1-score, não possui a maior Precisão.

5.3.1.2 Experimentos com *dataset* VICTOR-O100

Para o *dataset* VICTOR-O100, conforme indicado na Tabela 12, os melhores índices de F1-score, MCC e Precisão são obtidos a partir do experimento com a camada de *emebdding* pré-treinada, 3 páginas e 500 *tokens* como limites. De outro modo, o melhor *Recall* é obtido pelo experimento com a mesma configuração, porém com quantidade de páginas e *tokens* maiores, ou seja, HAN-GloVe-6-500-O100.

Quando se considera na comparação apenas os *pipelines* com camada de *embedding* especializada, verifica-se que o aumento da quantidade de *tokens* totais não aprimorou a performance, do mesmo modo que concentrar mais *tokens* e menos páginas, haja visto que o experimento com 6 páginas e 255 *tokens* alcançou os melhores índice dentre eles.

Experimento	**F-1 score**	**MCC**	**Precisão**	***Recall***
HAN-ESP-6-255-O100	0,7349	0,6959	0,6380	0,8950
HAN-ESP-3-500-O100	0,7258	0,6903	0,6201	0,9038
HAN-ESP-6-500-O100	0,7066	0,6713	0,6018	0,8921
HAN-GloVe-6-255-O100	0,7388	0,7070	0,6369	0,9120
HAN-GloVe-3-500-O100	**0,7513**	**0,7203**	**0,6538**	0,9066
HAN-GloVe-6-500-O100	0,6992	0,6527	0,5953	**0,9295**

Tabela 12 – Índices F1-score, Precisão, e *Recall* médios, e MCC para experimentos com a arquitetura HAN no *dataset* VICTOR-O100.

Em se tratando apenas daqueles que possuem camada de *embedding* pré-treinada, a concentração de mais *tokens* em menos páginas melhorou a performance, ao contrário do que ocorre com os *pipelines* com *embedding* especializado. Porém, em ambas categorias de camada de *embedding*, o aumento do número de *tokens* de 1500/1530 para 3000, degradou a performance, diminuindo 7% em termos de F1-score no caso da camada pré-treinada quando compara-se os experimentos HAN-GloVe-3-500 e HAN-GloVe-6-500.

De modo diverso ao que ocorre com os *pipelines* da Abordagem I, no caso do HAN, os valores médios de Precisão são menores que o *Recall*, indicando nesse caso, que os falsos positivos em relação às classes diferentes da classe “Outros” são maiores que os falsos negativos, ou seja, documentos da classe “Outros” estão sendo classificados de forma equivocada como das demais classes.

Analisando o desempenho entre as classes para o experimento HAN-GloVe-3-500-O100, a Tabela 13 mostra que a classe “Outros” apresenta os melhores índices para F1-score, Precisão e *Recall*, comportamento esperado em virtude de possuir mais de 95% dos exemplos empregados para treino. Além disso, verifica-se que para o *dataset* VICTOR-O100, a classe ‘’Despacho”, a semelhança do que ocorre no VICTOR-NO, também a presenta o pior desempenho para F1-score e Precisão, enquanto a classe ARE tem pior desempenho para *Recall*

Em virtude dos índices de Precisão na Tabela 13 serem relativamente menores em relação ao

Classe	F-1 score	Precisão	*Recall*
Acórdão	0,8300	0,7344	0,9543
ARE	0,6075	0,4923	0,7931
Despacho	0,5957	0,4549	0,8630
PRE	0,7302	0,6126	0,9036
Sentença	0,7610	0,6329	0,9542
Outros	**0,9833**	**0,9955**	**0,9713**
Média	0,7513	0,6538	0,9066

Tabela 13 – Índices separados por classe de F1-score, Precisão e*Recall*, para o experimento do *pipeline* HAN-GloVe-3-500-O100.

Recall, eles sugerem que a quantidade de falsos positivos para as demais classes diferentes da classe "Outros" é maior em relação ao de falso negativos, conforme explicitado no parágrafo anterior.

Na análise da Figura 22, que mostra a matriz de confusão para Precisão, é possível inferir que o maior problema para o experimento HAN-GloVe-3-500-O100 se deve ao fato de classificar documentos da classe "Outros" como sendo das demais classes, o que diminui o índice de *Recall* para ela, e consequentemente diminui também o índice de Precisão para as demais.

Como exemplo, na classe ''Despacho" que possui o menor F1-score, a taxa de falsos positivos em relação ao total de acertos é de aproximadamente 54%. Esse comportamento possivelmente é agravado pelo fato dessa classe, em conjunto com a ARE, possuir as menores quantidades de exemplos no conjunto de Treino, não obstante, são as duas que possuem o menor índice para F1-score no experimento em análise.

5.3.2 ULMFiT

Para os experimentos com ULMFiT, são consideradas duas possibilidades de uso do modelo de linguagem, a primeira aplica o modelo com ajuste fino realizado por (MENEZES-NETO; CLEMENTINO, 2022), portanto com documentos jurídicos distintos do *dataset* VICTOR, esse *pipeline* de experimentos será designado como "Genérico" (GEN). De outro modo, a segunda possibilidade realiza previamente à tarefa de classificação, o ajuste fino do modelo pré-treinado partindo do pré-treinamento do ULMFiT também disponibilizado por (MENEZES-NETO; CLEMENTINO, 2022), e dessa forma será denominado de "Especializado" (ESP).

A Figura 23 exemplifica os fluxos de ajuste do ULMFiT em cada fase, indicando os tipos de documentos empregados. Observa-se que a primeira fase, denominada "Pré-treinamento" aplica documentos de domínio geral, na segunda fase, correspondente ao ajuste fino, são consideradas duas possibilidades, genérico e especializado, após isso o classificador é ajustado para comparação dos efeitos de cada tipo de ajuste fino, bem como com os demais algoritmos experimentados. Portanto, ambas possibilidades partem do mesmo pré-treinamento, e divergem apenas nos documentos usados para o ajuste fino.

