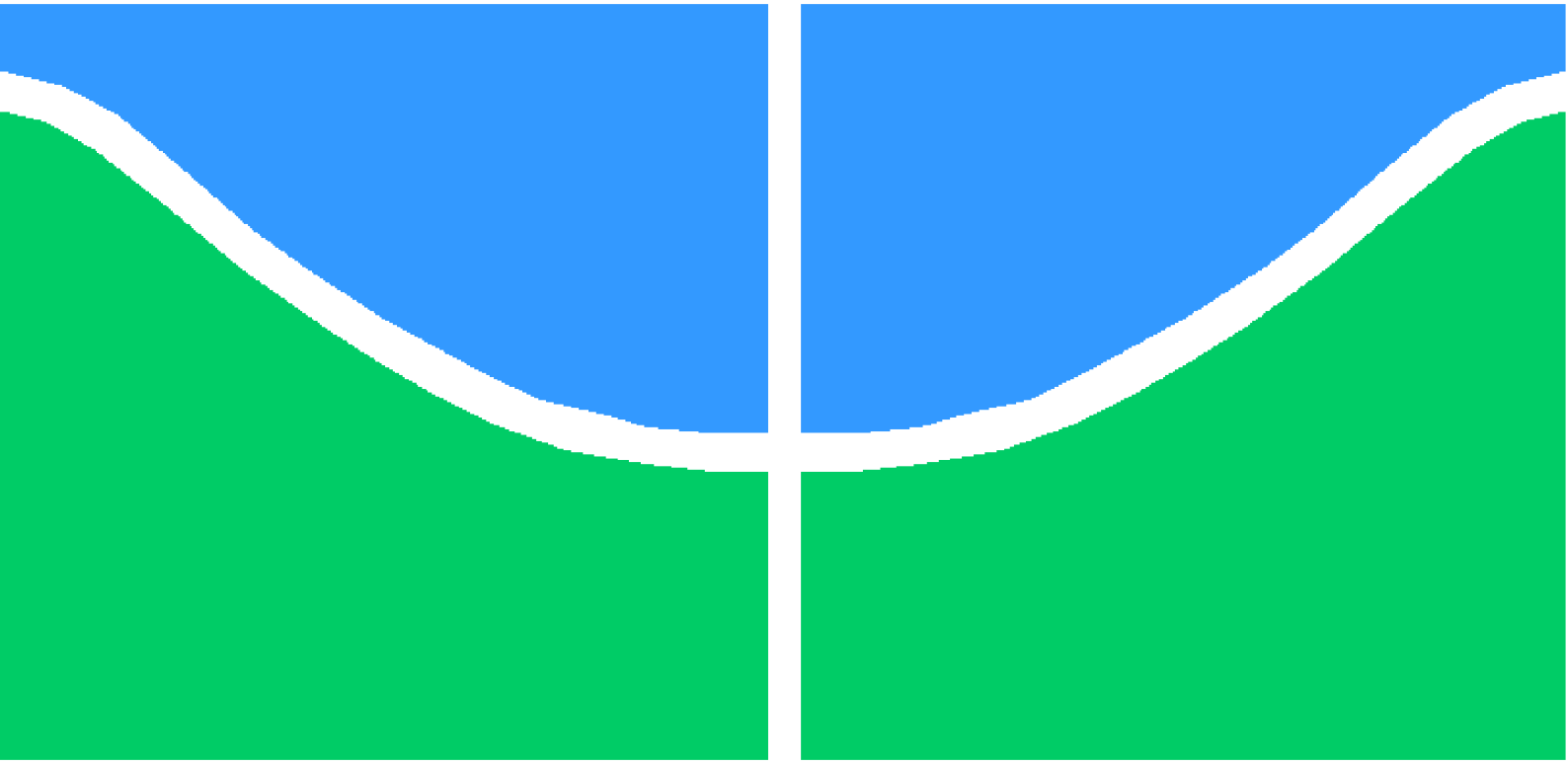


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Classificação de peças processuais jurídicas: Inteligência Artificial no Direito

Autor: Marcelo Herton Pereira Ferreira
Orientador: Doutor Nilton Correia da Silva

Brasília, DF
2018

Marcelo Herton Pereira Ferreira

Classificação de peças processuais jurídicas: Inteligência Artificial no Direito

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Doutor Nilton Correia da Silva

Brasília, DF

2018

Marcelo Herton Pereira Ferreira
Classificação de peças processuais jurídicas: Inteligência Artificial no Direito/ Marcelo Herton Pereira Ferreira. – Brasília, DF, 2018-
78 p. : il. (algumas color.) ; 30 cm.

Orientador: Doutor Nilton Correia da Silva

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.

1. Aprendizado de Máquina. 2. Classificação de documentos. I. Doutor Nilton Correia da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Classificação de peças processuais jurídicas: Inteligência Artificial no Direito

CDU 02:141:005.6

Marcelo Herton Pereira Ferreira

Classificação de peças processuais jurídicas: Inteligência Artificial no Direito

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 10 de dezembro de 2018:

Doutor Nilton Correia da Silva
Orientador

Doutor Fabricio Ataides Braz
Convidado 1

Mestre Tainá Aguiar Junquilho
Convidado 2

Brasília, DF
2018

Resumo

O Supremo Tribunal Federal tem a necessidade de separar as peças processuais jurídicas para facilitar a distribuição do processo internamente. Atualmente esta separação de volumes em peças e a classificação destas são feitas manualmente por uma equipe. A metodologia para o trabalho é a experimental. O processo de desenvolvimento é baseado na pesquisa e no desenvolvimento projetos de aprendizado de máquina, no qual foram utilizados para elaborar único processo. Nos documentos jurídicos deste trabalho, utilizou-se apenas a primeira página, removeu-se amostras com conteúdos duplicados, foram pré-processando. Não identificou-se correlações entre as categorias. Fez-se a implementação de 9 modelos neurais: CNN, CNN-rand, MLP, BLSTM, LSTM, BRNN, BLSTM-C, CNN-LSTM, VDCNN e suas respectivas parametrizações a fim de alcançar os maiores valores das métricas: Acurácia, Precisão e Revocação. Com modelo de base SVM Linear, obteve-se acurácia de 0,93, precisão de 0,93 e revocação de 0,92. O modelo neural com as melhores métricas foi o LSTM com o dado pré-processado, seus resultados foram acurácia 0,94, precisão 0,93 e revocação 0,95. Os documentos jurídicos são computacionalmente separáveis e pode-se escolher entre os modelos SVM Linear, BLSTM e CNN-rand, pois foram os que sofreram menos *overfitting*, possuem as melhores métricas no conjunto de teste e têm o tempo de predição entre 16ms a 72ms por documento.

Palavras-chave: Classificação de documentos; Aprendizado de máquina; Peças jurídicas.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AI	*Artifical Intelligence*
ARE	Agravo em Recurso Extraordinário
BLSTM	*Bidirectional Long Short-Term Memory*
BLSTM-C	*Bidirectional Long Short-Term Memory with Convolutional*
BoW	*Bag of words*
BRNN	*Bidirectional Recurrent Neural Network*
CF	Constituição Federal
CNJ	Conselho Nacional de Justiça
CNN	*Convolutional Neural Network*
CNN-LSTM	*Convolutional Neural Network with Long Short-Term Memory*
CNN-rand	*Convolutional Neural Network Randomized*
CPC	Código Processual Civil
CRISP-DM	*Cross-industry standard process for data mining*
GPAM	Grupo de Pesquisa em Aprendizado de Máquina
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
NLP	*Natural Language Processing*
OCR	*Optical Character Recognition*
PDF	*Portable Document Format*
PJe	Processo Judicial eletrônico
RBF	*Radial Basis Function*
RE	Recurso Extraordinário

RG	Repercussão Geral
RNN	*Recurrent Neural Network*
STF	Supremo Tribunal Federal
STJ	Superior Tribunal de Justiça
STM	Superior Tribunal Militar
SVM	Máquinas de suporte vetorial
TSE	Tribunal Superior Eleitoral
TST	Tribunal Superior do Trabalho
VDCNN	*Very Deep Convolutional Neural Network*

Lista de símbolos

$\odot$	Operação de Hadamard
δ	Delta
θ	Theta
$\hat{y}$	Valor predito em uma classificação
Σ	Somatório

Sumário

1 Introdução

Técnicas de Aprendizado de Máquina (do inglês *Machine Learning* ML) têm sido amplamente adotadas no mundo da computação para realizar tarefas em que há muitos dados disponíveis e existe algum tipo de relação entre eles (BRINK; RICHARDS; FETHEROLF, 2015).

Pode-se dizer que o ML é uma vertente da Inteligência Artificial (do inglês *Artificial Intelligence* AI), o qual possui um corpo de conhecimento específico ou um conjunto de técnicas para que a máquina possa aprender com os dados. Enquanto que a AI é um campo de estudo muito mais abrangente que engloba os campos de visão computacional, processamento de linguagem natural, robótica e outros (BRINK; RICHARDS; FETHEROLF, 2015).

Outro campo dentro da AI é o Processamento de Linguagem Natural (do inglês *Natural Language Processing* NLP), o qual é extremamente importante para aplicações modernas, pois trata-se da interpretação dos constructos da linguagem humana para serem processadas por computadores (GOLDBERG, 2017). Com a digitalização dos documentos, viabilizou-se o uso de técnicas, métodos para facilitar análises e extração de informações de conteúdos (OLIVEIRA; FILHO, 2017).

Alguns dos conjuntos de técnicas para o NLP são o diálogo ou sistemas de fala, análises textuais e recuperação de informações através de perguntas e respostas (ESLICK; LIU, 2005). As técnicas, que são úteis para lidar com documentos digitais e facilitar a separação de conteúdo destes, são a classificação de documentos, busca e recuperação de informações (OLIVEIRA; FILHO, 2017).

Ainda que haja um grande esforço para pesquisa nas áreas específicas de NLP, é inerente a todas elas as dificuldades de fazer com que a máquina consiga lidar com a enorme variabilidade da linguagem natural, ambiguidade e usos complexos da língua como figuras de linguagens (GOLDBERG, 2017).

O Supremo Tribunal Federal (STF) é um órgão público da esfera de Poder Jurídico do Brasil. A ele compete realizar a guarda da Constituição (1988), no qual julga casos em que os Artigos da Constituição possam ser violados ou mal interpretados, conflitos entre a União e os Estados incluindo o Distrito Federal, ações movidas por estados estrangeiros, razões contra o Conselho Nacional de Justiça (CNJ), conflitos entre os Tribunais de Justiça (BRASIL, 1988).

Cabe a ele julgar em recurso ordinário crimes políticos, *habeas corpus*, mandados de segurança e, em recurso extraordinário, infrações a Constituição, inconstitucionalidades

em tratados ou em leis, invalidar atos do governo caso contrariem a Constituição (BRASIL, 1988).

O Governo Brasileiro implantou a política de transparência pública, no qual todas as informações públicas devem ser disponibilizadas para que os cidadão pudessem acessar qualquer informação sobre as três esferas do poder: Judiciário, Executivo e Legislativo (BRASIL, 2011).

O sistema judiciário, assim como os outros dois poderes, teve um esforço para implantar todo o acesso a informação por meios eletrônicos, visto que a população em geral, quando quer buscar algo, utiliza os sites de busca como o Google [1] e DuckDuckGo [2] (RUSCHEL; ROVER; SCHNEIDER, 2011).

Logo então, o foco do sistema judiciário brasileiro foi no desenvolvimento dos *web site* de cada tribunal e na digitalização dos processos. O CNJ, que tem como uma de suas competências coordenar e auxiliar a digitalização de todos os 91 Tribunais de Justiça (RUSCHEL; ROVER; SCHNEIDER, 2011), colocou como metas para o desenvolvimento do parque tecnológico: a informatização todas as unidades judiciárias, interligação e implantação do processo eletrônico em parcela de suas unidades judiciárias. Com estas metas, esperava-se que todos os tribunais pudessem realizar a tramitação de processos por meio digital (BRASIL, 2009). Também, que o processo fosse representado da mesma forma em diferentes sistemas, para que a população pudesse ter acesso a eles através de mecanismos de consulta online (RUSCHEL; ROVER; SCHNEIDER, 2011).

Com isso, surgiu uma grande variabilidade de sistemas *online* mesmo com esforços do CNJ para realizar a unificação com o Modelo Nacional de Interoperabilidade (BRASIL, 2009). Muitos tribunais usam diferentes sistemas como PJe, Projudi e e-SAJ.

O Processo Judicial eletrônico (PJe) foi um dos sistemas desenvolvidos, o qual passou a ser adotado por diversos tribunais como uso obrigatório. O seu principal objetivo é manter um sistema que possibilite a transição dos processos, bem como o registro de peças processuais, além do acompanhamento de todos os acontecimentos independentemente do tribunal em que ele tramite. A proposta de sua solução é que ele fosse único para todos os tribunais (Wiki PJE, 2018).

O fato desses sistemas seguirem a lógica de processos físicos, quanto a classificação do tipo de peça jurídica, a montagem de volumes, à forma de peticionamento, ao processo de tramitação e aos meta-dados, causa problemas relacionados a dificuldade de comunicação entre sistemas, armazenamento de arquivos, reclamações quanto a usabilidade do sistema e a recuperação de informações dos dados torna-se mais difícil.

Esses sistemas possibilitam a tramitação de processos digitalizados, os quais estão

[1] Site :<https://google.com>
[2] Site: <https://duckduckgo.com>

sujeitos a problemas de identificação de texto, furos, manchas, desalinhamento da página e suas incorretas ordenações de conteúdos. A Figura 1 exibe uma página de um desses processos, a qual apresenta problema de parte do escaneamento estar espelhado, fazendo com que o texto fique no sentido contrário.

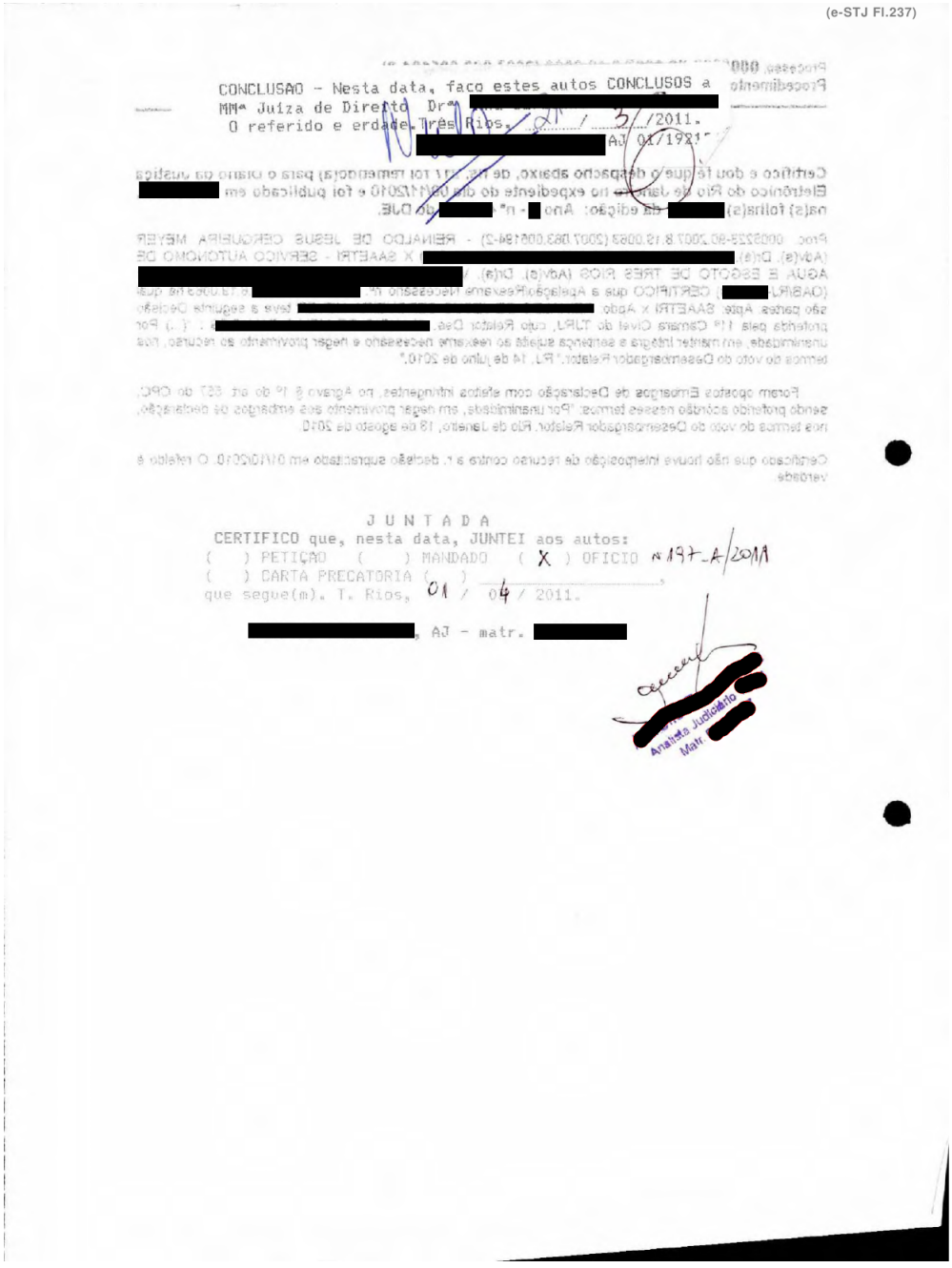


Figura 1 – Certidão de despacho digitalizada que possui parte de seu documento espelhado. Nomes pessoais e matrículas foram omitidos. Fonte: elaboração própria.

O Projeto Victor do STF tem o propósito de ser uma ferramenta que utiliza Inteligência Artificial para classificar novos processos em Repercussões Gerais (RG) (BRASIL, 2018). Esse projeto é focado na função principal do STF, que é a garantia da constitucionalidade das Leis Federais em conformidade com a Constituição Federal (1988) (BRASIL,

1988). É um projeto de parcerias públicas entre STF e Universidade de Brasília, sendo o projeto de AI mais relevante para o contexto jurídico brasileiro (BRASIL, 2018).

1.1 Problema e motivação

Como parte colaborativa para as pesquisas do Projeto Victor (BRASIL, 2018), este trabalho ajuda nas análises de mérito de um processo do STF, nas análises de admissibilidade, ajuda na distrição de processos para os gabinetes do Tribunal, no direcionamento das partes dos processos aos acessores e possibilita facilitar o trabalho na maior corte brasileira.

O STF é um órgão público central, lida com os casos de toda a Federação, advindos dos tribunais e juizados da segunda instância (BRASIL, 1988). Ele tem que lidar com as diferentes padronizações de marcadores nos volumes de processos e código das peças. Além disso, os servidores avaliam apenas as peças essenciais a sua atividade, por conseguinte, precisam encontrar entre os diversos documentos do processo os que serão utilizados por eles.

Os próprios advogados podem fazer a classificação do tipo de peça ao submeter no sistema eletrônico do STF, desde que obedeçam o tamanho máximo de 10 MB por PDF[3] (BRASIL, 2010). Por conta disso, há documentos que estão com apenas uma categoria carregando o conteúdo de várias peças.

