Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de Textos para Classificação de Processos Judiciais Trabalhistas

Ana Carolina Pereira Rocha

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Guilherme Novaes Ramos

Coorientador
Prof. Dr. Rômulo Soares Valentini

Brasília
2019

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

AR672m

ANA CAROLINA PEREIRA, ROCHA
MINERAÇÃO DE TEXTOS PARA CLASSIFICAÇÃO DE PROCESSOS JUDICIAIS TRABALHISTAS / ROCHA ANA CAROLINA PEREIRA; orientador GUILHERME NOVAES RAMOS; co-orientador RÔMULO SOARES VALENTINI. -- Brasília, 2019.
174 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2019.

1. JUSTIÇA DO TRABALHO. 2. MINERAÇÃO DE TEXTOS. 3. CLASSIFICAÇÃO DE TEXTOS. 4. PJE. 5. ASSUNTO. I. RAMOS, GUILHERME NOVAES, orient. II. VALENTINI, RÔMULO SOARES, co orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de Textos para Classificação de Processos Judiciais Trabalhistas

Ana Carolina Pereira Rocha

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Guilherme Novaes Ramos (Orientador)
CIC/UnB

Prof. Dr. Sandro José Rigo
Universidade do Vale do Rio dos Sinos

Prof. Dr. Thiago de Paulo Faleiros
Universidade de Brasília

Prof. Dr. Aletéia Patrícia Favacho de Araújo
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 04 de dezembro de 2019

Dedicatória

Dedico este trabalho a todas as mulheres que dedicam parte do seu tempo à construção de novos conhecimentos em busca de transformar o ambiente onde vivem, para que este seja o ambiente onde um dia sonharam viver.

Agradecimentos

Agradeço a todos que contribuíram, de forma direta ou indireta, para que este trabalho pudesse ser realizado. Agradeço em especial aos professores Guilherme e Rômulo por sua orientação e direcionamento.

Agradeço aos meus pais, por me ensinarem a dedicação aos estudos e ao crescimento profissional, e por toda sua paciência com minhas ausências nos momentos em que precisei me dedicar ao mestrado. Ao David Vieira, que me apoiou incondicionalmente no desenvolvimento deste trabalho, se colocando ao meu lado para estudar com entusiasmo os avanços na área de aprendizado de máquina. À toda a minha família, que vibrou comigo a cada conquista. À todos os meus amigos, pelo seu incentivo e por compartilharem comigo momentos de leveza e diversão em meio a esta empreitada. Aos colegas de mestrado, pelos momentos de aprendizado que passamos juntos.

Algumas pessoas também colaboraram diretamente neste trabalho. Agradeço à Dra. Natália Martins, Juíza do Trabalho e minha prima querida, que apoiou esta pesquisa trazendo direcionamentos e informações importantes. A Dra. Alciane Margarida, também Juiza do Trabalho, pelo apoio e por todos os ensinamentos. Ao professor e amigo Rogério Lopes, por me inspirar na caminhada, acreditando sempre em meu potencial, apresentando oportunidades de crescimento e desenvolvimento profissional. Ao colega Gustavo Orair, que sugeriu o estudo a classificação de assuntos nos processos trabalhista e apoiou esta pesquisa compartilhando seu vasto conhecimento. Aos colegas do Grupo de Negócio do PJe na Justiça do Trabalho, que contribuíram com seu conhecimento negocial neste trabalho.

À todos os colegas de trabalho, em especial aos colegas da equipe de Adminstração de Dados da SMPAD, que deram o apoio necessário para que eu pudesse me dedicar a este trabalho. Ao Alisson Wilker, Christiano Carvalho e demais gestores do CSJT por me concederem tempo para concluir este trabalho. Aos Juízes do Trabalho Maximiliano Carvalho, Fabiano Coelho e Fabiano Pfeilsticker por acreditarem nesta ideia e viabilizarem sua execução dentro das possibilidades da Coordenação do PJe.

Resumo

Atualmente existe uma grande quantidade de processos que tramitam na justiça trabalhista brasileira, o que demanda um alto esforço dos servidores do judiciário e demais envolvidos para dar tratamento a todas as demandas. É possível que a aplicação de técnicas de mineração de textos possa contribuir com a identificação automática de informações relevantes dos processos. Assim, este trabalho aplicou algoritmos de classificação em um conjunto de 241 mil documentos do tipo Recursos Ordinários, extraídos de processos do PJe instalado na Justiça do Trabalho, com o objetivo de encontrar o assunto principal de processos do 2^o grau, considerando 35 assuntos possíveis. Foram comparados os algoritmos Multinomial Naïve Bayes, Multi-Layer Perceptron, Random Forest e SVM. Identificou-se baixo desempenho dos modelos para se encontrar o assunto principal, que negocialmente, pode ser uma escolha subjetiva na maior parte dos processos. Nesta abordagem, chegou-se à uma micro precisão máxima de 46,03% com o Multi-Layer Perceptron. Ao fazer uma análise considerando o acerto dos modelos visando não apenas o assunto principal, mas avaliando se o modelo acertou qualquer um dos assuntos existentes no processo, chegou-se a uma micro precisão de 75,21% com o Random Forest. Assim, mostrou-se que é possível extrair conhecimento dos documentos para identificação de assuntos, embora a identificação do assunto principal tenha apresentado baixo desempenho.

Palavras-chave: justiça do trabalho, processo judicial eletrônico, pje, assunto, tema, classificação de textos, mineração de textos

Abstract

The number of lawsuits being processed in the Brazilian Labor Court system is growing every year, demanding more and more resources. This cost may be reduced by applying text mining techniques to the automatic identification of relevant information within a process. This work applied classification algorithms to a specific set of documents extracted from the Superior Court's system (PJe), aiming to find the main subject of a process within 35 possibilities. Multinomial Naïve Bayes, Multi-Layer Perceptron, Random Forest and SVM were compared at this task, and the best result for micro precision was 46,03%, achieved through Multi-Layer Perceptron. Extending the analysis to consider all the related subjects, instead of only the main one, Random Forest provided a micro precision of 75,21%, indicating that a machine learning approach is a feasible approach.

Keywords: labor court, electronic judicial process, classifier, text classification, pje, text mining

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A partir da evolução dos meios de armazenamento e processamento de dados computacionais e sua consequente redução de custos, somados aos avanços do ramo da ciência da computação, atualmente vive-se um cenário com uma elevada quantidade de informação. Há sobretudo, um grande volume de informações não estruturadas armazenadas em formas de textos, imagens, vídeos ou áudios. Estima-se que nos ambientes corporativos o montante de dados não estruturados represente de 85% a 90% de toda informação armazenada [1]. Enquanto a quantidade de dados está crescendo constantemente, a capacidade humana de interpretá-los não acompanha esta evolução [2]. Assim, novos ramos do conhecimento têm ganhado espaço e provido meios para se trabalhar com toda a informação que está sendo acumulada. Este ramo multidisciplinar é a ciência de dados, que tenta entender dados complexos de forma sistemática e os problemas do negócio relacionados, aplicando conhecimentos da área de estatística, informática, computação, inteligência artificial, comunicação entre outros [3]. Dentre as várias ferramentas utilizadas na ciência de dados, tem-se a mineração de textos, que busca extrair conhecimento de um conjunto de dados não estruturados [2].

Neste trabalho, aplica-se a mineração de textos no âmbito da Justiça do Trabalho (JT) de forma a tentar classificar processos trabalhistas baseado no conteúdo de seus documentos. O processo trabalhista é o instrumento pelo qual um cidadão (ou um conjunto de pessoas) requer formalmente ao Poder Judiciário que julgue determinada causa. Ele é composto por um conjunto de peças (ou documentos) processuais que, segundo um rito processual e uma burocracia pré-estabelecida, possibilita ao juízo competente determinar uma sentença [4].

