PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

JULIANA OBINO MASTELLA

UMA METODOLOGIA USANDO AMBIENTES PARALELOS PARA OTIMIZAÇÃO DA CLASSIFICAÇÃO DE TEXTOS APLICADA A DOCUMENTOS JURÍDICOS

Porto Alegre
2020

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA METODOLOGIA USANDO AMBIENTES PARALELOS PARA OTIMIZAÇÃO DA CLASSIFICAÇÃO DE TEXTOS APLICADA A DOCUMENTOS JURÍDICOS

JULIANA OBINO MASTELLA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. César A. F De Rose

Porto Alegre
2021

Juliana Obino Mastella

Uma metodologia usando ambientes paralelos para otimização da classificação de textos aplicada a documentos jurídicos

Tese/Dissertação apresentada como requisito parcial para obtenção do grau de Doutor/Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciencia da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 31 de agosto de 2020.

BANCA EXAMINADORA:

Prof. Dr. Sandro José Rigo (Unisinos)

Profa. Dra. Renata Vieira (PUCRS)

Prof. Dr. César Augusto F. De Rose (PPGCC/PUCRS - Orientador)

Ficha Catalográfica

M423m Mastella, Juliana Obino

Uma metodologia usando ambientes paralelos para otimização da classificação de textos aplicada a documentos jurídicos / Juliana Obino Mastella. – 2020.
67.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Fonticielha De Rose.

1. Classificação de textos. 2. Algoritmos de classificação. 3. PLN. 4. Parameter sweep. 5. Paralelismo. I. De Rose, César Augusto Fonticielha. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

DEDICATÓRIA

Dedico este trabalho a minha filha Diana Mastella de Castro e a meu marido Diego Dias de Castro.

“Quanto mais aumenta nosso conhecimento,
mais evidente fica nossa ignorância.”
(John F. Kennedy)

AGRADECIMENTOS

Agradeço primeiramente a meu orientador, prof. Dr. César Augusto Fonticielha De Rose, por ter acreditado em mim, por ter comprado este desafio comigo desde o início e por ter me apoiado em todas as fases desse Mestrado. Obrigada pelo aprendizado no caminho que trilhamos juntos nas disciplinas, fora delas e durante a dissertação. Ainda, muito obrigada por sempre ter sido disponível, presente, compreensivo e motivador quando foi necessário.

Agradeço aos meus colegas de grupo que me apoiaram em diferentes momentos, durante as disciplinas e a dissertação: Dionatrã Kirchoff, Matheus Lyra e Vinícius Meyer.

Agradeço à prof. Dra. Renata Vieira, pelos ensinamentos nas disciplinas e por todo apoio, compreensão, avaliação e críticas construtivas ao longo da elaboração dessa dissertação: disciplina de PLN, PEP e Seminário de Andamento.

Agradeço a meus colegas de trabalho e a minhas chefias: Juliana Bobsin, Carolina Mobüs e Luis Felipe Schneider por terem me apoiado desde o início na flexibilização de horário que se fez necessária para cursar algumas disciplinas do Mestrado. Agradeço ao Tribunal de Justiça do Rio Grande do Sul por ter contribuído para essa pesquisa com o fornecimento de dados e conhecimento que se fez necessário.

Agradeço a todos demais professores que tive no caminho, desde o colégio, graduação, mestrado, sem vocês certamente também não teria chegado até aqui. Cada um foi especial e teve sua importância na aquisição de conhecimento, gosto pelo estudo, curiosidade pelo novo.

Agradeço à Capes e à HP por terem me apoiado nesta trajetória, tornando este sonho possível investindo nos meus estudos.

Agradeço pelo apoio e pela compreensão de todos os amigos que estiveram comigo e pacientemente entenderam minha ausência por vezes necessária em função do estudo.

Agradeço a meus pais por sempre terem sido presentes em minha vida, por terem me apoiado nas minhas decisões e por terem me ensinado o valor que tem o estudo.

Por fim, mas certamente não menos importante, agradeço especialmente a meu amado marido, Diego, que sempre me apoiou nos estudos desde a graduação, entendendo a minha ausência que se tornou necessária por vários momentos em favor de um objetivo a ser alcançado, seja na graduação, seja no mestrado. Pela paciência que teve comigo e

suporte que me deu durante todas as fases desse caminho e pelo meu maior presente que em breve estará em nossos braços: nossa filha Diana.

UMA METODOLOGIA USANDO AMBIENTES PARALELOS PARA OTIMIZAÇÃO DA CLASSIFICAÇÃO DE TEXTOS APLICADA A DOCUMENTOS JURÍDICOS

RESUMO

Nos últimos anos testemunhou-se um crescimento exponencial do volume, da variabilidade e da velocidade com que novos dados são gerados. Sabe-se que a maior parte desses dados se apresenta de forma não-estruturada, o que aumenta ainda mais o desafio de analisar esses dados. Nesse cenário, a aplicação de técnicas de Processamento da Linguagem Natural (PLN) para classificação de textos de forma automática tem despertado o interesse de pesquisadores dos mais diversos domínios do conhecimento, dentre os quais pode-se destacar as Ciências Jurídicas. O Direito inerentemente depende da análise de um grande volume de informações textuais, o que o torna uma área com grande potencial para aplicação de técnicas de PLN. A escolha do algoritmo para solucionar um determinado problema de classificação de textos não é uma tarefa trivial. A qualidade e a viabilidade da abordagem de classificação escolhida dependerão do problema a ser resolvido, do volume e do comportamento dos dados, além da melhor utilização dos recursos computacionais disponíveis para que o resultado seja entregue em tempo adequado. Motivada pelo problema da classificação automática de textos jurídicos para aplicação a processos eletrônicos de um Tribunal Estadual Brasileiro, esta pesquisa propõe uma metodologia para otimizar a escolha de parâmetros do algoritmo de classificação de documentos jurídicos paralelizando o treinamento de Redes Neurais Recorrentes Bi-LSTM. Para aplicação a dados reais, 107.010 petições de um Tribunal Estadual Brasileiro, com classes previamente anotadas, foram submetidas ao treinamento de 216 Redes Neurais Recorrentes em paralelo. Ao final do treinamento, o modelo com melhor desempenho individual apresentou F1 = 0,846. Combinando-se os 4 melhores resultados individuais através de uma técnica Ensemble, pela regra da soma, não foi identificada melhora no desempenho (F1 = 0,826). Através do

treinamento em paralelo dos modelos, foi possível chegar a um resultado superior à maioria das parametrizações testadas (10% melhor do que a pior parametrização testada e 9,8% superior à média das combinações testadas) em aproximadamente 20 vezes menos tempo do que se levaria para testar todas as mesmas possibilidades de maneira sequencial.

Palavras-Chave: classificação de textos, Algoritmos de classificação, mineração de textos, classificação de documentos, documentos jurídicos, PLN, parameter sweep, paralelismo.

A METHODOLOGY USING PARALLEL ENVIRONMENTS TO OPTIMIZE TEXT CLASSIFICATION IN LEGAL DOCUMENTS

ABSTRACT

In last years it has been witnessed an exponential growth of data volume, data variability and data velocity. It is known that most of them are in an unstructured availability which intensify the data analysis challenge. Considering this scenario, the usage os Natural Language Processing (NLP) tools for text classification has been inspiring researchers from several knowlage domains, among them it can be highlighted the Legal Sciences. The justice in its root depends on analysis of huge text data volume which turns it into an important potential area for applying NLP tools. The choice of an algorithm for solving a specific text classification issue is not a trivial task. The picked classification approach quality and viability will depends on the issue to be solved, the data volume and the data behavior, in addition to the best use of available computational resources in order to results be delivered in time. Motivated by the problem of automatic classification of legal texts for application to electronic processes of a Brazilian State Court, this research proposes a methodology to optimize the choice of parameters for the classification algorithm of legal documents paralleling the training of Bi-LSTM Recurrent Neural Networks. For data application 107,010 petitions from a Brazilian State Court, with classes previously noted, underwent training of 216 Recurrent Neural Networks in parallel. At the end of training, the best individual performance was F1 = 0.846. Combining the 4 best models through an Ensemble technique resulted in a final model with lower performance than the best individual one (F1 = 0.826). Through the parallel training of models it was possible to reach a superior result to the majority of the tested parameterizations (10 % better than the worst parameterization tested and 9.8% better than the average) in approximately 20 times less time than it would take for test all the same possibilities sequentially.

Keywords: Text classification,classification algorithms, text mining, documents classification, legal documents, NLP, parameter sweep, parallelism.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos testemunhou-se um crescimento exponencial do volume, da variabilidade e da velocidade com que novos dados são gerados. Ao mesmo tempo em que esse fenômeno traz grandes oportunidades, ele impõe novos desafios a diversas áreas. A atual conjuntura permite a aplicação de técnicas de *Machine Learning* (ML) a um volume de dados sem precedentes, a partir dos quais torna-se possível identificar padrões refinados e ajustar modelos preditivos cada vez mais acurados. Em contrapartida, grande parte das abordagens tradicionais não foi desenvolvida para trabalhar de forma eficiente com tal volume de dados, o que às vezes inviabiliza a extração de informações relevantes em tempo hábil para a tomada de ação [49]. Adicionalmente, sabe-se que a maior parte desses dados se apresenta de forma não-estruturada, como imagens, áudios e textos, o que agrava ainda mais o desafio [48]. Esse cenário impulsionou o avanço e a aproximação de diferentes campos do conhecimento relacionados à análise de dados, dentre os quais pode-se citar: *High Performance Computing* (HPC), Estatística, ML, *Data Mining* (DM) e Processamento da Linguagem Natural (PLN).

O contexto apresentado fomenta o estudo de métodos para analisar grandes volumes de dados não-estruturados de forma eficiente. Uma das tarefas que tem despertado interesse é a classificação de textos de forma automática. Apesar de não haver unanimidade quanto às nomenclaturas e à quantidade de etapas do processo para classificação de textos, observa-se um fluxo geral padrão. Mironczuk et. al. (2018) [27] propõe um processo padrão com seis partes principais: (1) coleta de dados, (2) análise de dados para anotação de classes, (3) construção e ponderação de *features*, (4) seleção e projeção de *features*, (5) treinamento de um modelo de classificação e (6) avaliação da solução. Dado a complexidade de cada etapa, o escopo de interesse desta pesquisa será limitado a elementos da quinta etapa: escolha do método e treinamento do algoritmo de classificação. A figura 2.1 ilustra de forma esquematizada as fronteiras deste estudo em relação ao contexto apresentado.

Múltiplas abordagens veem sendo propostas para a classificação de textos. Dentre as clássicas, pode-se citar *Naive Bayes*, árvores de decisão e *Artificial Neural Networks* (ANN) [3, 28, 44, 20]. Mais recentemente, abordagens de *Deep Learning*(DL) como *Recurrent Neural Networks* (RNN), *Auto-encoder Neural Networks* (AENN), *Convolution Neural Networks* (CNN), *Long Short-Term Memory*(LSTM) vem ganhando destaque nessa área também [3, 20]. Além disso, há a possibilidade de combinar diferentes algoritmos de classificação utilizando um *Ensemble Method* (EM), que em muitos casos tem apresentado resultados superiores aos de cada técnica individualmente. Uma dificuldade importante em parte dessas abordagens é a falta de uma combinação de parâmetros única que atinja o melhor desempenho para todas as situações. Devido ao grande número de possibilidades

e o tempo de processamento que seria necessário para testar todas, muitas vezes o que ocorre é o especialista propor uma combinação de parâmetros que acredita ser a melhor de acordo com sua experiência, sem ter a certeza se poderia atingir melhor desempenho com algumas modificações.

Em termos práticos, no intuito de escolher a abordagem para solucionar um determinado problema de classificação de textos, outro aspecto importante a ser considerado é a disponibilidade de recursos computacionais. A qualidade e a viabilidade da abordagem de classificação escolhida dependerão, entre outros fatores, do prazo existente e da forma de utilização dos recursos computacionais disponíveis a fim de que o resultado seja entregue em tempo adequado[20].

As técnicas para classificação automática de textos têm despertado o interesse de pesquisadores dos mais diversos domínios do conhecimento, dentre os quais pode-se destacar as Ciências Jurídicas [20]. O Direito inerentemente depende da análise de um grande volume de informações textuais, motivo pelo qual tem-se explorado o potencial das técnicas de PLN em otimizar a realização de muitas dessas tarefas. Contudo, é importante considerar que a legislação varia conforme o país, estado, município jurisdicional, além de serem escritas no respectivo idioma de origem. Frente ao exposto, mostra-se relevante a realização de pesquisas com técnicas de PLN aplicadas a textos jurídicos do país de interesse, considerando-se o respectivo idioma.

Há diversos estudos envolvendo técnicas de PLN aplicadas ao meio jurídico. Em relação ao tipo de documentos jurídicos estudados, a maior parte dos estudos encontrados não foi muito específica em sua descrição, utilizando-se de termos genéricos como *legal documents*, *legal cases*. Entretanto, alguns deles foram mais específicos descrevendo a classificação de documentos como estatutos, regulamentos ou decisões administrativas [23], Leis sobre Pensão [31], políticas de privacidade [12], sentenças [36] ou Leis Financeiras[30].