Desta forma, foram identificados os seguintes problemas enfrentados pelo STF ao lidar com as peças de processos:

- Processos advindos de diferentes fontes possuem códigos identificadores de peças e marcadores de volumes não padronizados, fato que dificulta aos servidores de encontrar rapidamente as de seu interesse.
- Alguns marcadores nos volumes são errados ou incompletos, ou seja, não trazem informação suficiente para caracterizar uma peça específica.
- Peças duplicadas produzidas pela ressubmissão de um processo, por conta da adição do recurso de admissibilidade. Com esta duplicação, os servidores têm dificuldade de encontrar informações de interesse rapidamente.

Diante da problemática apresentada, definiu-se a pergunta de pesquisa: Como identificar automatizadamente o corpo de texto de diferentes tipos de peças processuais jurídicas, que sejam analisadas pelo STF para repercussão geral, em um único documento?

[3] Formato de arquivo criado pela empresa Adobe Systems Incorporated para unificar o compartilhamento de conteúdo. Disponível em: <https://acrobat.adobe.com/br/pt/acrobat/about-adobe-pdf.html>. Acesso em: 17-06-2018

1.2 Hipótese

Foram desenvolvidas duas hipóteses (GIL, 2002) para este trabalho baseadas na especificação do contexto e problemática do STF.

- As peças jurídicas avaliadas pelo STF para classificar um processo numa repercussão geral são computacionalmente separáveis.
- Não existem classes de peças diferentes com o mesmo conteúdo de texto.

1.3 Objetivo

Nesta seção, serão apresentados o objetivo geral e os objetivos específicos para a delimitação deste trabalho de conclusão de curso.

1.3.1 Objetivo geral

Classificar documentos jurídicos em diferentes categorias adotadas pelo STF utilizando técnicas de ML e NLP.

1.3.2 Objetivos específicos

- Levantar o estado da arte de classificação de documentos;
- Realizar análise exploratória dos dados;
- Construir um dicionário de palavras para o corpo textual dos processos.
- Elaborar arquitetura de pré-processamento dos dados;
- Realizar a transformação dos textos para representação computacional;
- Avaliar modelos para classificação de documentos;
- Modificar e evoluir modelos encontrados para ajustá-los ao problema do STF;
- Avaliar técnicas para validação de modelos.

1.4 Organização do trabalho

Este trabalho estrá organizado em sete capítulos. O primeiro é a Introdução onde contextualiza-se a problemática e definem-se os objetivos, o segundo capítulo é para descrição da metodologia científica adotada. O terceiro é a fundamentação teórica para o

desenvolvimento. O quarto é onde elabora-se uma breve análise dos dados. O quinto capítulo é onde apresenta-se os modelos propostos e os resultados de suas execuções. No sexto, faz-se uma discussão sobre os resultados obtidos e as referências adotadas. No sétimo e último, apresentam-se as conclusões do trabalho mediante todo o processo executado e os resultados coletados.

2 Metodologia

Este capítulo caracterizará o tipo de pesquisa e apresentará os aspectos metodológicos.

2.1 Caracterização da pesquisa

Este trabalho é uma pesquisa exploratória utilizando os métodos indutivo e experimental para o seu desenvolvimento. Uma pesquisa exploratória é utilizada quando deseja-se explorar as causas, os fatores influenciadores envolvidos e as fronteiras de um determinado problema. Usa-se da experimentação para controlar as variáveis envolvidas, tomar nota dos impactos delas para o problema (PRODANOV; FREITAS, 2013).

Métodos de pesquisa indutivos tem o objetivo de ampliar conhecimentos de uma determinada área, baseando-se na amostra de dados analisados. Diferentemente da dedução, em que chega-se a uma conclusão verdadeira. Este método tem o propósito de gerar uma probabilidade de a conclusão ser verdadeira (PRODANOV; FREITAS, 2013).

Na indução parte-se de casos particulares para a generalização de outros semelhantes (PRODANOV; FREITAS, 2013), logo, busca-se através da experimentação e observação dos métodos de classificação de documentos, alcançar o rotulamento de documentos do STF com as técnicas aqui exploradas.

2.2 Processo de desenvolvimento

A seguir será apresentado o processo para o desenvolvimento. Considerando-se que é uma pesquisa e um projeto de ML, mesclou-se ambos os processos para alcançar um processo final.

2.2.1 Projeto de pesquisa

Um projeto de pesquisa tem as etapas: **planejamento**, no qual formula-se os objetivos e o propósito para a pesquisa; **execução**, que representa o desenvolvimento do trabalho caracterizado pela coleta de dados; **análise** e proposições das conclusões; **documentação**, na qual é formulado o texto resultante do trabalho; **publicação** da pesquisa (PRODANOV; FREITAS, 2013).

Destas 4 fases, têm-se as principais atividades definidas por Prodanov e Freitas (2013) que são:

- **Formular e planejar a pesquisa**: nesta atividade, faz-se a escolha pelo assunto e levantamento bibliográfico para investigação do problema. Determinação dos objetos de estudo, além de averiguar assuntos correlatos com o escolhido (PRODANOV; FREITAS, 2013).

- **Escolher assunto e delimitar tema**: realizar a delimitação do assunto proposto, especificando-o para um tema o qual facilite o desenvolvimento e aprofundamento da pesquisa (PRODANOV; FREITAS, 2013).

- **Revisar a literatura**: esta atividade tem a proposição de situar o trabalho com as pesquisas já desenvolvidas sobre o tema e identificar qual é o estado da arte. Ademais, auxilia os leitores a compreenderem sobre o assunto tratado, de forma que entendam o desenvolvimento da pesquisa (PRODANOV; FREITAS, 2013).

- **Justificar trabalho**: identificar motivos, causas, razões da relevância e contribuição do trabalho (PRODANOV; FREITAS, 2013).

- **Definir do problema de pesquisa**: trata-se da reflexão sobre um problema específico do tema. Após esta reflexão, exprime-se em uma única frase concisa a problemática do trabalho (PRODANOV; FREITAS, 2013). Define-se, também, o enfoque que o trabalho terá de forma bastante objetiva e específica (GIL, 2002).

- **Determinar os objetivos geral e específicos**: caracteriza-se por estabelecer os itens que serão estudados e executados para responder a pergunta do problema de pesquisa. É neste momento que se explicita o que será realizado no trabalho.

- **Coletar dados**: etapa realizada para obter os valores das variáveis do problema (PRODANOV; FREITAS, 2013). Na pesquisa experimental, faz-se isto manipulando certas condições para observar os efeitos no objeto de estudo. Para isto, determina-se uma amostragem diante de toda a população dos dados para a coleta (GIL, 2002).

- **Tabular e apresentar os dados**: organizar os dados, realizar cálculos, construir gráficos, elaborar tabelas e o uso de outras técnicas que facilitem o entendimento dos dados.

- **Analisar e interpretar os dados**: utiliza-se de métodos estatísticos, visualização gráfica e da literatura para entender o fenômeno em pesquisas experimentais (GIL, 2002). Esta atividade exige a capacidade analítica, descritiva e crítica do pesquisador (PRODANOV; FREITAS, 2013).

- **Concluir ou realizar considerações finais**: deixa-se claro a vinculação entre os dados e resultados obtidos, tal qual transparecer se os objetivos estabelecidos foram alcançados ou não (GIL, 2002).

- **Redigir e apresentar o trabalho**: concretizar toda a pesquisa em formato textual, de forma que fique claro todas as etapas e bibliografias consultadas. Atividade realizada em paralelo com todas as outras (PRODANOV; FREITAS, 2013).

2.2.2 Projeto de Aprendizagem de Máquina

A gerência de projetos de ML é semelhante a um projeto de Ciência de dados. Porém este é mais completo, pois envolve mais características como a de extração de dados (Mineração de dados), a implantação do sistema e a sua utilização para oferecer dados para negócios inteligentes (CHAPMAN et al., 2000).

As atividades de Ciência de dados apresentadas na gerência de projetos CRISP-DM são ilustradas na Figura 2.

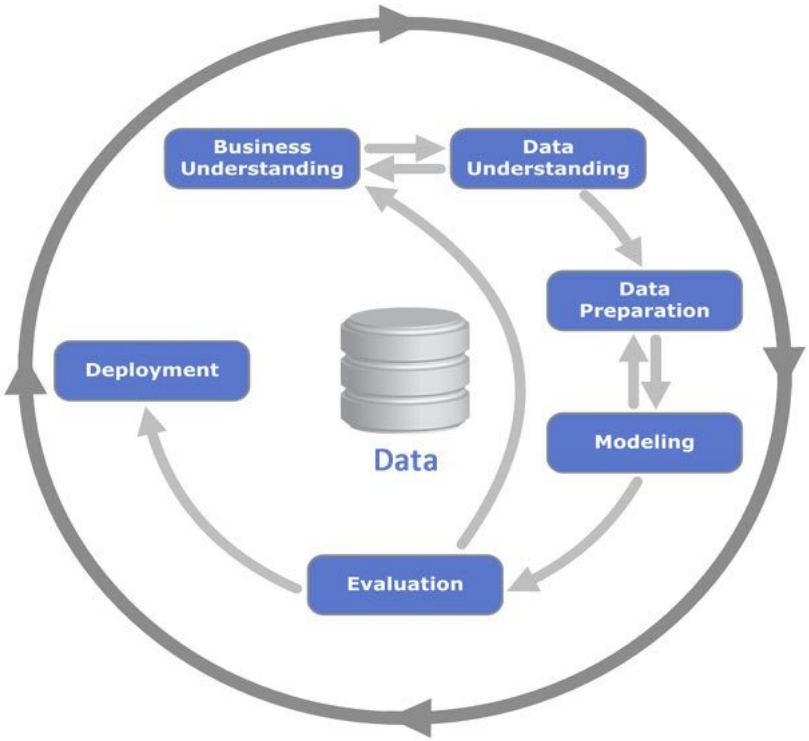


Figura 2 – Modelo de processo CRISP-DM. Fonte: Chapman et al. (2000, Página 10)

- **Entendimento do negócio**: atividade inicial que envolve o entendimento dos requisitos da organização, essa provê insumos para definir quais dados serão extraídos (CHAPMAN et al., 2000).

- **Entendimento dos dados**: explorar os dados almejando identificar suas características, problemas e possíveis informações ocultas (CHAPMAN et al., 2000).

- **Implantação**: a implantação é a ação de propor valor para a empresa ou para o consumidor através dos resultados obtidos do processo. Inclui, também, a entrega do modelo treinado (CHAPMAN et al., 2000).

A seguir serão apresentadas as atividades de um fluxo de trabalho para projetos de ML.

- **Obter dados históricos**: coletar dados históricos sobre o problema a ser resolvido, extrair as características dos dados, organizá-los estruturadamente para identificar dados faltantes e realizar um pré-processamento. Basicamente, trata-se de executar as atividades da engenharia de características, a qual é o conjunto de técnicas e ferramentas para transformação dos dados em um projeto de ML (BRINK; RICHARDS; FETHEROLF, 2015).
- **Construir um modelo**: construir ou utilizar um modelo que seja capaz de lidar bem com a natureza dos dados explorados na atividade de "Obter dados históricos". Este deve atender aos objetivos do projeto. Em problemas de classificação, o modelo deve classificar corretamente os rótulos de novas amostras (BRINK; RICHARDS; FETHEROLF, 2015).
- **Utilizar modelo**: com os dados resultantes da obtenção dos dados e da construção do modelo, faz-se uso do modelo para novos dados almejando a obtenção de respostas (BRINK; RICHARDS; FETHEROLF, 2015).
- **Validar modelo**: os modelos utilizados precisam passar pelo teste de lidar com novos dados. Através dos resultados obtidos destes testes, deve-se utilizar métricas adequadas a cada tipo de problema para identificar se o modelo teve bom resultado (BRINK; RICHARDS; FETHEROLF, 2015).
- **Otimizar o modelo**: após coletar o resultado das métricas, cabe ao desenvolvedor de um projeto de ML ter capacidade crítica, conhecimento do domínio e de técnicas para propor melhorias ao projeto. O foco nesta atividade é obter melhores resultados nas métricas. São três principais modos de melhorar: trocando os parâmetros do modelo, selecionando outro conjunto de características e/ou melhorando a engenharia de características (BRINK; RICHARDS; FETHEROLF, 2015).

2.2.3 Processo final

Em projetos de Ciência de dados que incluem ML, a melhor metodologia para o desenvolvimento é o CRISP-DM (CROWSTON; SALTZ; SHAMSHURIN, 2017). Portanto, será utilizada uma junção das atividades de um projeto de pesquisa com as de projeto de ML.

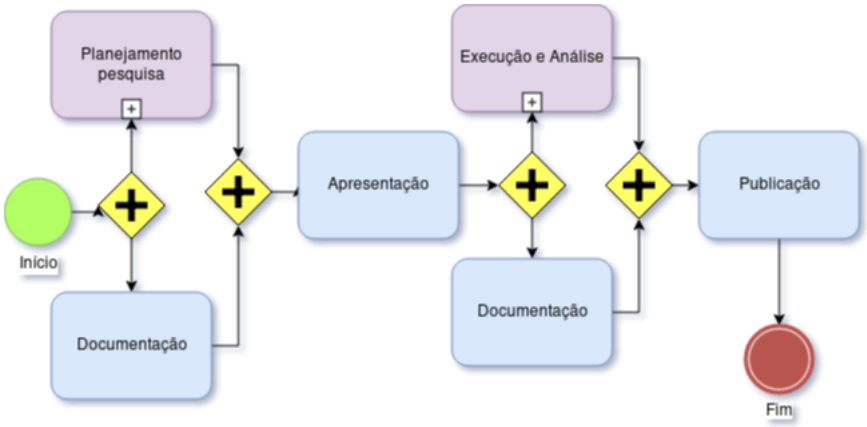


Figura 3 – Processo desenvolvimento da pesquisa. Fonte: elaboração própria

A Figura 3 traz a visão geral de como serão desenvolvidas as atividades do projeto. A tarefa de "Documentação" será executada em paralelo durante todo o projeto, exceto nas etapas de apresentação e publicação.

Das atividades da seção 2.2.1, temos que as de "Coletar dados", "Tabular e apresentar os dados", "Analisar e interpretar os dados" foram substituídas pelo fluxo de atividades de um projeto de ML. Esta substituição deve-se pela natureza do problema a ser resolvido, pois exige um fluxo diferenciado para análise dos resultados e obtenção dos dados. Essas atividades já foram definidas num projeto de ML descritas na seção 2.2.2.

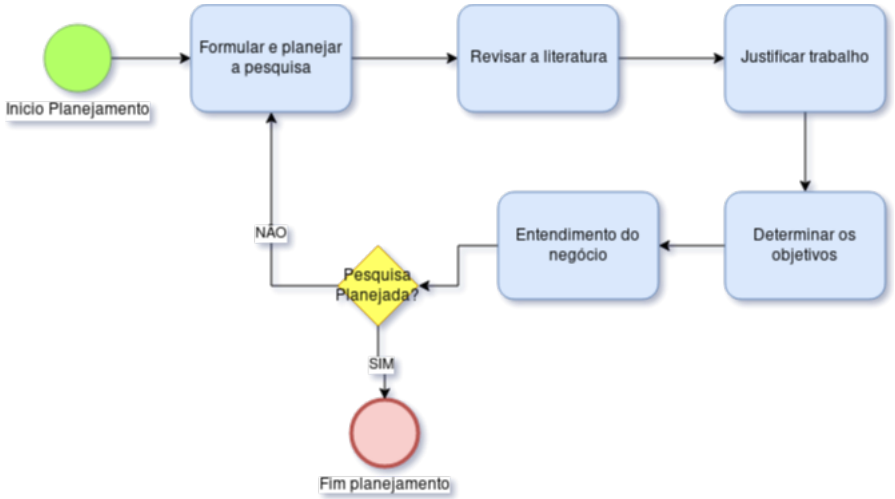


Figura 4 – Subprocesso planejamento da pesquisa. Fonte: elaboração própria

Além disso, na Figura 4, foi adicionada a atividade "Entendimento do negócio" para melhorar a compreensão do problema do ponto de vista de um trabalho de ML, o que facilitará, inclusive, o desenvolvimento da pesquisa (CROWSTON; SALTZ; SHAMSHURIN, 2017).

Na Figura 5, foi definido o fluxo de ML semelhante ao proposto por Brink, Richards e Fetherolf (2015). A única diferença entre eles são os propósitos de pesquisa, onde

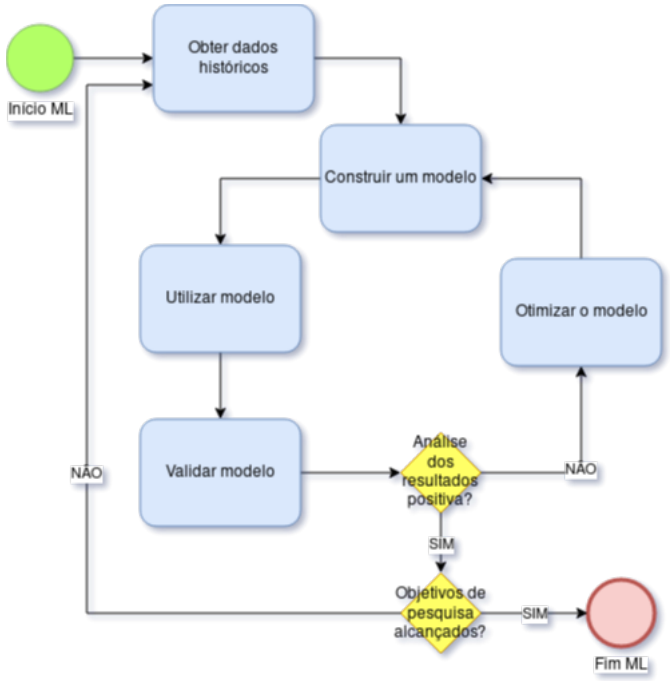


Figura 5 – Subprocesso execução e análise. Fonte: elaboração própria

adicionou-se mais uma condição: "Os objetivos da pesquisa foram alcançados?". Esta pergunta guiará o desenvolvimento do projeto, pois garante-se, mesmo com ótimos resultados no modelo, que a pesquisa tenha prosseguimento e seja possível alcançar os objetivos.

A atividade "Entendimento dos dados" foi removida do fluxo CRISP-DM, porque há uma outra atividade de "Obtenção dos dados" que possui equivalência. A outra de "Implantação" foi removida, pois ela é para projetos de Ciência de dados ou que terão um produto como entregável. A atividade equivalente a esta no trabalho científico é a de "Publicação dos resultados".

3 Referencial Teórico

Este capítulo apresentará o referencial necessário para entendimento do conteúdo técnico da solução e contexto do problema.

3.1 Classificação de documentos

Nas áreas de NLP, existem grupos de técnicas para lidar com documentos de texto, assim como em ML. Destas, serão abordados apenas os conjuntos para a classificação de corpos de textos.

3.1.1 Técnicas de pré-processamento

As técnicas de pré-processamento utilizadas para a classificação de textos são as de transformação do texto em símbolos, remoção de palavras recorrentes, a radicalização, normalização e criação de regras específicas para cada contexto utilizando de expressões regulares (OLIVEIRA; FILHO, 2017).