Atualmente, os processos trabalhistas tramitam de forma eletrônica no sistema Processo Judicial Eletrônico (PJe) instalado na Justiça do Trabalho JT[1]. O PJe é o sistema

[1]PJe instalado na Justiça do Trabalho: `http://www.pje.jus.br/wiki/index.php/`. Neste trabalho, sempre que a sigla PJe for utilizada, é uma referência ao PJe instalado na Justiça do Trabalho

que permite a prática processual de advogados, procuradores, magistrados, servidores e demais pessoas que participam de uma relação processual diretamente no sistema, permitindo que se junte os documentos processuais ao dossiê do processo. Além dos documentos em si, o sistema faz uso de metadados que auxiliam na organização interna dos processos e dos documentos. Entretanto, a maior parte da informação referente ao pedido processual e ao resultado final se ainda se encontra apenas no inteiro teor dos documentos elaborados pelos usuários, o que torna a leitura dos documentos, que é uma uma tarefa onerosa, mandatória em situações onde a simples existência (ou corretude) de dados estruturados estratégicos poderia dar as informações necessárias para determinada atividade.

Dentre os metadados armazenados sobre os processos, tem-se o assunto processual, que identifica quais são os temas tratados naquele processo. Um processo pode ser categorizado com mais de um assunto sendo que, de 1131 assuntos existentes, 877 assuntos estão disponíveis para o cadastro dentro da categoria de assuntos do Direito do Trabalho e do Direito Processual Civil e do Trabalho. Nem sempre esta informação é preenchida corretamente ou de forma completa, uma vez que a identificação dos assuntos exige que se conheça o rol de assuntos possíveis e demanda também o tempo do usuário, que deverá ler todas as peças processuais relevantes para fazer a correta vinculação aos assuntos pertinentes.

Por outro lado, o correto preenchimento dos assuntos traz uma série de vantagens. A primeira delas é a própria qualidade da informação, pois várias estatísticas são colhidas baseadas nesta informação de forma a dar visibilidade ao judiciário, conforme detalhado na Seção 1.1 . Este processo envolve complexidades próprias, de forma que tem-se uma ciência dedicada a este estudo: a Jurimetria, que aplica a ciência de dados ao Direito, fazendo um estudo qualitativo e quantitativo do comportamento judicial, possibilitando a previsão de comportamentos futuros com o objetivo de orientar políticas públicas. Além disso, a correta classificação do assunto tem impacto na organização do trabalho interno dos Tribunais, podendo ser utilizada para uma melhor organização do acervo de processos, na busca de demandas repetitivas ou na pesquisa jurisprudencial, montagem de pautas de audiência ou sessão, dentre outras atividades.

Assim, o objetivo deste trabalho é investigar se técnicas de classificação de textos, aplicadas em um conjunto de documentos processuais, são capazes de identificar o assunto principal do processo. Havendo um bom desempenho nesta atividade, mostra-se a viabilidade desta abordagem, apontando que modelos de aprendizado de máquina de classificação são capazes de reconhecer o assunto de um processo, podendo vir a ser incorporados ao PJe, de forma a facilitar a atividade do usuário por meio da sugestão dos assuntos dos novos processos e até mesmo de uma reclassificação de processos já armazenados.

1.1 Justificativa

Atualmente 877 assuntos diferentes do Direito do Trabalho e Direito Processual Civil e do Trabalho podem ser utilizados para classificar os processos trabalhistas no PJe, o que torna a escolha dos assuntos corretos uma tarefa complexa para os usuários. Não só é preciso conhecer e entender sobre os possíveis assuntos, como é preciso encontrar as opções corretas dentre todos os pedidos processuais, o que se apresenta como um problema do paradoxo da escolha, onde o usuário, em face à muitas opções, fica paralisado e acaba por fazer escolhas de forma menos cautelosa e muitas vezes errada [5]. Além disso, é comum que se tenha erros de classificação em dados inseridos pelos usuários [6, 7].

Como o preenchimento incorreto desta informação não causa prejuízo ao resultado final do processo, uma vez que se trabalha primordialmente com o conteúdo dos textos redigidos, advogados e servidores nem sempre se dão ao trabalho de preencher a informação corretamente e completamente. Assim, atualmente, este dado, embora preenchido, muitas vezes não representa a realidade total do processo em questão, ou seja, é um dado de baixa qualidade. Isso impacta negativamente os objetivos da criação das TPUs, conforme exposto no Manual de Utilização destas tabelas[2]. Dentre esses objetivos, ressalta-se: melhorar a gestão de pauta pelos órgãos judiciais; melhorar o controle de prevenção e a distribuição processual por competência em razão da matéria; identificar os assuntos mais frequentes nos processos judiciais, possibilitando melhor gestão do passivo pelos tribunais, além da adoção de medidas que previnam novos conflitos; assegurar, juntamente com outros instrumentos, a padronização de rotinas processuais e subsidiar a implantação diversos projetos corporativos no Poder Judiciário.

Além do trabalho de qualidade do dado em si, que visa manter uma base de dados com dados corretos, os estudos da área de jurimetria são fortemente impactados. A jurimetria, citada pela primeira vez em 1949 [8], é a disciplina que visa investigar o Direito por meio da aplicação de métodos estatísticos. Ela envolve três pilares operacionais: jurídico, estatístico e computacional, deslocando o centro de interesse do estudo do Direito do teórico para o operacional, apontando a importância dos fatores sociais, econômicos, geográficos, éticos, entre outros, na concretização das normas do direito. Para tanto, esta ciência é fortemente dependente dos dados armazenados em computadores e do poder computacional de análise, fazendo uso da metodologia estatística para descrever o comportamento humano, entender os fatores que influenciam na produção de normas e monitorar a reação que estas normas provocam em seus destinatários.

A partir destas informações, a jurimetria promove um entendimento do impacto gerado pelos estímulos criados pelas políticas públicas, dando visibilidade a um panorama geral

[2]Manual de Utilização das TPUs: `http://www.cnj.jus.br/sgt/versoes_tabelas/manual/Manual_de_utilizacao_das_Tabelas_Processuais_Unificadas.pdf`

para que se possa alcançar os objetivos socialmente desejados com o Direito. A relação que existe entre os estímulos que são criados pelo Direito e as condutas que posteriormente são observadas na população são a essência de uma ordem jurídica. Quanto mais próximo for o efeito esperado da conduta observada, mais bem-sucedida será a lei. A jurimetria presta um valioso auxílio aos agentes do Direito, permitindo por exemplo que um juiz compreenda possíveis consequências de suas decisões, advogados entendam fatores que influenciam em suas estratégias com seus clientes e os legisladores antecipem resultados de propostas de políticas públicas em pauta [9, 10, 11].

Várias são as regulamentações que ressaltam a importância da estatística e os trabalhos que fazem uso destes dados, e, portanto, podem ser impactados com a publicação de uma informação incorreta. O Departamento de Pesquisas Judiciárias (DPJ) do Conselho Nacional de Justiça (CNJ) tem como função colher dados para desenvolver pesquisas destinadas ao conhecimento da função jurisdicional brasileira, realizar análise e diagnóstico dos problemas estruturais e conjunturais dos diversos segmentos do Poder Judiciário e fornecer subsídios técnicos para a formulação de políticas judiciárias[3]. O Relatório Justiça em Números [12], elaborado anualmente pelo Sistema de Estatística do Poder Judiciário (SIESPJ) [4], também do CNJ, dá uma visão geral em números sobre o Poder Judiciário brasileiro, havendo até mesmo um prêmio CNJ de qualidade[5], que dentre vários objetivos pretende incentivar a produção de dados e o aprimoramento do SIESPJ. No âmbito da JT, o sistema e-Gestão[6] fornece estatísticas sobre a atividade judicante dos magistrados. Há ainda os relatórios estatísticos produzidos pelo Tribunal Superior do Trabalho (TST), que dentre várias informações, expõe os assuntos mais recorrentes da JT[7].

A disponibilização incorreta de informações nos sistemas é uma transgressão às regulamentações do Poder Judiciário citadas acima. Além de prejudicar a credibilidade das instituições que fornecem os dados, gera estatísticas que também serão consequentemente incorretas, o que impossibilita a elaboração de um planejamento estratégico adequado por parte dos órgãos públicos [8].