Diferentemente do movimento que já vem ocorrendo em muitos países desenvolvidos [13, 22], no Brasil o crescimento desta área ainda é muito recente, havendo poucas publicações nesse sentido. O Poder Judiciário Brasileiro enfrenta um cenário desafiador: um crescente aumento no ingresso de processos; uma limitada disponibilidade de servidores a ser mantida ou reduzida; culminando numa baixa vazão dos processos e, portanto, lentidão no ciclo de atendimento às demandas jurídicas da população. Diferentes melhorias veem sendo propostas e realizadas a fim de tornar o Sistema mais eficiente, uma delas é a ampla implantação do processo eletrônico em substituição aos processos físicos, que dentre outros benefícios facilita o acesso às informações processuais. Muitas das tarefas realizadas ainda são manuais, o que impacta negativamente na celeridade do processo e gestão otimizada dos recursos disponíveis. Uma das atividades iniciais é a classificação dos processos no momento da petição inicial em primeira instância. Além do tempo necessário para execução do preenchimento do sistema, muitas vezes essa etapa acaba

gerando retrabalho devido à classificação indevida. Nesses casos é necessário seu encaminhamento à área especializada que fará a reclassificação do processo, que por sua vez será redistribuído para o órgão competente de fato e só então o processo voltar a seguir o trâmite normal (Figura 3.1).

Motivado pelo problema da classificação automática de textos jurídicos para aplicação a processos eletrônicos de um Tribunal Estadual Brasileiro, espera-se entender quais os fatores mais relevantes a serem considerados no processo de decisões intermediárias que fazem parte da etapa de escolha e treinamento do algoritmo de classificação e como elas interferem no resultado desse modelo. Além disso, pretende-se verificar a viabilidade da utilização da técnica de paralelismo *parameter sweep* no intuito de otimizar a escolha dos parâmetros durante a fase de treinamento. Em relação às técnicas de ML para classificação de textos, serão testadas inicialmente algumas das principais abordagens, de acordo com a literatura, prosseguindo-se à otimização da respectiva parametrização para aquela, ou aquelas técnicas, que apresentarem-se com melhor desempenho inicial para o problema prático proposto. Portanto, ao final deste estudo almeja-se propor uma metodologia para otimizar a escolha de parâmetros do algoritmo de classificação de documentos jurídicos paralelizando o treinamento da(s) técnica(s) de ML que apresentarem melhor desempenho, em termos de F1, para o problema prático proposto.

2. REFERENCIAL TEÓRICO E ESTADO DA ARTE

Nesta seção serão descritos os principais conceitos relacionados ao tema desta pesquisa, além do Estado da Arte para classificação de textos. Inicialmente será apresentada uma visão geral quanto ao processo de modelagem para classificação de documentos a partir de dados não estruturados. Na sequência serão apresentadas em mais detalhes as possíveis técnicas de ML para a classificação de textos. Será apresentado o cenário atual em relação à aplicação dessas técnicas para classificação automática de documentos no contexto jurídico. Adicionalmente, será discorrido a respeito das características e desafios existentes nos ambientes comumente utilizados para processamento desses algoritmos.

2.1 Técnicas de ML para Classificação de Textos

2.1.1 Processo para Classificação de Textos

A classificação de textos é composta por diferentes etapas dispostas de forma encadeada. Dado a complexidade de elementos envolvidos, esse tema vem sendo estudado sob diferentes óticas. Conforme verificado por Mironcz et al.[27], apesar de não haver consenso quanto às nomenclaturas e à quantidade de passos desse processo, o Estado da Arte atual apresenta um fluxo padrão que pode ser representado por seis etapas principais: (1) coleta de dados, (2) análise de dados para anotação de classes, (3) construção e ponderação de *features*, (4) seleção e projeção de *features*, (5) treinamento de um modelo de classificação, e (6) avaliação da solução (Figura 2.1).

Na etapa inicial, a partir de uma ou múltiplas fontes, é realizada a coleta de dados. Na sequência, conforme a metodologia adotada e o volume de dados, as instâncias desse *dataset* bruto são totalmente ou parcialmente rotuladas com a respectiva classe, passando-se a ter um *dataset* rotulado.

O desempenho do modelo de classificação aplicado a dados reais está diretamente relacionado à quantidade e à qualidade que os exemplos do *dataset* rotulado possuem em refletir o comportamento da população alvo [17]. Na etapa seguinte são extraídas *features*. Esse é um dos elementos mais importantes do processo, pois mesmo que se tenha um *dataset* com ótima quantidade e qualidade de exemplos, é necessário que o método de representação do texto em forma estruturada seja capaz de refletir a informação não-estruturada original sem perder os respectivos elementos essenciais. Devido à importância e à complexidade envolvida, muitos esforços têm sido dedicados ao estudo desses métodos para representação de textos e extração de *features* [2, 42].

Dentre as técnicas clássicas pode-se citar a *Bag of Words*. Apesar dessa abordagem ser capaz de representar a sintática das palavras, ela não expressa o significado delas. Nesse contexto surgem muitos estudos envolvendo abordagens de *word embeddings* (WE) a fim de representar-se também a informação semântica das palavras. As WE são técnicas de aprendizado de *features* a fim de mapear palavras ou um conjunto de palavras de um vocabulário de dimensão N para um vetor de números reais com dimensionalidade m, sendo m « N. Dentre os métodos de WE que tem sido bastante utilizados, pode-se citar o *Word2Vec*, *GloVe* e *FastText* [20, 42].

Ao final da terceira etapa, geralmente os dados originais estarão representados por uma grande quantidade de *features*, formando assim uma matriz esparsa de grande dimensionalidade. Sabe-se que essas duas características são indesejadas para a execução da maioria dos algoritmos de classificação, pois além de aumentar o tempo de processamento, reduz o desempenho qualitativo, uma vez que dificulta a identificação de padrões de comportamento entre os casos. Em função disso, recomenda-se o emprego de técnicas para seleção e projeção de *features* no intuito de reduzir a dimensionalidade dessa matriz. Assim, o que no início era apenas um conjunto de textos e documentos, nesta fase estará representado por uma matriz de valores numéricos com dimensionalidade reduzida [45, 33, 1].

O processo descrito até aqui viabiliza a classificação de textos a partir de um arsenal de algoritmos de ML para classificação já utilizado no contexto de dados estruturados. Logo, a matriz gerada ao final da etapa anterior servirá como entrada para o treinamento de uma função de classificação, onde a classe anotada será a classe alvo. O resultado esperado é que o modelo treinado seja capaz de identificar a classe de novos textos de forma automática, acurada e precisa. Em síntese, o modelo final será uma fórmula cujo valor de saída será um vetor de n-probabilidades de um texto pertencer a cada uma das n-classes, a partir da entrada de um vetor de k-*features* extraídas do texto de entrada.

O último passo é a avaliação da solução final. Apesar de haver uma estrutura definida para a análise do resultado, seja ela: seleção de indicadores, definição do método de estimação, aplicação do método e comparação dos resultados [27], os valores absolutos dos resultados do desempenho ainda são muito dependentes do problema de classificação que está sendo testado. Assim, a possibilidade de generalização dos resultados e comparação entre diferentes estudos é bastante limitada [20].

As etapas do processo apresentado acima foram descritas apenas superficialmente, pois cada uma ainda possuiria uma gama de características próprias. Devido à complexidade do processo de classificação de textos, a maioria dos estudos da área explora uma dessas etapas, inclusive em casos de estudos de revisão [27]. Nesse mesmo sentido, o escopo desta pesquisa foi limitado à variação de dois elementos da quinta etapa: a escolha e o treinamento do algoritmo de ML para classificação de textos. Quanto às demais etapas, será adotada uma abordagem padrão alinhada ao respectivo Estado da Arte

a fim de garantir o controle das demais fontes de variabilidade que poderiam vir a intervir nos resultados.

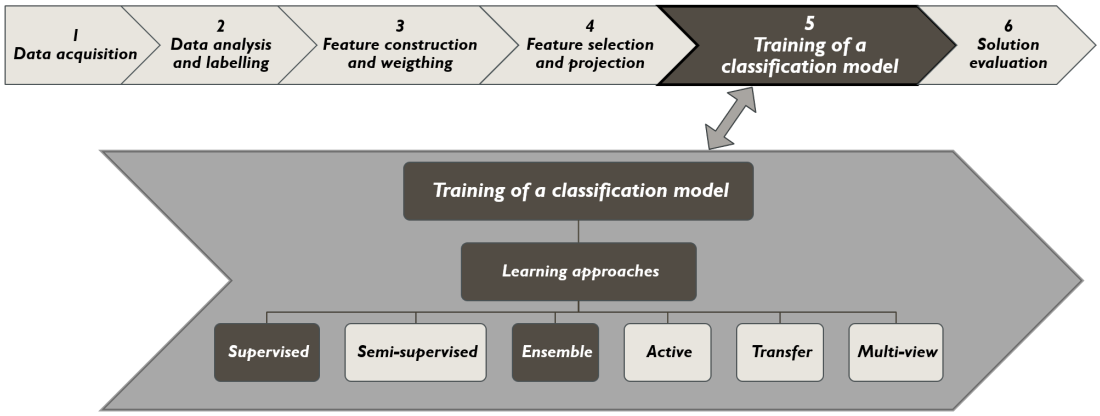

Figura 2.1 – Principais etapas do processo para classificação de textos, destacando etapa na qual o escopo deste estudo está inserido. Ilustração adaptada a partir do modelo proposto por Mironcz et al. 2018[27]

2.1.2 Escolha e treinamento do algoritmo de classificação

A escolha do algoritmo de classificação não é uma tarefa trivial e corresponde a uma das definições mais importantes do processo de classificação. Dentre os fatores a serem considerados, pode-se destacar: o tamanho da amostra rotulada; as *features* disponíveis; a quantidade de classes distintas e o balanceamento entre classes. Adicionalmente, pesquisas anteriores demonstram que não há uma configuração ótima que sirva para todos os casos, ela dependerá de um conjunto de elementos do problema a ser solucionado[17].

Nos casos em que as classes são conhecidas previamente, recomenda-se a utilização de técnicas de aprendizagem supervisionada. Diversos algoritmos de ML têm superado desempenhos prévios para classificação de textos, o que em grande parte se deve à capacidade desses algoritmos em captar padrões de relacionamento não-paramétricos. Em contraponto, a escolha da abordagem, do algoritmo de classificação e de seus hiperparâmetros a fim de maximizar o resultado torna-se uma tarefa desafiadora tendo em vista a existência de muitos graus de liberdade [20]. Há abordagens clássicas que ainda são bastante utilizadas, tais como *Naive Bayes*, árvores de decisão e *Artificial Neural Networks* (ANN) [3, 28, 44, 20]. Já dentre as técnicas de ML mais complexas, há estudos que propõem a utilização de *Recurrent Neural Networks* (RNN), *Auto-encoder Neural Networks* (AENN), *Convolution Neural Networks* (CNN)[3, 20].

Outra alternativa que tem apresentado excelentes resultados para problemas de classificação de textos são os Métodos Ensemble (ME). Esse tipo de abordagem chega a um resultado a partir da combinação de resultados individuais de diferentes classificadores

que formam uma espécie de comitê. Dentre as possíveis técnicas para se combinar os resultados individuais, pode-se citar: *bagging*, *boosting*, *Ad-aBoost, stacked generalisation* e *mixtures of experts* [3].

As técnicas de *Deep Learning*(DL) têm atingido o estado da arte em diversas áreas que utilizam dados não estruturados, como: análise de imagens, reconhecimento de fala. O mesmo tem sido observado para a compreensão de textos [48]. Apesar disso, sabe-se que a qualidade do resultado de muitas dessas técnicas, especialmente daquelas de aprendizagem, depende da disponibilidade de uma grande quantidade de exemplos para treino a fim de que possam ser identificados padrões de comportamento associados às classes. Além disso, paradoxalmente, ao mesmo tempo que para atingir resultados ótimos é necessário um grande volume de dados, quando essa não é a principal limitação, o gargalo ocorre justamente devido às dificuldades em lidar com um grande volume de dados.

2.1.3 Naive Bayes

Dentre os métodos clássicos e probabilísticos, temos o Classificador de Naive Bayes. Essa técnica é baseada no Teorema de Bayes, partindo da premissa que os atributos de um dado exemplo x são independentes entre si dado uma determinada classe y_i. Dessa forma, a probabilidade condicional de um exemplo x pertencer à classe y_i pode ser calculada a partir do produtório das probabilidades condicionais dos atributos dado a mesma classe. Assim, o classificador de Naive Bayes pode ser descrito pela equação 2.1[41]. Apesar da sua simplicidade perante a outras técnicas, dependendo do problema, esse algoritmo pode já gerar uma solução satisfatória do ponto de vista qualitativo a um baixo custo computacional.

$$P(y_i|x) \propto P(y_i) \prod_{i=1}^{d} P(x^j|y_i) \tag{2.1}$$

2.1.4 Random Forest

Random Forest é uma técnica Ensemble que combina várias árvores de decisão individuais, com indução do topo até as folhas. Em cada nó, a árvore de decisão amostra aleatoriamente N dentre os atributos e escolhe aquele que otimiza a divisão de instâncias segundo um critério pré-definido. Assim, tanto a variável quanto o respectivo valor que otimiza a divisão em um determinado nó serão os valores de retorno daquele nó[37]. Dessa forma, ao final teremos várias árvores de decisão diferentes, cujo resultado conjunto será dado por voto majoritário[5]. Existem diferentes formas de se parametrizar essa técnica,

sendo também possível amostrar as instâncias, trabalhando-se com subamostras em cada nó ao invés da amostra completa.

2.1.5 Redes Neurais Recorrentes de Memória de Longo Prazo Bidirecionais

No grupo das técnicas de DL que tem atingido o estado da arte em diversas áreas de pesquisa, dentre elas a classificação de textos, há as Redes Neurais Recorrentes que são redes especializadas em processamento de dados sequenciais. Devido a sua natureza sequencial, elas são capazes de manter por mais tempo informações obtidas em passos anteriores, propagando-as, garantindo assim uma espécie de memória de longo prazo. Essa propriedade da rede torna-se de extrema relevância a fim de facilitar a captura de contexto das palavras. Esse, por sua vez, é essencial para que a semântica dos textos seja interpretada adequadamente. Entretanto, a memória em Redes Neurais Recorrentes *Vanilla* vai sendo neutralizada à medida em que avançamos os seus passos, caracterizando uma memória de curto prazo. Já redes LSTM [16] são uma variação de Redes Neurais Recorrentes onde é possível garantir uma memória de longo prazo ao modelo, contornando a limitação apresentada anteriormente.