3.1.1.1 Transformação em símbolos

A transformação em símbolos, trata-se de criar símbolos das palavras identificadas no texto (MANNING; RAGHAVAN; SCHÜTZE, 2008). A complexidade de como transformar a sentença numa lista de símbolos vai depender do contexto. Pode-se transformar endereços de rede, recursos de *web site* em representações únicas (MANNING; RAGHAVAN; SCHÜTZE, 2008). Na Tabela 1 é apresentado a separação da sentença em símbolos.

Tabela 1 – Transformação de palavras em símbolos.

Original	**Símbolos textuais**
juiz federal relator formar incisar iii **lei nº 11419** dezembro resolução trf região março conferência autenticidade documento disponível endereçar eletrônico **www.stf.org.br**	"juiz", "federal", "relator", "formar", "incisar", "iii", **"LEI_11419"**, "dezembro", "resolução", "trf", "região", "março", "conferência", "autenticidade", "documento", "disponível", "endereçar", "eletrônico", **"SITE"**

Fonte: elaboração própria.

3.1.1.2 Remoção de palavras recorrentes

Ocorre, em documentos de texto, a repetição de algumas palavras que não trazem valor na classificação dos documentos. Essas podem ser removidas do texto, com isto

diminui-se a quantidade de dados a serem processadas sem perder propriedades estatísticas e semânticas do texto (MANNING; RAGHAVAN; SCHÜTZE, 2008).

Na Tabela 2 é apresentado a remoção das palavras: da, de, a, está.

Tabela 2 – Remoção de palavras recorrentes.

Original	**Palavras removidas**
juiz federal relator forma inciso iii **da** LEI_11419 **de de** dezembro resolução trf região março **a** conferência autenticidade documento **está** disponível endereço eletrônico SITE	juiz federal relator forma inciso iii LEI_11419 dezembro resolução trf região março conferência autenticidade documento disponível endereço eletrônico SITE

Fonte: elaboração própria.

3.1.1.3 Radicalização e Normalização

Estas técnicas auxiliam na classificação dos textos, pois ressaltam propriedades estatísticas de uma palavra no texto. A tarefa de radicalização é remover sufixos e prefixos, verificar palavras compostas e substituí-las apenas pelo seu radical. Já a normalização, é transformar palavras que tenham diferentes formas com o mesmo significado para uma única representação (SINGH; GUPTA, 2016). A Tabela 3 apresenta um exemplo de aplicação destas técnicas.

Tabela 3 – Aplicação de radicalização e normalização.

Original	**Normalizado**	**Normalizado e Radicalizado**
juiz federal relator **forma inciso** iii da LEI_11419 de de dezembro resolução trf região março a conferência autenticidade documento **está** disponível endereço eletrônico SITE	juiz federal relator **formar incisar** iii da LEI_11419 de de **dezembro resolução** trf região março o **conferência** autenticidade documento estar disponível endereçar eletrônico SITE	juiz federal relator form incis iii da lei_11419 de de dezembr resolu trf regiã marc o conferent autent document estar dispon enderec eletrôn sit

Fonte: elaboração própria.

3.1.1.4 Expressões regulares

Quando deseja-se encontrar algum padrão específico em um texto, utiliza-se de regras de formação para capturar todos os grupos que se encaixaram. Esta técnica também auxilia a realizar substituições, remover trechos do texto. Ela diminui a complexidade de código para realizar manipulações em *strings* (GOYVAERTS; LEVITHAN, 2012). A Tabela 4 mostra um exemplo da aplicação das expressões regulares presentes no Apêndice A.

Tabela 4 – Aplicação de expressões regulares.

Original	**Processado com Expressão Regular**
Juiz Federal Relator, na\nforma do artigo 1º , inciso III, da Lei 11.419, de 19 de dezembro de 2006 e Resolução TRF 4ª\nRegião nº 17, de 26 de março de 2010. A conferência da autenticidade do documento está\ndisponível no endereço eletrônico http://www.jfpr.jus.br/ gedpro/verifica/verifica.php,	juiz federal relator forma inciso iii da LEI_11419 de de dezembro resolução trf região março a conferência autenticidade document está disponível endereço eletrônico SITE

Fonte: elaboração própria.

3.1.2 Representações de textos

Quando faz-se o processamento de linguagem natural, utiliza-se diferentes formas de representação para o texto ao invés de *string*. As técnicas utilizadas são as de *one-hot-encoder*, *bag of words* (BoW) e *word embedding*.

3.1.2.1 One-hot-encoder

Esta técnica consiste em transformar cada palavra num único vetor, no qual a posição da palavra no dicionário de dados receberá o valor 1. Desta forma, uma sentença se tornará uma matriz, como no exemplo abaixo (BRINK; RICHARDS; FETHEROLF, 2015).

$$\begin{aligned} Frase = & \quad [0 \quad 1 \quad 0 \quad 0] \\ de = & \quad [1 \quad 0 \quad 0 \quad 0] \\ exemplo = & \quad [0 \quad 0 \quad 1 \quad 0] \\ simples = & \quad [0 \quad 0 \quad 0 \quad 1] \end{aligned}$$

3.1.2.2 Bag of Words - BoW

Diferentemente do *One-hot-encoder*, o BoW transforma toda a sentença em apenas um vetor. Cada posição deste, possuirá a quantidade de vezes que um símbolo apareceu. Ela permite que possam ser removidas palavras com alta ou baixa incidência (BRINK; RICHARDS; FETHEROLF, 2015).

$$\begin{aligned} Frase\ de\ exemplo\ simples = & \quad [1 \quad 1 \quad 1 \quad 1 \quad 0] \\ Frase\ de\ frase\ repetida = & \quad [2 \quad 1 \quad 1 \quad 0 \quad 1] \end{aligned}$$

3.1.3 Técnicas de ML

As principais técnicas para aprendizado de máquina para classificação de texto são Máquinas de Vetores de Suporte (do inglês *Suporte Vector Machine* SVM), K Vizinhos Próximos, Multinomial Naive Bayes.

Para este trabalho, utiliza-se apenas o modelo SVM. Isto porque serve para lidar com vetores que possuem muitas características. Além disso, seu algoritmo possibilita extrair dos vetores de suporte, os quais são as dimensões mais importantes para realizar a separabilidade dos dados (HEARST et al., 1998) e possui boa performace para classificar corpos textuais maiores (WANG; MANNING, 2012).

3.1.3.1 Máquinas de Vetores de Suporte

Este algoritmo recebe os vetores de entrada, com uma alta dimensionalidade, e faz transformações neles utilizando seus núcleos para que seja possível aplicar métodos lineares, mesmo em problemas não-lineares (HEARST et al., 1998).

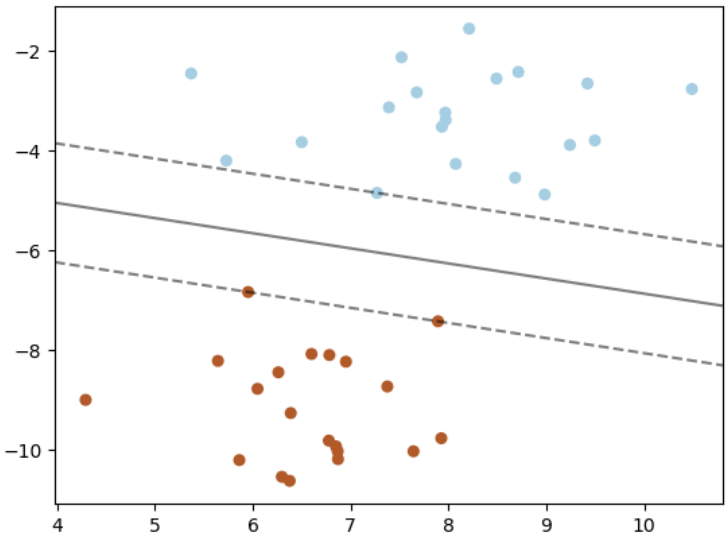


Figura 6 – Funcionamento do algoritmo SVM. Fonte: Pedregosa et al. (2011, Acesso em 10 de Junho de 2018.)

A função de classificação do SVM é baseada em hiperplanos. O algoritmo busca otimiza-los de forma que ele fique ortogonal às linhas que separam as duas classes, como é possível ver na Figura 6. Chama-se estes vetores de separação de suporte (SMOLA; SCHÖLKOPF, 2004).

$$f(x) = sign(\sum_{i=1}^{t} v_i \cdot k(x, x_i) + b) \tag{3.1}$$

A Equação 3.1 é a de classificação utilizada neste algoritmo. A função k é a que representa o núcleo, na implementação de Pedregosa et al. (2011) existe disponível o: linear, polinomial, função de base radial (do inglês *Radial Basis Function* RBF) e o de sigmoide.

O valor de x em 3.1 representa o vetor de entrada a ser classificado. Já o x_i representa os vetores de suporte. Os pesos v_i, são calculados através de uma solução quadrática, no qual são a combinação linear dos padrões de entrada (SMOLA; SCHÖLKOPF, 2004). Os valores de b, são limiares para maximização das margens entre duas classes distintas (HEARST et al., 1998).

3.1.3.2 Redes neurais

Um dos algoritmos desenvolvidos para realizar o aprendizado de máquina, foi baseado em um neurônio humano (GOLDBERG, 2017). O neurônio, chamado de perceptron possui sensores de retina (entrada), os sinais recebidos se propagam para uma área de associação, onde são combinados. O sinal propagado para as ligações a frente, se e apenas se, o valor dele for maior que um limitar θ (RASCHKA, 2015).

Como na figura 7, há pesos para os vetores de entrada, que são todos somados e, em seguida, passam pela função de ativação para gerar o resultado do perceptron (RASCHKA, 2015).

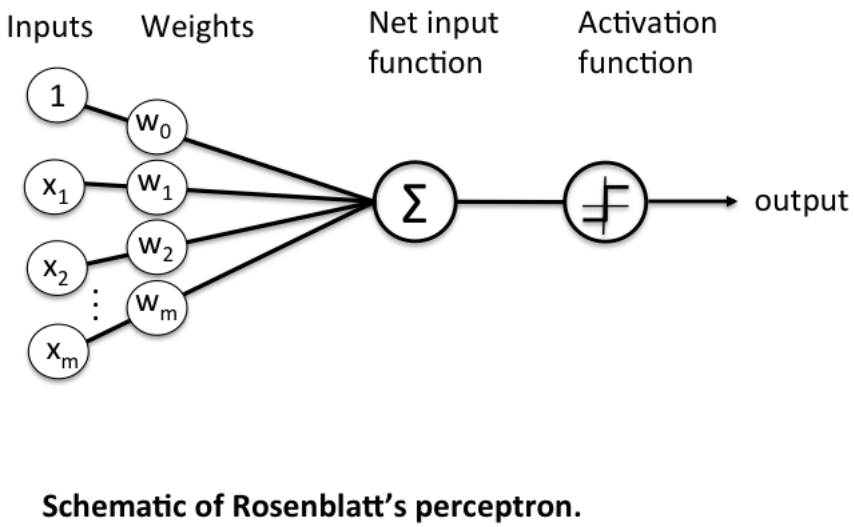


Figura 7 – Ilustração esquemática de um perceptron de Rosenblatt. Fonte: Raschka (2015, Acesso em 10 de Junho de 2018).

Agrupou-se os neurônios para criar uma rede, na qual ela possui diferentes camadas.

Entrada: esta camada não possui funções de ativação, sua função é propagar os dados de entrada para a camada seguinte.

Camadas ocultas: são as camadas que possuem funções de ativações. Uma camada oculta pode possuir vários neurônios.

Saída: esta é a ultima camada de uma rede neural, ela que irá retornar o resultado do processamento realizado pelas camadas ocultas.

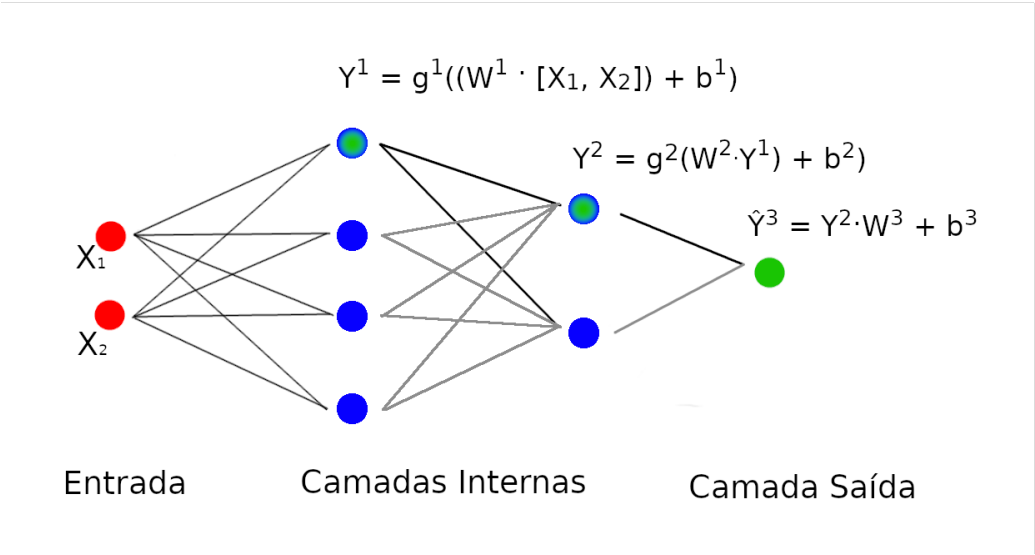


Figura 8 – Representação de uma rede neural com duas camadas internas, dois neurônios de entrada e apenas um de saída. Fonte: elaboração própria

Os neurônios com a coloração verde na figura 8, indicam que apenas estes foram ativados durante o processamento.

O fluxo comum da execução de uma rede neural é o recebimento dos dados de entrada na primeira camada, em seguida há a propagação dos valores para uma camada oculta, após o cálculo de ativação, os dados são propagados novamente para a próxima camada, até que ao final, chega-se na camada de saída (GOLDBERG, 2017).

O processo de propagação (do inglês *feed-forward propagation*) é representado na Figura 8, no qual há propagação dos valores com o produto entre o vetor x e W (GOLDBERG, 2017). A representação de vetores da computação para os pesos são feitas de forma que o neurônio da camada seguinte i tem o peso associado a posição j ($W[i][j]$) (NIELSEN, 2015).

Para modelos com mais camadas do que foi ilustrado na Figura 8, pode-se generalizar a equação lá apresentada. Em cada camada i, o vetor W^i possui dimensão $d^i_{in} \times d^i_{out}$ dimensões. Os vetores de viés b^i terão $1 \times d^i_{out}$. O valor de N é igual ao número de camadas na rede. Na Equação 3.2, cada camada da rede neural poderá ter sua própria função de ativação (GOLDBERG, 2017).

$$x^i = \begin{cases} x & \text{se for camada de entrada} \\ \hat{y}^{i-1} & \text{caso contrário} \end{cases}$$

$$y^i = g^i(x^i W^i + b^i) \tag{3.2}$$
$$\hat{y} = y^{N-1} W^N$$

Para ajustar os valores dos pesos e vetor de viés W^i e b^i, calcula-se o quanto os neurônios estão corretos em relação ao distanciamento entre o $\hat{y}$ e y. Utiliza-se da função de erro $L(\hat{y}, y)$ (GOLDBERG, 2017), para calcular o erro na camada de saída. Este deve ser propagado para a camada anterior, com isto, todas as camadas vão ajustando recursivamente seus pesos e vetores de viés (NIELSEN, 2015). Este processo chama-se de retro-propagação (do inglês *back-propagation*) apresentado na Figura 9.

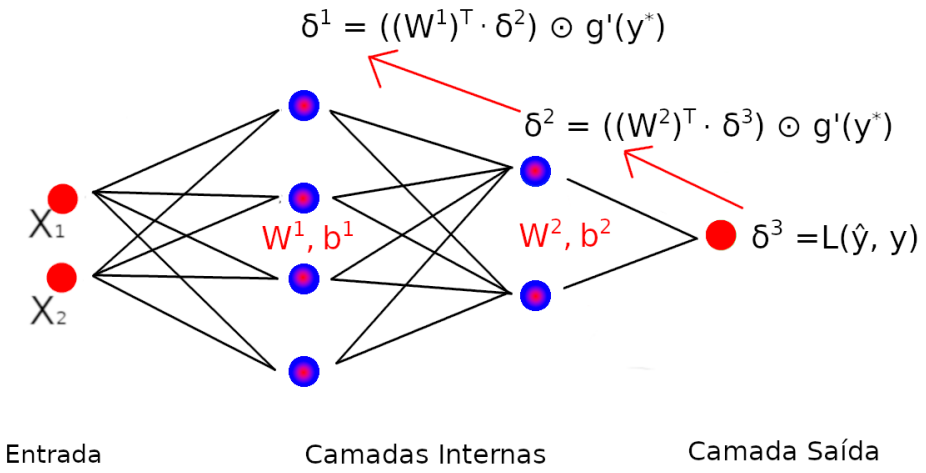


Figura 9 – Simplificação da retro-propagação em Rede Neural. Fonte: elaboração própria

Na Figura 9, o valor de g' é a derivada para a função de ativação, executada no valor de y^*, o qual é o resultado da operação de $x^i W^i + b^i$ sem aplicar a função de ativação g (NIELSEN, 2015).

A operação de $\odot$ é o produto de Hadamard, ela representa multiplicar os elementos de cada posição de um vetor a_n pelo do valor de b_n resultando em um outro vetor c_n (NIELSEN, 2015). A Equação 3.3 ilustra essa operação entre os vetores.

$$\begin{bmatrix} 3 \\ 7 \end{bmatrix} \odot \begin{bmatrix} 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 3 & * & 4 \\ 7 & * & 9 \end{bmatrix} = \begin{bmatrix} 12 \\ 63 \end{bmatrix} \tag{3.3}$$

Com a função de δ^i, faz-se a atualização dos pesos e vetores de viés. A estrutura é recursiva, de forma que, para calcular o δ^i, é necessário ter calculado o valor de δ^{i+1}. A prova matemática para a retro-propagação e a definição da função $L(\hat{y}, y)$ encontra-se em Nielsen (2015).

A retro-propagação é utilizada juntamente com uma função de gradiente. Com esta combinação, faz-se os ajustes nos pesos das camadas, de forma que, a cada nova predição de $\hat{y}$, é feita uma retro-propagação calculando os valores mínimos para a função $L(y, \hat{y})$ (NIELSEN, 2015).

Com as técnicas apresentadas, é possível implementar uma rede neral Multicamadas de Perceptron (do inglês *Multilayer Perceptron* - MLP). Existem outros tipos de arquiteturas, como as Redes Neurais Recorrentes (RNN), as Redes Neurais Convolucionais (CNN). Elas podem ser utilizadas como camadas internas de uma MLP, bem como podem existir camadas de mescla (do inglês *merge*) e de junção (do inglês *pooling*) (GOLDBERG, 2017).

3.1.4 Técnicas de Aprendizado Profundo

O aprendizado de máquina profundo é quando utiliza-se várias camadas ocultas em uma rede neural para realizar o aprendizado. Não é restrito ao uso de apenas de um tipo de camada ou função de ativação para a construção de uma rede profunda.