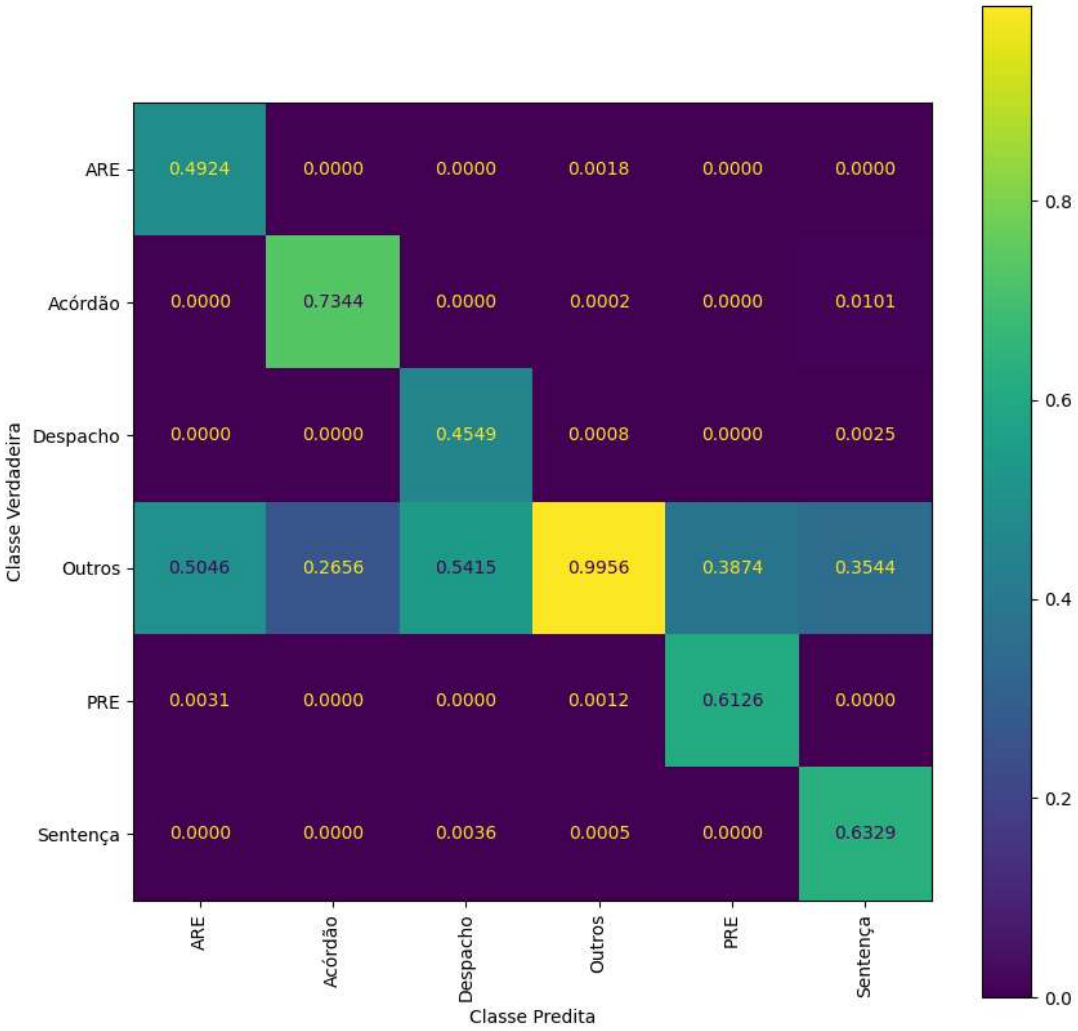


Figura 22 – Matriz de confusão para Precisão, relacionado ao experimento HAN-GloVe-3-500-O100

Para a efetivação do ajuste fino do ULMFiT com os dados do conjunto de Treino do *dataset* VICTOR-O100, tal como descrito em (ARAUJO; CAMPOS; SOUSA, 2020b), é aplicado o ajuste fino discriminativo com taxa de aprendizado máxima de 10^{-3}, *batch size* de 32, *weight decay* 0.1, e 6 épocas no total, sendo a primeira época apenas na camada mais externa, e as demais incluindo todas as camadas.

O ajuste do classificador segue o descrito em (MENEZES-NETO; CLEMENTINO, 2022), com taxa máxima de aprendizado de 10^{-3}, obtida a partir do algoritmo do "*LR range test*"(SMITH, 2015), função de perda Entropia Cruzada, otimizador Adam (KINGMA; BA, 2017) com betas $0,8$ e $0,7$ e *weight decay* $0,1$, função de ativação ReLU (AGARAP, 2018), ocorrendo durante 20 épocas no máximo, usando o algoritmo *One Cycle Policy* (SMITH; TOPIN, 2019), sendo as primeiras 4 épocas ajustando apenas a camada de classificação, em seguida realizando o descongelamento gradual das camadas do *encoder* de 4 em 4 épocas até a 12ª, e a partir da 13ª até a 20ª época com todas as camadas descongeladas. Nessa última fase, o algoritmo de *Early Stopping* determina o fim do treinamento, conforme indicado no início da Seção 5.3.

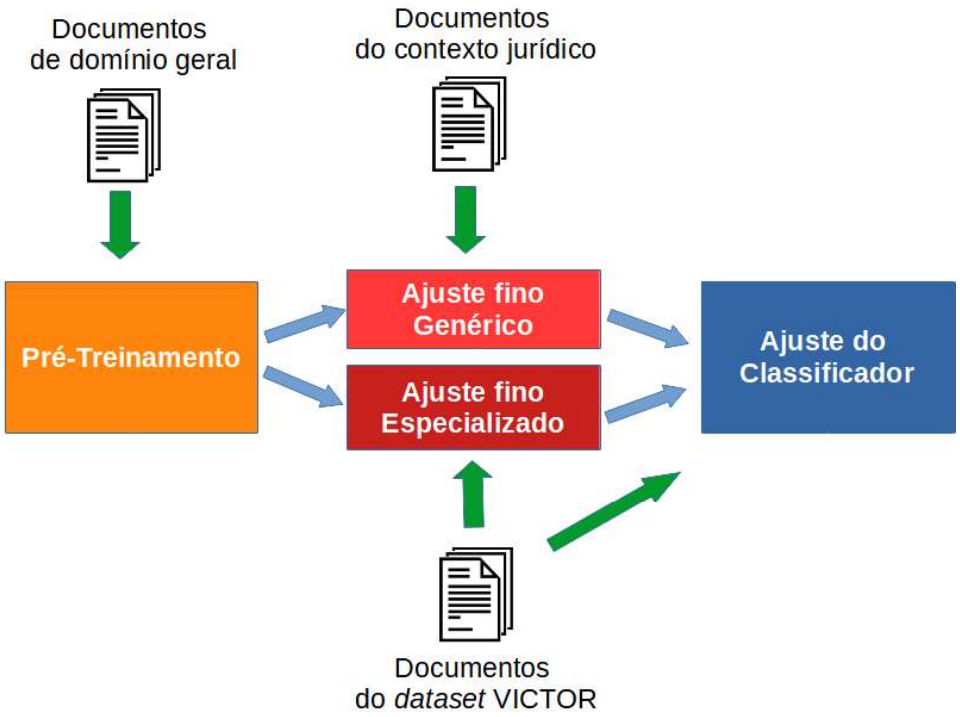


Figura 23 – Fluxo e documentos empregados no ajuste do ULMFiT, a partir do pré-treinamento até o classificador.