Uma variação de redes LSTM que tem chamado a atenção para classificação de textos são as redes LSTM Bidirecionais (Bi-LSTM), em que a entrada de informações sequenciais é percorrida simultaneamente em ambas as direções:*backward* e *forward*. Esse comportamento permite que a rede capte mais informações do contexto em que cada *token* está inserido, portanto geralmente apresenta melhores resultados do que uma simples camada unidirecional LSTM. Essa representação mais rica advém da combinação dos estados das duas camadas LSTM,que geralmente são concatenadas [40]. Entretanto, essa técnica só é aplicável em situações que se tem previamente as informações anteriores e posteriores a cada *token*, o que não seria verdade no caso de um modelo a ser treinado para desenvolver uma conversação em que, portanto, só se teria a informação passada, por exemplo. Já no caso da classificação de textos torna-se uma alternativa bastante interessante, já tendo sido explorada em estudos anteriores inclusive para a classificação de textos jurídicos atingindo resultados satisfatórios[4].

2.1.6 Métricas de Avaliação

No contexto de ML, quando se trabalha com um problema de classificação com modelos supervisionados, avalia-se a qualidade do modelo confrontando-se a classificação mais provável segundo o algoritmo final *versus* a classificação real previamente anotada.

Para tanto, há algumas métricas comumente usadas e que foram adotadas neste estudo: Accuracy, Precision, Recall e F1.

Considerando *True Positivies*(TP) como os casos que foram classificados corretamente na classe k;*False Positives*(FP) como os que foram erroneamente classificados na classe k; *False Negatives*(FN) como os casos que não foram classificados na classe k, mas deveriam ter sido; e *True Negatives* como os casos que corretamente não foram classificados na classe k, tem-se [41]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.5}$$

2.2 Classificação Automática de Textos Aplicada a Documentos Jurídicos

As técnicas para classificação automática de textos têm despertado o interesse de pesquisadores dos mais diversos domínios de conhecimento, dentre os quais pode-se destacar as Ciências Jurídicas [20]. O Direito inerentemente depende da análise de um grande volume de informações textuais, motivo pelo qual tem-se explorado o potencial das técnicas de PLN em otimizar a realização de muitas dessas tarefas. Contudo, é importante considerar, além das diferenças de idiomas, que a legislação varia conforme o país, estado ou mesmo município jurisdicional.

Os estudos mais recentes quanto ao uso de técnicas de PLN a textos jurídicos, independentemente da nacionalidade, têm apresentado melhores resultados ao adotar variações de abordagens de DL como RNN, Redes Bi-LSTM, método BERT, ou CNN [50, 8, 22, 13, 15, 35, 30], adotando abordagens supervisionadas, ensemble ou ainda *Multi-Task*. A Tabela 2.1 apresenta um resumo dos resultados de estudos anteriores identificados ao se fazer uma busca por artigos no idioma inglês, nas bases de busca ACM, IEEE e *Science Direct* e *Web of Science*, cujo objetivo fosse a classificação de textos jurídicos. Para consolidar as informações na tabela, foi considerado sempre como representante do estudo aquele modelo que apresentou maior desempenho.[23, 31, 12, 36, 25, 30, 7, 8, 22, 13, 4, 50, 15, 35].

Dentre as particularidades de cada estudo, pode-se citar o tipo de documento jurídico considerado, sendo que a maior parte dos estudos encontrados não foi muito es-

pecífica em sua descrição, utilizando-se de termos genéricos como *legal documents*, *legal cases*. Entretanto, alguns deles foram mais específicos descrevendo os documentos como estatutos, regulamentos ou decisões administrativas [23], Leis sobre Pensão [31], políticas de privacidade [12], sentenças [36] ou Leis Financeiras[30]. Além do tipo de documento, também há diferença no idioma dos textos considerados para classificação e no tamanho da amostra.

Apesar de haver muitas publicações que se propuseram a estudar diferentes aspectos para aplicação de técnicas de PLN para classificação automática de textos, aparentemente ainda não há um volume de publicações tão amplo para o domínio jurídico frente ao potencial de aplicações que poderia contribuir para a área. Adicionalmente, dentre esses estudos, poucos se referem a modelos específicos para a legislação brasileira.

Tabela 2.1 – Resultado final dos algoritmos de classificação com melhor desempenho por estudo dentre os trabalhos anteriores identificados cujo objetivo foi a classificação de textos jurídicos.

Estudo	*Accuracy*	*Precision*	*Recall*	F1
[23]		0,90	0,50	
[31]	0,92	0,90	0,88	0,89
[13]		0,67	0,63	0,65
[35]		0,65	0,76	0,70
[12]				0,91
[36]	0,51			
[25]	0,76			
[30]	0,97			0,95
[7]		0,85	0,89	0,86
[8]	0,94			0,94
[22]	0,91			
[4]		0,92	0,95	0,93
[50]	0,82			0,82
[15]		0,95	0,96	0,96

No estudo de Elnaggar et al. (2018)[13], por exemplo, os pesquisadores da universidade alemã *Technische Universit* avaliaram a utilização de uma abordagem de *Multi-Task Deep Learning* (MTDL) para solucionar tarefas de tradução, *summarization* e classificação de textos para documentos jurídicos. Os autores justificaram a escolha da técnica devido à escassez de documentos jurídicos em alemão devidamente anotados, ao explorar-se um aprendizado por transferência no intuito de contornar essa limitação, viabilizando dessa forma a utilização de um modelo DL Estado da Arte, mesmo com a escassez de dados anotados. No intuito de gerar um modelo genérico para as três tarefas, eles utilizaram o modelo proposto pela Google, MultiModel, o qual adota uma abordagem *sequence-to-sequence* baseada em uma Rede Neural Convolucional para resolução de múltiplas tarefas de PLN, visualização computacional e reconhecimento de fala. Mesmo considerando a es-

pecificidade do tipo de problema e idioma investigados, os resultados do estudo apontaram que a abordagem *multi-task deep learning* pode ser útil para a resolução de problemas de PLN aplicados ao domínio jurídico. Segundo os autores, tendo apresentado desempenho do modelo final que superaram os resultados Estado da Arte para aquele momento. No caso do modelo de classificação o resultado final apresentou valores de F1, *Recall* e *Precision* correspondentes a respectivamente: 0,65; 0,63 e 0,67.

Em relação a outros países, foram identificados poucos estudos relacionados à aplicação de técnicas de PLN a documentos jurídicos, em português, especificamente para a legislação brasileira. A título de exemplo, Fernandes et al. [15] possuem publicação recente que aborda a aplicação de técnicas de PLN para extração de informação de documentos contendo decisões do Supremo Tribunal Federal (STF). O objetivo do estudo foi, dado uma determinada apelação encaminhada a instâncias superiores, identificar se havia e, caso positivo, qual seria a decisão tomada pelo STF. Para tanto, foram utilizados dados de jurisprudência quanto a apelações da base de dados do Tribunal de Justiça do Rio de Janeiro. Quanto à metodologia empregada, foram avaliadas abordagens de ML e DL, individualmente e de forma combinada. O melhor resultado correspondeu ao modelo *Bidirectional Long Short-Term Memory network* (Bi-LSTM) em combinação com campos condicionais aleatórios, atingindo dessa forma um F1 score de 94,79%

Dentre os estudos encontrados, Braz et. al. (2018)[4] apresentaram aquele cujo objetivo mais se assemelha ao deste trabalho. Os autores exploraram a aplicação de técnicas de ML para classificação de documentos ingressos na Suprema Côrte Brasileira: Supremo Tribunal Federal (STF). Esse estudo apresentou resultados preliminares, onde foram avaliados 6.814 documentos, previamente anotados manualmente por 4 advogados especialistas, a serem classificados em 6 possíveis classes de documento mutuamente exclusivas: acórdão, despacho, sentença, Recurso Extraordinário, Agravo de Recurso Extraordinário ou outros. Foi proposta uma arquitetura Bidirectional Long Short-Term Memory network (Bi-LSTM) para resolução dessa tarefa, sendo relatado um desempenho do modelo final correspondente a um score F1 de 84% F1 para o dataset utilizado sem pré-processamento. Inicialmente foram extraídos de aproximadamente 1000 tokens dos documentos, aos quais foi aplicada uma técnica de *word embeddings* (WE) reduzindo a dimensão para vetores com 100 dimensões formando a camada de entrada para a *Bi-LSTM*. Em relação ao plano amostral de treinamento, foi adotada uma estratégia de 70%, 20%, 10% respectivamente para amostras de treino, validação e teste.

2.3 Ambientes de Execução

Em termos práticos, no intuito de escolher a abordagem para solucionar um determinado problema de classificação de textos, outro aspecto importante a ser considerado é

a disponibilidade de recursos computacionais. Por exemplo, um dos principais motivadores para serem aplicadas técnicas para redução da dimensionalidade e seleção de variáveis é a redução da complexidade de memória e melhora no tempo computacional para execução das etapas seguintes [20].

Técnicas clássicas mais simples como o classificador Naive Bayes, Algoritmo de Rochio, entre outros, às vezes podem apresentar resultados qualitativos tão bons quanto algoritmos mais complexos, entretanto a um baixo custo computacional e menor demanda de memória. Em especial no caso das técnicas de DL, é necessário um maior volume de dados do que para outras técnicas a fim de realizar o treinamento do modelo. Por sua vez, o grande volume de dados envolvido, somado à complexidade do algoritmo acaba por aumentar consideravelmente a demanda computacional para execução dos algoritmos de DL [39].

Em função do alto custo computacional relacionado às técnicas de ML e DL que tem apresentado melhores desempenhos para problemas de diversas áreas, surge outra área de interesse no intuito de orientar o melhor uso dos recursos disponíveis para execução desses algoritmos. Assim muitos pesquisadores têm se dedicado a estudar arquiteturas de DL para aprimorar a robustez e poder computacional. No caso das CNN, um dos tipos de arquitetura de DL, é comum a utilização de *pooling* a fim de diminuir a complexidade computacional ao reduzir o tamanho da saída entre uma camada e outra na rede. Uma das vantagens das técnicas de DL é a capacidade de processamento em paralelo, execução de mais de uma tarefa ao mesmo tempo[20, 13].

Além das melhorias desenvolvidas para paralelizar os algoritmos de DL em si, outra abordagem que se torna interessante para emprego dessas técnicas seria o conceito de paralelismo para *parameter sweep*(PS) no intuito de buscar a melhor configuração de parâmetros em menor tempo. Isso se torna relevante a partir do momento em que não há uma configuração única de parâmetros dos modelos de DL que sempre atinja os melhores resultados, isso dependerá de cada caso. Conforme reportado por Chirigati et. al. [10], uma importante característica de *workflows* de estudos científicos é o intenso processamento de dados, sendo o PS um ótimo exemplo, avaliando de forma iterativa o comportamento de um determinado modelo frente à entrada de diferentes valores de parâmetros. Diversas áreas científicas tem se beneficiado da simulação de diferentes cenários de parâmetros em busca do melhor resultado em menor tempo, via PS. Algumas delas são: bioinformática ([46]), exploração de petróleo ([32]), estudo de Abalos Sísmicos([6]), saúde pública e epidemiologia ([43, 11]), sistemas para pesquisa de informação ([9]), entre outras. O que todos esses estudos possuem em comum é a inexistência de uma resposta única para a melhor combinação de parâmetros de entrada e a independência entre os diferentes modelos a serem testados, o que por sua vez possibilita o processamento em paralelo das diversas soluções via PS. Complementarmente, há esforços no sentido de tentar facilitar a execução desses estudos de simulação para um grande espaço de parâmetros ([34]).

Em relação ao PS, uma das formas que teríamos para classificar esses *workflows* seria de acordo com a forma com que as atividades consomem e produzem dados, sendo 4 os tipos básicos: *Map*, *Split Map*, *Reduce* e *Join*(Figura2.2). Atividades que executam a função *Map* seriam aquelas que produzem uma única saída para cada entrada de dados, ou seja, seria um modelo de razão *input/output* 1:1. Já no caso das atividades *Split Map*, seriam quando são produzidas umas série de saídas para cada entrada de dados, ou seja, seria um modelo de razão *input/output* 1:n. Por sua vez, atividades que executam a função *Reduce* seriam aquelas que a partir de várias entradas produzem uma única saída, ou seja, seria um modelo de razão *input/output* n:1. Por último, as atividades com função *Join* seriam aquelas em que há o consumo de uma série de dados que podem vir de um número fixo de m atividades diferentes para produzir uma única série de saída de dados, ou seja, seria um modelo de razão *input/output* n:n.

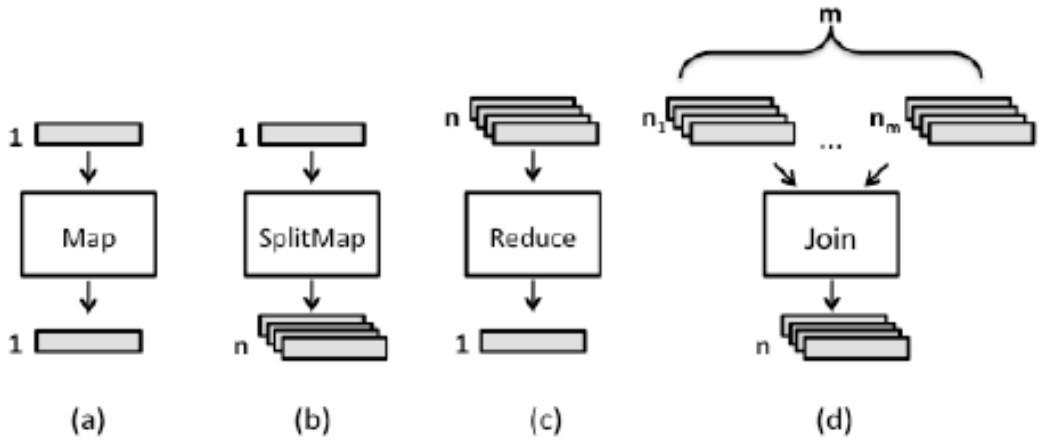

Figura 2.2 – Atividades básicas utilizadas em fluxos de trabalho *parameter sweep* (PS), conforme reportado por Chirigati et. al. [10].