Quanto ao aspecto negocial, a elevada quantidade de processos e documentos jurídicos a serem apreciados pelos servidores da justiça trabalhista traz lentidão ao tempo de tramitação do processo. Atualmente, de acordo com o Relatório Justiça em Números do

[3]Lei de criação do DPJ: http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/lei/L11364.htm

[4]Leis do SIESPJ: https://atos.cnj.jus.br/atos/detalhar/atos-normativos?documento=188, https://atos.cnj.jus.br/atos/detalhar/atos-normativos?documento=110f

[5]Prêmio CNJ de Qualidade: https://atos.cnj.jus.br/atos/detalhar/atos-normativos?documento=2920/

[6]Lei de criação do e-Gestão: https://hdl.handle.net/20.500.12178/4549

[7]Relatórios: http://www.tst.jus.br/web/estatistica/jt/assuntos-mais-recorrentes,http://www.tst.jus.br/web/estatistica/jt/relatorio-geral

[8]Notícia CNJ: https://www.cnj.jus.br/nao-se-faz-gestao-judiciaria-sem-producao-de-conhecimento-dest

ano base de 2018 [12], o tempo médio de tramitação de um processo até a elaboração da sentença é de 9 meses (fase de conhecimento) no 1^o grau, 5 meses no 2^o grau e 1 ano e dois meses no 3^o grau. À cada documento peticionado pelos advogados em um processo, cabe uma apreciação e um retorno por parte dos tribunais, na forma de outro documento. A redação dos documentos por parte dos magistrados e servidores, quando não se trata de modelos de documentos comuns como despachos ou mandados, normalmente envolverá uma elaboração em relação ao contexto do processo e jurisprudência (conjunto de decisões colegiadas sobre interpretações da lei) aplicada naquele tipo de demanda, envolvendo, portanto, uma pesquisa sobre documentos da mesma natureza que já foram redigidos em processos similares, ou seja, processos que trataram de um mesmo assunto. Isto é importante para que se tenha uma homogeneidade na forma de tratar os processos de maneira geral.

Assim, a correta classificação dos assuntos processuais pode auxiliar na busca de processos similares, reduzindo o tempo que os servidores levam para recuperar a informação que precisam para embasar a redação dos documentos em cada processo, podendo ter um impacto positivo no tempo em que um processo fica parado aguardando um retorno dos tribunais. Por exemplo, para que um servidor ou magistrado que trabalha na redação de votos em um gabinete possa elaborar um voto sobre processo que trate do assunto "Assédio moral", ele precisará fazer um procedimento análogo à "revisão da literatura" no meio acadêmico, recorrendo a processos que já trataram deste tema para buscar leis, súmulas, acórdãos dentre outros elementos que foram utilizados na elaboração da decisão final de um magistrado. Isso não só facilita o trabalho da elaboração do documento mas auxilia a resguardar o princípio da segurança jurídica [13], que visa trazer estabilidade e previsibilidade das consequências das decisões emanadas pelos Tribunais, o que tende a diminuir o ajuizamento de novas demandas para casos com fatos já julgados, o que, dentre outras benesses, reduz o congestionamento processual na Justiça Trabalhista, contribuindo para a redução do número de ações ajuizadas [14].

Em termos práticos, se hoje um servidor leva aproximadamente 2 horas para fazer uma pesquisa para encontrar os processos que trataram de "Assédio moral", considerando que, o servidor da justiça do trabalho no 2^o grau (por exemplo) deve atuar em média com uma carga de 282 processos por ano (entre pendentes e julgados)[12], e esse tempo possa ser reduzido pela metade pela correta classificação dos assuntos, ganhar-se-ia 282 horas de trabalho do servidor. Em uma jornada de 7 horas diárias, isto equivaleria a 40 dias de trabalho que poderiam ser utilizados em outra atividade ou outros processos. O valor aproximado de 2 horas foi sugerido por uma servidora do TST que trabalha com a redação de votos, uma vez que não se encontrou trabalhos publicados que abordassem a quantidade de tempo despendida na tarefa de elaboração de documentos de decisão.

Outra tarefa que pode ser facilitada pelo processo de classificação automática dos assuntos é a triagem inicial, que acontece quando o processo chega no gabinete e precisa ser direcionado para a equipe que trabalha com aquele tipo de processo. Se ao chegar no gabinete o assunto do processo já estiver categorizado de forma automática e correta, o direcionamento do processo fica facilitado, poupando ao servidor que faz a triagem a leitura do inteiro teor da petição inicial ou do recurso, possibilitando até mesmo a criação de mecanismos automáticos para a distribuição dos processos.

Cita-se ainda como benefício da correta classificação de assuntos a possível facilitação no processo de identificação de demandas repetitivas. Esta atividade ganhou relevância no âmbito da Justiça depois da publicação do novo Código do Processo Civil[9], que em seu Artigo 976 estabelece a instauração do processo de Incidente de Resolução de Demandas Repetitivas quando houver questões contenham controvérsia sobre uma mesma questão apresentando risco de ofensa à isonomia e à segurança jurídica. A classificação correta dos assuntos pode auxiliar na identificação de demandas que tratam de causas similares, direcionando o usuário tanto na classificação de novos casos que versem sobre tema que já é tratado como uma demanda repetitiva, quanto na busca de processos já julgados que trataram sobre esta causa.

Explicada a importância da correta classificação dos assuntos, ressalta-se que este trabalho está alinhado com a gestão estratégica do Conselho Superior da Justiça do Trabalho (CSJT), atuando na melhoria contínua do processo de trabalho e se mostrando como uma possibilidade de inovação para o PJe. Dentre os indicadores impactados pela classificação automática de assuntos processuais tem-se o Índice de Satisfação Interna com o Sistema do Processo Judicial Eletrônico e o Índice de Satisfação Externa com o Sistema do Processo Judicial Eletrônico [10]. Nesse sentido, o resultado deste estudo poderá servir de insumo para o desenvolvimento do projeto Classificador de Temas do CSJT, que já tem Termo de Abertura de Projeto (disponível no Anexo I) aprovado pela Coordenação Nacional Executiva (CNE) do PJe para a implementação futura[11]. A utilização de um algoritmo de classificação de textos que possa fazer uma categorização automática do assunto traz um leque de possibilidades de tornar o PJe um sistema ainda mais inovador, que explora de forma inteligente os benefícios que a tecnologia pode trazer para os advogados, servidores e magistrados.

[9]CPC: `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13105.html`

[10]Lista de indicadores: `http://www.csjt.jus.br/c/document_library/get_file?uuid=93f2f26f-e3f9-4561-8993-cdb9d0e1799d&groupId=5625802`

[11]O projeto Classificação de Assuntos do PJe estava previsto para ser desenvolvido no primeiro semestre de 2019, mas devido à outras demandas prioritárias, o projeto precisou ser adiado. A nova gestão, do biênio 2020-2021, deverá reavaliar a prioridade do projeto.

1.2 Objetivos

O objetivo da pesquisa é testar a hipótese de que algoritmos de classificação de texto são capazes de classificar os processos judiciais trabalhistas quanto ao assunto por meio da investigação da aplicação desses algoritmos em documentos judiciais da justiça trabalhista e de seus resultados de classificação.

1.2.1 Objetivos específicos

Com o propósito de se alcançar o objetivo definido para este estudo, estabelece-se os seguintes objetivos específicos:

1. Entender como está organizada a Justiça do Trabalho e sua relação com o sistema PJe
2. Realizar uma análise exploratória da distribuição de processos, assuntos e documentos.
3. Aplicar algoritmos de classificação em documentos de processos do PJe.
4. Fazer uma busca de hiper-parâmetros para os algoritmos de classificação selecionados.
5. Comparar o desempenho de diferentes algoritmos neste domínio.
6. Disponibilizar um artefato de software que automatize o processo da escolha do melhor modelo dado um conjunto de documentos.