Tal como observado no item 3.5.2.4, por se tratar de um modelo bidirecional, o ULMFiT sugere que o ajuste fino e ajuste do classificador sejam realizados em ambas direções do documento, ou seja, começando do início do texto para o fim (forward), e do fim para início (backward). Dessa forma, os ajustes do ULMFiT são sempre em dois, e cada um resulta em um experimento diferente com resultados diferentes, e sendo designados por FW, BW e FWBW, ou seja, *forward*, *backward* e a junção dos dois, respectivamente.

Assim, as designações dos experimentos seguem a seguinte lógica: ULMFiT-FW-ESP-NO indica o experimento que usa a metodologia ULMFiT, no sentido *forward* (FW), com o ajuste fino especializado para o *dataset* VICTOR-NO; ou então ULMFiT-FWBW-GEN-O100, que indica o *pipeline* que aplica a junção dos resultados de BW e FW, partindo do modelo de ajuste fino genérico. Do mesmo modo como aplicado nos *pipelines* com HAN, aqui também há a limitação de 1530 *tokens*, com exceção de experimentos com a limitação de 3000 *tokens* realizados especificamente para o *dataset* VICTOR-O100, quando for o caso, a denominação irá incluir no final “3000”, por exemplo, ULMFiT-FW-ESP-O100-3000.

5.3.2.1 Experimentos ULMFiT com *dataset* VICTOR-NO

A Tabela 14 mostra os resultados dos experimentos com a metodologia ULMFiT para o *dataset* NO. Aqui, mais uma vez os *pipelines* que empregam ajuste especializado nos parâmetros do modelo levam vantagem de modo geral, a exceção do ULMFiT-FW-ESP-NO que obteve o pior desempenho diante dos demais. De qualquer modo, o experimento ULMFiT-BW-ESP-NO

possui os melhores índices de F1-score, MCC, Precisão e *Recall* inclusive quando comparados aos *pipelines* da Abordagem I, mas ainda assim com índices bem próximos com diferença apenas na terceira casa decimal em relação ao experimento TF-IDF_N3_SVM-NO.

Outro registro interessante que se observa na Tabela 14 em relação ao F1-score, é o fato que os *pipelines backwards* (BW) superam os *forwards* (FW) e os combinados (FWBW), para os ajustes finos genérico e especializado.

Experimento	F-1 score	MCC	Precisão	Recall
ULMFiT-FW-GEN-NO	0,9197	0,9003	0,9257	0,9189
ULMFiT-BW-GEN-NO	0,9699	0,9622	0,9705	0,9693
ULMFiT-FWBW-GEN-NO	0,9698	0,9634	0,9685	0,9715
ULMFiT-FW-ESP-NO	0,9149	0,8926	0,9145	0,9161
ULMFiT-BW-ESP-NO	**0,9790**	**0,9748**	**0,9789**	**0,9793**
ULMFiT-FWBW-ESP-NO	0,9767	0,9714	0,9762	0,9772

Tabela 14 – Índices F1-score, Precisão, e *Recall* médios, e MCC para experimentos com metodologia ULMFiT no *dataset* VICTOR-NO.

Ainda em relação ao *dataset* VICTOR-NO, a Tabela 15 e Figura 24, exibem os valores de F1-score, Precisão e *Recall* para cada classe, e a matriz de confusão para Precisão, respectivamente.

Classe	F-1 score	Precisão	*Recall*
Acórdão	**0,9975**	**0,9949**	**1.0000**
ARE	0,9706	0,9659	0,9754
Despacho	0,9622	0,9655	0,9589
PRE	0,9783	0,9832	0,9734
Sentença	0,9867	0,9848	0,9885
Média	0,9790	0.9789	0.9792

Tabela 15 – Índices de F1-score, Precisão e*Recall*, separados por classe para o experimento de *pipeline* ULMFiT-BW-ESP-NO.

Na análise detalhada do experimento com melhores índices, ou seja, ULMFiT-BW-ESP-NO, a Tabela 15 mostra que a classe com os menores valores de F1-score, Precisão e *Recall* é a "Despacho", enquanto a classe "Acórdão" apresenta o melhor desempenho. Essa característica pode ser observada na Figura 24, que mostra a classe "Despacho" e "PRE" com mais falsos positivos, refletindo portanto, no menor F1-score quando comparadas às demais.

5.3.2.2 Experimentos ULMFiT com *dataset* VICTOR-O100

Para o *dataset* VICTOR-O100, a Tabela 16 exibe os resultados dos experimentos nos mesmos moldes do VICTOR-NO, com a inclusão do experimento que aumenta o limite de *tokens* para 3000 exclusivamente para o *pipeline* com ajuste fino especializado.

Constata-se na Tabela 16, que mais uma vez os *pipelines* com ajuste fino especializado superam o genérico em termos de F1-score e MCC, a semelhança do que ocorre com os

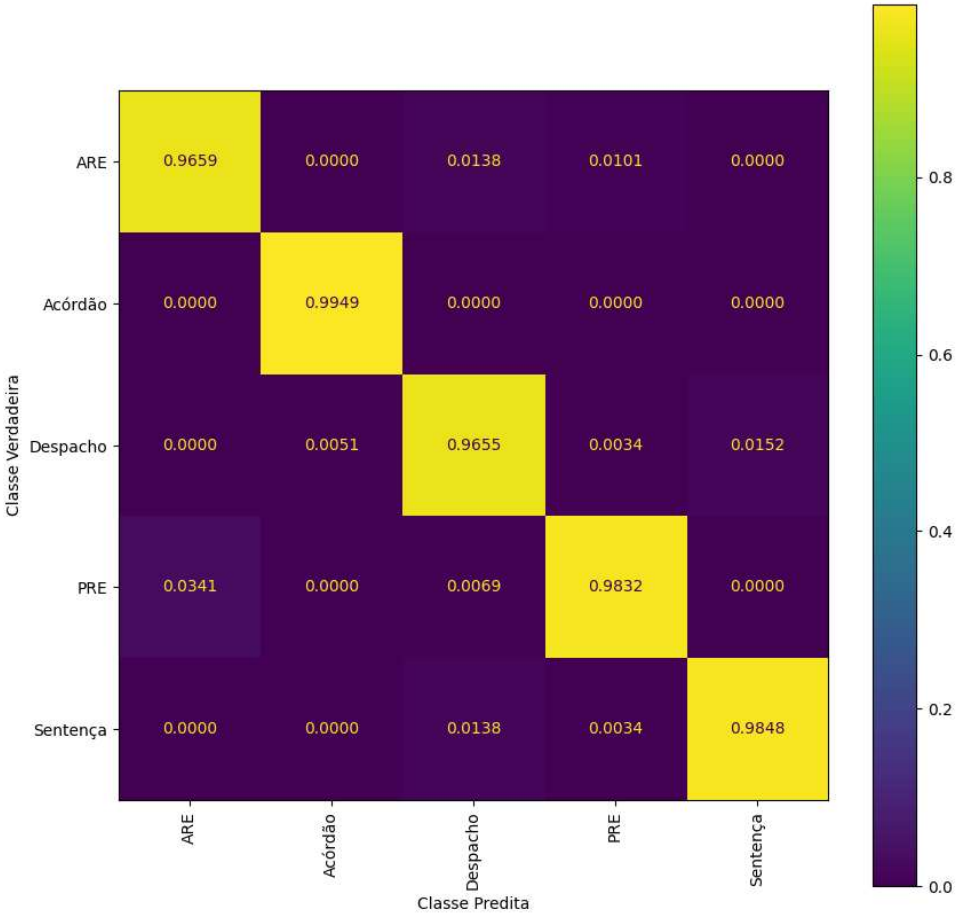


Figura 24 – Matriz de confusão para Precisão, relacionado ao experimento ULMFiT-BW-ESP-NO.