Adicionalmente, há diferentes padrões de combinações dessas atividades básicas apresentadas anteriormente para formar os *workflows* PS. Segundo Chirigati et. al. [10], é possível observar 5 padrões: *Sequence*, *Thread Split*, *Thread Merge*, *Parallel Split* and *Synchronization*(Figura2.3). O primeiro padrão, *Sequence* (Figura2.3a) se refere à sequência de atividades *Map*. Já o *Thread Split* (Figura2.3b) corresponde à combinação sequencial de atividades *Split Map* e *Map*, em que a primeira etapa gera uma série de dados, sendo que cada um deles por sua vez será consumido por uma atividade *Map* (1:1). Já o padrão *Thread Merge* corresponde à sequência de atividades *Map* e uma *Reduce*(Figura2.3c). Em que a primeira etapa um conjunto de atividades *Map* gera n saídas que por sua vez serão consumidas por uma etapa *Reduce* que produzirá uma única saída. Outro padrão encontrado é o *Parallel Split*(Figura2.3d), em que há um *broadcast* do resultado de um *Map* específico para n núcleos *Map*, que recebem cópias do primeiro resultado, cada um gerando um particular resultado de forma concorrente. Por último, há o padrão *Synchronization* (Figura2.3e), em que na primeira etapa n ativiades *Map* geram n resultados, que por

sua vez são consumidos conjuntamente numa etapa *Join*, portanto gerando n resultados a partir de uma etapa que consolida os resultados das atividades prévias. Nesse último padrão, é importante que os dados para consumo estejam todos prontos para que a etapa *Join* inicie.

Em relação às métricas usualmente utilizadas para quantificar os benefícios da técnica PS versus processamento sequencial, a maioria está baseada em alguma espécie de análise do tempo de processamento em comparação ao número de núcleos físicos utilizados. Chirigati et. al. [10] propõe algumas métricas que também serão adotadas neste estudo.

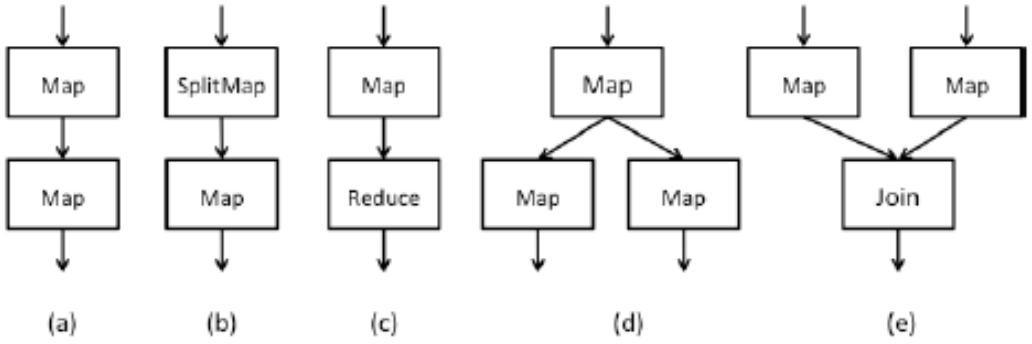

Figura 2.3 – Padrões de combinações das atividades básicas em fluxo de trabalho *parameter sweep* (PS), conforme reportado por Chirigati et. al. [10].

- Tempo de Execução Paralela (T_E): tempo total de execução do *workflow* (em minutos);
- Tempo de Execução Sequencial (T_1): tempo total de execução dos processos de maneira sequencial (em minutos);
- Speedup (S): valor que relaciona o equivalente do tempo de execução sequencial sobre o de execução em paralelo;
- Eficiência (E): valor entre 0 e 1 que representa o quão eficientemente os núcleos disponíveis são utilizados para a execução do *workflow* completo;
- Núcleos (*cores*) (c): total de núcleos físicos utilizados no *workflow* PS;

Em que:

$$S = \frac{T_1}{T_E} \tag{2.6}$$

$$E = \frac{S}{c} \tag{2.7}$$

3. CLASSIFICAÇÃO DE TEXTOS JURÍDICOS

3.1 Definição do Problema

Diferentemente do movimento que já vem ocorrendo em muitos países desenvolvidos [13, 22], no Brasil o avanço de pesquisas sobre aplicação de modelos de ML e PLN a documentos jurídicos ainda é muito recente, havendo poucas publicações. O Poder Judiciário Brasileiro enfrenta um cenário desafiador: um crescente aumento no ingresso de processos, uma limitada disponibilidade de servidores a ser mantida ou reduzida, culminando numa baixa vazão dos processos e, portanto, lentidão no ciclo de atendimento às demandas jurídicas da população. Percebe-se que o modelo de tramitação atual não é sustentável. Diferentes melhorias vêm sendo propostas a fim de tornar o Judiciário Brasileiro mais eficiente, uma delas é a ampla implantação do processo eletrônico em substituição aos processos físicos que, dentre outros benefícios, facilita o acesso às informações processuais.

O fluxo processual é bastante complexo, ainda possuindo muitas etapas manuais. Um exemplo é a determinação da classe processual que está associada à forma que o processo irá tramitar de acordo com a petição inicial. A classe determinará qual o procedimento judicial ou administrativo adequado ao pedido. Até 2007 não havia padronização das classes processuais, dessa forma havendo diferentes nomenclaturas conforme a unidade da federação, instância (1º Grau, 2º Grau) ou esfera (Estadual, Federal). Essas diferenças implicavam em diversos problemas: dificuldade em apurar estatísticas padronizadas para auxiliar no planejamento estratégico do Judiciário Brasileiro; um mesmo processo ao migrar de instância ser classificado de forma diferente, entre outros. A fim de sanar esses e outros problemas, foi criado o Conselho Nacional de Justiça (CNJ), órgão responsável por orientar, padronizar e fiscalizar os tribunais estaduais, federais, eleitorais e militares. Uma importante contribuição desse órgão foi a criação das TPUs (Tabelas Processuais Unificadas), criadas pela Resolução CNJ n. 46, de 18 de dezembro de 2007[26], a fim de padronizar as classes, os assuntos e os movimentos processuais.

Ocorre que, para que os benefícios esperados com essa mudança sejam efetivados, é necessária a correta utilização das TPUs para classificar os processos, o que acaba sendo uma tarefa manual e complexa, dependendo muito também da interpretação de quem classificará. Além da dificuldade operacional existente para a classificação do processo, caso ele seja classificado de forma equivocada, será distribuído para o local errado que, por sua vez, o retornará a uma central para reclassificação desse processo. Após, o processo será redistribuído para o órgão competente, seguindo a tramitação normal. A

ilustração 3.1 demonstra de forma simplificada esse fluxo do processo eletrônico, desde a sua abertura pelo advogado até sua correta distribuição.

Assim, caso fosse possível classificar esses processos de forma automática, essa informação não precisaria ser preenchida pelo advogado, o que melhoraria a experiência dele ao ingressar com o processo, viabilizando uma classificação de forma mais rápida e objetiva. Além disso, reduziria o retrabalho e possível retardo do andamento processual devido a erro na classificação.

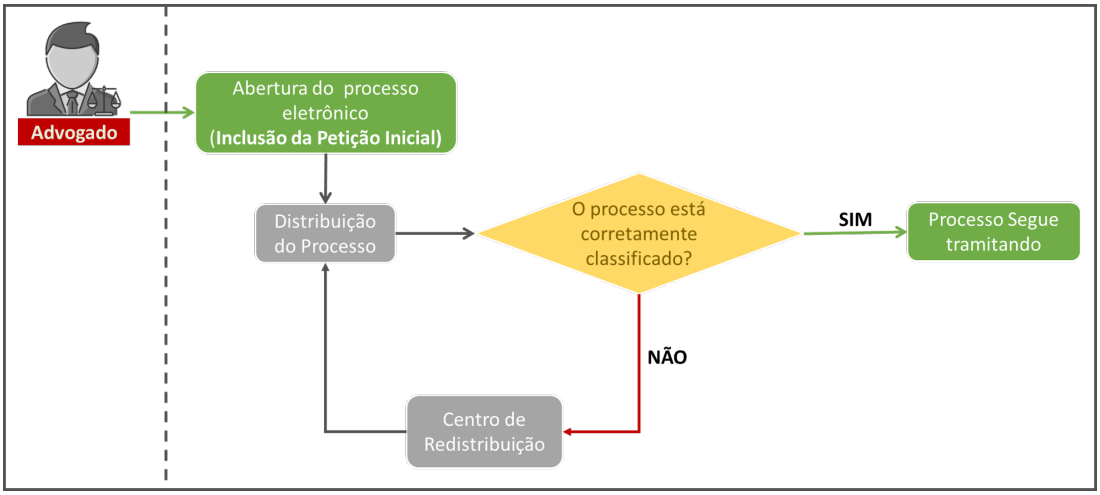

Figura 3.1 – Representação simplificada do fluxo atual para abertura de um processo eletrônico em 1ª instância. Destaque para a etapa de verificação da respectiva classe CNJ fornecida pelo advogado junto à petição inicial, e seus possíveis encaminhamentos.

Motivado por esse cenário, o presente estudo se propôs a investigar a classificação automática de documentos de petições de processos eletrônicos cíveis de um Tribunal Estadual. Foram utilizados 107.010 documentos, já previamente anotados com a respectiva classe CNJ. A anotação prévia considerada para este estudo foi extraída diretamente do sistema judicial, cujos registros da classe foram inseridos manualmente pelos técnicos judiciários das comarcas. Os documentos passaram por uma etapa de saneamento inicial e posteriormente foram distribuídos, de forma aleatória, para a amostra de treino (70%), amostra de validação (20%) ou amostra de teste (10%). A partir do mesmo *dataset* de treino, inicialmente foram processados diferentes algoritmos de classificação para documentos (Naive Bayes, Random Forest, Bi-Directional LSTM). Na sequência, foram aferidas estatísticas qualitativas a respeito da classificação final de cada modelo: F1, Recall, Accuracy e Precision. A partir desses resultados preliminares, optou-se por seguir investigando a parametrização exclusivamente das Redes Bi-LSTM.

3.2 Amostragem

Os dados utilizados neste trabalho compreenderam inicialmente 107.010 petições do Tribunal de Justiça do RS (TJRS) que foram compartilhadas para fins de pesquisa sem a possibilidade de publicação de qualquer nome ou outro dado sensível, além de terem sido excluídos processos sigilosos. Adicionalmente, foram coletadas as respectivas classes CNJ iniciais dos processos correspondentes. A Classe CNJ foi utilizada como variável alvo para fins de treinamento supervisionado e posterior aferição das métricas de desempenho.

Critérios de inclusão: foram consideradas apenas petições vinculadas a processos eletrônicos de 1º Grau e Juizados Especiais tramitados pelo sistema eThemis de primeiro grau, com data de propositura compreendida de janeiro de 2014 a dezembro de 2019, que possuíssem a respectiva petição digitalizada. Nesse universo avaliado, foi observado alto desbalanceamento entre classes, havendo grande concentração em uma única classe CNJ, detendo 70,3% das petições encontradas. Ainda, 16,67% das classes observadas detinham 94,89% do total de petições nesse universo.

Na sequência foi realizada a recuperação desses arquivos que estavam armazenados em outra estrutura acessível via serviço, descompactação dos mesmos para o formato original em pdf, na sequência sendo extraído o texto desses arquivos utilizando o *framework* PyPDF2, python 3.7.6. Parte dos arquivos estavam digitalizados como imagem e, nesses casos, foram desconsiderados para o presente estudo. Além disso, em outros casos não foi possível extrair o conteúdo textual de forma satisfatória, portanto também foram descartados. Ao final dessa etapa, restaram 59.032 documentos com o respectivo texto extraído de forma satisfatória para o desenvolvimento do estudo, previamente classificados em uma dentre 10 classes CNJ conforme a distribuição final da amostra ilustrada na tabela 3.1.

Tabela 3.1 – Análise descritiva da amostra final.

Classe CNJ	**Descrição**	**Freq. Abs.**	**Freq. Rel.(%)**
436	Procedimento do Juizado Especial Cível	47.856	81,07
159	Execução de Título Extrajudicial	5.696	9,65
261	Carta Precatória Cível	2.602	4,41
7	Procedimento Comum Cível	1.862	3,15
156	Cumprimento de sentença	310	0,53
120	Mandado de Segurança Cível	277	0,47
1111	Execução de Título Judicial	166	0,28
1116	Execução Fiscal	141	0,24
12119	Inc. de Desconsideração de Personalidade Jurídica	119	0,20
37	Embargos de Terceiro Cível	3	0,01
-	Total	59.032	100,00

3.2.1 Pré-Processamento

Após a etapa inicial, foi realizado um pré-processamento dos textos, contendo as seguintes etapas:

1. *Tokenização*: O texto foi dividido em tokens;
2. Exclusão de termos desnecessários e capitalização: todas palavras foram padronizadas para minúsculo, foram excluídas todas as pontuações e *stopwords*.

A Figura 3.2 apresenta um exemplo de petição que foi utilizada no presente estudo como texto de entrada a ser classificado. Complementarmente, a Figura 3.3 ilustra esse mesmo texto após a execução da etapa de pré-processamento. Esse mesmo tratamento foi aplicado a todas as 107.010 petições.