1.3 Contribuições esperadas

Nesta dissertação, será apresentado um estudo que dará uma explicação da estrutura organizacional da Justiça do Trabalho, da utilização do PJe pelos Tribunais Regionais do Trabalho (TRTs) e o fluxo geral dos processos. Estas informações compõe um relatório do funcionamento da Justiça do Trabalho com o PJe, que contém não só as informações gerais de seu funcionamento, mas também uma análise exploratória à nível nacional que permitirá identificar a distribuição de tipos de documentos e assuntos em 2º grau, bem como aspectos dos textos contidos nestes documentos. Este relatório poderá servir de insumo para trabalhos futuros da área da computação e também da área do direito.

Serão apresentados ainda aspectos relevantes em relação à tarefa de classificação de texto por meio de técnicas de mineração de dados e Processamento de Linguagem Natural, além de serem contemplados alguns trabalhos publicados que tratam de problemas similares fazendo uso da classificação de textos.

Em seguida, serão apresentados os passos necessários para a preparação de um conjunto de textos específicos do PJe para que possam ser utilizados nos algoritmos de classificação de textos; a busca de hiper-parâmetros para estes algoritmos; o resultado da aplicação das melhores configurações encontradas para cada um desses algoritmos; e um comparativo do desempenho de um conjunto de algoritmos de classificação de texto aplicados a documentos judiciais trabalhistas, que mostrará qual tipo de modelo se mostra mais adequado para o problema em questão.

O código utilizado para esta atividade será entregue como um artefato de software que, a partir de uma entrada pré-definida, fará o pré-processamento dos textos, treinará diferentes modelos em diferentes configurações, e elegerá o modelo de melhor desempenho, tendo como saída o modelo vencedor. Este artefato poderá ser utilizado em trabalhos futuros para tarefas de classificação, podendo ser adaptado para incorporar melhorias, novos algoritmos e novas formas de pré-processamento e representação textual.

Por fim, serão apontadas as conclusões obtidas com este trabalho e espaços não explorados para o desenvolvimento de trabalhos futuros.

O restante do texto está organizado da seguinte forma: No Capítulo 2 é apresentada uma revisão da literatura, onde buscou-se encontrar trabalhos que trataram problemas parecidos; no Capítulo 3, aborda-se os principais conceitos teóricos necessários ao entendimento do trabalho; no Capítulo 4 apresenta-se os resultados obtidos e no Capítulo 5 faz-se um apanhado das conclusões obtidas no trabalho e aponta trabalhos futuros a serem realizados neste contexto.

Capítulo 2

Trabalhos relacionados

Esta revisão da literatura busca encontrar trabalhos onde se tenha abordado a classificação de textos. Buscou-se, sobretudo, estudos que envolveram a língua portuguesa no contexto de documentos jurídicos, embora alguma dessas premissas tenham sido relaxadas para alguns trabalhos citados. Os trabalhos foram recuperados das bases de publicações do Web of Science[1], Scopus[2] e Google Acadêmico[3], com termos os termos "classificação de textos jurídicos", "mineração de textos jurídicos", "classificação de textos", "mineração de textos", e outras variações, fazendo uso destes termos em português e em inglês.

Começando pelos trabalhos já publicados envolvendo os documentos da Justiça do Trabalho do Brasil, encontra-se o estudo realizado por um servidor do Tribunal Regional do Trabalho (TRT) da 2ª Região [15] em 2011, onde extraiu-se as ementas das jurisprudências, que são o resumo do que foi decidido em um acordão, com suas premissas e justificativas. De posse de 43.013 ementas, por meio da ferramenta Weka, aplicou-se os algoritmos J4.8 (árvore de decisão), Naïve Bayes (NB) e Sequential Minimal Optimization (SMO), que é uma otimização do Support Vector Machine (SVM), de forma individual e também combinados em um *ensemble*. Foram escolhidas 10 categorias para análise, e utilizados apenas 1.000 documentos para a criação do modelo de cada categoria, sendo 500 documentos da categoria de interesse e os demais de outras categorias, configurando uma abordagem binária balanceada para tratar o problema multiclasse. Após o pré-processamento, utilizou-se uma matriz TF-IDF para a representação de cada texto. Não foi informado o tamanho médio dos textos, apenas informou-se que eles são um resumo do acordão, indicando um texto de tamanho reduzido em relação ao documento de inteiro teor do acordão. Notou-se que o desempenho de cada algoritmo é bastante dependente da categoria, apresentando elevadas taxas de erro em determinadas categorias, e em ou-

[1] `https://www.webofknowledge.com`
[2] `https://www.scopus.com`
[3] `https://scholar.google.com.br/`

tras não. Considerando a acurácia média dentre todas as categorias, o melhor modelo individual foi o J4.8, que apresentou o valor 88,60% e o ensemble apresentou o valor de 84,80%.

Outro trabalho mais antigo sobre a Justiça do Trabalho, de 2007 [16], envolveu também a categorização de documentos de processos trabalhistas. Neste estudo, analisou-se 104 partes diferentes de sentenças e acórdãos que trataram de 4 assuntos diferentes. Cada parte extraída contem entre mil e duas mil palavras. Os documentos utilizados eram de processos físicos, à época ainda não havia PJe. Foram removidas as palavras de pouca relevância semântica, também chamadas de *stopwords*, e aplicou-se a radicalização. Foram gerados vetores de *bag-of-words* (BOW) para a representação dos textos. Os modelos utilizados para a classificação foram criados na ferramenta Text-Miner Software Kit (TMSK), onde escolheu-se a utilização de algoritmos do modelo Linear por Ordenação, Naïve Bayes, e com a ferramenta Rule Induction Kit for Text (RIKTEXTI) utilizou-se um modelo de Regras de Decisão. O modelo linear apresentou os melhores resultados em 3 das quatro categorias, com a métrica F-Measure variando entre 86,88% e 94,30% entre os diferentes assuntos, e o Naïve Bayes apresentou o melhor desempenho para a categoria de honorários, apresentando F-Measure de 95,31%.

Partindo para outros trabalhos publicados envolvendo documentos jurídicos na língua portuguesa, tem-se o trabalho [17] , de 2018, que mostra a pesquisa de apoio realizada para a construção do Projeto Victor do STF [4], cujo propósito é a classificação do tipo de cada peça processual em processos em Repercussão , uma vez que tem-se um documento único com várias peças processuais anexas. Trabalhou-se com 5 tipos de documentos processuais diferentes, configurando-se um problema multiclasse. Foi necessária a aplicação de Reconhecimento Óptico de Caracteres (OCR) em algumas peças, todas estavam em formato PDF. No pré-processamento, foram removidos caracteres especiais, números, e-mails e links; aplicou-se a radicalização, normalização e a remoção de palavras recorrentes. O tamanho do vocabulário foi restringido devido à elevada quantidade de erro do processo de OCR. Não foi informado o tamanho do vocabulário utilizado, nem o tamanho médio dos textos (embora nota-se que a camada de *embeddings* das redes neurais contenham apenas 1.000 neurônios, indicando o tamanho do vocabulário usado nestes modelos). Foram identificadas as citações de leis, artigos e decretos. Ao todo, foram usados 6.814 documentos, que foram classificados manualmente por uma equipe de 4 advogados, uma vez que é sabido que a classificação existente na base não é confiável.

Neste trabalho, o modelo SVM foi utilizado como modelo base para uma comparação de diversos modelos de redes neurais, envolvendo redes neurais densas (MLP), recorrentes (LSTM, BLSTM, BRNN), convolucionais (CNN, CNN-rand, VDCNN) e mistas (BLSTM-

[4]Notícia STF: `http://www.stf.jus.br/portal/cms/verNoticiaDetalhe.asp?idConteudo=380038`

C, CNN-LSTM). Utilizou-se a linguagem Python, com Keras e Tensorflow. Para o SVM utilizou-se uma representação BOW, e para os demais, utilizou-se vetores de palavras. O melhor modelo foi o LSTM, com acurácia de 94.13%, precisão de 93,34% e revocação de 95,01%. O SVM apresentou resultado inferior à LSTM mas melhor que a maior parte das redes neurais testadas, chegando a 93,11% de acurácia, 93,33% de precisão e 95.01% de revocação.