Experimento	F-1 score	MCC	Precisão	Recall
ULMFiT-FW-GEN-O100	0,5955	0,5560	0,4815	0,8746
ULMFiT-BW-GEN-O100	0,5958	0,5397	0,4918	0,8618
ULMFiT-FWBW-GEN-O100	0,6324	0,5913	0,5187	0,8951
ULMFiT-FW-ESP-O100	0,7711	0,7257	0,6934	0,8910
ULMFiT-BW-ESP-O100	**0,8135**	**0,7825**	**0,7460**	0,9002
ULMFiT-FWBW-ESP-O100	0,8106	0,7774	0,7439	0,8987
ULMFiT-FW-ESP-3000-O100	0,7289	0,6806	0,6356	0,8847
ULMFiT-BW-ESP-3000-O100	0,7704	0,7393	0,6734	0,9168
ULMFiT-FWBW-ESP-3000-O100	0,7452	0,7192	0,6388	**0,9182**

Tabela 16 – Índices F1-score, Precisão, e *Recall* médios, e MCC para experimentos com a metodologia ULMFiT em relação ao *dataset* VICTOR-O100.

classificadores da Abordagem I quando comparamos essa especificidade. Nesse caso, a diferença no F1-score entre o melhor desempenho com ajuste genérico e especializado, chega a ser 28% superior para o segundo, demonstrando o ganho em se especializar os parâmetros da rede com os documentos da tarefa de classificação final. Além disso, constata-se mais uma vez, a semelhança do que ocorre com os *pipelines* do HAN, que o aumento no número de *tokens* máximo de 1530 para 3000 não aprimorou o resultado.

Observa-se também na Tabela 16, a tendência dos experimentos com ULMFiT *backward* superarem aqueles com *forward*, este último na verdade o sentido natural de leitura do texto. Essa diferença fica evidente quando comparamos o experimento com melhor desempenho geral, ULMFiT-BW-ESP, e a sua versão *forward* (ULMFiT-FW-ESP), onde o primeiro apresenta um resultado de F1-score 5% superior ao segundo. Do mesmo modo, os experimentos que combinam FW e BW apresentam resultados intermediários quando comparados às suas versões individuais, a exceção do experimento com ajuste fino genérico (ULMFiT-FWBW-GEN), o qual apresentou índices superiores.

Mais uma vez, da mesma forma como ocorre com os *pipelines* com HAN, e ao contrário dos classificadores que empregam AM Tradicional (Abordagem I), os valores de Precisão médio para os experimentos com ULMFiT são menores que os valores de *Recall*, evidenciando esse comportamento comum que diferencia os classificadores da Abordagem II em relação à Abordagem I.

Na análise do desempenho por classe do experimento ULMFiT-BW-ESP, que possui os melhores índices de forma geral, a Tabela 17 mostra que, mais uma vez, de forma análoga ao que ocorre com os classificadores da Abordagem I e experimentos com redes hierárquicas (HAN), a classe "Outros" supera as demais em todos os índices.

Classe	**F-1 score**	**Precisão**	***Recall***
Acórdão	0.8904	0.8233	0.9695
ARE	0.6403	0.5540	0.7586
Despacho	0.7365	0.6543	0.8425
PRE	0.7825	0.6979	0.8904
Sentença	0.8423	0.7515	0.9580
Outros	**0.9887**	**0.9950**	**0.9824**
Média	0.8135	0.7460	0.9002

Tabela 17 – Índices separados por classe de F1-score, Precisão e*Recall*, para o experimento ULMFiT-BW-ESP-O100 em relação ao *dataset* VICTOR-0100.

Simetricamente ao que ocorre nos experimentos com melhores índices para VICTOR-O100 do *pipeline* com HAN e classificadores embasados em AM Tradicional, as classes com menor desempenho em relação ao F1-score são "Despacho" e "ARE", porém, conforme apontado na Tabela em análise e nas Tabelas 9 e 13, no caso dos *pipelines* da Abordagem I o valor médio de *Recall* é menor que o valor médio de Precisão, situação inversa ao que ocorre para os experimentos com melhores resultados para *pipelines* HAN e ULMFiT. Essa diferença sugere que o experimento TF-IDF_N3_RL tende a classificar documentos das demais classes como sendo "Outros", enquanto o HAN-GloVe-3-500 e ULMFiT-ESP-BW tendem a classificar os documentos do tipo "Outros" como sendo as demais classes.

A Figura 25, que mostra a matriz de confusão para Precisão em relação ao experimento ULMFiT-BW-ESP-O100, exemplifica bem o fato do índice médio de Precisão ser inferior ao

Recall, pois verifica-se que documentos da classe “Outros” estão sendo preditos como das demais classes, sendo que para o caso da classe ‘’ARE” (Agravo em Recurso Extraordinário), esse índice atinge *0,4460*, ou seja, cerca de 45% dos documentos preditos para essa classe, na verdade são documentos do tipo “Outros”.