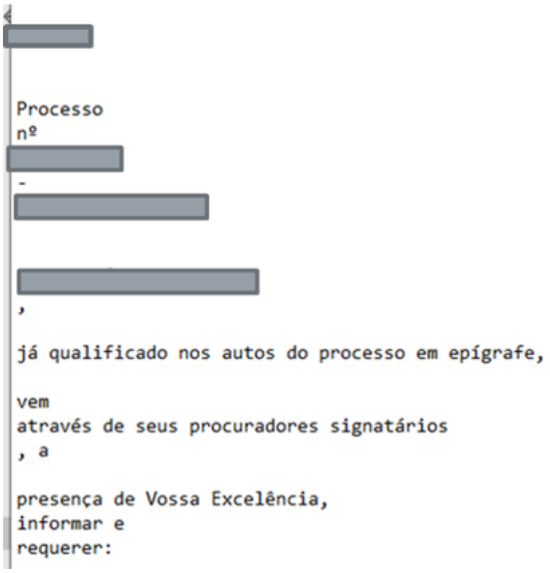

Figura 3.2 – Exemplo de petição que foi utilizada como texto de entrada a ser classificado.

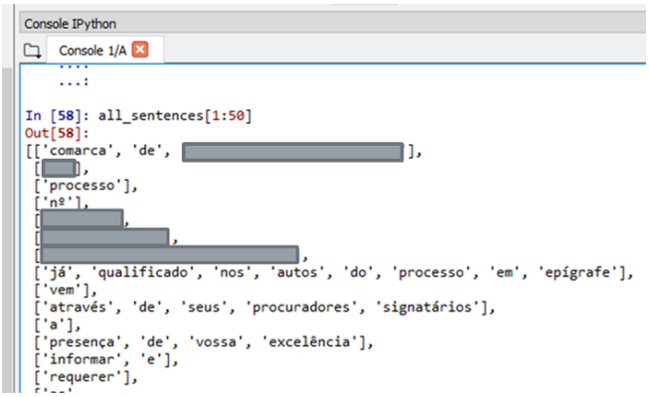

Figura 3.3 – Exemplo de tokens extraídos do texto ilustrado na Figura 3.2.

3.3 Testes Preliminares

3.3.1 Variação dos Algoritmos de Classificação

Inicialmente foram testados sequencialmente três algoritmos de classificação a fim de verificar se faria sentido continuar investigando-os com diferentes parametrizações no modelo proposto de *parallel parameter sweep* e posterior método Ensemble, considerando o tempo de processamento necessário assim como os resultados em termos qualitativos pelas métricas de desempenho F1, accuracy, recall e precision. Nesta etapa, escolheu-se representantes de três grupos diferentes de modelos de classificação, sejam eles: probabilístico clássico - Naive Bayes; árvores de decisão - Random Forest; Estado da Arte Deep Learning - Bi-LSTM.

3.3.2 Treinamento

O treinamento foi desenvolvido em duas fases, sendo a primeira comum a todos algoritmos de classificação e a segunda compreendendo o treinamento em si, fase em que foram testados diferentes cenários, variando-se o algoritmo de classificação e as respectivas parametrizações, além do tamanho de amostra.

Após o pré-processamento, a amostra foi dividida aleatoriamente em três grupos: treino (70%), análise (20%) e teste (10%). Todas as etapas de treinamento foram conduzidas com a amostra de treino e análise e, posteriormente, testadas na amostra de teste. A etapa de treino para representação dos textos em formato de vetores numéricos densos foi realizada utilizando-se o *framework* gensim, Word2Vec, algoritmo skip-gram. Para tanto, foi utilizada a seguinte parametrização: dimensionalidade = 200, contagem mínima de palavras=20, *workers* =4, janela de 10 palavras para contexto, *negative*=20, épocas=30.

Logo após, foram testados sequencialmente os três métodos de classificação com três tamanhos de amostra para fins de aferição do tempo de processamento por meio de projeção do tempo que seria necessário para a amostra completa. Os tamanhos de amostra utilizados foram: 1.000, 5.000 e 10.000 textos, que, ao final, projetando-se o tempo necessário para um total de 59.032. Cada modelo foi testado variando-se os respectivos parâmetros e avaliando-se o consequente impacto na qualidade do modelo final, assim como no tempo de processamento.

Ao se ajustar um modelo Ensemble, um dos critérios importantes a ser considerado é a diversidade entre os modelos individuais. Portanto, estando pouco correlacionados entre si e podendo assim serem capazes de captar aspectos diferentes na amostra,

somando esforços para uma classificação final mais acurada[3, 47]. Considerando que, na sequência deste trabalho, pretende-se ajustar um modelo Ensemble, nesta etapa escolheu-se representantes de três grupos diferentes de modelos de classificação, são eles: probabilístico clássico - Naive Bayes; árvores de decisão - Random Forest; Deep Learning - Bi-LSTM.

3.3.3 Ambiente de Execução

A execução do experimento inicial foi realizada em um notebook ASUS, INTEL CORE i7-6500U CPU 2.5Hz, 8GB RAM, NVIDIA GEFORCE 940M. Frameworks TensorFlow, Keras, Scikit-learn e Gensim, python 3.7.4.

3.3.4 Parametrização dos Algoritmos de Classificação

Dentre os métodos clássicos e probabilísticos temos o Classificador de Naive Bayes. Esse é o mais simples dentre os modelos testados; entretanto, dependendo do problema pode já gerar uma solução satisfatória do ponto de vista qualitativo a um baixo custo computacional. Esse modelo foi utilizado como *baseline* para os demais. O parâmetro variado neste algoritmo foi apenas o tamanho da amostra (1.000,3.000,5.000,10.000).

Random Forest é uma técnica Ensemble que combina várias árvores de decisão individuais com indução do topo até as folhas. Foi utilizado o algoritmo Random Forest, do framework Scikit-learn, python. Diferente do modelo proposto pelo estudo original[5], nessa implementação, ao final não é retornada a classificação por voto majoritário, mas sim a média das probabilidades estimadas individualmente. Esse algoritmo foi testado usando-se como critério de otimização a medida de entropia, também para amostras de 1000, 5000 e 10.000 petições. Os parâmetros variados foram: *n_estimators*(20% ou 30% da amostra de treino) e *class_weight*("balanced"ou "None"). O primeiro refere-se ao total de árvores individuais a serem geradas, utilizando-se amostragem aleatória simples com reposição. O segundo parâmetro refere-se à forma como é feita essa amostragem, ou se obedecendo à distribuição dos dados em relação à classe, ou se realizando de forma balanceada entre as classes garantindo um número de representantes igual para cada uma. Os demais parâmetros foram mantidos no seu padrão, como, por exemplo, utilizando-se subamostras das instâncias da amostra de treino.

Considerando a forte dependência da ordem em que as palavras são concatenadas nos textos amostrados, assim como o contexto em que estão inseridas, optou-se por trabalhar com Redes Neurais Recorrentes de Memória de Longo Prazo, do inglês Long Short-Term Memory (LSTM). Adicionalmente, considerando que se tem acesso ao docu-

mento completo, deu-se preferência à técnica de Redes Neurais Recorrentes de Memória de Longo Prazo Bidirecionais (Bi-LSTM) no intuito de aproveitar ao máximo a informação disponível. Nesta primeira fase foram utilizadas redes LSTM Bidirecionais [38] com uma arquitetura de 4 camadas: *embedding* Word2Vec previamente treinado, garantindo que fosse o mesmo modelo para todas as técnicas de classificação; uma camada Bi-LSTM com dropout 0.15; na sequência, mais duas camadas densas cujas funções de ativação corresponderam a respectivamente Relu e Sigmoid. Para otimização dos parâmetros foi testado o otimizador de Adam *default* do método que segue os parâmetros citados no artigo original[18], função de perda pelo critério da entropia cruzada. Em relação aos demais parâmetros da rede, arbitrariamente testaram-se diferentes números de neurônios nas camadas Bi-LSTM e na camada Densa escondida: 32 e 64. Adicionalmente, variou-se o total de épocas a serem processadas: 10 e 30.

3.3.5 Resultados Preliminares

As Tabelas 3.2 e 3.3 apresentam os resultados encontrados inicialmente. Nesta fase foram realizados testes com amostras reduzidas com o intuito de variar-se os parâmetros de cada modelo, identificando como eles interferem na qualidade do resultado. A Tabela 3.2 apresenta o resultado médio para as métricas de qualidade aferidas em cada um dos classificadores, já a Tabela 3.3 apresenta o tempo médio de processamento para cada um dos tamanhos de amostra assim como a projeção do tempo necessário para treinar-se com a amostra completa, no mesmo ambiente de treinamento. Para projeção, com base no comportamento observado nas amostras menores, realizou-se uma regressão linear simples para estimar o tempo total necessário para um processamento com a amostra completa nos modelos Naive Bayes e Random Forest. Já para o Bi-LSTM foi ajustado uma regressão quadrática de acordo com o comportamento observado.

A partir desses experimentos preliminares, percebe-se a dificuldade em encontrar-se a melhor configuração de forma manual para os modelos mais complexos, como é o caso da Rede Neural Recorrente Bi-LSTM que possui vasta gama de hiperparâmetros a serem especificados. Variou-se o total de épocas, parecendo que, para esse problema em específico, 30 seriam suficientes para atingir melhores resultados, ao aumentar-se o número de épocas a partir desse patamar não se identificou grande incremento nos resultados. Todavia, não se sabe o que poderia ocorrer com a execução na amostra completa, talvez fossem necessárias mais épocas para estabilizar a qualidade do resultado ao utilizarmos um maior volume de dados. Outro parâmetro variado foi o total de neurônios na camada Bi-LSTM e na camada densa escondida, foram testados os valores 32 e 64. O desempenho em termos qualitativos e o tempo total de processamento se mostraram diretamente relacionados ao incremento no total de neurônios das camadas escondidas, assim como no total de épocas.

No modelo Random Forest, obteve-se resultados superiores ao utilizar-se o parâmetro *class_weight* = "balanced". A amostra era bastante desbalanceada, o que pode explicar esse resultado encontrado. Classificar eventos raros é uma dificuldade já discutida na literatura [24], sendo o balanceamento de amostras uma das estratégias a fim de melhorar a qualidade das estimativas, assim os resultados encontrados vão ao encontro desses estudos. Adicionalmente, ao aumentar-se o parâmetro *n_estimators*, total de árvores individuais, também obteve-se uma melhora no resultado. No entanto, não se chegou a um número de árvores em que pareceu estabilizar ou reduzir a qualidade do resultado.

Comparando-se os diferentes modelos de classificação testados, observou-se que em relação à *accuracy* os modelos Bi-LSTM e Naive Bayes apresentaram, em média, o mesmo desempenho e também foram superiores ao *Random Forest*. Contudo, o modelo Naive Bayes atingiu o mesmo patamar de *accuracy* a um custo computacional bem menor, evidenciado pelo tempo total de processamento significativamente mais baixo. Por outro lado, sabe-se que há grande desbalanceamento entre as classes, logo talvez a *accuracy* não seja a métrica mais indicada neste caso, pois pode estar inflacionada pelo acerto da classe predominante. Já a medida F1 acaba sendo mais robusta do que a *accuracy*, penalizando mais o erro nas classes menos frequentes ao realizar uma média ponderada entre *precision* e *recall*. Essa diferença é captada pela métrica F1 superior no modelo Bi-LSTM em relação aos demais, indicando que apresentou um modelo capaz de acertar uniformemente todas as classes a serem preditas, diferente do que ocorreu nos demais modelos.

Baseado nos resultados iniciais, chegou-se à conclusão de que, para o problema e algoritmos de classificação propostos, inicialmente, faria mais sentido seguir explorando as diferentes parametrizações apenas do algoritmo Estado da Arte: Redes Bi-LSTM. Além disso, definiu-se que a amostra seria balanceada para continuidade dos treinamentos. Portanto, na etapa seguinte além de utilizar-se a amostra completa, balanceada, foi explorada uma abordagem de PS para testar em paralelo diferentes combinações de parâmetros para essa rede, objetivando chegar-se a um melhor resultado em menor tempo.

Tabela 3.2 – Comparação dos Algoritmos de Classificação - Desempenho na classificação

Métrica	**Naive Bayes**	**Random Forest**	**Bi-LSTM**
Accuracy	0,822	0,808	0,822
Precision	0,962	0,619	0,824
Recall	0,217	0,268	0,822
F1	0,354	0,374	0,823

Tabela 3.3 – Comparação dos Algoritmos de Classificação - Tempo de Execução (em segundos)

Tamanho da Amostra	**Naive Bayes**	**Random Forest**	**Bi-LSTM**
1.000	0,01	27,77	263,54
5.000	0,02	46,94	340,61
10.000	0,05	74,69	570,53
estimativa para 59.032	0,26	323,03	41.777,49

4. METODOLOGIA UTILIZANDO AMBIENTES PARALELOS PARA OTIMIZAÇÃO DA CLASSIFICAÇÃO DE TEXTOS

A partir dos resultados preliminares, definiu-se que o escopo a ser explorado na etapa seguinte seria limitado à variação de parâmetros das Redes Bi-LSTM. Uma dificuldade importante nessas técnicas de *Deep Learning* é a falta de uma combinação de parâmetros única que atinja o melhor desempenho para todas as situações. Devido ao grande número de possibilidades e o tempo de processamento que seria necessário para testar todas, muitas vezes, o que ocorre é o especialista propor uma combinação de parâmetros que acredita ser a melhor de acordo com sua experiência, sem ter a certeza se poderia atingir melhor desempenho com algumas modificações. Dito isso, a qualidade do modelo final pressupõe grande conhecimento e experiência por parte do especialista, o que em muitos contextos não é tão simples de se ter disponível.