Outro estudo de 2016 mostra a aplicação de técnicas de classificação de texto nas notificações recebidas por um escritório de advogacia, que são classificadas manualmente [18]. Neste trabalho, recuperou-se 5.471 documentos escritos em português, classificados entre 8 categorias distintas. Removeu-se as *stopwords* e as palavras menos significativas de acordo com o valor apresentado na matriz TD-IDF, que foi a forma de representação de textos utilizada. A classe com menos exemplos continha 121 documentos, enquanto a mais populosa continha 1.715 documentos. Não foi informado se foi aplicada alguma técnica de balanceamento e nem o tamanho médio dos textos. Os algoritmos utilizados foram o K-Nearest Neighbor, SVM, Naïve Bayes e Complement Naïve Bayes. A métrica utilizada para avaliação foi a curva ROC (Característica de Operação do Receptor, ou *Receiver Operating Characteristic*), onde o melhor resultado foi apresentado pelo SVM, com valor 0,846. Este modelo também foi o de maior acurácia, apresentando o valor de 84,53%.

Envolvendo estudos aplicados a um contexto jurídico mas com textos escritos em outras línguas, tem-se uma publicação recente de 2019 [19], com um estudo de caso bastante semelhante ao desta dissertação, para o qual será dada maior atenção. Este estudo comparou o desempenho de métodos mais atuais do estado da arte usados nas tarefas de classificação envolvendo NLP com o desempenho de modelos tradicionais para fazer a classificação de documentos contendo o julgamento de processos da Suprema Corte de Singapura. O trabalho se propõe a fazer uma classificação multirrótulo, com 6.227 documentos escritos em inglês, contendo em média 6.968 palavras. O pré-processamento passou pela tokenização, remoção de stopwords e lematização com a biblioteca spaCy do Python.

O problema inicialmente continha 51 classes principais, mas como havia um desbalanceamento de dados, foi limitado às 30 classes principais mais frequentes. Diferentemente do estudo apresentado nesta dissertação de mestrado, neste estudo os autores não tinham uma tabela padronizada de classes. As classes eram escritas livremente, havendo erros de digitação, diferentes nomes para áreas de mesmo interesse, áreas que são subáreas de outras áreas, apresentando uma hierarquia, de forma que foi preciso fazer um trabalho de organizar e normalizar estas informações, ainda que tenha sido declarado que há uma subjetividade nas escolhas feitas pelos pesquisadores. Para escolher as classes primárias,

os autores explicaram que desconhecem uma ontologia padrão para a classificação no Direito de Singapura, então eles usaram como referência a Árvore de Assuntos da Academia de Direito de Singapura, chegando a 51 classes principais. Mencionou-se que outras 252 classes consideradas muito específicas e com poucos casos foram agrupadas em uma classe 'Outros'. Para lidar com o problema do desbalanceamento, os autores utilizaram um método de amostragem iterativa estratificada [20], que é um método apropriado para um problema multirrótulo, que leva em consideração a distribuição de classes, e pares de classes. Foram separados conjuntos com 10%, 50% e 100% dos dados para os testes.

Neste trabalho, os modelos de classificação escolhidos foram: um classificador *dummy* que classifica como 1 qualquer classe que tenha probabilidade maior que 1/31, um classificador baseado na ocorrência de palavras chaves relacionadas, ambos usados com *baselines*; dois classificadores com LSA e LinearSVM (OneVsRest), um com 100 tópicos e outro com 250; três classificadores com vetores de palavra GloVe, cada um diferenciando-se na forma de gerar o vetor final do texto, onde um usou a média dos vetores, outro usou *max-pooling* e o último gerado a partir de uma Rede Neural rasa (CNN); dois classificadores com o modelo de linguagem Bert, um com uma arquitetura mais simples e o outro com a uma arquitetura mais complexa; e um classificador com o modelo de linguagem ULMFit com ajuste fino. Um ponto importante a ser mencionado é que os modelos BERT passaram por ajuste fino com apenas a parte inicial de cada documento devido a uma limitação de sua arquitetura inicial, competindo em desvantagem. Os modelos foram testados em sua configuração padrão, não havendo a busca dos melhores parâmetros. Os modelos foram avaliados quanto às macro métricas e micro métricas. De maneira geral, na maior parte das análises, o conjunto com 100% dos dados apresentou melhores resultados. Os modelos de *baseline* apresentaram métricas bastante inferiores aos demais classificadores nos conjuntos de 50% e 100%, com exceção da revocação, onde o valor foi similar aos demais classificadores.

Considerando-se apenas o conjunto de 100,00% e desconsiderando-se os modelos de *baseline*, analisa-se as métricas. Quanto à F-Measure, o pior resultado foi do ULMFit, enquanto o melhor valor foi apresentado pela combinação LSA de 250 tópicos, com o Linear SVM, apresentando macro F-Measure de 63,20% e micro F-Measure de 73,3% Quanto à precisão, o pior valor novamente foi do ULMFit, e novamente a combinação de LSA com Linear SVM apresentou o melhor resultado, tendo 83,40% na macro precisão com 250 tópicos, e 83,70% na micro precisão com 100 tópicos, superando todos classificadores com modelos de linguagem ou vetores de palavras com pelo menos 10,00% a mais na precisão. Quanto à revocação, os piores modelos foram o de LSA com Linear SVM, enquanto os melhores resultados foram apontados pelo modelo de vetores de palavras com a CNN, com macro revocação de 62,30% e micro revocação de 68,8%. Isto mostra

a relevância de se ter claro a prioridade de cada métrica para que se faça a escolha do classificador que tenha resultados mais adequados ao problema que se quer resolver.

O estudo mostrou que os modelos do estado da arte de Processamento de Linguagem Natural, testados em sua configuração padrão inicial, na maior parte das vezes não apresenta melhores resultados que os métodos tradicionais. Fica a expectativa de que com mais investimento em busca de hiperparâmetros, ajuste fino para melhorar o benefício advindo da transferência de conhecimento (*transfer learning)* e a adaptação para textos longos, eles possam apresentar melhores resultados neste domínio jurídico.

Em um estudo de 2017 [21], os autores fizeram uso de técnicas de mineração de texto para classificar documentos jurídicos chineses. Fez-se uso de um conjunto de 6.735 textos jurídicos previamente classificados em 13 categorias. Destes textos, extraiu-se com expressões regulares apenas a seção que é interessante para o objetivo da classificação, e depois aplicou-se técnicas de pré-processamento de texto. Vale notar que além da lista de *stopwords* chinesa comum, os autores construíram uma lista de *stopwords* jurídicas, que eram muito frequentes e pouco relevantes para a tarefa de classificação. As palavras do texto foram representadas por meio e vetores TF-IDF, onde se representa a relevância de cada palavra perante o conjunto de documentos, e para a redução da dimensionalidade aplicou-se os métodos Princial Component Analysis (PCA) e uma variação do Singular Vector Decomposition (SVD). Utilizou-se os algoritmos NB, Árvores de Decisão, Random Forest e SVM para a tarefa de classificação, e eles foram comparados por meio das métricas de acurácia, precisão, revocação e F-Measure. O melhor resultado foi do SVM, com métrica F-Measure de 87,00%.

Em [22], tentou-se predizer a decisão de um juiz para um determinado processo na Corte Europeia de Direitos Humanos. Criou-se um modelo de classificação binária, onde a entrada é o texto com a problemática do processo e o resultado esperado é a suposta decisão do juiz. Este trabalho usou documentos escritos em inglês, nos quais se removeu as *stopwords*, transformando o texto em letras minúsculas, para posteriormente extrair-se n-gramas que foram analisados para aplicação do processo de extração de tópicos. Os dados foram passados para o modelo de classificação em função dos tópicos extraídos e também dos n-gramas, criando-se uma matriz BOW, e tópicos extraídos. O modelo utilizado para a classificação foi o SVM, com o qual chegou-se a um modelo com acurácia de 79,00%.