3.1.4.1 Redes convolucionais

Este tipo de arquitetura, baseia-se na aplicação da operação de convolução nos dados de entrada e, em seguida, realizar junções. Uma CNN é definida como extratora de características de dados textuais por realizar as etapas de convolução e junção repetidas vezes (GOLDBERG, 2017).

O uso dessa rede não se limita apenas a extração de características, pode ser utilizada também, sozinha, para classificar textos utilizando abordagem de rede neural muito profunda com uma última camada de classificação linear (CONNEAU et al., 2017).

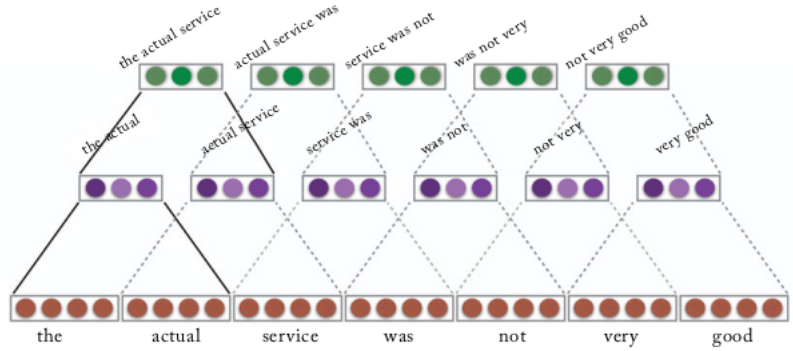


Figura 10 – Aplicação de CNN em texto. Fonte: Goldberg (2017, Página 160)

Considera-se uma sentença $x_{1:n} = x_1, x_2, ..., x_n$ e um filtro w que pode ser aplicado a intervalo de h palavras. Através da operação de convolução é possível gerar uma nova

característica, dada pela aplicação de uma função de ativação g (KIM, 2014):

$$c_i = g(w \cdot x_{i:i+h-1} + b) \tag{3.4}$$

Na Figura 10, é apresentado graficamente o processo da aplicação de filtros de convolução, seguidos da operação de junção em 2 camadas. A partir da Equação 3.4, gera-se um novo conjunto de características extraídas do texto. Após isso, utiliza-se de uma camada de *max pooling* para obter o maior valor daquele novo sub conjunto (KIM, 2014).

$$c = [c_1, c_2, ..., c_{n-h+1}]$$

$$\hat{c} = \begin{cases} maior(c) \\ media(c) \\ k - maiores(c) \end{cases} \tag{3.5}$$

O resultado de $\hat{c}$, é o maior valor significativo para o filtro w aplicado sobre uma sentença entrada x. Em uma rede convolucional, tem-se vários filtros que possibilitam processar (KIM, 2014) e identificar os n-gramas mais importantes (GOLDBERG, 2017) dos corpos textuais.

3.1.4.2 Redes recorrentes

As redes neurais recorrentes possibilitam capturar características sequenciais dos dados. Sendo capaz de receber um vetor de entrada com tamanho variado e entregar um tamanho fixo na camada de saída (GOLDBERG, 2017).

Dado um vetor $x_{1:n} = x_1, x_2, ..., x_n$ de entrada na $f(x_{i:n})$, será retornado um vetor $\hat{y}_{1:m} = \hat{y}_1, \hat{y}_2, ..., \hat{y}_m$, com o valor de m fixado. Há um vetor de estados s que carrega a informação do processamento dos neurônios anteriores. Uma rede neural recorrente é recursiva, pois o neurônio i, realizará o seguinte processamento $s_i = RNN_i(x_i, s_{i-1})$ (MIKOLOV et al., 2010).

Na figura 11, as funções de R, O em cada perceptron são, respectivamente, as funções f e g da equação 3.6. O θ, são os valores de hiperparametrização da rede.

Para utilizar a RNN como método de classificação, deve-se aplicar uma função de ativação g. Na equação 3.6, tem-se também a função f, que é uma função de ativação responsável por juntar o vetor de estado s_i com o vetor de entrada x_i (MIKOLOV et al.,

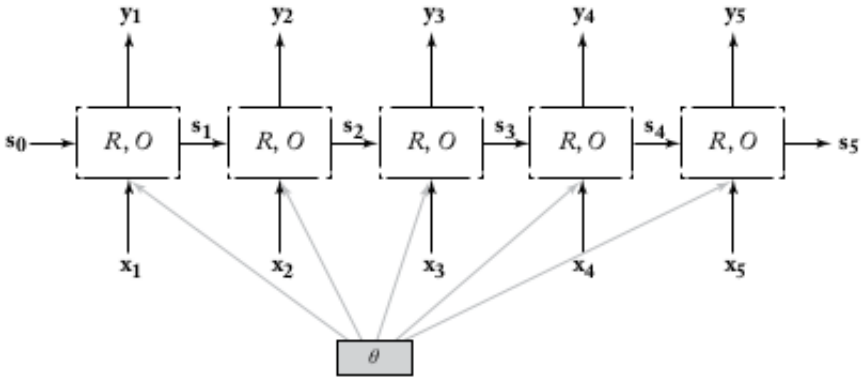


Figura 11 – Arquitetura recursiva da RNN. Fonte: Goldberg (2017, Página 166)

2010).

$$s_i = \begin{cases} i = 0 & f(S, x_i) \\ i \neq 0 & f(s_{i-1}, x_i) \end{cases} \qquad \hat{y}_i = g(s_i) \tag{3.6}$$

Na Equação 3.6, o valor de S representa um hiper-parâmetro para a rede neural. Este é iniciado com um vetor de zeros $[0, 0, ..., 0]$.

3.1.5 Métodos de avaliação

A seguir serão apresentados o método de validação cruzada para modelos de classificação e as métricas para determinar o desempenho.

3.1.5.1 Validação cruzada

Para realizar a validação do modelo de classificação, há o modelo de validação cruzada. Este consiste em dividir o conjunto de dados em validação (30%) e treino (70%). A Figura 12 apresenta o processo para realizar a validação de um modelo classificador (BRINK; RICHARDS; FETHEROLF, 2015).

De acordo com BRINK; RICHARDS; FETHEROLF (2015), os passos para a validação são:

1. Realizar a divisão dos dados em conjunto de treino e teste.
2. Utilizar o conjunto de treino para a construção do modelo.
3. Utilizar o modelo para predizer o conjunto de dados de teste.
4. Comparar os resultados da predição com os rótulos da base de teste e gerar as métricas.

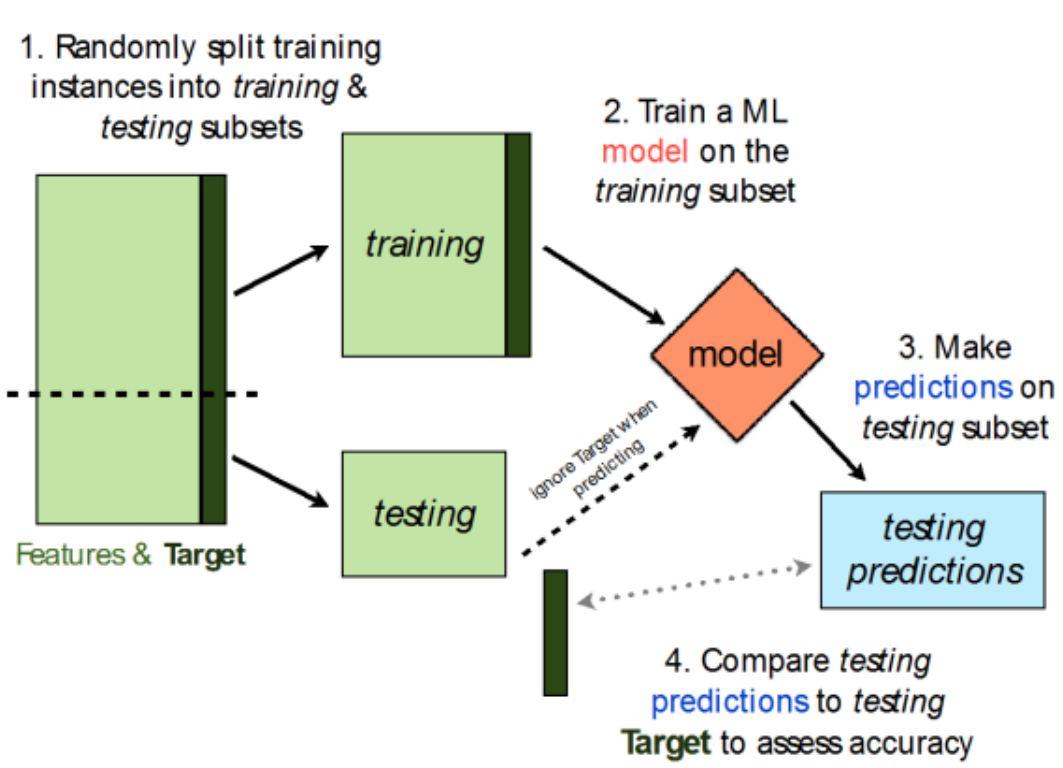


Figura 12 – Validação cruzada de modelos. Fonte: Brink, Richards e Fetherolf (2015, Página 82)

3.1.5.2 Métricas

Em modelos de classificação, é possível determinar com exatidão o resultado esperado em categorias. Com isso, é possível determinar o Verdadeiro Positivo, Verdadeiro Negativo que é quando o modelo acerta a predição. Para quando ele erra, há o Falso Positivo e Falso Negativo (BRINK; RICHARDS; FETHEROLF, 2015).

A Tabela 5 mostra a relação de Verdadeiros/Falsos Positivos/Negativos quando há mais de uma classe.

Tabela 5 – Matriz de confusão

Classe (N)	A	B	C	
A	VP_A	eB_A	eC_A	T_A
B	eA_B	VP_B	eC	T_B
C	eA_C	eB_C	VP_C	T_C
	P_A	P_B	P_C	$T = T_A + T_B + T_C$ $= P_A + P_B + P_C$

Fonte: elaboração própria.

Os valores de eA_N, eB_N e eC_N representam erros na predição, enquanto que os de VP_A, VP_B e VP_C, os acertos. Ao somar-se os valores da coluna A na Tabela 5, tem-se o resultado do $VP_A + eA_B + eA_C$. Isso indica o quanto o modelo está predizendo a classe A.

Ao mesmo tempo que somando-se a linha A, tem-se $VP_A + eB_A + eC_A$, a qual representa o total de valores de A.

Com esta tabela, é possível obter valores numéricos para as métricas de acurácia (ou eficiência), revocação (ou sensibilidade), taxa de verdadeiros negativos, taxa de falsos positivos e negativos, precisão. Além de outras métricas que utilizam estas para seus cálculos (RODRÍGUEZ; CASTAÑO; SAMBLÁS, 2016).

As métricas a seguir serão utilizadas a fim de validar a qualidade e evolução de performance dos modelos. Todas as métricas a seguir são descritas por RODRÍGUEZ; CASTAÑO; SAMBLÁS (2016):

Revocação: trata-se da taxa de acertos para a classe N. A fórmula é: $R = \frac{VP_N}{T_N}$.

Precisão: também conhecido como "acurácia da classe N", representa a proporção de VP para os valores preditos em N. $P = \frac{VP_N}{P_N}$

Taxa geral de acerto: também conhecido como "acurácia do modelo", representa o quanto ele acerta de todas as classes. $G = \frac{\sum VP_N}{T}$

A revocação e precisão são para analisar pontualmente o comportamento do modelo com cada classe. Enquanto que a taxa geral de acerto irá proporcionar a informação da capacidade do modelo em aprender como separar as peças. Além disso, a matriz de confusão (Tabela 5) será utilizada para verificar quais classes são mais confundidas pelo modelo.

3.2 Sistema Jurídico Brasileiro

A estrutura do sistema jurídico brasileiro é estabelecida pela Constituição Federal de 1988. Ela prevê quais são os órgãos que compõem o Poder Judiciário, as competências que eles possuem, onde estão localizados, o tamanho dos representantes das instâncias superiores e como se faz a investidura aos cargos (BRASIL, 1988). Além da Constituição, para um estudo mais completo, são necessárias as Lei n.º 8.457/1992, que estabelece a primeira instância da Justiça Militar (BRASIL, 1992) e a n.º 12.665/2012, a qual define as Turmas Recursais para segunda instância de Juizados Especiais (BRASIL, 2012).

Para ter um entendimento completo sobre as peças jurídicas e suas origens, é preciso ter conhecimento sobre como estão organizados os órgãos do Judiciário, pois são através deles que a população vivencia um processo (AMENDOEIRA JR, 2012).

Para organizar melhor o trabalho do sistema judiciário, separou-se a justiça comum em algumas áreas para lidar com temas específicos: os assuntos trabalhistas, os eleitorais e uma especialização para lidar com as Leis Militares (AMENDOEIRA JR, 2012).

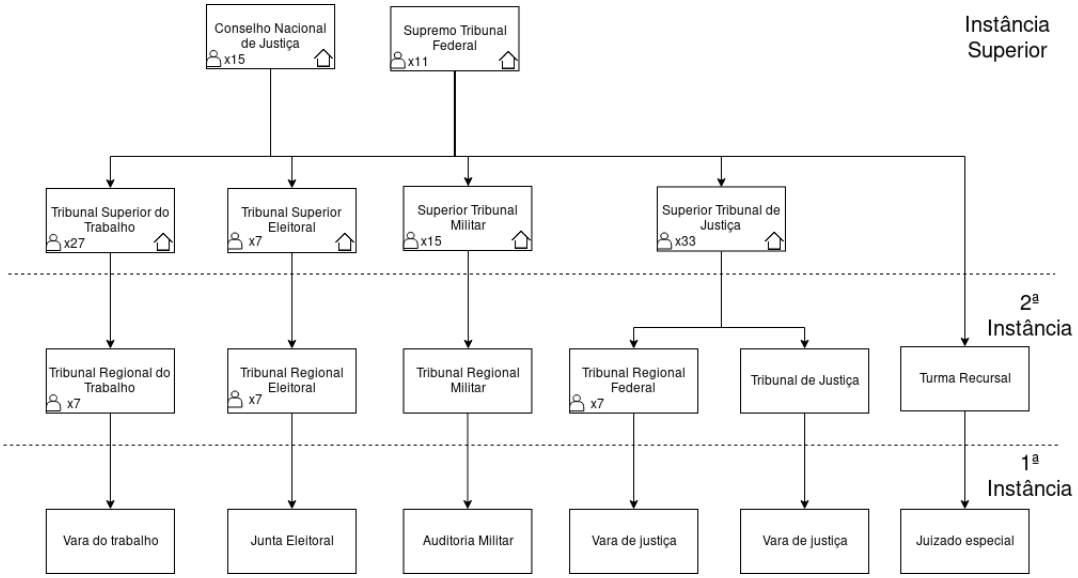


Figura 13 – Organização do sistema jurídico brasileiro. Fonte: elaboração própria

3.2.1 Instância superior - Tribunais Superiores e STF

Como apresentado na Figura 13, são tribunais superiores Supremo Tribunal Federal, Superior Tribunal de Justiça (STJ), Tribunal Superior Eleitoral (TSE), Tribunal Superior do Trabalho (TST) e Superior Tribunal Militar (STM) (BRASIL, 1988).

O STF e o STJ não se encaixam em nenhuma das justiças comum ou especializadas. A função primordial do STF é o controle de constitucionalidade das Leis Federais em conformidade com a CF (1988) e a do STJ é o controle de legalidade das Leis do Brasil em conformidade com as Leis Federais (BRASIL, 1988). Eles não são chamados de terceira instância devido ao princípio da dupla jurisdição. A correta denominação é instância superior, pois em Recurso Extraordinário, julgam as teses jurídicas e não os fatos do processo (AMENDOEIRA JR, 2012).

Os tribunais TST, TSE e STM compõe também a instância superior, mas estes julgam apenas os recursos de sua área de especialização na justiça (BRASIL, 1988).

Todos possuem sua sede na capital do Brasil e o número mínimo de representantes está estabelecido diretamente na CF (1988). Na Figura 13, são representados estes valores para cada um deles. Estes Tribunais também possuem a competência para estabelecer o número de servidores nas instâncias inferiores (BRASIL, 1988).

3.2.2 2ª instância - Tribunais

Além dos Tribunais de segunda instância da justiça especial: o Tribunal Regional Eleitoral, o Tribunal Regional do Trabalho e o Tribunal Regional Militar, tem-se os Tribunais que compõe a justiça comum o Tribunal Regional Federal e os Tribunais Estaduais de Justiça (BRASIL, 1988). Há também as Turmas Recursais que representam a Segunda

Instância para os juizados especiais (BRASIL, 2012).

Nestes Tribunais, julgam-se os Recursos Ordinários, as provas, as evidências, diferentemente dos Tribunais de instância superior (AMENDOEIRA JR, 2012).

3.2.3 1ª instância

Para se entender a organização da justiça de primeira instância, é preciso definir como estão estruturadas a separação da jurisdição dos Tribunais.

Cada Estado e o Distrito Federal dividem-se em comarcas (ou foros), no qual cada um destes possue uma ou mais varas especializadas para as justiças comum e do trabalho (AMENDOEIRA JR, 2012). Para a Justiça Militar, o território nacional está separado em circunscrições, onde cada uma possui sua Auditoria Militar (BRASIL, 1992). Já para a Justiça Eleitoral, existem juntas eleitorais que subdividem os Estados e o Distrito Federal em zonas distintas (BRASIL, 1988).

A seguir são apresentadas as definições:

Circunscrição é uma divisão geográfica administrativa para restringir a atuação de um tribunal (GUIMARÃES, 2012, p. 71).

Comarca é uma circunscrição sob jurisdição de juízes, na qual o território esta subdividido (GUIMARÃES, 2012, p. 75).

Vara é uma repartição judiciária com seu domínio a cargo de um juiz (GUIMARÃES, 2012, p. 259).

3.3 Código Processual Civil (CPC)

O CPC (Lei n.º 13.105/2015) é a lei que define como devem ser estruturados os processos civis. Ele é o meio pelo qual o juiz concretiza as leis, considerando os interesses entre as partes envolvidas.

O processo civil possui duas categorias de procedimentos adotados: o comum e especial. Neste trabalho tratar-se-á apenas do comum. A seguir, será apresentado para o entendimento das peças jurídicas avaliadas pelo STF, suas características em meio ao processo civil.