Complementarmente, sabe-se que seguidamente há redundância de estruturas de processamento disponíveis, como máquinas com múltiplos *cores*, que não são aproveitadas em sua plenitude, já que pressupõem que a orientação para sua utilização seja passada via *script*. Conforme apresentado no Capítulo 2, a técnica de *Parameter Sweep* (PS) seria uma abordagem de paralelismo para avaliar de forma iterativa o comportamento de um determinado modelo frente à entrada de diferentes valores de parâmetros. A implementação do PS é uma forma de paralelismo relativamente simples, pois não há dependência e necessidade de controles para sincronização dos processamentos que são executados em paralelo. Assim, acredita-se que a adoção do PS para testar diferentes parametrizações do modelo de classificação, como no caso de Redes Bi-LSTM, seja uma excelente alternativa para otimizar a utilização dos recursos computacionais disponíveis e disponibilizar de maneira mais simples melhores resultados em menor tempo a mais pessoas que queiram realizar a classificação de textos.

Considerando o exposto, neste capítulo será apresentada a metodologia proposta assim como o estudo de caso realizado.

4.1 Apresentação da Metodologia

A partir dos resultados preliminares e da motivação apresentada, definiu-se o escopo a ser testado, utilizando-se a técnica de *Parameter-Sweep*(PS) para avaliar em paralelo diferentes combinações de parâmetros para redes Bi-LSTM com a amostra completa anteriormente apresentada. A Metodologia proposta é um fluxo composto por 4 fases (Figura 4.1):

1. Pré-processamento - Responsável pela preparação dos textos e pela formação do corpus;
2. Treinamento do Modelo de Representação - Etapa em que é realizado o treinamento do modelo de representação do texto em formato numérico, via técnica de Word Embeddings, viabilizando a utilização do algoritmo de classificação na fase seguinte;
3. Treinamento em Paralelo dos Modelos de Classificação - Execução do treinamento dos modelos de classificação via *Parallel Parameter Sweep*, aproveitando a arquitetura computacional disponível, com variação dos parâmetros de entrada;
4. Método Ensemble - combinação entre os melhores resultados individuais no intuito de aprimorar ainda mais o modelo final.

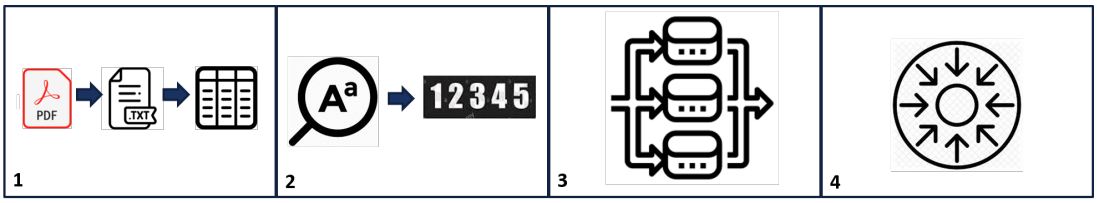

Figura 4.1 – Fluxo geral da Metodologia proposta em 4 fases: (1) Pré-processamento, (2) Treinamento do Modelo de Representação, (3) Treinamento em Paralelo dos Modelos de Classificação, (4) Método Ensemble.

4.1.1 Fase 1: Pré-processamento

Inicialmente, propõe-se que seja realizado um pré-processamento dos textos para formação do corpus, contendo as seguintes etapas:

1. Extração do texto: no caso de os textos originais estarem armazenados no formato de pdf, primeiramente é realizada uma extração do conteúdo textual para o formato txt;
2. Balanceamento da amostra: no caso de a amostra apresentar desbalanceamento acentuado, realizar alguma técnica de balanceamento da amostra. Lidar com a identificação de eventos raros é um problema já discutido na literatura, havendo diferentes abordagens sugeridas para lidar com o desbalanceamento no intuito de melhorar o desempenho na identificação desses grupos [21, 24]. Dentre as possíveis alternativas, pode-se trabalhar com diferentes métodos de amostragem, como exemplo temos *undersampling* e *oversampling*.
3. *Tokenização*: na sequência, o texto extraído é dividido em tokens;
4. Exclusão de termos desnecessários e capitalização: todas palavras são padronizadas para minúsculo e são excluídas todas as pontuações e *stopwords*.

4.1.2 Fase 2: Treinamento do Modelo de Representação

Na etapa seguinte são extraídas *features* do texto. Esse é um dos elementos mais importantes do processo, pois mesmo que se tenha um *dataset* com ótima quantidade e qualidade de exemplos, é necessário que o método de representação do texto em forma estruturada seja capaz de refletir a informação não-estruturada original sem perder os respectivos elementos essenciais. Devido à importância e à complexidade envolvida, muitos esforços têm sido dedicados ao estudo desses métodos para representação de textos e extração de *features* [2, 42]. Conforme apresentado no Capítulo 2, recomenda-se a utilização de abordagens de *word embeddings* (WE) a fim de representar-se também a informação semântica das palavras.

4.1.3 Fase 3: Treinamento em Paralelo dos Modelos de Classificação

Nesta etapa, é escrita uma função que será executada em paralelo, de acordo com a arquitetura computacional disponível, variando-se os parâmetros de entrada. Esta função conterá as seguintes etapas:

1. Carregamento do modelo de representação previamente treinado;

2. Separação das amostras em três grupos: treino (70%), análise (20%) e teste (10%);

3. Treinamento do modelo de classificação com os respectivos parâmetros de entrada nas amostras de treino e de análise;

4. A partir da amostra de teste, coleta das estatísticas de desempenho: F1, *recall, accuracy, precision*, tempo de processamento;

Conforme apresentado no Capítulo 2, a característica de independência dos dados de entrada dos modelos permite que se realize uma abordagem de treinamento via PS, a qual otimiza a utilização dos recursos computacionais disponíveis. Caso não fosse adotada essa abordagem e se realizasse um treinamento sequencial, parte da arquitetura disponível ficaria ociosa. Assim, torna-se uma proposta viável e mais eficiente, facilitando a identificação da melhor combinação de parâmetros, em última análise gerando melhores resultados em menor tempo.

4.1.4 Fase 4: Método Ensemble

Outra abordagem que também pode ser adotada no intuito de buscar melhores resultados é a combinação de diferentes resultados de classificação individuais, sejam eles da mesma técnica ou não. Esses métodos são denominados *Ensemble* e há diferentes formas com que os resultados individuais podem ser combinados. Segundo Faceli *et. al.*, 2015, [14] há diferentes maneiras de classificar essas técnicas, sendo uma delas como Métodos de Votação versus Métodos de Seriação. O primeiro seria a forma mais frequentemente utilizada para combinação de predições, em que a seria formado uma espécie de comitê em que cada classificador individual entraria como um voto. Essa votação seria dada de forma uniforme, ou seja, quando todos os votos possuem mesmo peso na classificação final; ou ainda poderia ser realizada de forma ponderada, onde cada classificador base possuiria um peso associado, o qual pode variar ao longo do tempo, de forma a reforçar o resultado obtido por um bom classificador. Já o segundo pode ser visto como uma melhoria no processo de votação, ao combinar-se as probabilidades de pertencer a cada uma das classes ao invés de puramente as etiquetas atribuídas aos indivíduos, Kittler (1998)[19] explora em seu estudo diferentes funções dentre as mais comuns para combinação dessas probabilidades, dentre elas a Regra da Soma (4.1), regra da média (4.2) e regra do produto (4.3).

Regra da Soma:

$$S_k = \sum_{i=1}^{m} P_{ik} \tag{4.1}$$

Regra da Média:

$$S_k = \sum_{i=1}^{m} \frac{P_{ik}}{m} \tag{4.2}$$

Regra do Produto:

$$S_k = \prod_{i=1}^{m} P_i \tag{4.3}$$

Onde P_{ik} representa a probabilidade dada pelo classificador i de pertencer à classe k. Após a aplicação de qualquer uma dessas funções, a classe final pelo método Ensemble será dada pela classe que maximize S_k. Conforme o estudo de Kittler [19], a junção pela Soma se demonstra mais robusta, conservadora, sendo menos suscetível a erros. Já a junção pelo produto se mostra mais arriscada, mais suscetível a erros, mas podendo apresentar melhores resultados dependendo do caso.

4.2 Estudo de Caso

4.2.1 Implementação

A execução do experimento foi realizada no Laboratório de Alto Desempenho da PUCRS (LAD-PUCRS), na máquina Pantanal, a qual possui a seguinte configuração: Intel Xeon Gold 5118 CPU 2.30GHz, 24(48) cores, Sistema Operacional Linux, Ubuntu 16.04.2 LTS. Em relação à linguagem, a implementação foi conduzida utilizando-se python 3.6 e os *frameworks*: TensorFlow, Keras, Scikit-learn, Multiprocessing e Gensim.

4.2.2 Fase 1: Pré-processamento

A etapa de pré-processamento da amostra utilizada neste trabalho foi descrita com mais detalhes no subitem "Amostragem", do Capítulo 3. Às etapas descritas anteriormente, foi adicionada a etapa 2 para balanceamento da amostra. Uma vez que parte das categorias possuía volume muito limitado, conforme demonstrado na Tabela 3.1, no intuito de não desperdiçar informação, optou-se por trabalhar com a técnica de *oversampling*. Na abordagem escolhida, todos os casos raros são aproveitados e, por sua vez, são novamente amostrados aleatoriamente até atingir-se um equilíbrio de proporção com as outras classes. Apesar dessa estratégia minimizar os problemas advindos do desbalanceamento (como o acerto muito alto apenas para a classe dominante), é possível que seja embutido um viés no padrão identificado para a classe de menor prevalência, já que os mesmos casos são replicados. Principalmente no caso das classes com menor número de representantes.

4.2.3 Fase 2: Treinamento do Modelo de Representação

A etapa de treino para representação dos textos em formato de vetores numéricos densos foi realizada utilizando-se o *framework* gensim, Word2Vec, algoritmo skip-gram. Para tanto, foi utilizada a seguinte parametrização: dimensionalidade = 200, contagem mínima de palavras=20, *workers* =24, janela de 10 palavras para contexto, *negative*=20, épocas=30. Apesar desta etapa não ser o foco deste estudo, fica a sugestão de serem exploradas variações da técnica adotada nessa etapa em estudos futuros, substituindo o modelo Word2Vec por outros métodos de representação propostos como *GloVe*, *FastText* [20, 42], entre outros. Esta etapa foi comum a todos os modelos de classificação treinados,

sendo armazenado o modelo treinado para posterior carregamento e utilização a cada uma das combinações de parâmetros dos modelos de classificação treinados.

4.2.4 Fase 3: Treinamento em Paralelo dos Modelos de Classificação

Após o carregamento do modelo de representação Word2Vec previamente treinado, a cada execução a amostra foi dividida aleatoriamente em três grupos: treino (70%), análise (20%) e teste (10%). Todos os treinamentos dos modelos de classificação foram conduzidos com a respectiva amostra de treino e análise e posteriormente testadas na amostra de teste.

Seguindo o modelo proposto nos testes iniciais, foram utilizadas redes LSTM Bidirecionais [38] com uma arquitetura de 4 camadas: *embedding* Word2Vec previamente treinado, garantindo que fosse o mesmo modelo para todas as parametrizações; uma camada Bi-LSTM variando-se o valor de parâmetros para o total de nós, épocas e taxa de *drop-out*; na sequência, mais duas camadas densas cujas funções de ativação corresponderam a respectivamente Relu e Sigmoid. Para otimização dos parâmetros foi testado o otimizador de Adam *default* do método que segue os parâmetros citados no artigo original[18], função de perda pelo critério da entropia cruzada. Em relação aos parâmetros que foram variados, definiu-se o intervalo de 10 a 110 para o total de épocas, com valores igualmente espaçados adotando-se um intervalo de 20, ou seja: 10, 30, 50, 70, 90, 110 épocas. Já quanto aos nós, os limites considerados foram de 12 a 132, com intervalo de 24, ou seja: 12, 36, 60, 84, 108, 132 nós. Por último, também se variou a taxa de *drop-out* considerada no intuito de mitigar as chances de *overfitting* do modelo, os valores considerados para tanto variaram de 0,10 a 0,50, igualmente espaçados em 0,10, ou seja: 0,00, 0,10, 0,20, 0,3, 0,4, 0,5. Os parâmetros propostos, combinados entre si, geraram as 216 possíveis combinações avaliadas.

Ao final de cada execução das Redes Bi-LSTM, utilizando-se a amostra de teste, foram medidas as estatísticas de qualidade da classificação: F1, *recall*, *precision* e *accuracy*. Além disso, foi medido o tempo total dos treinamentos e coleta das estatísticas, desde o modelo 1 até o final da coleta de estatísticas do modelo 216, para fins de cálculo do *speed-up* e eficiência, conforme descrito no Capítulo 2. Por fim, foi medido o tempo para execução de cada combinação de parâmetros, a fim de medir-se a mediana dos tempos e estimar-se o tempo total sequencial.

Utilizando as nomenclaturas de atividades básicas para PS propostas por Chirigati et. al. [10], descritas no Capítulo 2, o *workflow* desse estudo foi composto por uma combinação de *Joins* em paralelo, onde cada célula apresentou a entrada de n = 3 parâmetros e a saída de n = 4 estatísticas de qualidade da classificação, que por sua vez, posteriormente foram consolidadas em um arquivo de saída com uma etapa *Reduce*, n:1. A Figura

4.2 ilustra o fluxo de PS adotado nesta pesquisa. Ao total foram testados em paralelo 216 combinações de parâmetros, utilizando-se uma máquina com 24 núcleos físicos para tanto.