No trabalho [23], criou-se um modelo de classificação para as revisões dos usuários sobre estabelecimentos listados no sítio TripAdvisor[5]. Foi abordado o problema de se ter uma base de dados com registros previamente categorizados, onde há erro de cadastro das categorias. O pré-processamento envolveu a tokenização, radicalização, remoção de núme-

[5]TripAdvisor: `https://www.tripadvisor.com/`

ros e pontuações e transformação em letras minúsculas. Os textos foram representados em uma matriz TF-IDF. Com o uso da linguagem Python, utilizou-se os algortimos Stochastic Gradient Descent Classifier (SGDClassifier), Random Forest, Ada Boost, Linear Support Vector Classification (Linear SVC), Regressão Logística e Multinomial Naïve Bayes. Inicialmente, o problema apresentava 6 classes, que foram simplificadas em 2, tornando o problema de classificação binário. Não foi informado o tamanho médio doas textos, mas no exemplo dado, o texto continha 64 palavras, indicando um tamanho relativamente pequeno. Foram removidos os textos com menos de 8 palavras, por serem considerados pouco informativos. Devido à baixa qualidade dos rótulos apresentados inicialmente na base, a F-measure máxima foi alcançada pelo Multinomial Naïve Bayes, atingindo 77,00%. Para reduzir a quantidade de dados errados no conjunto de treinamento, buscou-se encontrar um centroide baseado nos vetores TD-IDF dos documentos que representasse cada uma das classes, e escolheu-se apenas os elementos mais próximos do centroide para compor um conjunto confiável para treinamento, o que elevou a métrica do Multinomial Naïve Bayes F-Measure de 77,00% para 90,00%. Maiores informações sobre a redução de ruídos em um conjunto de dados podem ser encontradas na pesquisa consolidada no estudo [6] e em outros trabalhos relacionados [24, 25].

Outro trabalho que se parece com a proposta deste estudo por envolver um elevado número de classes se refere à classificação denúncias recebidas pela Controladoria Geral da União [26]. A diferença se dá pelo tamanho dos textos, que é limitado a 2.048 caracteres, e o teor da mensagem, que não é de cunho jurídico de maneira geral. Como pré-processamento, removeu-se números, espaços em branco duplicados, pontuações, transformou-se todo o conteúdo em letras minúsculas, removeu-se stopwords e fez-se a radicalização. Inicialmente, o problema de classificação continha 94 classes, mas devido aos baixos resultados, o escopo foi reduzido para 64 classes. Foram testados os algoritimos SVM, Random Forest, Naïve Bayes e Árvore de decisão (C4.5). Esses algoritmos não se mostraram escaláveis à medida que se aumentava a quantidade de classes, e apresentaram desempenho abaixo do esperado, chegando a um máximo de 59,00% de precisão com o SVM. Assim, resolveu-se abordar o problema como uma classificação multirrótulo, aumentando as chances de acerto do classificador. Buscando-se resolver a questão da escalabilidade, o modelo de classificação foi criado com uma combinação da Codificação Adaptativa de Huffman com a Descrição de Comprimento Mínimo (CAH+MDL). Mostrou-se que esse algoritmo apresentou taxa de revocação maior que o SVM, mas uma precisão menor. Com a abordagem multirrótulo e o CAH-MDL, somados a um novo pré-processamento de texto que fez uso apenas dos termos mais significativos, reduzindo a dimensionalidade da matriz de 165.203 termos para apenas 1.350 termos, que proporcionou maior redução da dimensionalidade, conseguiu-se uma precisão de 84,00%.

De maneira geral, com base nos trabalhos analisados, nota-se o reconhecido desempenho do SVM e o frequente uso de árvores de decisão, Naïve Bayes e diferentes tipos de redes neurais. Como forma de representação dos textos, o TF-IDF, BOW e LSA são amplamente utilizados, sendo o TF-IDF foi a forma mais utilizada, não sendo possível afirmar se pela facilidade de uso ou eficácia e popularidade da técnica. Assim, este trabalho se propõe à aplicação destas técnicas para a classificação de assunto dos processos da JT.

Os trabalhos encontrados que foram desenvolvidos dentro do contexto da JT brasileira [15, 16] foram publicados há 8 e 12 anos respectivamente, não havendo conhecimento de publicações mais recentes envolvendo a classificação de textos com documentos de processos da JT, de forma que este trabalho vem a retomar estes estudos em uma época onde um maior poder computacional já se encontra disponível. Além disso, a consolidação e ampla utilização do PJe permite a uniformização dos dados dentre os 24 TRTs, o que facilita a análise de dados massivos.

Capítulo 3

Referencial teórico

De forma a trazer um entendimento do contexto deste trabalho e das técnicas utilizadas para abordar o problema, neste capítulo serão apresentados a forma como a Justiça do Trabalho está organizada; o sistema PJe, onde tramitam os processos judiciais trabalhistas; os principais conceitos sobre a classificação de textos; e o modelo de referência para desenvolvimento de atividade de mineração de dados conhecido como CRISP-DM.

3.1 Justiça do Trabalho

O Poder Judiciário brasileiro é dividido em cinco esferas principais: a Justiça do Trabalho trata das ações judiciais entre empregados e empregadores; a Justiça Eleitoral trata das questões relacionadas à realização das eleições; a Justiça Militar (Estadual e Federal) julga os militares; e a Justiça Comum, que é dividida entre Estadual e Federal, e lida com as demais questões. A JT, foco deste trabalho, é organizada em três graus ou instâncias de apreciação.

A primeira delas é composta pelas Varas de Trabalho, onde atuam os juízes do trabalho. A competência destes órgãos julgadores é dada pela localização onde se prestou o trabalho (independentemente do local de contratação). Segundo o relatório Justiça em Números de 2019 [12], ao final de 2018 existiam 1587 Varas de Trabalho.

A segunda instância é composta por 24 Tribunais Regionais do Trabalho, onde os desembargadores julgam os recursos contrários às decisões providas nas Varas de Trabalho bem como outras ações de competência originárias deste grau. Os 26 estados brasileiros e o Distrito Federal se dividem entre as 24 regiões. Os TRTs são classificados pelo SIESPJ quanto ao porte, que leva em consideração diversos dados estatísticos relacionados a despesas, processos em trâmite, número de magistrados e servidores e trabalhadores auxiliares. Na Figura 3.1, extraída do relatório Justiça em Números de 2017 [12], pode-se analisar o porte de cada região.

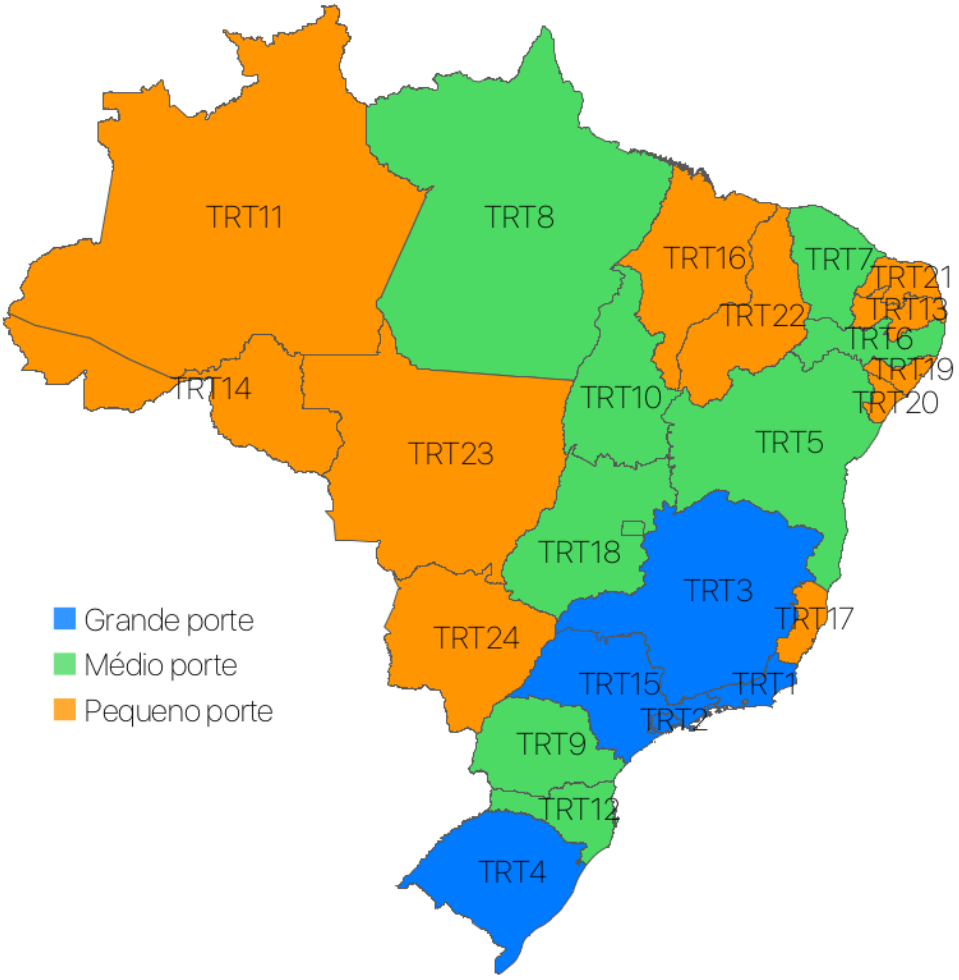

Figura 3.1: Porte dos Tribunais Regionais (Fonte: [12]).