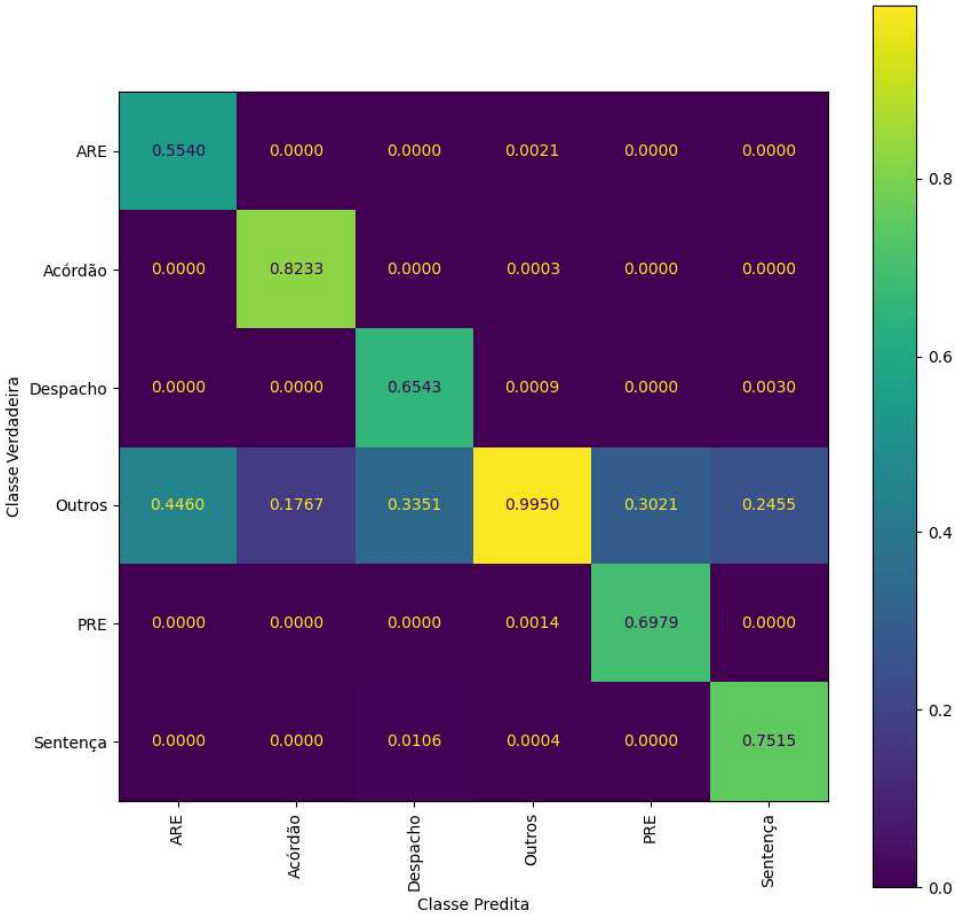


Figura 25 – Matriz de confusão para Precisão, relacionado ao experimento ULMFiT-BW-ESP-O100.

No Capítulo 6 são apresentadas as principais conclusões e contribuições da dissertação, além de propor trabalhos futuros sob o mesmo tema.

6 CONCLUSÕES E TRABALHOS FUTUROS

Primeiramente, convém citar que a maioria dos *pipelines* propostos alcançaram resultados satisfatórios em termos de F1-score e MCC, com resultados superiores a $0,7$, sendo que no caso do *dataset* VICTOR-NO o valor mínimo alcançado para F1-score foi de $0,9$, portanto, com níveis compatíveis aos trabalhos relacionados analisados no Capítulo 2. A exceção se aplica ao ULMFiT com ajuste fino genérico no *dataset* VICTOR-O100, que obteve índices abaixo $0,6$ para ambas medidas. Vale destacar, que o *dataset* VICTOR-O100 possui grande desbalanceamento entre as classes, onde os documentos do tipo "Outros" são cerca de 95% do total, o que em si traz enorme dificuldade no processo de aprendizado dos algoritmos.

Dessa forma, os *pipelines* das duas abordagens propostas são viáveis para aplicação no processo de classificação de documentos jurídicos do *dataset* VICTOR, que por se caracterizar por documentos da mais alta corte do país, recebe os mais diversos assuntos jurídicos de todos os Estados da Federação, o que confere certa diversidade aos exemplos usados para treinamento e teste dos experimentos propostos.

Em relação a Abordagem I e as diversas formas de representação computacional dos documentos com: TF-IDF, Word2Vec, FastText e BERT, percebe-se desempenho superior do TF-IDF em relação aos demais, principalmente quando se trata do *dataset* VICTOR-O100. Além disso, conclui-se também, que o nível de pré-processamento maior dos documentos com a inclusão da lematização e *bigram* para o TF-IDF, melhorou os resultados para ambos *datasets*, notadamente no caso do VICTOR-O100. Ao comparar exclusivamente o desempenho do classificador em si, percebe-se que o SVM possui índices superiores na maioria das comparações, porém o melhor resultado para o *dataset* O100 é obtido por meio da Regressão Logística.

Os resultados dos experimentos com SVM e Regressão Logística quando se aplica a representação computacional dos documentos com Word2Vec e FastText, evidenciam a superioridade quando adota-se os vetores de *embedding* especializados (ESP) treinados exclusivamente com os documentos do *dataset* VICTOR, em comparação aos vetores genéricos (GEN) disponibilizados por (HARTMANN et al., 2017a). Especialmente em relação ao *dataset* VICTOR-O100, para o caso mais extremo a superioridade chegou a aproximadamente 10% para o experimento Word2Vec-ESP_SVM-O100 em relação ao seu paralelo genérico. Já em relação ao *dataset* VICTOR-NO, as diferenças são bem menores, sendo a situação mais extrema para o *pipeline* FastText-(ESP/GEN)_RL-NO, com aproximadamente 2% de superioridade do especializado em relação ao genérico.

Resultados semelhantes na comparação da utilização de níveis diferentes de pré-processamento para o TF-IDF, bem como vetores de *embedding* genéricos e especializados para FastText e Word2Vec, são obtidos no trabalho (NOGUTI; VELLASQUES; OLIVEIRA, 2020), que usa os mesmos classificadores nos experimentos, porém com *dataset* de petições eletrônicas recebidas no Ministério Público Estadual do Paraná, onde os textos possuem características diversas que

não apenas jurídicas, além da média de *tokens* por documento ser menor que 100. Já no caso do *dataset* VICTOR, os documentos possuem natureza estritamente jurídica, bem como média de *tokens* por documento é aproximadamente 570. De qualquer modo, a semelhança nos resultados indica que a especialização dos vetores traz vantagens em ambas situações e deve ser considerada, pesando-se na escolha o fato da necessidade de maior tempo e recurso computacional para o ajuste dos vetores especializados em comparação aos vetores genéricos que já estão prontos para uso.

Ainda em relação aos *pipelines* da Abordagem I, é importante destacar a representação computacional do documento com o BERT, que mesmo não apresentando os melhores índices de F1-score, os resultados evidenciam que é possível aplicá-la nos *pipelines* propostos mesmo com a limitação de 512 *tokens* por documento. Em todo caso, a aplicação do modelo de linguagem BERT com SVM e Regressão Logística, salvo melhor juízo, não tem sido muito explorada na literatura para experimentos com documentos jurídicos do Brasil, o que mostra a relevância da aplicação desse Modelo de Linguagem em conjunto com Aprendizado Máquina Tradicional. Além disso, a adoção do modelo BERT-*large*, técnicas que mesclem o BERT com mecanismo de atenção (CHALKIDIS et al., 2020), ou mesmo com redes neurais recorrentes como em (MENEZES-NETO; CLEMENTINO, 2022), podem ser exploradas para possível aprimoramento dos resultados.