3.3.1 1ª instância

O processo civil dá-se início por meio de uma peça chama de Petição Inicial. E através deles, a parte faz seus pedidos ao juiz e identifica quem é a outra parte, que por

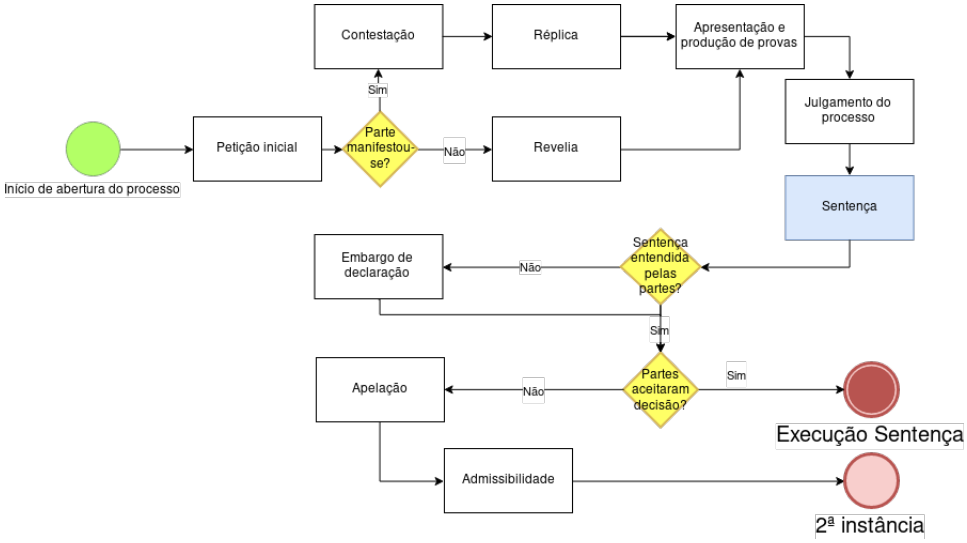


Figura 14 – Processo judiciário - 1ª Instância. Fonte: elaboração própria

sua vez, possui o direito de se manifestar ou não (BRASIL, 2015). Chama-se esta fase de postulatória (GONÇALVES, 2016).

A etapa seguinte é a ordinatória, caracterizada pelo direito de réplica. Depois disso, inicia-se a fase instrutória caracterizada pela produção de provas (GONÇALVES, 2016).

A etapa decisória tem o ato da sentença (GONÇALVES, 2016). Deste, é gerada a peça que descreve a decisão tomada pelo juiz mediante os fatos apresentados pelas partes. Uma Sentença é inalterável pelo próprio juiz que a proferiu, poderá apenas ser esclarecida pelo recurso de Embargo de Declaração (BRASIL, 2015).

Após a sentença, as partes têm o direito garantido pelo duplo grau de jurisdição de realizar a apelação. O processo será enviado a um órgão segunda instância caso esteja em conformidade com a lei (GONÇALVES, 2016).

O Artigo n.º 489 do CPC (Lei n.º 13.105/2015) define que existem em todas as sentenças três elementos fundamentais:

> I - o relatório, que conterá os nomes das partes, a identificação do caso, com a suma do pedido e da contestação, e o registro das principais ocorrências havidas no andamento do processo;
>
> II - os fundamentos, em que o juiz analisará as questões de fato e de direito;
>
> III - o dispositivo, em que o juiz resolverá as questões principais que as partes lhe submeterem. Brasil (2015)

3.3.2 2ª Instância

Garantido pelo princípio da dupla jurisdição, os processos que cumprem com os requisitos formais de uma apelação são direcionados ao Tribunal de segunda instância

competente para sua análise. Um Acórdão é o resultado do voto de três ou mais desembargadores, a decisão que mais receber votos é a final, portanto deve ser ímpar o número de juízes na sessão.

Após essa fase decisória, há o momento para as partes ou o Ministério Público elaborarem os recursos. Na Figura 15, são as atividades do fluxo, após o procedimento de primeira instância.

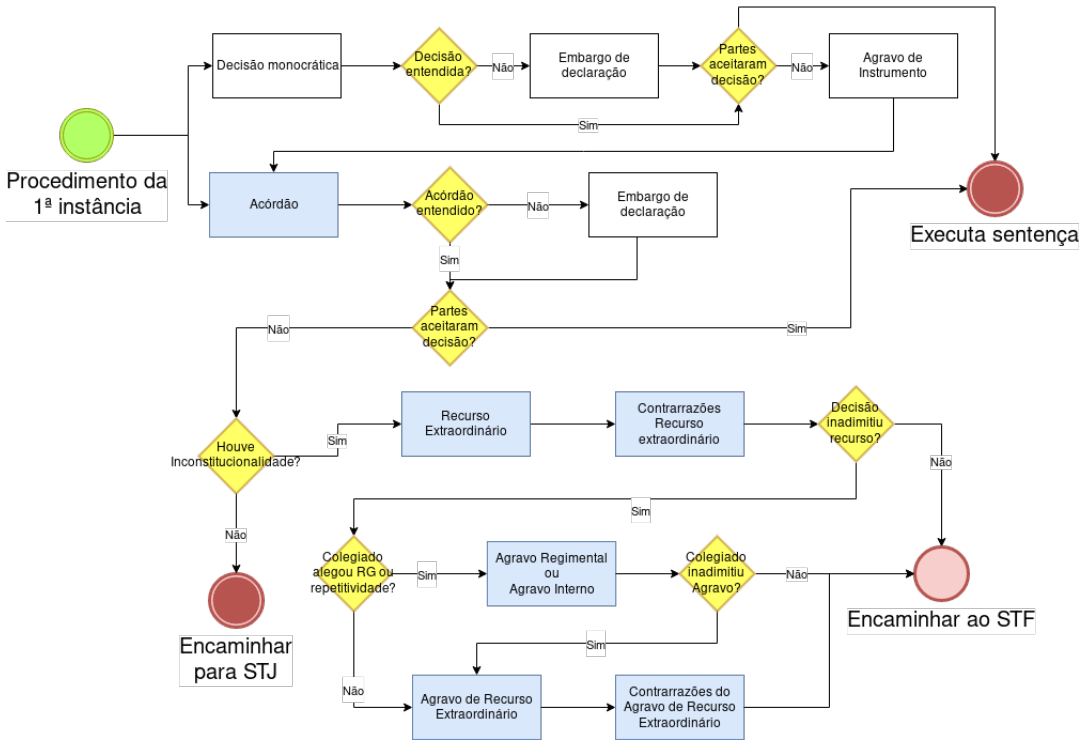


Figura 15 – Processo judiciário - 2ª Instância. Fonte: elaboração própria

O Agravo Interno ou Regimental, deve ter em seu conteúdo apenas a informação sobre a questão agravada e encaminhado diretamente ao tribunal competente.

Para Recurso Extraordinário (RE), este deve ser apresentado ao Presidente ou Vice-Presidente do Tribunal em que foi recorrida a decisão. O RE deve conter as informações sobre as razões do pedido, a exposição do quê ocorreu associado a regra do direito e discorrer sobre a validade da interposição do recurso. Quando houver recursos simultâneos ao STJ e STF, o processo deve, primeiramente, ser encaminhado ao STJ (BRASIL, 2015).

As características procedurais de um Agravo em Recurso Extraordinário (ARE) são iguais às de um RE. A distinção entre ambas as peças são seus conteúdos, no qual o RE será relacionado a alguma inconstitucionalidade e o ARE será um contraponto à inadmissão do recurso.

3.3.3 Instância superior e o Supremo Tribunal Federal

A decisão proferida pelos ministros do STF sobre a Repercussão Geral (RG) são irrecorríveis. Caso haja RG, ou seja, ele cumpre os requisitos do § 1º do Artigo n.º 1.035 do CPC (Lei n.º 13.105/2015) "Para efeito de repercussão geral, será considerada a existência ou não de questões relevantes do ponto de vista econômico, político, social ou jurídico que ultrapassem os interesses subjetivos do processo" - Brasil (2015) o processo é distribuído para que os Ministros do Tribunal possam julgar a questão do RE.

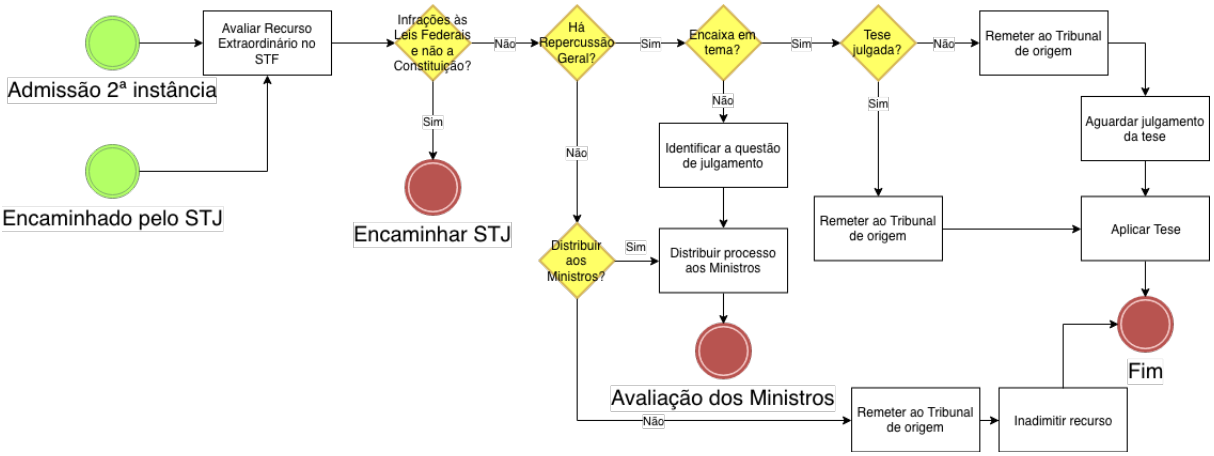


Figura 16 – Processo judiciário - Instância Superior e STF. Fonte: elaboração própria

Na figura 16, a atividade de Avaliação dos Ministros é composta por um outro conjunto de atividades descritas no Regimento Interno do STF (BRASIL, 2016) e na Constituição Federal (1988).

Existem outras atividades desenvolvidas pelo STF que não foram apresentadas na Figura 16. Estas atividades estão relacionadas ao Regimento Interno (BRASIL, 2016) deste Tribunal e o que eles fazem com as peças. Ou seja, não há produção de novos documentos internamente os quais sejam analisados para definir se há ou não Repercussão Geral.

4 Os dados

Neste capítulo serão tratadas as características dos dados usados neste trabalho.

4.1 Seleção dos dados

Os dados foram processados e extraídos no desenvolvimento do projeto Victor[1]. Dessa forma, eles são os mesmos que apresentados por Silva et al. (2018) e Braz et al. (2018). Além disso, há confiabilidade nos rótulos dos documentos processados, pois estes foram rotulados por especialistas em direito.

As 5 categorias de documentos Agravo de Recurso Extraordinário, Acordão, Despacho de Admissibilidade, Recurso Extraordinário e Sentença foram escolhidas por serem as peças avaliadas pelo STF para definir se um processo se encaixa em algum tema. Os pricipais motivos obtidos através de conversas informais com representantes do STF foram:

- Acordãos possuem uma síntese de alto nível da tese jurídica tratada no processo, que é de extrema importância para avaliação do STF.
- Sentença é o documento de menor peso, avaliado apenas em alguns casos específicos nos quais não foi possível determinar o tema com as outras 4 peças e busca-se por mais informações da tese jurídica tratada no processo.
- Despacho de Admissibilidade é utilizado principalmente para averiguar se o processo está em acordo com as regras de submissão do CPC (Lei n.º 13.105/2015).
- Recurso Extraordinário é a principal peça avaliada, pois ela é o artefato gerado para ser encaminhado ao STF. Neste artefato, tem-se os motivos pelos quais recorreu-se ao julgamento de segunda instância, incluindo o(s) artigo(s) da CF 1988 que foram infringidos.
- Agravo de Recurso Extraordinário é utilizando quando presente no processo, pelo mesmo motivo que o RE.
- Outro é a categoria para agrupar o restante das peças que não se encaixam em nenhuma das anteriores.

[1] Victor é o Projeto de Pesquisa & Desenvolvimento de aprendizado de máquina (machine learning) sobre dados judiciais das repercussões gerais do Supremo Tribunal Federal - STF.

4.2 Extração dos textos

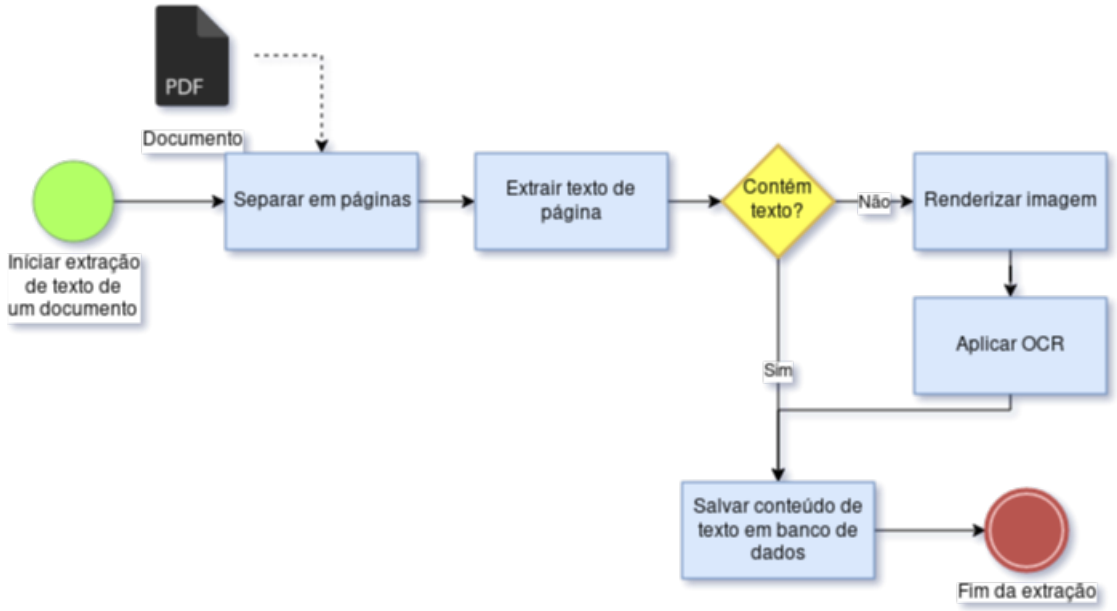


Figura 17 – Procedimento para extração de textos. São aceitos volumes ou peças separadas. O texto extraído é salvo em uma base de dados. Fonte: Silva et al. (2018, p. 2)

O STF disponibilizou os processos e suas peças no formato PDF. Nos arquivos recebidos há volumes e peças já separadas.

As etapas necessárias para realizar a extração de documentos estão representadas na Figura 17, onde utilizou-se da ferramenta XpdfReader[2] para extrair o texto das páginas de um PDF que estava em formato digital ou digitalizado com uma camada de texto.

Quando encontrada uma página sem texto, transformou-se esta em uma imagem para, em seguida, aplicar um reconhecedor ótico de caracteres (do inglês *Optical Character Recognition* OCR).

4.3 Tratamento dos textos

Utilizou-se apenas parte dos dados para realizar a análise. Fez-se a limpeza utilizando expressões regulares (GOYVAERTS; LEVITHAN, 2012) a fim de:

- Remover caracteres especiais, tais como: # @ $\backslash n$ 0xCE;
- Remover números misturados com letras;
- Remover números;

[2] Executou-se os comandos **pdftotext** e **pdftopng**. Disponível em: <https://www.xpdfreader.com/pdftotext-man.html>. Acesso em: 17-06-2018.

- Capturar leis, artigos e decretos;
- Remover espaços em branco;
- Remover e-mail e *link*.

Em seguida, aplicou-se as técnicas de radicalização, normalização (SINGH; GUPTA, 2016) e remoção de palavras recorrentes (MANNING; RAGHAVAN; SCHÜTZE, 2008). Ao analisar os dados, detectou-se a presença de nomes próprios, símbolos e sequências de caracteres sem sentido para a língua portuguesa, *latim* e para o contexto Jurídico Brasileiro.

Ao processar os dados, fez-se uma restrição no tamanho do vocabulário adquirido pelo modelo de transformação do texto ao invés de utilizar palavras pré-determinadas. isto devido à variabilidade de palavras com erros de OCR ser alta. Utilizou-se as palavras contidas nos próprios documentos pois elas podem ser representativas para este conjunto de texto, além de que os mesmos erros podem aparecer em outros documentos que sofreram o mesmo tratameno de digitalização.

Diferentemente do modelo de tratamento de dados para este contexto (SILVA et al., 2018), não foi aplicado a normalização das palavras, ficando elas no formato final como apresentado na Tabela 1.

4.4 Características dos dados

Como abordado na Seção 1.1, os rótulos das peças separadas dos volumes advindos do STF não são confiáveis. Por conta disso, houve um novo rotulamento destes documentos, classificando-os apenas em 5 categorias e Outros, apresentados na Tabela 6.

A quantidade de documentos obtidos são apresentados na Tabela 6. Nela, percebe-se que as peças de Agravo, Sentença, Petição de Agravo, Despacho de Agravo e Recurso Extraordinário não estão uniformemente distribuídas.

Tabela 6 – Amostras do conjunto de dados, contagem disponível em cada conjunto: Treino, Validação e Teste e seus respectivos totais nos conjuntos.

Tipo	**Treino**	**Validação**	**Teste**	**Total**
Agravo de Recurso Extraordinário	640	183	92	915
Acordão	573	164	82	819
Despacho	388	111	55	554
Outro	1961	561	280	2802
Recurso Extraordinário	440	125	63	628
Sentença	767	219	110	1096
Total	4769	1363	682	**6814**

Fonte: Silva et al. (2018, p. 2)

Após o tratamento dos dados, fez-se a remoção de documentos com o mesmo conteúdo que tivessem classificações diferentes. As categorias que se confundiram foram as tuplas Outros e Acórdão, Agravo de Recurso Extraordinário e Recurso Extraordinário, Despacho e Outros, Acórdão e Sentença, e por último Petição de Agravo e Outros.

Foram feitos quatro tipos de processamentos nos dados coletados: a transformação em símbolos e o *Bag Of Words*, com os dados crus e também pré-processamento. As análises dos documentos para o dado com a simples transformação em símbolos são apresentados na Tabela 7.

Tabela 7 – Métricas dos documentos.

Nome	**Sem pré-processamento**	**Pré-processado**
Menor número de símbolos	10	20
Maior número de símbolos	945	790
Média de símbolos por documento	262,88 ± 116,93	145.83 ± 75.50
Total de símbolos	1.253.677	695.485
Tamanho do vocabulário	42.870	18.670

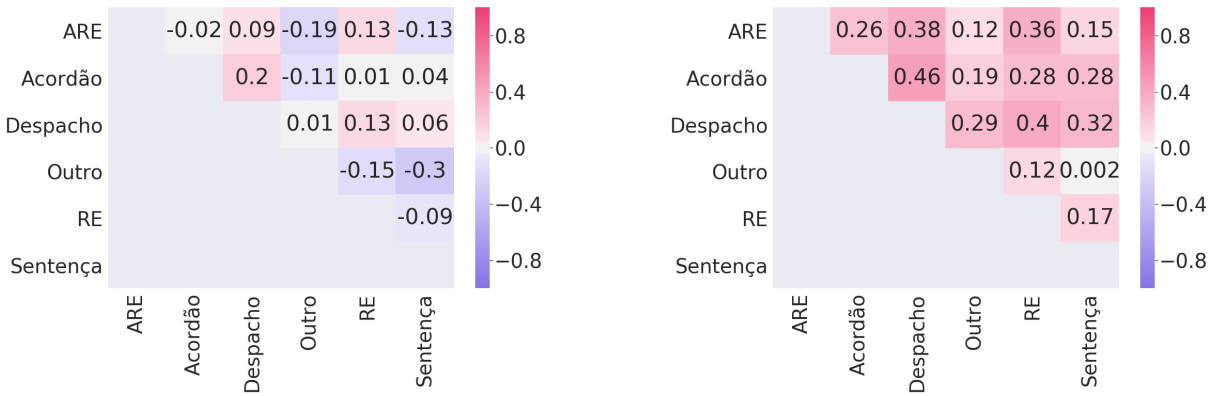
Fonte: elaboração própria.