A última instância é o Tribunal Superior do Trabalho (TST), que, como órgão máximo, atua como revisor das decisões das instâncias anteriores, além de atuar nas causas de competência originária desta corte. Sua função principal é a de uniformizar as decisões sobre ações trabalhistas de forma a consolidar a jurisprudência deste ramo do judiciário.

De forma a organizar e dar direcionamento para todos os ramos da justiça, em 2005 criou-se o Conselho Nacional da Justiça (CNJ)[1], que tem atuação em todo o território brasileiro e a missão de desenvolver políticas judiciárias para promover a unidade e efetividade de todo o Poder Judiciário, buscando valores de justiça e paz social. Este órgão zela pela autonomia do Poder Judiciário, definindo seu planejamento estratégico, planos de metas e programas de avaliação, além de prestar serviços aos cidadãos (recebe reclamações, petições,etc), definir e avaliar indicadores pertinentes à atividade jurisdicional, entre outros. Assim, todas as justiças são regidas pelos atos normativos e recomendações

[1]CNJ: `http://www.cnj.jus.br/sobre-o-cnj/quem-somos-visitas-e-contatos`

do CNJ.

Além do CNJ, cada ramo da justiça tem o seu Conselho próprio. Assim, na JT foi criado o Conselho Superior da Justiça do Trabalho[2] (CSJT), que exerce a supervisão administrativa, orçamentária, financeira e patrimonial da JT de 1^o e 2^o graus. Além de atuar na supervisão, este órgão promove a integração e o desenvolvimento da JT de 1^o e 2^o graus. Na Figura 3.2, tem-se um esboço de como estes órgãos estão estruturados dentro da JT.

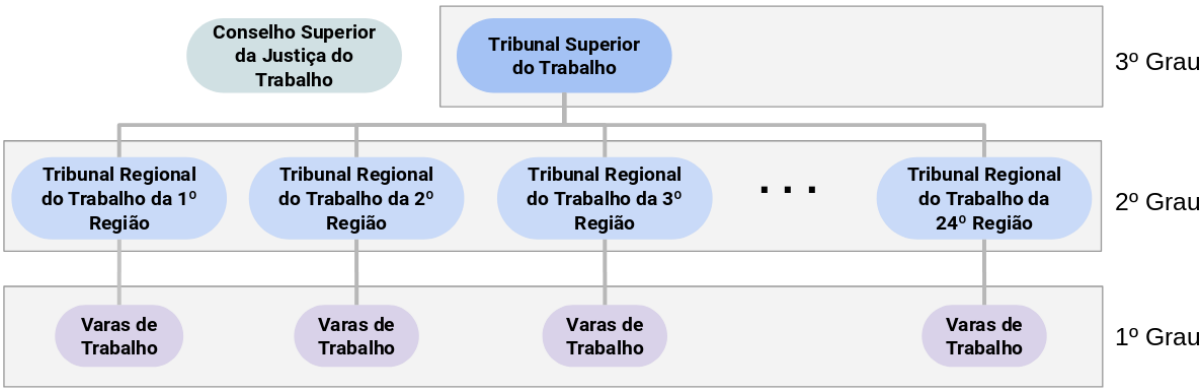

Figura 3.2: Organização da Justiça do Trabalho.

3.2 Processo Judicial Eletrônico da Justiça do Trabalho (PJe)

Estabelecidos os órgãos competentes, os cidadãos que necessitam de julgamento em alguma causa recorrem à abertura dos processos judiciais. Conforme estudo publicado [14], o processo judicial é o canal pelo qual o Estado concretiza a prestação jurisdicional, ou seja, resolve a lide de forma imparcial, entregando às partes envolvidas a solução para o litígio. Atualmente, existe um esforço de informatização de todas as justiças, de forma a viabilizar que os processos jurídicos tramitem da forma eletrônica: *"o que se busca com o processo eletrônico é evoluí-lo de tal forma a alcançar o denominado 'i-processo', em que elementos de inteligência artificial (uso de metadados e algoritmos) servem de ferramenta para auxílio da decisão judicial, notadamente ante a conexão do processo ao mundo virtual de informações."* [14]. Ainda neste trabalho, o autor afirma que o processo eletrônico garante a lisura dos procedimentos envolvidos, além de promover o aumento da segurança jurídica e de viabilizar a rápida resposta do Judiciário.

[2]CSJT: `http://www.csjt.jus.br/missao-visao-valores`

O PJe foi lançado[3] em 2011 pelo CNJ, e vários ramos da justiça já aderiram ao sistema desde então. De acordo com o Relatório Justiça em Números 2019 [12], observa-se que é crescente a quantidade de processos que tramitam em meio eletrônico. Conforme o documento Caderno do Pje[4], publicado pelo CNJ em 2016 , 54 Tribunais dos 90 existentes já haviam implantado o sistema até o momento da publicação do relatório.

No momento do lançamento do PJe, o CNJ distribuiu um sistema único, e cada ramo da justiça fez as alterações necessárias para que fossem atendidas as suas necessidades, e assim foi criado o PJe, especializado para este ramo da justiça. Algumas informações foram padronizadas entre todas as justiças, com o objetivo de promover a uniformização taxonômica e terminológica de classes processuais, movimentação e assuntos, assim, criou-se as Tabelas Processuais Unificadas (TPUs) do Poder Judiciário, de forma que todos os novos processos devem ser criados seguindo esta padronização de nomenclatura.

Analisando-se os dados quantitativos de processo[5], tem-se o cenário mostrado na Figura 3.3. Enquanto, na primeira instância, a quantidade de processos recebidos anualmente até 2018 se posicionou entre 1,7 e 2,8 milhões aproximadamente, na segunda instância este intervalo foi de 640 mil a 1,15 milhão e na terceira instância, esse quantitativo esteve sempre abaixo de 400 mil. Estes dados são bastante expressivos no que se refere à quantidade de demandas em cada instância, mostrando que a primeira instância recebe a grande maior parte dos processos, a segunda instância recebe uma quantia inferior à metade dos dados recebida na primeira, e a terceira lida com um número ainda menor de processos. Importante notar ainda uma queda repentina do quantitativo de processos recebidos na primeira instancia partir de 2017. Isso se deu pela publicação da reforma trabalhista[6], que alterou várias disposições das Consolidações das Leis do Trabalho (CLT), editada em 1943 [7]. Este dado ainda não se refletiu[8] nas demais instâncias devido à natureza recursal das mesmas, que continuam recebendo processos que já estavam em andamento no 1^o grau.