Relativamente aos experimentos da Abordagem II, os *pipelines* com HAN apresentam resultados diversos em relação a utilização da camada de *embedding* GloVe ou especializada (ESP) em relação a cada *dataset*. No caso do VICTOR-NO há pequena superioridade do melhor caso especializado em relação ao GloVe, provavelmente pelo fato de ter 95% menos documentos que o *dataset* O100, o ajuste dos parâmetros pré-treinados da camada GloVe durante o treinamento do algoritmo não apresentou o mesmo impacto para o *dataset* NO quando comparado ao O100, dado que o experimento com a camada de *embedding* GloVe obtém melhores índices de F1-score e MCC. Em relação ao aumento do número de *tokens* empregados no treinamento de 1530 para 3000 no caso do *dataset* VICTOR-O100, verifica-se que não trouxe ganhos para os resultados, além de tornar o processo de treinamento mais demorado.

Uma contribuição importante dos experimentos com HAN, salvo melhor juízo, é que não foram encontrados trabalhos de classificação de documentos com *datasets* do domínio jurídico do Brasil com essa técnica, apesar de ser empregado em estudos de outros países, a exemplo de (CHALKIDIS et al., 2020) com documento jurídicos em inglês, e (LIU; TU; SUN, 2019) com documentos em chinês.

Ainda na esteira dos algoritmos da Abordagem II, a metodologia ULMFiT obtém resultados interessantes, e demonstra a importância do processo de ajuste fino dos parâmetros do modelo de linguagem com os documentos da tarefa final de classificação, uma vez que os resultados mostram a superioridade desse *pipeline* em relação ao que recebeu o ajuste fino com documentos jurídicos de Tribunal e contexto diferentes do *dataset* VICTOR. Nesse ponto, é possível fazer um paralelo

com os *pipelines* da Abordagem I, que também revelam melhoras nos índices de comparação quando aplicam vetores de *embeddings* especializados. Esse benefício gerou inclusive resultados excelentes para o *dataset* VICTOR-NO, uma vez que os documentos empregados no ajuste fino especializado contavam também com os documentos da classe "Outros", ou seja, o ajuste fino mesmo sendo realizado com documentos que extrapolam o *dataset* da tarefa final, mas que guarde semelhança de contexto com ele, auxiliam na melhora do desempenho.

Cabe destacar ainda, conforme visualiza-se no gráfico da Figura 26, que há diferença em relação aos valores de Precisão e *Recall* nas duas abordagens, onde na Abordagem I há uma tendência de valores maiores para Precisão quando comparada ao *Recall*, e na Abordagem II esse comportamento se inverte. A linha traceja em vermelhor no gráfico da Figura 26 tem o objetivo apenas de mostrar a clara separação entre os aglomerados de pontos referentes a cada abordagem.

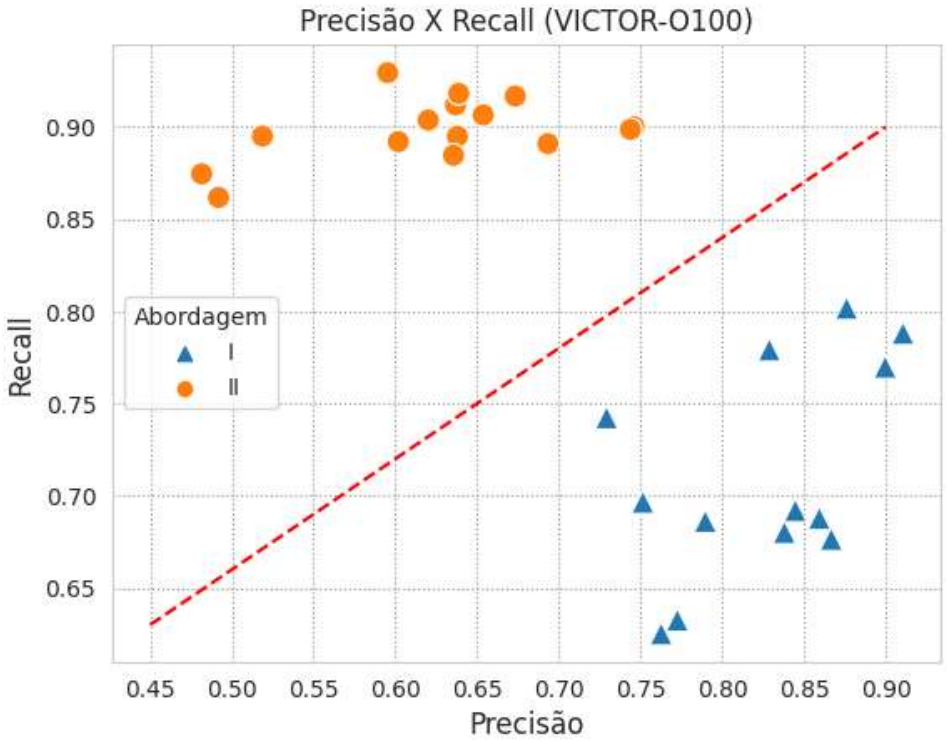


Figura 26 – Gráfico de Precisão e *Recall* para os *pipelines* das Abordagens I e II em relação aos experimentos com *dataset* VICTOR-O100.

A observação em relação a separação dos pontos no gráfico da Figura 26 é relevante, pois indica que os *pipelines* da Abordagem I tendem a ter maior número de Falsos Negativos, ou seja, documentos das demais classes estão sendo rotuladas como "Outros", situação que se inverte para a Abordagem II, onde a quantidade de Falsos Positivos é maior, indicando que os documentos do tipo "Outros" estão sendo preditos como das demais classes. Essa diferença em relação à Precisão e *Recall* nas Abordagens, se torna importante no momento de aplicação prática da técnica de classificação automática de documentos para o *dataset* VICTOR, pois a

área responsável pelo conhecimento de negócio precisa definir qual tipo de erro é mais tolerado, ou seja, mais Falsos Positivos ou mais Falsos Negativos para as classes diferentes de "Outros".

Outro ponto interessante em relação aos *pipelines* experimentados, no que diz respeito às análises dos melhores resultados de cada um deles para os índices separados por classe, é que todos revelam maior dificuldade no processo de classificação das classes "ARE" e "Despacho", onde ambas obtêm os menores índices de F1-score em relação às demais. Porém, a forma como esse erro ocorre é diferente, pois nos algoritmos da Abordagem II os índices de Precisão são menores quando comparados ao *Recall*, enquanto que no experimento TF-IDF-N3_RL ocorre de forma inversa. A dificuldade maior na classificação dessas duas classes também é verificada nos resultados do trabalho (ARAUJO et al., 2020a).

Ao final, conforme mencionado no início do capítulo, todos os *pipelines* experimentados são possíveis de utilização para o processo de classificação de documentos do *dataset* VICTOR, sendo que, de forma objetiva e olhando apenas para os índices, o experimento com representação computacional baseada em TF-IDF e Regressão Logística obtém o melhor resultado de F1-score e MCC para o *dataset* VICTOR-O100, enquanto o ULMFiT com ajuste fino especializado para o *dataset* VICTOR-NO. Porém, em muitos casos os resultados foram bem próximos, com diferenças apenas na segunda ou terceira casas decimais de índices que variam de 0 a 1 para F1-score, e -1 a 1 para MCC. Essa conclusão é análoga e reforça os resultados do trabalho (ARAUJO; CAMPOS; SOUSA, 2020b) que compara classificação de documentos do Diário Oficial do Distrito Federal com SVM e ULMFiT, onde os dois alcançam resultados bem próximos, porém com pequena vantagem para o SVM em termos de F1-score médio.