Antes de treinar um modelo classificador, fez-se uma correlação entre as classes com o **BoW relativo**. Ou seja, o corpo de texto de cada tipo de documento foi concatenado, logo após, a contagem de palavras em cada tipo foi dividido pelo total de ocorrências dela em todos os documentos. Na Equação 4.1, o i representa o tipo de documento e o j representa a palavra pertencente ao dicionário. O t representa todo o corpo textual.

$$BoWrelative_{i,j} = \frac{BoW_{i,j}}{BoWt_j} \times 100 \tag{4.1}$$

A partir desse resultado, fez-se a correlação de *spearman* para gerar uma matriz triangular. Os valores para o coeficiente de *spearman* têm os intervalos entre $[-1, 1]$, sendo a diagonal principal igual a 1. Os dados observados na Figura 18 mostram que não há correlação entre os documentos, na qual o maior valor presente é a correlação entre Acórdão e a peça Despacho.

[3] O Grupo de Pesquisa em Aprendizado de Máquina (GPAM) <http://gpam.unb.br/> está envolvido na tarefa de classificar peças para o STF. Esta imagem foi gerada para a exploração de dados realizada no projeto. Elaborado pelos membros: Davi Alves Bezerra e Davi Benevides Gusmão.



(a) Correlação do texto sem pré-processamento

(b) Correlação do texto com pré-processamento

Figura 18 – Mapa de calor para correlação entre os tipos peças. Fonte: GPAM [3]

5 Modelos e resultados

Neste capítulo, serão apresentados os modelos neurais utilizados e a parametrização realizada tanto para eles quanto para o SVM.

5.1 Modelo de ML

O modelo SVM foi o escolhido para servir de linha de base para comparação com os demais modelos. Visto que o BoW e SVM juntos possuem bons resultados para classificação de textos mais longos (WANG; MANNING, 2012).

Para parametrizar o modelo, foi utilizada a base de validação, portanto, escolheu-se o modelo com os parâmetros que apresentasse melhores resultados neste conjunto de dados. Vale notar que, ao mesmo tempo em que busca-se ótimos resultados na validação, os modelos com casos de *overfitting*, ou seja, o modelo prediz com uma alta taxa a base de treino, foram desconsiderados. Além disso, os parâmetros foram variados de acordo com o disponível na implementação do Pedregosa et al. (2011).

Para encontrar os melhores parâmetros do modelo, seguiu-se as recomendações de Wang e Manning (2012). Utilizou-se os núcleos Linear, RBF e Polinomial de segunda ordem. Combinados a ele fez-se um balanceamento de classes e seleção da contante C em 1.0 padrão (PEDREGOSA et al., 2011) e 0.1 para maior controle do *overfitting* (WANG; MANNING, 2012). Foi utilizado 4 variações diferentes do dado para o SVM: Formato de símbolos pré-processado e não pré-processado, BoW também pré-processado e não pré-processado. Os melhores resultados para seleção de parâmetros estão na Tabela 8, para os demais não listados, utilizou-se os valores padrão da implementação.

Tabela 8 – Resultados no conjunto de validação e sua respectiva parametrização e processamento de dados.

Acurácia	Transformação	Pré-processado	Núcleo	C	Grau	Balanceamento[1]
0,5003	símbolos	X	polinomial	1,0	2	-
0,4167	símbolos	-	RBF	0,1	-	X
0,9178	**BoW**	**X**	**linear**	**0,1**	**1**	**X**
0,9126	BoW	-	linear	0,1	1	X

Fonte: elaboração própria.

[1] O balanceamento foi computado utilizando o método da biblioteca do Scikit Learn de Pedregosa et al. (2011) *sklearn.utils.class_weight.compute_class_weight*

5.2 Modelos neurais

Modelos hierárquicos como Hierarchical Attention Network (YANG et al., 2016) e CNN/LSTM-GRNN (TANG; QIN; LIU, 2015) foram inviabilizados, pela dificuldade de determinar quando uma sentença se inicia no documento, visto que o processo de extração de dados não deixou o conteúdo livre de caracteres especiais ou lixos. Assim, utilizar separadores de sentença pré-treinados não foi possível pela quantidade sentenças irregulares resultantes. Os terminadores de períodos comuns geraram confusão em muitos documentos.

Para todas as redes utilizadas, o vetor de enviesamento foi inicializado com o valor 0, a função de otimização utilizada foi a *Adam* com a taxa de aprendizado de 0.001 e valores *beta1* e *beta2* com 0.9 e 0.9999 respectivamente.

Não foram utilizadas funções de regularização, decaimento da taxa de aprendizado, nem diferenciação dessa taxa entre diferentes camadas.

A MLP foi utilizada em todas as arquiteturas como camada de saída com 6 neurônios, uma unidade para cada categoria de documento. Utilizou-se também a função de ativação *softmax*, e a função para cálculo de *loss* foi o *categorical cross entropy*.

Escolheu-se o tamanho 256 para os mini lotes (do inglês *mini batch*) de treinamento com 30 épocas (do inglês *epochs*). Salvou-se vários estados intermediários da rede neural durante o treinamento com os pesos que geraram a melhor acurácia. Dessa forma, na hora de coletar o resultado na base de treino, fez-se o carregamento destes melhores pesos.

5.2.1 *Multilayer Perceptron*

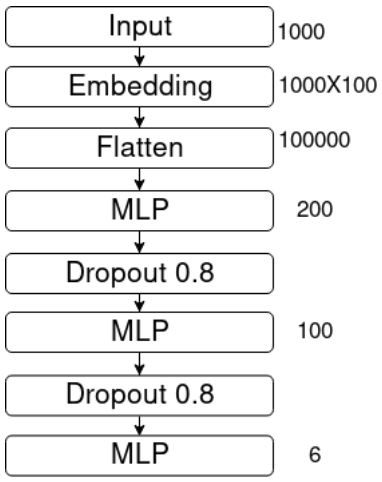
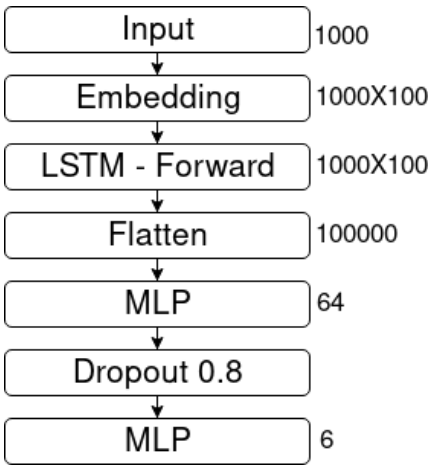


Figura 19 – Representação gráfica do modelo neural denso com três camadas ocultas, uma de entrada e uma de saída. Fonte: elaboração própria.

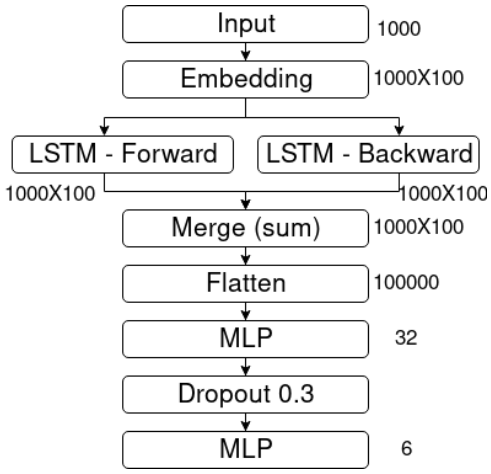
Elaborou-se um modelo de *Multilayer Perceptron* (**MLP**), como apresentado na Figura 19 para servir de base de comparação dos demais algorítmos. No diagrama está

presente duas camadas ocultas de MLP com 200 e 100 células cada, seguidas de *dropout* de 0.8. A função de ativação utilizada foi a ReLu..

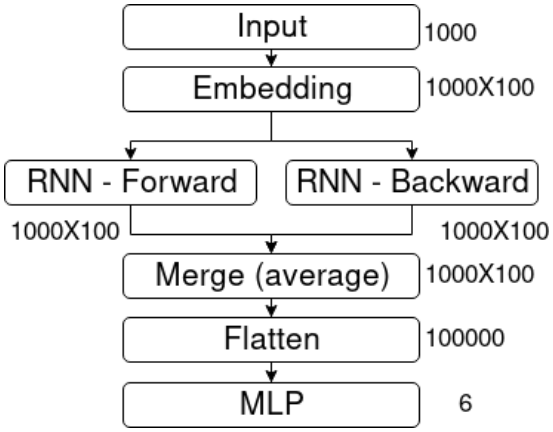
5.2.2 Recorrente



(a) Modelo baseado em LSTM (HOCHREITER; SCHMIDHUBER, 1997).



(b) Modelo baseado em *Bidirectional* LSTM (BRAZ et al., 2018).



(c) Modelo baseado em *Bidirectional* RNN (SCHUSTER; PALIWAL, 1997).

Figura 20 – Representações gráficas para os modelos neurais recorrentes.

Os modelos recorrentes foram utilizados na sua forma padrão, em que a saída de cada neurônio retorne um valor para a próxima camada, ao invés de apenas um número ao final da sequência da camada (GOLDBERG, 2017), ou que a saída de uma célula se torne a entrada da próxima (GOLDBERG, 2017) na mesma camada, nem fez-se uso das técnicas de atenção (GOLDBERG, 2017) as quais fazem o modelo manter um vetor adicional para priorizar um conjunto de palavras na saída da rede.

Para o modelo *Long Short-Term Memory* (**LSTM-pre**) (HOCHREITER; SCHMIDHUBER, 1997) foi necessário o texto pré-processado, pois sem ele, o modelo não foi capaz de generalizar todas as classes, predizendo sempre uma ou duas. Utilizou-se 100

células com uma taxa de *dropout* de 0.5. Foi adicionado mais uma rede MLP com 64 neurônios. A representação das camadas deste modelo estão na Figura 20a.

Os modelos Bidirecionais Recorrentes possuem a sua estrutura semelhante, a diferenciação é o algorítmo da célula de cada um deles (GRAVES; SCHMIDHUBER, 2005). Utiliza-se da propriedade do modelo recorrente, e treina-se duas camadas, uma com os dados na sequência direta e outra reversa. Dessa forma, consegue-se obter contexto de x_{i-1} e x_{i+1} na unidade da posição x_i.

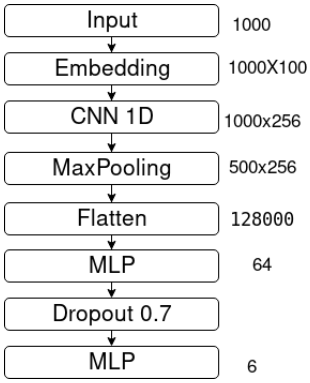
Para a arquitetura do modelo da *Bidirectional Recurrent Neural Network* (**BRNN-pre**) (SCHUSTER; PALIWAL, 1997), utilizou-se 100 unidades recorrentes, retornando os valores de saída de cada neurônio com um fator de *dropout* de 0.2 a fim de minimizar o *overfitting* no treino com muitos *epochs*. Ao modelo foi adicionada uma camada de *Embedding* com vetor de saida de tamanho 100. A função de mescla, foi a média aritmética dos vetores de saída. O modelo está representado na Figura 20c. Para o resultado desta arquitetura, foi necessário utilizar o texto pré-processado, pois sem ele, a rede não conseguia generalizar todas as classes.

Ao modelo *Bidirectional Long Short-Term Memory* (**BLSTM**) (BRAZ et al., 2018), foi adicionado uma camada de MLP com *dropout* de valor 0.3. Foram utilizados 100 células de memorização em cada LSTM. A mescla dos resultados foi feito com a média dos vetores de saída. A representação das camadas deste modelo estão na Figura 20b.

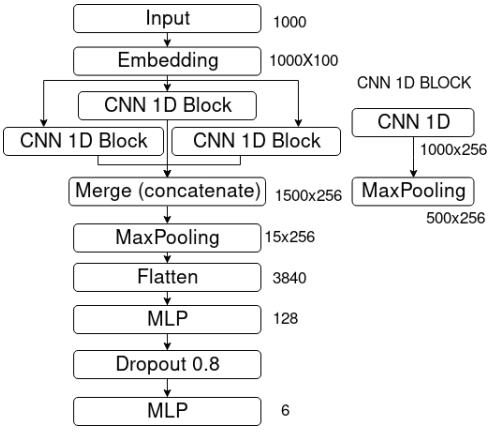
5.2.3 Convolucional

O modelo *Convolutional Neural Network* (**CNN**) (SILVA et al., 2018) é uma implementação simples do modelo com apenas uma convolução seguida de um *max pooling*, seguida da rede MLP com 64 neurônios e com dropout de 0.7. O propósito deste modelo é servir como linha de base para os modelos convolucionais mais complexos utilizados. Diferentemente do modelo proposto apresentado por SILVA et al. 2018, foi utilizado apenas a primeira página, isso fez com que a contagem de símbolos é inferior a 1000, assim o tamanho do vetor de entrada foi reduzido. Também foi adicionada uma camada escondida de MLP com *dropout* como apresentado na 21a, os demais parâmetros foram mantidos.

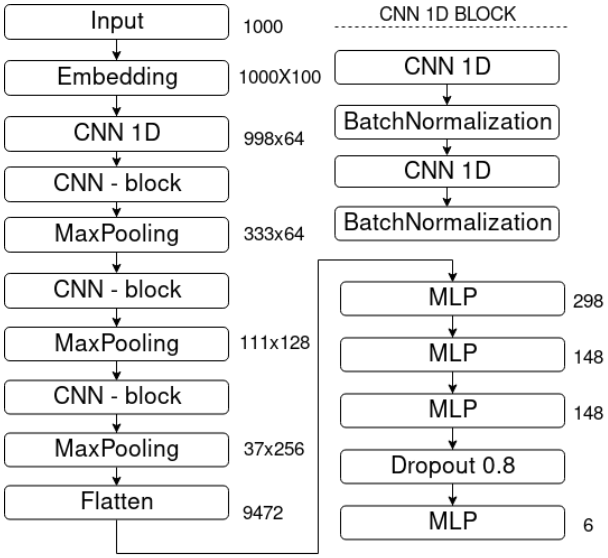
O modelo *Convolutional Neural Network* (**CNN-rand**) (KIM, 2014) foi modificado adicionando-se um dropout de 0.4 em cada canal de convolução. Realizar o treinamento do zero dos *embeddings* de uma CNN não geram os melhores resultados (KIM, 2014), entretanto não encontrou-se um modelo de *embedding* na língua portuguesa publicado em artigo que pudesse ser retreinado. A convolução foi feita com núcleos de tamanhos 3,4 e 5 e um total de 256 filtros, concatenando zeros para manter o tamanho do vetor ao final da convolução. O *max pooling* foi obtendo o maior valor de uma janela de tamanho



(a) Modelo baseado em CNN (SILVA et al., 2018).



(b) Modelo baseado em CNN (KIM, 2014).

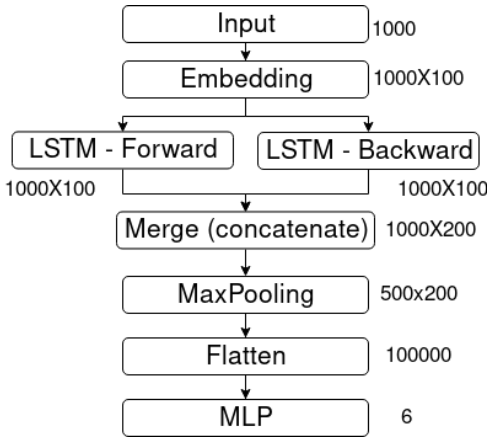


(c) Modelo baseado em CNN (CONNEAU et al., 2017).

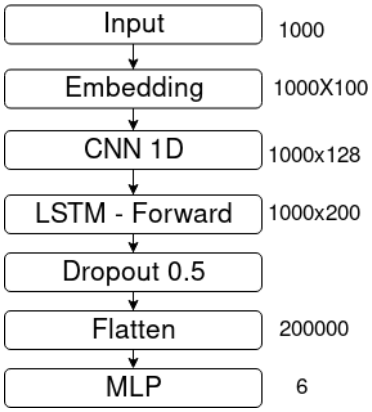
Figura 21 – Representações gráficas para os modelos neurais convolucionais.

2. Após a concatenação o *max pooling* aplicado foi com uma janela de tamanho 100. A representação deste modelo está na Figura 21b.

O *Very Deep Convolutional Neural Network* (**VDCNN**) (CONNEAU et al., 2017) foi reparametrizado para trabalhar com a carga de dados deste trabalho, portanto, foi removido um bloco de convolução, utilizando-se apenas 64, 128 e 256 filtros. Cada bloco desse foi montado com duas convoluções de núcleo com tamanho 3 e adicionada uma normalização, para garantir que os desvios padrões das taxas de ativação sejam próximas de 1. O *max pooling* dos blocos de convolução, são com janela de tamanho 2 e avanço de 3. O modelo está representado na Figura 21c.



(a) Modelo baseado em BLSTM-C (LAI et al., 2015).



(b) Modelo baseado em CNN-LSTM (ZHOU et al., 2015).

Figura 22 – Representações gráficas para os modelos neurais combinados entre convolucionais e recorrentes.

5.2.4 Modelos mistos

O modelo *Bidirectional Long Short-Term Memory-Convolutional* (**BLSTM-C**) (LAI et al., 2015) é uma BLSTM que possui um *max pooling* para extrair as melhores features da saída das LSTM, portanto na mesca dos resultados, é utilizado a concatenação. O modelo está representado na Figura 22a.

O *Convolutional Neural Network Long Short-Term Memory* (**C-LSTM**) (ZHOU et al., 2015) é uma proposta de utilizar a CNN como extratora de características e utilizar este mapa de características na LSTM, assim como representado na Figura 22b. A convolução possui 128 filtros com núcleo de tamanho 2, a LSTM tem 200 células.

5.3 Resultados dos modelos

A seguir nesta seção, serão apresentados os resultados do treinamento e predição do conjunto de teste. Nas tabelas e gráficos estão presentes as métricas coletadas para avaliar a performance do modelo e outras auxiliares.

5.3.1 Treinamento e validação

A execução dos modelos acima propostos foi no *hardware* NVidia[2] TitanXP 12 GB 3840 Cuda Cores a 1.5 GHz, Intel Core i5 7400, 48 GB DDR4 2133MHz, obteve-se os resultados da Tabela 9. Para gerar os tempos no conjunto de treinamento e de teste foram consideradas as quantidades de amostras apresentadas na Tabela 6.