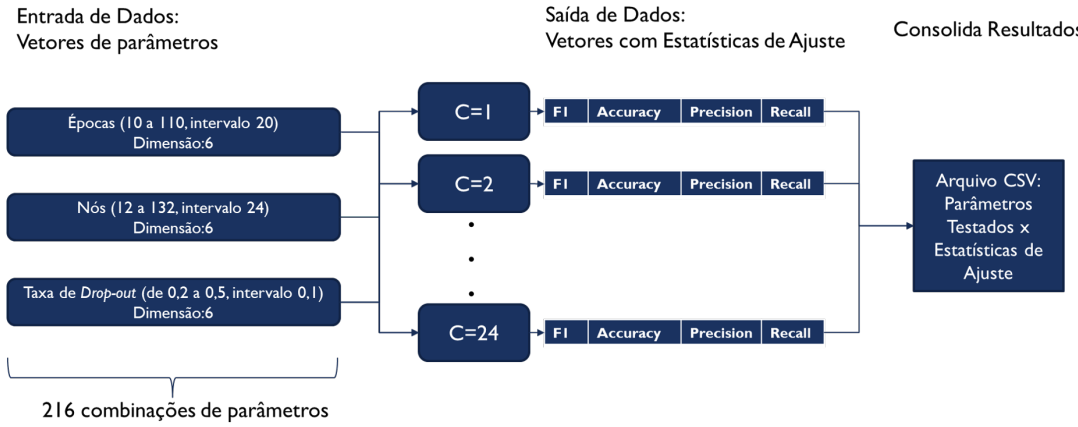

Figura 4.2 – Fluxo de trabalho *parameter sweep* (PS) aplicado neste estudo.

Conforme apresentado no Capítulo 2, a característica de independência dos dados de entrada dos modelos permite que se realize uma abordagem de treinamento via PS a qual otimiza a utilização dos recursos computacionais disponíveis. Caso não fosse adotada essa abordagem e se realizasse um treinamento sequencial, parte da arquitetura disponível ficaria ociosa. Assim, torna-se uma proposta viável e mais eficiente, facilitando a identificação da melhor combinação de parâmetros, em última análise gerando melhores resultados em menor tempo.

4.2.5 Fase 4: Método Ensemble

Uma vez que foram processados 216 modelos de classificação, mesmo que fossem de mesma técnica, existiam parâmetros de entrada e, portanto, também modelos finais diferentes. Ao final do treinamento, selecionou-se aqueles 4 com maior desempenho em termos de F1, combinando-os pela regra da soma supramencionada (Equação 4.1) a fim de verificar se seria possível formar um modelo de classificação melhor pelo método Ensemble do que aquele com melhor desempenho individual. Foram escolhidos os 4 melhores devido a seu desempenho superior em comparação com a grande maioria dos modelos.

5. AVALIAÇÃO DE RESULTADOS E TRABALHOS RELACIONADOS

5.1 Análise Estatística

O experimento apresentado para fins de análise estatística foi considerado como um experimento Fatorial de 3 fatores: (1) Épocas, (2) Nós, (3) *Drop-out*, sendo cada um com 6 níveis testados conforme ilustrado na Tabela 5.1. A combinação entre os níveis dos fatores gerou um total de 216 tratamentos testados, sem réplicas, ou seja, uma repetição para cada combinação de níveis dos fatores. Os testes estatísticos foram conduzidos avaliando-se inicialmente a hipótese nula (h0) de que nenhum fator produzisse alteração estatisticamente significativa na qualidade da classificação dos textos, em termos de F1, contra a hipótese alternativa(h1) de que pelo menos 1 dos fatores interferisse no resultado de F1. Foram considerados para teste tanto o efeito principal dos 3 fatores como o efeito cruzado de primeira ordem entre eles, para caso fosse identificado algum efeito de interação entre eles. Em caso de se rejeitar a h0 no primeiro teste, ele seria complementado com uma análise de regressão para cada um dos fatores, no intuito de se encontrar o ponto de ótimo para cada fator considerando-se os limites testados, ajustando-se uma regressão ou superfície de resposta apropriada conforme proposto por Montgomery ([29]). Os testes estatísticos foram executados no *software* estatístico R 3.6.2. Para fins de significância foi adotado um $\alpha = 0,05$.

Tabela 5.1 – Delineamento Experimental Análise Estatística

Fator	**Nível**	**Descrição**
Épocas	1	10
Épocas	2	30
Épocas	3	50
Épocas	4	70
Épocas	5	90
Épocas	6	110
Nós	1	12
Nós	2	36
Nós	3	60
Nós	4	84
Nós	5	108
Nós	6	132
Drop-out	1	0,00
Drop-out	2	0,10
Drop-out	3	0,20
Drop-out	4	0,30
Drop-out	5	0,40
Drop-out	6	0,50

A Tabela 5.2 apresenta os resultados da Análise de Variância ajustada no intuito de verificar se pelo menos um dos fatores testados, ou seja, os parâmetros da rede Bi-LSTM variados na metodologia PS, interfere significativamente no resultado da classificação em termos de F1. Conforme pode ser verificado na tabela, considerando a variação dos parâmetros explorados, só foi identificada diferença significativa na qualidade da classificação ao se variar o número de nós, não sendo identificada diferença estatisticamente significativa no resultado ao se alterar os demais parâmetros (total de épocas e taxa de *drop-out*, assim como não chegou a ser identificado efeito significativo de qualquer interação de primeira ordem entre os fatores.

Tabela 5.2 – Tabela ANOVA

Fonte de Variação	**Graus Liberdade**	**SQ**	**SQM**	**Estatística F**	**p-valor**
Épocas	5	0,0008	0,00016	1,164	0,331
Nós	5	0,0071	0,00143	10,650	<0,001
Drop-out	5	0,0012	0,00023	1,730	0,132
Épocas*Nós	25	0,0039	0,00015	1,152	0,297
Épocas**Drop-out*	25	0,0037	0,00015	1,106	0,346
Nós**Drop-out*	25	0,0033	0,00013	0,995	0,479
Resíduos	125	0,0167	0,00013		

Seguindo a análise estatística para o efeito principal do total de nós, único que apresentou diferença estatisticamente significativa, dado que os níveis eram quantitativos e não categóricos, segundo Montgomery([29]), é mais recomendável e informativo complementar-se com o ajuste de uma regressão do que com um teste de comparações múltiplas entres as categorias testadas. No ajuste da regressão, foram testados os efeitos até um polinômio de grau 4, momento em que o efeito de regressão deixou de ser significativo. Assim, ao final, sendo estatisticamente significativo ($p<0,05$) o efeito linear, quadrático e cúbico, conforme resultado do ajuste de regressão polinomial de grau 3 apresentado na Tabela 5.3. A regressão ajustada apresentou R^2 de 0,2, o que se pode considerar relativamente baixo, demonstrando que a equação ajustada seria responsável por explicar apenas aproximadamente 20% da variabilidade observada em função do total de Nós. Aditivamente, verifica-se que o máximo da função é atingido com o menor número testado para o parâmetro, 12 nós. Em estudos futuros, recomenda-se ampliar o limite inferior do total de Nós a fim de verificar o comportamento fora desse intervalo. Essa observação é acrescentada uma vez que as estimativas de um modelo de regressão não devam ser extrapoladas para além dos limites testados.

Tabela 5.3 – Análise complementar à ANOVA, ajuste de Regressão Polinomial.

Coeficiente	Estimativa	Erro Padrão	t-valor	p-valor
Intercepto	0,778	0,0071	109,49	<0,001
Nós	-0,0302	0,0081	-3,73	<0,001
Nós²	0,0074	0,0026	2,86	<0,01
Nós³	-0,0006	0,0002	-2,36	0,019

5.2 Apresentação dos Resultados

A Tabela 5.4 resume os resultados encontrados ao se processar a Metodologia PS proposta. Observa-se que foi possível identificar uma combinação de parâmetros 9,8% melhor do que a média das combinações testadas e 10,3% melhor do que a pior delas, isso em termos da métrica F1 a qual é mais robusta considerando o desbalanceamento inerente à amostra. Para tanto, o fluxo proposto foi capaz de atingir um *speed-up* de 19,8, com a utilização de 24 *threads* para os 24 núcleos físicos. Em complemento, considerando as informações expostas, foi atingida uma eficiência de 82,6%. Em termos absolutos, para termos a noção de tempo, o experimento que necessitaria de aproximadamente 74 dias de processamento sequencial (considerando o tempo mediano dos processamentos individuais) foi executado em aproximadamente 3,7 dias.

Tabela 5.4 – Resultados da Metodologia *Parameter Sweep* para 216 combinações de parâmetros para treinamento de Redes Bi-LSTM. Utilizando amostra de 39.032 textos jurídicos.

Estatística	Accuracy	Precision	Recall	F1
Mínimo	0,819	0,819	0,819	0,739
Média	0,821	0,821	0,821	0,743
Mediana	0,820	0,820	0,820	0,739
Máximo	0,846	0,846	0,846	0,816

Na sequência, adotando-se uma combinação conservadora para combinação dos 4 melhores modelos de Redes Neurais Bi-LSTM, utilizou-se a regra da soma apresentada no Capítulo 2, obteve-se os resultados apresentados na Tabela 5.6. Os resultados dos melhores modelos individuais são apresentados na Tabela 5.5. Percebeu-se que as melhorias em termos de desempenho esperadas não foram observadas, portanto dado o aumento de complexidade num modelo Ensemble versus o melhor modelo individual, para o caso específico avaliado no presente estudo, ainda faria mais sentido se recomendar o uso do melhor resultado individual. Essa recomendação faria mais sentido tanto pelo desempenho superior na qualidade da classificação dos textos quanto por ser um modelo mais parcimonioso, menos complexo para interpretação e processamento.

Tabela 5.5 – Resultados da Metodologia *Parameter Sweep* Resultado dos 10 modelos com melhor desempenho dentre os 216 treinados.

Modelo: Épocas, Nós, *Drop-out*	**Accuracy**	**Precision**	**Recall**	**F1**
Modelo75: 50,12,30	0,846	0,846	0,846	0,846
Modelo172: 110,12,30	0,844	0,844	0,844	0,844
Modelo78: 50,12,20	0,842	0,842	0,842	0,842
Modelo148: 90,12,30	0,841	0,841	0,841	0,841
Modelo74: 50,12,50	0,840	0,840	0,840	0,840
Modelo103: 70,12,40	0,839	0,839	0,839	0,839
Modelo40: 30,12,40	0,836	0,836	0,836	0,836
Modelo152: 90,36,30	0,835	0,835	0,835	0,835
Modelo154: 90,36,20	0,830	0,830	0,830	0,830
Modelo122: 90,12,0	0,825	0,825	0,825	0,825

Tabela 5.6 – Resultado do Método Ensemble a partir da combinação dos 4 melhores modelos individuais identificados na aplicação da Metodologia *Parameter Sweep*.

Métrica	**Valor**
Accuracy	0,808
Recall	0,808
Precision	0,845
F1	0,826

5.3 Trabalhos Relacionados

Os estudos mais recentes quanto ao uso de técnicas de PLN a textos jurídicos, independentemente da nacionalidade, têm apresentado melhores resultados ao adotar variações de abordagens de DL como RNN, Redes Bi-LSTM, método BERT, ou CNN [50, 8, 22, 13, 15, 35, 30], o que vai ao encontro do direcionamento do presente estudo ao se prosseguir avaliando apenas as Redes Bi-LSTM e se descartando os demais modelos avaliados inicialmente (Naive Bayes e Random Forest). A Tabela 5.7 apresenta um resumo dos resultados de estudos anteriores apresentados no Capítulo 2 na Tabela 2.1. Fica como ressalva para interpretação o fato de nem todos os estudos reportaram todos os indicadores em seu resultado, dessa forma tendo sido utilizados para cálculo da média e desvio padrão todos os representantes de cada indicador quando existissem. Ao todo, para consolidação da informação nessa tabela, foram considerados 14 artigos [23, 31, 12, 36, 25, 30, 7, 8, 22, 13, 4, 50, 15, 35].

Quando comparamos os resultados do presente estudo (Tabelas 5.4 e 5.5) com os reportados anteriormente para classificação de textos jurídicos, observa-se que o desempenho parece ter ficado próximo à média observada por outros estudos, conforme informação sumarizada na Tabela 5.7. Dessa forma, acredita-se que os resultados estejam coerentes e satisfatórios para o que se propôs inicialmente e frente a estudos similares. Apesar disso,

cabe destacar que nenhum dos estudos anteriores, aos quais estamos comparando, trabalharam exatamente com o mesmo tipo de documento jurídico ou classificação de processo que o presente estudo, portanto ainda sendo importante considerar as peculiaridades de cada um ao se fazer uma análise dos valores apresentados.

Tabela 5.7 – Resumo do resultado final dos algoritmos de classificação de textos jurídicos com melhor desempenho de trabalhos anteriores, a partir dos dados da Tabela 2.1

Métrica	Média(DP)
Accuracy	0,79(0,21)
Precision	0,78(0,14)
Recall	0,69(0,16)
F1	0,82(0,13)

Dentre as particularidades de cada estudo, pode-se citar o tipo de documento jurídico considerado, sendo que a maior parte dos estudos encontrados não foi muito específica em sua descrição, utilizando-se de termos genéricos como *legal documents*, *legal cases*. Entretanto, alguns deles foram mais específicos descrevendo os documentos como estatutos, regulamentos ou decisões administrativas [23], Leis sobre Pensão [31], políticas de privacidade [12], sentenças [36] ou Leis Financeiras[30]. Além do tipo de documento, também há diferença no idioma dos textos considerados para classificação e no tamanho da amostra.

Já em relação às melhorias advindas do emprego da técnica de paralelismo PS, considerando o impacto no tempo de processamento, *speed-up* e eficiência, fica difícil a comparação com estudos anteriores. Não foram identificados artigos que tratassem especificamente a respeito da utilização da técnica do PS no intuito de otimizar a busca pela melhor combinação de parâmetros para um modelo de Deep Learning para classificação de textos.