Portanto, além dos números em si, é necessário levar em consideração no momento da escolha do algoritmo de classificação a disponibilidade de recursos computacionais e o tempo à disposição, uma vez que os classificadores lineares em conjunto com representação computacional TF-IDF apresentam menor necessidade de computação em comparação ao ULMFiT e HAN, que por sua vez requerem processamento com GPU, tanto para treinamento, quanto para o processo de inferência. Ademais, mesmo os experimentos da Abordagem I, mas que usam a transformação de documentos a partir de modelos neurais, exigem maiores recursos computacionais no processo de especialização dos vetores de *embedding*, o que não descarta a utilização dos classificadores tradicionais com vetores pré-treinados, dado que os experimentos realizados mostram que é possível obter índices superiores a $0,7$ em termos de F1-score nesse último caso.

6.1 Contribuições

Além das contribuições dos resultados e conclusões da pesquisa em si, durante o Mestrado foram realizadas duas publicações com revisões sistemáticas em relação ao tema, a primeira *Text Classification in Law Area: a Systematic Review* no *Symposium on Knowledge Discovery, Mining*

and Learning (KDMILE), e sua versão estendida sob mesmo título no *Jornal de Informação e Gerenciamento de Dados (JOURNAL OF INFORMATION AND DATA MANAGEMENT - JIDM)* da Sociedade Brasileira de Computação.

Também, o repositório com os códigos Python em formato Jupyter Notebook[1] empregados para pré-processamento, representação computacional dos documentos, treinamento dos modelos e ajuste fino dos modelos de linguagem ULMFiT e BERT, estão disponíveis publicamente em <https://github.com/vtsimoes/class_victor_dataset> para auxiliar outros pesquisadores e estudantes da área, pois contém todos os passos usados nos experimentos e são resultados de pesquisa por bibliotecas e códigos *Python* para serem empregados na área de Processamento de Linguagem Natural, sendo um bom ponto de partida na pesquisa e experimentos, principalmente para aqueles que estão iniciando na área, uma vez que os códigos no formato de *Notebooks* se tornam mais didáticos e de fácil entendimento.

6.2 Trabalhos Futuros

Como primeira possibilidade e sugestão de trabalhos futuros, está a necessidade de se implementar processo de busca de hiperparâmetros para os algoritmos de *Deep Learning* HAN e ULMFiT, nesse último caso principalmente para a fase de ajuste do classificador. Além disso, ainda em relação ao HAN, há a possibilidade de trabalhar ao nível de sentenças e não de páginas, o que demanda a coleta de documentos do *dataset* VICTOR em formato PDF para extração desse texto sem a exclusão dos sinais de pontuação indicativos de sentença.

Em relação à estrutura do texto dos documentos, entender melhor as diferenças entre eles, aprimorando o pré-processamento com a inclusão de técnicas de mineração de texto, e a partir daí extrair características que auxiliem o classificador, com o objetivo de melhorar o desempenho.

Qualitativamente, é interessante o estudo do impacto que a aplicação de *pipelines* de classificação de documentos jurídicos representam no dia a dia dos órgãos de justiça, indicando as possíveis melhoras em termos de processos laborais dos setores ora responsáveis por esse serviço, bem como aprimoramentos e celeridade dos procedimentos internos desses órgãos, uma vez que, possivelmente, a partir da aplicação de técnicas de classificação automática, haverá maior disponibilidade de recursos humanos para tarefas com maior grau de cognição.

Como última sugestão, incluir na comparação modelos de linguagens mais complexos, como os *Large Language Models* (LLM), a exemplo do GPT (BROWN et al., 2020), bem como o BERT com ajuste simultâneo dos parâmetros do modelo e do classificador neural, o que não foi possível ser realizado nesse trabalho em virtude da limitação de memória disponível para GPU no ambiente *Google Colab Notebook*. Ou seja, novos algoritmos e técnicas, principalmente em relação ao *Deep Learning*, podem ser empregados para alcançar melhores resultados.

[1] <https://jupyter.org/>

REFERÊNCIAS

AGARAP, A. F. Deep learning using rectified linear units (relu). **CoRR**, abs/1803.08375, 2018. Disponível em: <http://arxiv.org/abs/1803.08375>.

ARAUJO, P. H. Luz de et al. VICTOR: a dataset for Brazilian legal documents classification. In: **Proceedings of the 12th Language Resources and Evaluation Conference**. Marseille, France: European Language Resources Association, 2020a. p. 1449–1458. ISBN 979-10-95546-34-4. Disponível em: <https://www.aclweb.org/anthology/2020.lrec-1.181>.

ARAUJO, P. H. Luz de; CAMPOS, T. E. de; SOUSA, M. Magalhães Silva de. Inferring the source of official texts: Can svm beat ulmfit? In: QUARESMA, P. et al. (Ed.). **Computational Processing of the Portuguese Language**. Cham: Springer International Publishing, 2020b. p. 76–86. ISBN 978-3-030-41505-1.

BIRD, S.; LOPER, E.; KLEIN, E. **Natural Language Processing with Python**. [S.l.]: O'Reilly Media Inc, 2009. ISBN 0596516495.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, MIT Press, Cambridge, MA, v. 5, p. 135–146, 2017. Disponível em: <https://aclanthology.org/Q17-1010>.

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

BROWN, T. B. et al. Language models are few-shot learners. **CoRR**, abs/2005.14165, 2020. Disponível em: <https://arxiv.org/abs/2005.14165>.

CHALKIDIS, I.; ANDROUTSOPOULOS, I.; ALETRAS, N. Neural legal judgment prediction in English. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy: Association for Computational Linguistics, 2019. p. 4317–4323. Disponível em: <https://www.aclweb.org/anthology/P19-1424>.

CHALKIDIS, I. et al. LEGAL-BERT: The muppets straight out of law school. In: **Findings of the Association for Computational Linguistics: EMNLP 2020**. Online: Association for Computational Linguistics, 2020. p. 2898–2904. Disponível em: <https://www.aclweb.org/anthology/2020.findings-emnlp.261>.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: <https://doi.org/10.1145/2939672.2939785>.

CHEN, Y.-C. **A Tutorial on Kernel Density Estimation and Recent Advances**. 2017.

CHO, K. et al. On the properties of neural machine translation: Encoder–decoder approaches. In: **Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 103–111. Disponível em: <https://aclanthology.org/W14-4012>.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: **NIPS 2014 Workshop on Deep Learning, December 2014**. [S.l.: s.n.], 2014.