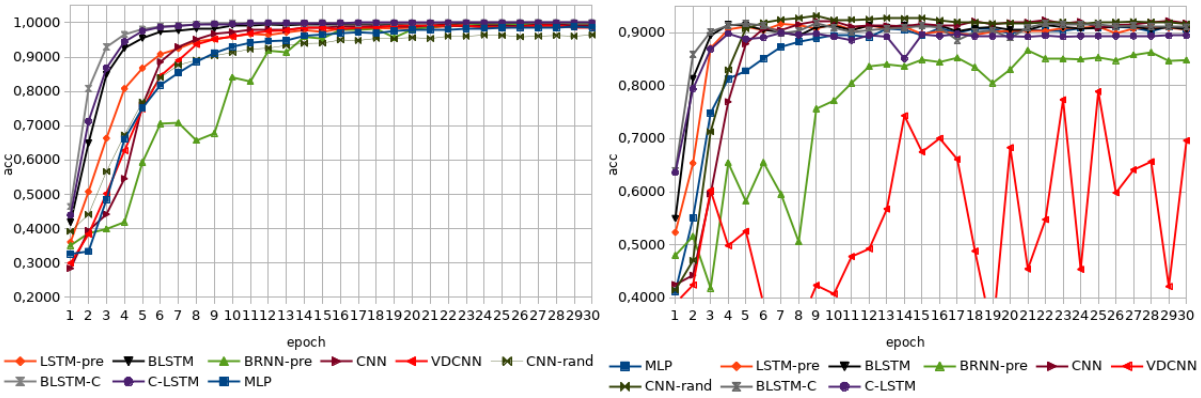
[2] Esta placa de vídeo foi doação da compania NVidia ao projeto GPAM para realização em pesquisas de ML.

Tabela 9 – Dados da execução dos modelos classificadores nos conjuntos de treino e validação.

Modelo	Melhor Epoch	Treinamento	Teste	pré-processamento
MLP	**28**	**0m 1s 2ms 822us**	**0s 121us**	**0**
LSTM-pre	9	38s 10ms 071us	1s 1ms	4s 58ms
BLSTM	3	91s 17ms 925us	1s 1ms	0
BRNN-pre	20	34s 61ms 0us	7s 10ms	4s 58ms
CNN	21	34s 6ms 756us	2s 4ms	0
VDCNN	24	44s 18ms 455us	1s 2ms	0
CNN-rand	20	91s 21ms 546us	5s 7ms	0
BLSTM-C	9	96s 20ms 229us	2s 3ms	0
C-LSTM	6	91s 21ms 447us	6s 8ms	0
SVM Linear-pre	-	6s 72ms 0us	1s 31ms	4s 58ms
SVM Linear	-	11s 100ms 0us	2s 100ms	0

Fonte: elaboração própria.

A utilização das bibliotecas de aprendizado profundo Keras[3] e de processamento Tensforflow[4] que possuem implementações otimizadas para placa de vídeo, possibilitaram que todos os tempos de execução na Tabela 9 fossem abaixo de 1 minuto e 40 segundos. Ressalta-se que para os modelos SVM Linear, LSTM e BRNN, os quais utilizaram dados pré-processados, deve-se considerar o tempo adicional de tratamento do dado. Dessa forma os tempos para esses modelos são $10s\ 13ms$, $42s\ 68ms\ 071us$ e $38s\ 119ms$ respectivamente, para seus tempos de teste adiciona-se $580ms$ ficando $1s\ 621ms$, $1s\ 581ms$ e $7s\ 590ms$.



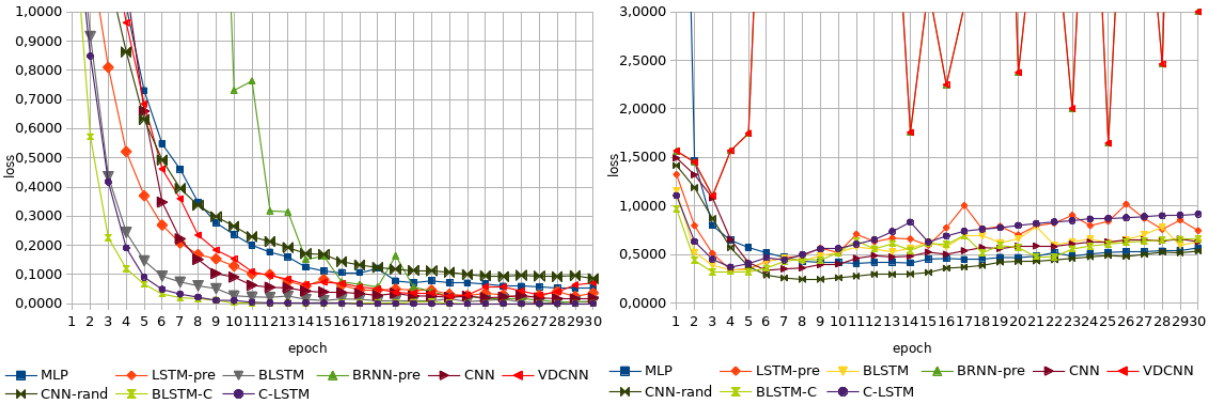
(a) Conjunto de treino (b) Conjunto de validação

Figura 23 – Acurácia dos modelos neurais durante a fase de treinamento e validação, separados por cada *epoch*. Fonte: elaboração própria.

Para os modelos neurais foram coletadas as informações de *loss* e acurácia utilizados para parametrizar todas as redes, o resultado final da execução dos modelos com

[3] Biblioteca que facilita a criação de diferentes arquiteturas neurais, faz uso de *backends* de processamento tanto em GPU quanto CPU. Disponível em: <https://keras.io/>. Acessado em: 25 de Novembro de 2018

[4] Um *framework open source* de aprendizado de máquina que viabiliza computação numérica de alta performace. Disponível em: <https://www.tensorflow.org/>. Acessado em: 25 de Novembro de 2018.



(a) Conjunto de treino (b) Conjunto de validação

Figura 24 – *Loss* dos modelos neurais durante a fase de treinamento e validação, separados por cada *epoch*. Fonte: elaboração própria.

seus melhores parâmetros estão na Figura 23 e *loss* na Figura 24.

Foi realizado um corte nas Figuras 24a e 24b para que as diferenças mais próximas dos intervalos de 0.5 e 0.01 do *loss* se tornassem mais visíveis nos gráficos. Para o *loss* no treino os valores nos *epoch* de 1 a 9 para a BRNN que ficaram omitidas devido ao corte são: $3,3539$, $3,7614$ $4,4127$, $4,6006$, $1,7363$, $1,1438$, $1,0280$, $1,8636$, $2,3836$. Pelo fato de se mostrar inconstante e sem tendência os erros da VDCNN, os valores no gráfico de *loss* de validação não serão apresentados.

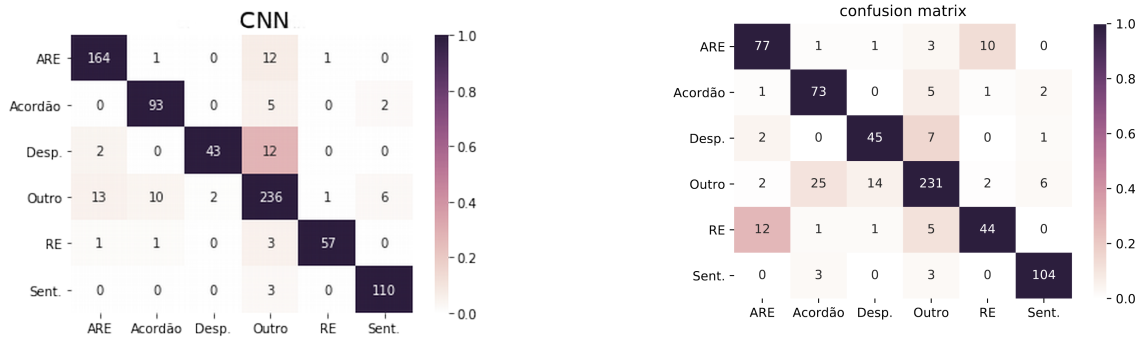
5.3.2 Performance final

Para a verdadeira avaliação da performance dos modelos, serão consideras as três métricas de revocação, precisão e acurácia apresentadas na Tabela 10. Outra métrica a se levar em consideração é o tempo necessário para predizer as amostras do conjunto de teste, estes tempos são apresentados na Tabela 9.

Para obter-se as métricas dos modelos CNN (SILVA et al., 2018) e BLSTMN (BRAZ et al., 2018) que utilizaram o mesmo conjunto de dados, fez-se uso das matrizes de confusão na Figura 25a e 25b para o cálculo. Os resultados foram adicionados à Tabela 10.

Dos modelos neurais apresentados, apenas o VDCNN apresentou grandes oscilações na acurária e nos valores de *loss*. No conjunto de treino, a acurárcia dos modelos apresentavam bons resultados a partir da 10ª iteração, o que refletiu também no conjunto de validação. A partir deste ponto, percebe-se que eles começaram a indicar *overfitting*, pois o *loss* começou a aumentar na validação e se aproximar de 0 sem que houvessem mudanças significativas no valor da acurácia enquanto que na validação o *loss* passou a aumentar.

A utilização das técnicas de pré-processamento mostraram diferença significativa



(a) Matriz de confusão CNN. Fonte: (SILVA et al., 2018, p. 3)

(b) Matriz de confusão BLSTM. Fonte: (BRAZ et al., 2018, p. 3)

Figura 25 – Matrizes de confusão para os modelos de CNN e BLSTM que utilizaram o mesmo conjunto de dados.

Tabela 10 – Métricas coletadas da execução de todos os modelos no conjunto de teste. Junto aos resultados obtidos por outros autores na mesma base de dados.

Modelo	**Acurácia**	**Precisão**	**Revocação**
DNN	0,9164	0,9095	0,9258
LSTM-pre	**0,9413**	**0,9334**	**0,9501**
BLSTM	0,9326	0,9311	0,9289
BRNN-pre	0,8651	0,8428	0,8569
CNN	0,9326	0,9265	0,9316
VDCNN	0,8255	0,8153	0,8117
CNN-rand	0,9384	0,9286	0,9422
BLSTM-C	0,9179	0,9146	0,9201
C-LSTM	0,9208	0,9169	0,9168
SVM Linear	0,9311	0,9300	0,9277
CNN (SILVA et al., 2018)	0,9035	0,9202	0,8965
BLSTM (BRAZ et al., 2018)	0,8416	0,8112	0,8357

Fonte: Elaboração própria.

apenas nos modelos recorrentes. Com o texto cru, os modelos não foram capaz de ajustar seus pesos para todas as classes. Com uma quantidade menor de símbolos presentes no texto, como apresentado na Tabela 7, foi possível que as características de memorização e adição de contexto fossem melhor aproveitadas.

6 Classificação de textos jurídicos

Neste capítulo, serão apresentadas as análises dos resultados encontrados.

6.1 Qualidade da *pipeline*

A *pipeline* de extração de dados possui uma complexidade muito alta. O fato de ser necessário dividir os documentos que vieram no formato de volumes, fez com que a confiabilidade nos rótulos das peças fosse baixo.

Por conta disso, foi necessário trabalho humano para reclassificar adequadamente os documentos. Além disso, o processo de extração de dados possibilitou que um mesmo documento, passasse pela pipeline mais de uma vez, ocasionando duplicação do seu registro no banco de dados.

Mesmo sendo confiável o rótulo gerado pelos especilistas, mudanças nos padrões estabelecidos por eles, fez com que documentos com o mesmo conteúdo recebessem classificações distintas, por exemplo um documento *2017080910001* poderia receber o rótulo Outros e também Despacho de Admissibilidade.

Apesar dos problemas registrados com a *pipeline*, a matriz de correlação utilizando *Spearman*, mostrou que as categorias de documentos não possuem correlação entre si. Com a aplicação da limpeza dos dados, observou-se na Figura 18, que a correlação entre as categorias aumentou, mas que ainda assim não é significativa. Esta não correlação é um sinal positivo, indicando que os conjunto de documentos possuem características distintas e que indicam ser separáveis.

6.2 Análise dos modelos

Da Tabela 8, percebe-se que os parâmetros sugeridos por Wang e Manning (2012) apresentaram os melhores resultados nos diferentes processamentos do texto. Ressalva-se, entretanto que, para o contexto em questão, a técnica BoW sobressaíu-se a de Unigramas. Além de que, a função de núcleo linear de (SMOLA; SCHÖLKOPF, 2004) obteve performance superior às demais quando fez-se uso do BoW.

Contrapondo ainda Wang e Manning (2012), a performance do modelo linear foi melhor do que o núcleo polinomial de grau 2. O fato do unigrama ter resultado inferior não impede de a combinação do SVM bi-grama (WANG; MANNING, 2012) se sobressair, entretanto, pode ser um indício que para este conjunto de dados o BoW seja o melhor processamento para o SVM.

Na Figura 23a, fica claro que do *epoch* 20 em diante, a maior parte dos modelos neurais encontram o problema do *overfitting* e mesmo com o uso em diferentes modelos da rede da técnica de *dropout* (SRIVASTAVA et al., 2014), não foi o suficiente para mitigar o problema. Com relação a isto, os modelos que se sobressaíram foi o CNN-rand (KIM, 2014), o MLP, o LSTM (HOCHREITER; SCHMIDHUBER, 1997) e o BLSTM (BRAZ et al., 2018), em que estes modelos não ultrapassaram o valor de acurácia de 98% no conjunto de treinamento, mantendo bons resultados no de validação.

Notoriamente o modelo VDNN (CONNEAU et al., 2017) não obteve boa performance, sendo o mais instável e que gerou os piores resultados. Quanto maior a profundidade da rede de VDNN, mais dados são necessários para seu treinamento (CONNEAU et al., 2017), e como neste trabalho têm-se disponível apenas 4 mil amostras, a consequência foi instabilidade. Nos experimentos em que a arquitetura fora proposta, a quantidade de dados era superior, sendo o menor número de amostras para treino 120.000 no conjunto de dados *AG's new*. Dessa forma, este modelo profundo mostrou-se pouco efetivo na separação de documentos jurídicos com o número de amostras atuais, ainda que tenha chegado em 82% na acurácia e 81 % em precisão, revocação no conjunto de teste.

O modelo que mostrou melhor performance entre os neurais foi o LSTM com os textos pré-processados, chegando a 94% de acurácia, 93% de precisão e 95% de revocação. Além de que seu tempo para predizer as amostras de teste foram na faixa de 1,58 segundos. A combinação de contextos das palavras predecessoras (SCHUSTER; PALIWAL, 1997) não trouxeram tanto ganho de informação com os dados pré-processados. O grande diferencial da BLSTM e BLSTM (BRAZ et al., 2018), foi o fato de serem mais robustas e não demandar o mesmo pré-processamento, sendo sua performance muito próximo ao da LSTM.

Entre os modelos convolucionais, o destaque foi a CNN-rand que foi robusta ao *overfitting* e apresentou bons resultados. Ainda assim, não se obteve a melhor performance deste modelo por sua inicialização ter sido totalmente randômica (KIM, 2014). O uso de um *embedding* inicial que se reajustasse durante o treinamento aumentaria sua performance (KIM, 2014), podendo até superar o LSTM. Entretanto, para a língua portuguesa não foi encontrado um *embedding* confiável treinado, optou-se por utilizar a forma randômica do modelo.

O CNN-rand também mostrou-se superior a implementação da CNN (SILVA et al., 2018). A modificação da CNN em adicionar uma camada de MLP com a técnica de *dropout*, reduzindo o tamanho do vocabulário melhorou em 3,2% na acurácia, 0,67% na precisão e 3,91% na revocação comparado a sua implementação original.

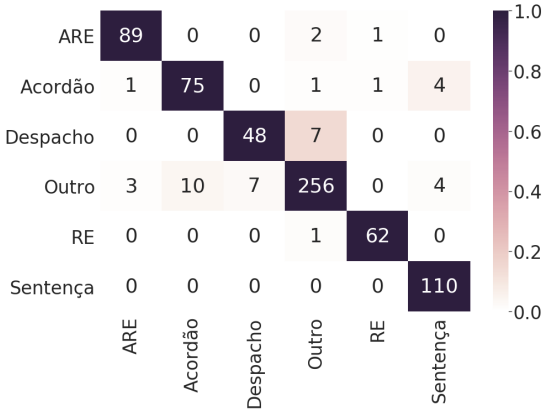
Assim como as modificações na CNN melhoraram a performance, o modelo BLSTM foi melhor do que o Braz et al. (2018). A adição de uma camada de MLP com *dropout* e o uso da saída de cada neurônio da LSTM na próxima camada, trouxeram o ganho de

10,8% na acurácia, 14,78% na precisão e 11,15% na revocação.

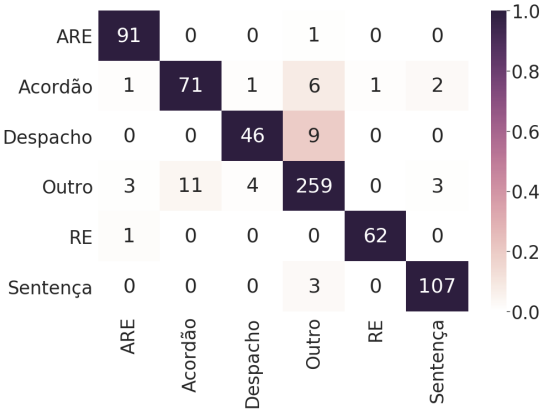
O modelo CNN-LSTM não obteve performance maior do que o LSTM, BLSTM, nem SVM como esperado (ZHOU et al., 2015). O BLSTM-C também não foi melhor do que o BoW + SVM e CNN (LAI et al., 2015). Assim sendo, neste contexto é melhor escolher e parametrizar uma das duas arquiteturas Recorrente ou Convolucional isoladamente.

6.3 Melhores modelos

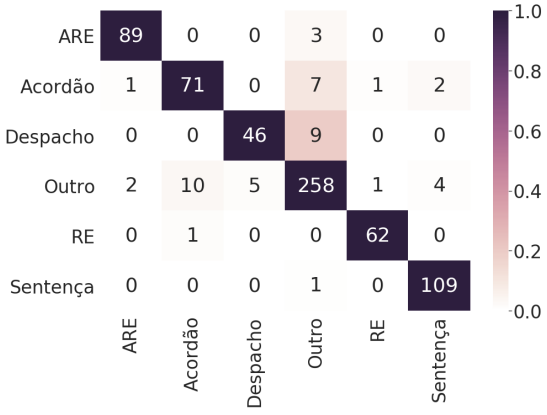
Ainda que o estado da arte de classificação de textos tenha avançado bastante na área de modelos neurais, utilizando ainda modelos de atenção (YANG et al., 2016; TANG; QIN; LIU, 2015), modelos profundos (CONNEAU et al., 2017) e demais, o modelo de aprendizado de máquina raso, SVM Linear, performou bem fazendo frente aos modelos do estado da arte, para este conjunto de dados.



(a) Modelo CNN-rand



(b) Modelo BLSTM



(c) Modelo SVM Linear

Figura 26 – Matriz de confusão dos principais modelos classificadores que obtiveram boas métricas. Fonte: elaboração própria.

Na Figura 26, são apresentadas as matrizes de confusão dos modelos que apresentaram as melhores métricas em relação a: **tempo de predição**, **acurácia**, **precisão** e

revocação. A partir destas três matrizes pode-se afirmar que os documentos de peças jurídicas do STF são computacionalmente separáveis, utilizando-se da primeira página de cada um deles.