6. CONCLUSÃO

Através do treinamento em paralelo dos modelos, utilizando-se a técnica de PS, foi possível chegar a um resultado superior à maioria das parametrizações testadas (10% melhor do que a pior parametrização testada e 9,8% superior à média dos resultados encontrados) em aproximadamente 20 vezes menos tempo do que levaria para testar todas as mesmas possibilidades de maneira sequencial. Assim, entende-se que seja recomendável uma abordagem de PS no intuito de buscar a melhor parametrização de Redes Bi-LSTM para classificação de textos jurídicos em menos tempo, ao aproveitar-se melhor os recursos computacionais disponíveis. Ainda, para estudos futuros, pressupõe-se que seria possível acelerar mais o processo de PS ao adotar-se uma arquitetura de multicomputadores, trabalhando-se com o processamento de maneira distribuída.

Em relação à classificação automática de textos jurídicos, verificou-se a viabilidade de automatizar a classificação das petições a partir do conteúdo textual contido nelas com um desempenho satisfatório, sendo atingido um F1 de 0,846 na melhor Rede Bi-LSTM individual. Suplementarmente, foi apresentada uma proposta de combinação dos melhores modelos individuais através de uma técnica Ensemble que não foi capaz de majorar esse desempenho individual, atingindo-se um F1 de 0,826. Portanto, especificamente para o problema apresentado no presente trabalho, seria mais recomendável manter-se apenas o melhor modelo individual do que combinar-se os melhores resultados. Parece que talvez pela baixa variabilidade entre os melhores modelos finais, ao combinar-se eles entre si, não foi possível captar particularidades diferentes no comportamento da amostra, resultando num desempenho inclusive levemente inferior ao apresentado pelo melhor resultado individual.

Em relação ao problema jurídico utilizado para análise prática, mostrou-se ser possível automatizar grande parte das classificações dos processos, o que seria capaz de acelerar o andamento processual e, em última análise, liberar os servidores que ora estiveram envolvidos com a classificação dos processos para desempenhar outras atividades. Assim, conjectura-se que o modelo proposto por este estudo seja uma das possibilidades a ser aplicada ao meio jurídico visando a uma melhora na gestão dos recursos públicos brasileiros.

6.1 Sugestões para Estudos Futuros

Quanto a estudos futuros, sugere-se que sejam explorados outros modelos de representação de textos diferentes do Word2Vec. Seria interessante explorar não apenas o efeito principal de cada modelo de representação, como a combinação desses com os de-

mais parâmetros, pois pode ser que a melhor combinação de parâmetros para maximizar a qualidade final do modelo de classificação seja diferente conforme o modelo de representação adotado na entrada. Além disso, seria interessante incluir uma etapa de extração para conversão dos pdfs de imagem para texto, de forma a aproveitar uma maior parte da amostra disponibilizada. Ainda em relação às peculiaridades do problema prático explorado neste estudo, como sabe-se que a combinação de parâmetros e desempenho final do modelo de classificação dependem também do conjunto de dados da entrada, seria interessante avaliar a aplicação da metodologia proposta a outros corpus.

Por último, sugere-se que seja explorado outro nível de paralelismo, adotando-se uma arquitetura de multicomputadores, trabalhando-se com processamento de maneira distribuída, a fim de otimizar ainda mais a busca pela melhor combinação de parâmetros. Acredita-se que ainda há espaço para melhoria no aproveitamento de estruturas redundantes disponíveis e, consequentemente, no tempo de resposta para que seja indicada a melhor combinação de parâmetros para o modelo de classificação a ser treinado.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Agnihotri, D.; Verma, K.; Tripathi, P. “Variable global feature selection scheme for automatic classification of text documents”, *Expert Systems with Applications*, vol. 81, 2017, pp. 268 – 281.

[2] Altınel, B.; Ganiz, M. C. “Semantic text classification: A survey of past and recent advances”, *Information Processing & Management*, vol. 54, 2018, pp. 1129 – 1153.

[3] Anderlucci, L.; Guastadisegni, L.; Viroli, C. “Classifying textual data: shallow, deep and ensemble methods”, *arXiv e-prints*, vol. abs/1902.07068, 2019, pp. arXiv:1902.07068.

[4] Braz, F. A.; da Silva, N. C.; de Campos, T. E.; Chaves, F. B. S.; Ferreira, M. H. S.; Inazawa, P. H.; Coelho, V. H. D.; Sukiennik, B. P.; de Almeida, A. P. G. S.; de Barros Vidal, F.; Bezerra, D. A.; Gusmao, D. B.; Ziegler, G. G.; Fernandes, R. V. C.; Zumblick, R.; Peixoto, F. H. “Document classification using a bi-lstm to unclog brazil’s supreme court”, *CoRR*, vol. abs/1811.11569, 2018.

[5] Breiman, L. “Random forests”, *Machine Learning*, vol. 45, 2001, pp. 5–32.

[6] Callaghan, S.; Maechling, P.; Small, P.; Milner, K.; Juve, G.; Jordan, T. H.; Deelman, E.; Mehta, G.; Vahi, K.; Gunter, D.; Beattie, K.; Brooks, C. “Metrics for heterogeneous scientific workflows: A case study of an earthquake science application”, *The International Journal of High Performance Computing Applications*, vol. 25, 2011, pp. 274–285.

[7] Chalkidis, I.; Androutsopoulos, I.; Michos, A. “Extracting contract elements”. In: Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law, 2017, pp. 19–28.

[8] Chalkidis, I.; Kampas, D. “Deep learning in law: early adaptation and legal word embeddings trained on large corpora”, *Artificial Intelligence and Law*, vol. 27, 2019, pp. 1–28.

[9] Chen, S. J.; Wang, X.; Qin, Z.; Metzler, D. “Parameter tuning in personal search systems”. In: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 97–105.

[10] Chirigati, F.; Silva, V.; Ogasawara, E.; de Oliveira, D.; Dias, J.; Porto, F.; Valduriez, P.; Mattoso, M. “Evaluating parameter sweep workflows in high performance computing”. In: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, 2012, pp. 1 – 10.

[11] Chunduri, S.; Ghaffar, M.; Lahijani, M. S.; Srinivasan, A.; Namilae, S. “Parallel low discrepancy parameter sweep for public health policy”. In: Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2018, pp. 291–300.

[12] Costante, E.; Sun, Y.; Petković, M.; den Hartog, J. “A machine learning solution to assess privacy policy completeness: (short paper)”. In: Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, 2012, pp. 91–96.

[13] Elnaggar, A.; Gebendorfer, C.; Glaser, I.; Matthes, F. “Multi-task deep learning for legal document translation, summarization and multi-label classification”. In: Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference, 2018, pp. 9–15.

[14] Faceli, K.; Lorena, A. C.; Gama, J.; de Carvalho, A. C. “Inteligência Artificial:Uma Abordagem de Aprendizado de Máquina”. LTC, 2015.

[15] Fernandes, W. P. D.; Silva, L. J. S.; Frajhof, I. Z.; da Franca Couto Fernandes de Almeida, G.; Konder, C. N.; Nasser, R. B.; de Carvalho, G. R.; Barbosa, S. D. J.; Lopes, H. C. V. “Appellate court modifications extraction for portuguese”, *Artificial Intelligence and Law*, 2019, pp. 1–34.

[16] Hochreiter, S.; Schmidhuber, J. “Long short-term memory”, *Neural Computation*, vol. 9, 1997, pp. 1735–1780.

[17] Kadhim, A. I. “Survey on supervised machine learning techniques for automatic text classification”, *Artificial Intelligence Review*, vol. 52, 2019, pp. 273–292.

[18] Kingma, D. P.; Ba, J. “Adam: A Method for Stochastic Optimization”, *arXiv e-prints*, 2014, pp. arXiv:1412.6980.

[19] Kittler, J. “Combining classifiers: A theoretical framework”, *Pattern Analysis and Applications*, vol. 1, 1998, pp. 18–27.

[20] Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. “Text classification algorithms: A survey”, *Information*, vol. 10, 2019.

[21] Leevy, J. L.; Khoshgoftaar, T. M.; Bauder, R. A.; Seliya, N. “A survey on addressing high-class imbalance in big data”, *Journal of Big Data*, vol. 5, 2018, pp. 1–30.

[22] Li, P.; Zhao, F.; Li, Y.; Zhu, Z. “Law text classification using semi-supervised convolutional neural networks”, *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 309–313.

[23] Lu, Q.; Conrad, J. G.; Al-Kofahi, K.; Keenan, W. “Legal document clustering with built-in topic segmentation”. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, 2011, pp. 383–392.

[24] Maalouf, M.; Trafalis, T. B. “Rare events and imbalanced datasets: an overview”, *International Journal of Data Mining, Modelling and Management*, vol. 3, 2011, pp. 375–388.

[25] Mastropaolo, A.; Pallante, F.; Radicioni, D. P. “Legal documents categorization by compression”. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, 2013, pp. 92–100.

[26] Ministra Ellen Gracie, C. N. J. “Resolução nº 46, de 18 de dezembro de 2007: cria as tabelas processuais unificadas do poder judiciário e dá outras providências”, 2007.

[27] Mirończuk, M. M.; Protasiewicz, J. “A recent overview of the state-of-the-art elements of text classification”, *Expert Systems with Applications*, vol. 106, 2018, pp. 36 – 54.

[28] Mohammad, A. H. “Arabic text classification: A review”, *Modern Applied Science*, vol. 13, 2019.

[29] Montgomery, D. C. “Design and Analysis of Experiments”. John Wiley & Sons, Inc., 2013.

[30] Neill, J. O.; Buitelaar, P.; Robin, C.; Brien, L. O. “Classifying sentential modality in legal language: A use case in financial regulations, acts and directives”. In: Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law, 2017, pp. 159–168.

[31] Nguyen, L.-M.; Bach, N. X.; Shimazu, A. “Supervised and semi-supervised sequence learning for recognition of requisite part and effectuation part in law sentences”. In: Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing, 2011, pp. 21–29.

[32] Ogasawara, E.; de Oliveira, D.; Valduriez, P.; Dias, J.; Porto, F.; Mattoso, M. “An algebraic approach for data-centric scientific workflows”, *PVLDB*, vol. 4, 2011, pp. 1328–1339.

[33] Pinheiro, R. H.; Cavalcanti, G. D.; Ren, T. I. “Data-driven global-ranking local feature selection methods for text categorization”, *Expert Systems with Applications*, vol. 42, 2015, pp. 1941 – 1949.

[34] Ponce, E.; Stephenson, B.; Lenhart, S.; Day, J.; Peterson, G. D. “Papas: A portable, lightweight, and generic framework for parallel parameter studies”. In: Proceedings of the Practice and Experience on Advanced Research Computing, 2018.

[35] Rabelo, J.; Kim, M.-Y.; Goebel, R. “Combining similarity and transformer methods for case law entailment”. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, 2019, pp. 290–296.

[36] Ramrakhiyani, N.; Pawar, S.; Palshikar, G. K. “A system for classification of propositions of the indian supreme court judgements”. In: Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation, 2013.

[37] Rokach, L. “Ensemble-based classifiers”, *Artificial Intelligence Review*, vol. 33, 2010, pp. 1–39.

[38] Schuster, M.; Paliwal, K. K. “Bidirectional recurrent neural networks”, *IEEE Transactions on Signal Processing*, vol. 45, 1997, pp. 2673–2681.

[39] Severyn, A.; Moschitti, A. “Learning to rank short text pairs with convolutional deep neural networks”. In: SIGIR, 2015.

[40] Shabanian, S.; Arpit, D.; Trischler, A.; Bengio, Y. “Variational bi-lstms”, *arXiv preprint arXiv:1711.05717*, 2017, 1711.05717.

[41] Shalev-Shwartz, S.; Ben-David, S. “Understanding Machine Learning:From Theory to Algorithms”. Cambridge University Press, 2014.

[42] Sinoara, R. A.; Camacho-Collados, J.; Rossi, R. G.; Navigli, R.; Rezende, S. O. “Knowledge-enhanced document embeddings for text classification”, *Knowledge-Based Systems*, vol. 163, 2019, pp. 955 – 971.

[43] Srinivasan, A.; Sudheer, C. D.; Namilae, S. “Optimizing massively parallel simulations of infection spread through air-travel for policy analysis”. In: Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2016, pp. 136–145.

[44] Stein, R. A.; Jaques, P. A.; Valiati, J. F. “An analysis of hierarchical text classification using word embeddings”, *Information Sciences*, vol. 471, 2018, pp. 216–232.

[45] Uysal, A. K.; Gunal, S. “A novel probabilistic feature selection method for text classification”, *Knowledge-Based Systems*, vol. 36, 2012, pp. 226 – 235.

[46] Walker, E.; Guiang, C. “Challenges in executing large parameter sweep studies across widely distributed computing environments”. In: Proceedings of the 5th IEEE Workshop on Challenges of Large Applications in Distributed Environments, 2007, pp. 11–18.

[47] Whitaker, L. I. K. . C. J. “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy”, *Machine Learning*, vol. 51, 2003, pp. 181–207.

[48] Zhang, Q.; Yang, L. T.; Chen, Z.; Li, P. “A survey on deep learning for big data”, *Information Fusion*, vol. 42, 2018, pp. 146–157.

[49] Zhou, L.; Pan, S.; Wang, J.; Vasilakos, A. “Machine learning on big data: Opportunities and challenges”, *Neurocomputing*, vol. 237, 2017, pp. 350–361.

[50] Škrlj, B.; Kralj, J.; Lavrac, N.; Pollak, S. “Towards robust text classification with semantics-aware recurrent neural architecture”, *Machine Learning and Knowledge Extraction*, vol. 1, 2019, pp. 575–589.