SULEA, O. et al. Exploring the use of text classification in the legal domain. **CoRR**, abs/1710.09306, 2017b. Disponível em: <http://arxiv.org/abs/1710.09306>.

SULEA, O.-M. et al. Predicting the law area and decisions of French Supreme Court cases. In: **Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017**. Varna, Bulgaria: INCOMA Ltd., 2017a. p. 716–722. Disponível em: <https://doi.org/10.26615/978-954-452-049-6_092>.

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

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000. Disponível em: <https://www.sciencedirect.com/science/article/pii/S002200009791504X>.

GOLDBERG, Y. A primer on neural network models for natural language processing. **J. Artif. Int. Res.**, AI Access Foundation, El Segundo, CA, USA, v. 57, n. 1, p. 345–420, sep 2016. ISSN 1076-9757.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In: **Anais do XI Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana**. Porto Alegre, RS, Brasil: SBC, 2017. p. 122–131. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/stil/article/view/4008>.

HARTMANN, N. S. et al. Portuguese word embeddings evaluating on word analogies and natural language tasks. In: **Anais do XI Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana**. Porto Alegre, RS, Brasil: SBC, 2017. p. 122–131. Disponível em: <httpssol.sbc.org.brindex.phpstilarticleview4008>.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/0893608089900208>.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Melbourne, Australia: Association for Computational Linguistics, 2018. p. 328–339. Disponível em: <https://www.aclweb.org/anthology/P18-1031>.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: **Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37**. [S.l.]: JMLR.org, 2015. (ICML'15), p. 448–456.

JOULIN, A. et al. Bag of tricks for efficient text classification. In: **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics:**

Volume 2, Short Papers. Valencia, Spain: Association for Computational Linguistics, 2017. p. 427–431. Disponível em: <https://aclanthology.org/E17-2068>.

KIM, H. K.; KIM, H.; CHO, S. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. **Neurocomputing**, v. 266, 05 2017.

KIM, Y. Convolutional neural networks for sentence classification. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1746–1751. Disponível em: <https://aclanthology.org/D14-1181>.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017.

LIU, Z.; TU, C.; SUN, M. Legal cause prediction with inner descriptions and outer hierarchies. In: ____. [S.l.: s.n.], 2019. p. 573–586. ISBN 978-3-030-32380-6.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. Cambridge, UK: Cambridge University Press, 2008. ISBN 978-0-521-86571-5. Disponível em: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>.

MANNING, C. D.; SCHÜTZE, H. **Foundations of Statistical Natural Language Processing**. Cambridge, Massachusetts: The MIT Press, 1999. Disponível em: <http://nlp.stanford.edu/fsnlp/>.

MAVERICK, G. V. Computational analysis of present-day american english. henry kucera, w. nelson francis. In: . [S.l.: s.n.], 1969. p. 71–75.

MCCLOSKEY, M.; COHEN, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In: BOWER, G. H. (Ed.). Academic Press, 1989, (Psychology of Learning and Motivation, v. 24). p. 109–165. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0079742108605368>.

MENEZES-NETO, E. Jacob de; CLEMENTINO, M. B. M. Using deep learning to predict outcomes of legal appeals better than human experts: A study with data from brazilian federal courts. **PLOS ONE**, Public Library of Science, v. 17, n. 7, p. 1–20, 07 2022. Disponível em: <https://doi.org/10.1371/journal.pone.0272287>.

MERITY, S.; KESKAR, N. S.; SOCHER, R. **Regularizing and Optimizing LSTM Language Models**. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1708.02182>.

MERITY, S. et al. Pointer sentinel mixture models. **CoRR**, abs/1609.07843, 2016. Disponível em: <http://arxiv.org/abs/1609.07843>.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: BENGIO, Y.; LECUN, Y. (Ed.). **1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings**. [s.n.], 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

MOTA, C. et al. Classificação de páginas de petições iniciais utilizando redes neurais convolucionais multimodais. In: **Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2020. p. 318–329. ISSN 0000-0000. Disponível em: <httpssol.sbc.org.brindex.phpeniacarticleview12139>.

NEGRI, R. **Reconhecimento de Padrões: Um estudo dirigido**. [S.l.: s.n.], 2021. ISBN 9786555061635.

NOGUTI, M. Y.; VELLASQUES, E.; OLIVEIRA, L. S. Legal document classification: An application to law area prediction of petitions to public prosecution service. In: **2020 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2020. p. 1–8. ISSN 2161-4407.

POLO, F. et al. Legalnlp - natural language processing methods for the brazilian legal language. In: **Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2021. p. 763–774. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/18301>.

ŘEHŮŘEK, R.; SOJKA, P. Software Framework for Topic Modelling with Large Corpora. In: **Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks**. Valletta, Malta: ELRA, 2010. p. 45–50.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. [S.l.]: Prentice Hall, 2010.

SAVAS, C.; DOVIS, F. The impact of different kernel functions on the performance of scintillation detection based on support vector machines. **Sensors**, v. 19, n. 23, 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/23/5219>.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM COMPUTING SURVEYS**, v. 34, p. 1–47, 2002.

SILVA, A. C.; MAIA, L. C. G. The use of machine learning in the classification of electronic lawsuits: An application in the court of justice of minas gerais. In: CERRI, R.; PRATI, R. C. (Ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2020. p. 606–620. ISBN 978-3-030-61377-8.

SILVA, N.; BRAZ, F.; CAMPOS, T. de. Document type classification for brazil's supreme court using a convolutional neural network. In: . [S.l.: s.n.], 2018. p. 7–11.

SMITH, L. N. No more pesky learning rate guessing games. **CoRR**, abs/1506.01186, 2015. Disponível em: <http://arxiv.org/abs/1506.01186>.

SMITH, L. N.; TOPIN, N. Super-convergence: very fast training of neural networks using large learning rates. In: PHAM, T. (Ed.). **Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications**. SPIE, 2019. v. 11006, p. 1100612. Disponível em: <https://doi.org/10.1117/12.2520589>.

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

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

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

YANG, Z. et al. Hierarchical attention networks for document classification. In: **Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. San Diego, California: Association for Computational Linguistics, 2016. p. 1480–1489. Disponível em: <https://aclanthology.org/N16-1174>.

ZAHEER, M. et al. Big bird: Transformers for longer sequences. **Advances in Neural Information Processing Systems**, v. 33, 2020.

ZHANG, W.; YOSHIDA, T.; TANG, X. A comparative study of tf*idf, lsi and multi-words for text classification. **Expert Systems with Applications**, v. 38, n. 3, p. 2758–2765, 2011. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417410008626>.

ZONG, C.; XIA, R.; ZHANG, J. **Text Data Mining**. Springer Singapore, 2021. ISBN 978-981-16-0100-2. Disponível em: <https://doi.org/10.1007/978-981-16-0100-2>.