Os três modelos confundiram-se de forma semelhante, com as classes Outro e Acordão, Outro e Despacho. Após isto, a confusão foi em classes distintas, como Outro e Sentena na Figura 26a; Acordão e Outro na Figura 26b e 26c. De maneira geral, o maior número de confusões foram associadas a categoria Outro. Repara-se que das categorias que apresentaram maior confusão entre si, não são as que apresentaram maiores valores no coeficiente de *spearman* da Tabela 18.

7 Conclusão

O processo elaborado para projeto de pesquisa que envolva pesquisa em Aprendizado de Máquina, mostrou-se efetivo para o planejamento e execução da pesquisa.

O estudo feito para os diferentes tipos de peças, envolveu o entendimento da organização do Poder Judiciário brasileiro e o do CPC (Lei 13.105/2015). O responsável pela confecção da peça, o tipo de conteúdo que ela contém, a quem é destinada e quais são as peças necessárias para a avaliação de Repercussão Geral foram detalhadas para ter um bom entendimento do domínio do problema.

A hipótese apresentada na Seção 1.2, de que não há textos iguais para peças diferentes, mostrou-se falsa após a aplicação do tratamentos de dados exposto no Apêndice A. No qual detectou-se documentos duplicados que foram removidos do conjunto de dados. Além disso, percebeu-se um aumento considerável ao comparar as matrizes de correlação de *spearman*, onde houve um aumento geral na correlação direta de 0.24 a 0.31 entre as classes.

Foram experimentados diferentes modelos para a classificação de textos, o modelo SVM Linear obteve performance equiparável aos modelos mais robustos e complexos como o CNN-rand e BLSTM. O SVM alcançou os valores de 93% de acurácia, 93% de precisão e 92% de revocação, empatando na acurácia com CNN-rand e BLSTM, sendo melhor do que a precisão do CNN-rand e empatando com a revocação do BLSTM.

Alguns modelos, apesar de apresentarem boa performance no conjunto de teste, foram desconsiderados por exigirem pré-processamento de dados ou por sofrerem de *overfitting* mesmo utilizando de técnicas de redução de *overfitting* como o *dropout* (SRIVASTAVA et al., 2014). O não uso de normalização dos pesos com *l2* também poderia ter auxiliado a reduzir o *overfitting*.

Dois objetivos do trabalho não foram alcançados. O primeiro: "elaboração de um dicionário para o contexto jurídico" não foi necessário, pois os modelos utilizados foram capazes de generalizar bem com dicionário extraído dos próprios documentos; O segundo: "levantar o estado da arte para a classificação de documentos" não foi alcançado, pois restringiu-se a busca por métodos que envolvessem principalmente modelos neurais. O resultado desta busca foi apresentada diretamente como modelos de referência no Capítulo 5.

Este trabalho possui lacunas não exploradas como a utilização ou criação de um *Word Embedding* para o contexto jurídico e utilizá-lo nos modelos como sugerido por Kim (2014). Outra lacula é relacionada às técnicas de agrupamento de modelos como o

Deep Forest (ZHOU; FENG, 2017), *XGBoost* (CHEN; GUESTRIN, 2016), dentre outros, no qual faz-se uso de vários modelos ao mesmo tempo para melhorar a acertividade, ou reduzir *overfitting* e a variância.

Além disso, os modelos neurais são caixas-pretas, ou seja, não se sabe como ele utiliza os dados para predizer uma categoria. Para trabalhos futuros, também propõe-se o uso de técnicas como *SHapley Additive exPlanations* (LUNDBERG; LEE, 2017), *Anchors as High-Precision Explanations* (RIBEIRO; SINGH; GUESTRIN, 2018) e LIME (RIBEIRO; SINGH; GUESTRIN, 2016), as quais tem a proposta de avaliar de forma agnóstica modelos preditivos e identificar quais dados da entrada possuem maior influência no resultado.

Referências

AMENDOEIRA JR, S. *Manual de direito processual civil: Teoria geral do processo e fase de conhecimento em primeiro grau de jurisdição*. 2. ed. São Paulo: Editora Saraiva, 2012. v. 1. ISBN 978-85-02-12710-4. Citado 3 vezes nas páginas 40, 41 e 42.

BRASIL. *Constituição da República Federativa do Brasil*. [S.l.]: Senado, 1988. Citado 9 vezes nas páginas 17, 18, 19, 20, 40, 41, 42, 45 e 47.

BRASIL. *Lei Nº 8.457, de 4 de setembro de 1992*. Brasília, DF, 1992. Disponível em: <https://www.planalto.gov.br/ccivil_03/leis/l8457.htm>. Citado 2 vezes nas páginas 40 e 42.

BRASIL. *Lei Nº 12.665, de 13 de junho de 2012*. Brasília, DF, 2012. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12665.htm>. Citado 2 vezes nas páginas 40 e 42.

BRASIL. *Lei Nº 13.105, de 16 de março de 2015. Código de Processo Civil*. [s.n.], 2015. Disponível em: <www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13105.htm>. Citado 3 vezes nas páginas 43, 44 e 45.

BRASIL. Congresso Nacional. *Lei Nº 12.527, de 18 de novembro de 2011*. Brasília, DF, 2011. Citado na página 18.

BRASIL. Conselho Nacional de Justiça. *Termo de acordo de cooperação técnica Nº 058/2009*. Brasília, DF, 2009. Citado na página 18.

BRASIL. Supremo Tribunal Federal. *Resolução Nº 427, de 20 de abril de 2010*. Brasília, DF, 2010. Citado na página 20.

BRASIL. Supremo Tribunal Federal. *Regimento Interno*. Brasília, DF, 2016. Citado na página 45.

BRASIL. Supremo Tribunal Federal. *Inteligência artificial vai agilizar a tramitação de processos no STF*. 2018. Disponível em: <http://www.stf.jus.br/portal/cms/verNoticiaDetalhe.asp?idConteudo=380038>. Citado 2 vezes nas páginas 19 e 20.

BRAZ, F. A. et al. Document classification using a Bi-LSTM to unclog Brazil's supreme court. In: *32st Conference on Neural Information Processing Systems (NIPS)*. Montreal, CA: [s.n.], 2018. Citado 6 vezes nas páginas 47, 55, 56, 60, 61 e 64.

BRINK, H.; RICHARDS, J. W.; FETHEROLF, M. *Real-World Machine Learning*. Meap. [S.l.]: Manning, 2015. Citado 6 vezes nas páginas 17, 26, 27, 31, 38 e 39.

CHAPMAN, P. et al. *CRISP-DM 1.0: Step-by-step data mining guide*. CRISP-DM consortium, 2000. Disponível em: <https://www.the-modeling-agency.com/crisp-dm.pdf>. Citado 2 vezes nas páginas 25 e 26.

CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, p. 785–794, 2016. ArXiv: 1603.02754. Disponível em: <http://arxiv.org/abs/1603.02754>. Citado na página 68.

CONNEAU, A. et al. Very Deep Convolutional Networks for Text Classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, 2017. p. 1107–1116. Disponível em: <http://aclweb.org/anthology/E17-1104>. Citado 4 vezes nas páginas 36, 57, 64 e 65.

CROWSTON, K.; SALTZ, J. S.; SHAMSHURIN, I. Comparing Data Science Project Management Methodologies via a Controlled Experiment. In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. Mānoa, Hawaii: [s.n.], 2017. p. 10. ISBN 978-0-9981331-0-2. Citado 2 vezes nas páginas 26 e 27.

ESLICK, I.; LIU, H. Langutils: A Natural Language Toolkit for Common Lisp. In: *Proceedings of the International Conference on Lisp*. Stanford, California: [s.n.], 2005. Citado na página 17.

GIL, A. C. *Como elaborar projetos de pesquisa*. 4ª. ed. São Paulo, SP: Atlas S.A., 2002. ISBN 85-224-3169-8. Citado 2 vezes nas páginas 21 e 24.

GOLDBERG, Y. *Neural Network Methods for Natural Language Processing*. [S.l.]: Morgan & Claypool, 2017. ISBN 978-1-62705-295-5. Citado 8 vezes nas páginas 17, 33, 34, 35, 36, 37, 38 e 55.

GONÇALVES, M. V. R. *Direito processual civil esquematizado*. 6. ed. São Paulo: Saraiva, 2016. Citado na página 43.

GOYVAERTS, J.; LEVITHAN, S. *Regular Expressions Cookbook*. Second. [S.l.]: O'Reilly Media, 2012. ISBN 978-1-4493-1943-4. Citado 2 vezes nas páginas 30 e 48.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, v. 18, n. 5, p. 602–610, jul. 2005. ISSN 0893-6080. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608005001206>. Citado na página 56.

GUIMARÃES, D. T. *Dicionário compacto jurídico*. 16. ed. São Paulo, SP: Rideel, 2012. ISBN 978-85-339-2023-1. Citado na página 42.

HEARST, M. A. et al. Support vector machines. *IEEE Intelligent Systems and their applications*, v. 13, n. 4, p. 18–28, 1998. Citado 2 vezes nas páginas 32 e 33.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Citado 2 vezes nas páginas 55 e 64.

KIM, Y. Convolutional Neural Networks for Sentence Classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1746–1751. Disponível em: <http://aclweb.org/anthology/D14-1181>. Citado 5 vezes nas páginas 37, 56, 57, 64 e 67.

LAI, S. et al. Recurrent Convolutional Neural Networks for Text Classification. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. Austin, Texas: AAAI Press, 2015. (AAAI'15), p. 2267–2273. ISBN 978-0-262-51129-2. Disponível em: <http://dl.acm.org/citation.cfm?id=2886521.2886636>. Citado 2 vezes nas páginas 58 e 65.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017. p. 4765–4774. Disponível em: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>. Citado na página 68.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN 0-521-86571-9 978-0-521-86571-5. Citado 3 vezes nas páginas 29, 30 e 49.

MIKOLOV, T. et al. Recurrent neural network based language model. In: *Eleventh Annual Conference of the International Speech Communication Association*. [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 37 e 38.

NIELSEN, M. A. *Neural networks and deep learning*. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/chap2.html>. Citado 3 vezes nas páginas 34, 35 e 36.

OLIVEIRA, E.; FILHO, D. B. Automatic classification of journalistic documents on the Internet. *Transinformação*, v. 29, p. 245 – 255, 2017. ISSN 0103-3786. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-37862017000300245&nrm=iso>. Citado 2 vezes nas páginas 17 e 29.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 3 vezes nas páginas 32, 33 e 53.

PRODANOV, C. C.; FREITAS, E. C. d. *Metodologia do trabalho cientifico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico*. 2ª. ed. Novo Hamburgo, RS: Editora Feevale, 2013. ISBN 978-85-7717-158-3. Citado 3 vezes nas páginas 23, 24 e 25.

RASCHKA, S. *Single-Layer Neural Networks and Gradient Descent*. 2015. Disponível em: <https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html>. Citado na página 33.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should I trust you?": Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. [S.l.: s.n.], 2016. p. 1135–1144. Citado na página 68.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. Anchors: High-Precision Model-Agnostic Explanations. In: *The Thirty-Second AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018. p. 1527–1535. Citado na página 68.

RODRÍGUEZ, L. C.; CASTAÑO, E. P.; SAMBLÁS, C. R. Quality performance metrics in multivariate classification methods for qualitative analysis. *TrAC Trends in Analytical Chemistry*, v. 80, p. 612–624, 2016. ISSN 0165-9936. Citado na página 40.

RUSCHEL, A. J.; ROVER, A. J.; SCHNEIDER, J. Governo eletrônico: o judiciário na era do acesso. In: *CALLEJA, P.L. (Org). La Administración Electrónica como Herramienta de Inclusión Digital*. Zaragoza, Espanha: Zaragoza: Prensas Universitarias de Zaragoza, 2011, (LEFIS series; 13). p. 59 – 79. ISBN 978-84-15274-66-7. Citado na página 18.

SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, v. 45, n. 11, p. 2673–2681, nov. 1997. ISSN 1053-587X. Citado 3 vezes nas páginas 55, 56 e 64.

SILVA, N. C. da et al. Document type classification for Brazil's supreme court using a Convolutional Neural Network. In: *10th International Conference on Forensic Computer Science and Cyber Law*. São Paulo, SP: [s.n.], 2018. p. 4. Disponível em: <http://icofcs.org/2018/papers-accepted.html>. Citado 8 vezes nas páginas 47, 48, 49, 56, 57, 60, 61 e 64.

SINGH, J.; GUPTA, V. Text Stemming: Approaches, Applications, and Challenges. *ACM Comput. Surv.*, v. 49, n. 3, p. 45:1–45:46, set. 2016. ISSN 0360-0300. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/2975608>. Citado 2 vezes nas páginas 30 e 49.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and Computing*, v. 14, n. 3, p. 199–222, ago. 2004. ISSN 1573-1375. Disponível em: <https://doi.org/10.1023/B:STCO.0000035301.49549.88>. Citado 3 vezes nas páginas 32, 33 e 63.

SRIVASTAVA, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=2627435.2670313>. Citado 2 vezes nas páginas 64 e 67.

TANG, D.; QIN, B.; LIU, T. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 1422–1432. Disponível em: <http://aclweb.org/anthology/D15-1167>. Citado 2 vezes nas páginas 54 e 65.

WANG, S.; MANNING, C. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *50th Annual Meeting of the Association for Computational Linguistics*, n. 50, p. 5, 2012. Citado 3 vezes nas páginas 32, 53 e 63.

Wiki PJE. *Wiki PJe*. 2018. Disponível em: <http://www.pje.jus.br/wiki/index.php/Página_principal>. Citado na página 18.

YANG, Z. et al. Hierarchical Attention Networks for Document Classification. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016. p. 1480–1489. Disponível em: <http://aclweb.org/anthology/N16-1174>. Citado 2 vezes nas páginas 54 e 65.

ZHOU, C. et al. A C-LSTM Neural Network for Text Classification. nov. 2015. Disponível em: <https://arxiv.org/abs/1511.08630>. Citado 2 vezes nas páginas 58 e 65.

ZHOU, Z.-H.; FENG, J. Deep Forest: Towards An Alternative to Deep Neural Networks. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, 2017. p. 3553–3559. ISBN 978-0-9992411-0-3. Disponível em: <https://www.ijcai.org/proceedings/2017/497>. Citado na página 68.

Apêndices

APÊNDICE A – limpeza dos dados

O código abaixo foi utilizado para aplicar as expressões regulares, as normalizações e a transformação em símbolos. Este código foi desenvolvido pelo Grupo de Pesquisa em Aprendizado de Máquina da Universidade de Brasília e foi adicionado aqui por estar em um repositório de código privado.

```
import re
from spacy.lang import pt
import nltk
from nltk.stem.snowball import SnowballStemmer

try:
    nltk.word_tokenize('some word')
except:
    nltk.download('punkt')

try:
    nltk.corpus.stopwords.words('portuguese')
except:
    nltk.download('stopwords')
finally:
    STOP_WORDS = pt.STOP_WORDS.union(
        set(nltk.corpus.stopwords.words('portuguese'))
    )


class CorpusHandler:

    def __init__(self):
        pass

    @staticmethod
    def clean_number(document, **kwargs):
        return re.sub(r'\s\d+\s', ' ', document)

    @staticmethod
    def clean_email(document, **kwargs):
```

```
        local_part = r"[0-9a-zA-Z!#$%&'*+-/=?^_`{|}~.]+"
        domain = r"[a-zA-Z][a-zA-Z0-9-.]*[a-zA-Z]\.\w{2,4}"
        document = re.sub(r"\s{}@{}".format(
                local_part, domain),
            ' EMAIL ', document)
        return document

    @staticmethod
    def clean_site(document, **kwargs):
        # Addaption for rules of
        # https://tools.ietf.org/html/rfc3986#section-3 to
        # stf's documents
        scheme = r"[a-zA-Z][a-zA-Z0-9+-.]*://?"
        host_ip = r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
        host_name = r"[a-zA-Z]+(\.[a-zA-Z0-9-_~]+)+"
        www = r'www\.{}'.format(host_name)
        scheme_host = r'({}{}|{}{}|{}{}|{})'.format(
            scheme, host_ip, scheme, www, scheme, host_name, www
        )
        port = r"(:\d+)?"
        resource = r"(/[a-zA-Z0-9-._~!$&'/*+,;=]*)?"
        query = r"(\?[a-zA-Z0-9-._~!$&'/*+,;=]*)?"
        fragment = r"(#[a-zA-Z0-9-._~!$&'/*+,;=]*)?"
        document = re.sub(r"{}{}{}{}{}".format(
                scheme_host, port, resource, query, fragment),
            ' SITE ', document)
        return document

    @staticmethod
    def transform_token(document, **kwargs):
        word_number = r'(n.?|numero|numero|n.?|no.?)?'
        # Law number: 00/YEAR or 00
        number_law = r'([0-9]+)((\s|\.)+\d+)?'
        matchs = [('LEI', 'lei'), ('ARTIGO', r'art\.|art\w*'),
                  ('DECRETO', 'decreto'),]

        for word, regex in matchs:
            document = re.sub(r'{}\s*{}\s*{}'.format(
                    regex, word_number, number_law),
```

```
                r'{}_\2'.format(word),
                document, flags=re.I)

        return document

    @staticmethod
    def remove_small_big_words(document, **kwargs):
        # remove 2 chars
        document = re.sub(r'\s\w{0,2}\s', '␣', document)
        # remove bigger words ex.: infrainconstitucionalidade
        document = re.sub(r'\s\w{30,}\s', '␣', document)
        return document

    @staticmethod
    def remove_letter_number(document, **kwargs):
        # Remove wor00 00wo 00wor00 wo00rd and keep WORD_000
        return re.sub(r'([A-Z]+_\d+)|[^␣]*\d+[^␣]*', r'\1', document)

    @staticmethod
    def clean_document(document, **kwargs):
        # Replace 0.0 for 00
        document = re.sub(r'(\d)\.(\d)', r'\1\2', document)

        # Remove all non alphanumeric
        document = re.sub(r'\W', '␣', document)

        return document

    @staticmethod
    def clean_spaces(document, **kwargs):
        # Remove multiple spaces
        document = re.sub(r'\s+', '␣', document)
        document = document.strip()
        return document

    @staticmethod
    def clean_alphachars(document, **kwargs):
        return re.sub(r'[^␣]*[^-_\'u\'i\'o\~o\^o\'e\^e\~a'
          '\'a\`a\^aa-z0-9\{\c␣c}␣]+[^␣]*',
```

```
                ' ', document)

    @staticmethod
    def tokenize(document, **kwargs):
        """Transform string documents in array of tokens"""
        if isinstance(document, str):
            document = document.split()
        return document

    @classmethod
    def remove_stop_words(cls, document, stop_words=[],
        extra_stop_words=[], **kwargs):
        tokens = cls.tokenize(document)
        words = set(stop_words) or STOP_WORDS
        if extra_stop_words != []:
            words = words.union(set(extra_stop_words))
        document = " ".join(filter(lambda x: x not in words, tokens))
        return document

    stemmer = SnowballStemmer("portuguese")
    @classmethod
    def snowball_stemmer(cls, document, **kwargs):
        """Use nltk Snowball Stemmer to stemmize words"""
        tokens = cls.tokenize(document)
        document = ' '.join([cls.stemmer.stem(word) for word in tokens])
        return document
```