3.2.1 Distribuição do sistema do PJe para os Tribunais

O CSJT, desde que o PJe foi aderido pela JT, é o responsável pela evolução do sistema, de forma que é este o órgão que distribui a versão do PJe a ser usada pela Justiça

[3]Notícia CNJ: http://www.cnj.jus.br/tecnologia-da-informacao/processo-judicial-eletronico-pje

[4]Caderno PJe: http://www.cnj.jus.br/files/conteudo/arquivo/2016/09/551be3d5013af4e50be35888f297e2d7.pdf

[5]Estatística TST: http://www.tst.jus.br/web/estatistica/jt/recebidos-e-julgados

[6]Reforma Trabalhista: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/L13467.htm

[7]CLT: http://www.planalto.gov.br/ccivil_03/decreto-lei/del5452.htm

[8]Notícia TST: http://www.tst.jus.br/noticias/-/asset_publisher/89Dk/content/primeiro-ano-da-reforma-trabalhista-efeitos?inheritRedirect=false

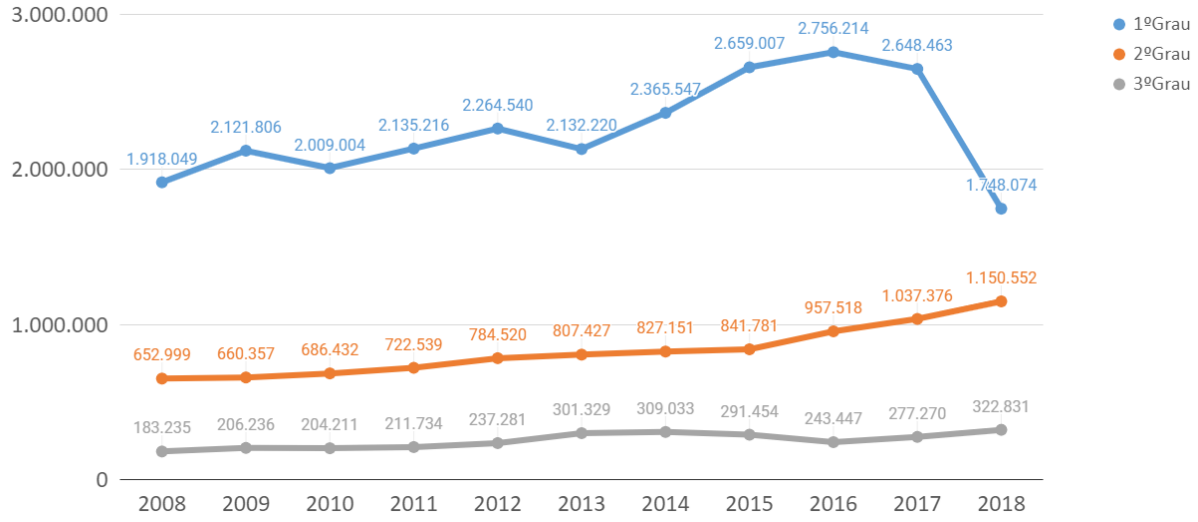

Figura 3.3: Quantidade de processos por instância.

do Trabalho para todos os TRTs e para o TST. Estes órgãos, por sua vez, além de contribuírem ativamente para o desenvolvimento da ferramenta junto ao CSJT, recebem as novas versões do PJe e providenciam a infraestrutura necessária para que o sistema possa ser disponibilizado aos usuários. Atualmente, em cada TRT há uma instalação do sistema para atender ao 1^o grau e uma para atender ao 2^o, e são utilizadas duas bases de dados diferentes, uma para cada grau. Considerando 24 TRTs, com 1^o e 2^o grau, e o TST, como 3^o grau, tem-se 49 bases de dados distintas e independentes.

Vistos os principais pontos referentes à organização e processos da Justiça do Trabalho, bem como do PJe, será apresentada na próxima Seção os conceitos básicos referentes à mineração de textos.

3.3 Aprendizado de máquina

Conforme falado anteriormente, é crescente a quantidade de dados armazenados, o que traz a necessidade de novas técnicas de estudo para se trabalhar com estas informações. A ciência de dados [3], que agrega várias ferramentas para esta análise, se propõe a extrair conhecimento e também inteligência a partir de um vasto conjunto de dados. O conhecimento pode ser construído por meio da identificação de associações, correlações, relações de dependência ou causalidade entre diversos dados, bem como da comprovação de teorias por meio da análise dos dados. Esta análise resulta em novas informações sobre estes elementos, o que compõe uma inteligência específica sobre as forças que regem uma massa de dados, possibilitando que não só se entenda o que aconteceu no passado,

mas também que se possa tentar prever o que virá a acontecer no futuro, observadas determinadas características aprendidas com o dados analisados previamente.

Dentre as diversas ferramentas utilizadas nesta ciência tem-se a área de Aprendizado de Máquina (ML) (*Machine Learning*), um campo da Inteligência Artificial (IA) que estuda formas de sistematizar conhecimentos específicos, construídos a partir da observação de um conjunto de dados coletados, de forma que as máquinas possam entender o conhecimento adquirido [27]. Seu objetivo principal é extrair informações úteis a partir de dados de forma automática [28]. Assim, ela está fortemente ligada ao processo de descoberta de conhecimento em bases de dados (*Knowledge Data Discovery* - KDD) e seu processo de mineração de dados, que engloba o conjunto de técnicas utilizadas para fazer a análise de um grande conjunto de dados, extraindo padrões e novos conhecimentos [29].

Assim, técnicas de modelagem que fazem uso do conhecimento estatístico, computacional e matemático, geram modelos que podem ser simples ou extremamente complexos, capazes de explicitar a relação entre as características observadas (chamadas de variáveis independentes, comumente denominada X) e aquela que se quer entender (chamada de variável dependente ou variável alvo, comumente denominada Y) [27]. A relação entre X e Y pode ser brevemente colocada na forma da Equação 3.1, onde f representa a função desconhecida que mapeia a variável de entrada X com a saída Y. O símbolo ε representa o erro que está associado à função f . O desafio desta ciência está em reduzir o erro associado à esta função. Outra nomenclatura utilizada para a função f é o que chamamos de modelo de ML. [27]

$$\mathrm{Y}= f(\mathrm{X}) + \varepsilon \tag{3.1}$$

Os problemas de ML podem ter dois objetivos principais: a predição ou a descrição. Na predição, faz-se uso dos modelos construídos para se fazer previsões de Y, enquanto na descrição busca-se entender em detalhe como se dá o relacionamento das variáveis independentes com a variável alvo, buscando por exemplo entender as variáveis mais relevantes para a identificação de Y, se o relacionamento se dá de forma positiva ou negativa, de maneira linear ou não, entre outros aspectos. No que se refere à abordagem de aprendizado utilizada, tem-se dois tipos principais: aprendizado supervisionado e aprendizado não supervisionado. No primeiro, parte-se de um conjunto de dados previamente categorizados, para a partir daí, extrair um conhecimento que possa ser aplicado em um novo conjunto de dados. No segundo, busca-se extrair conhecimento de dados que não estão previamente rotulados. Em ambos os casos, é importante se trabalhar com uma amostra de dados que seja grande o suficiente e diversificada o suficiente para ser representativa da população que se estuda [27].

De acordo com [27], dentre as principais atividades englobadas no aprendizado supervisionado, cita-se a classificação, onde busca-se aprender formas de atribuir categorias aos elementos avaliados; e a regressão, onde tenta-se identificar uma formula matemática capaz de fazer a predição de um valor numérico. Já nos métodos não supervisionados, destaca-se o agrupamento (*clustering*) de dados, que busca encontrar grupos de dados com características em comum; e também a redução de dimensionalidade, que busca representar dados de alta dimensionalidade em espaços dimensionais menores. Já no livro [28], cita-se ainda a atividade de associação, uma atividade não supervisionada que busca encontrar regras de associação entre elementos; e a detecção de anomalias, que busca formas de identificar dados que fogem à um padrão, podendo ser abordada de maneira supervisionada ou não supervisionada.

Neste trabalho será explorada a tarefa de classificação, uma técnica de aprendizado supervisionado para predição. Neste tipo de abordagem, conforme explicado, busca-se aprender com dados previamente rotulados, tentando-se estabelecer formas de identificar cada uma das classes de dados. Naturalmente, nesta técnica, espera-se que a base de dados esteja classificada corretamente, de forma a se construir o conhecimento correto [7]. Àqueles dados incorretos, chamamos de ruído (*noise*). A utilização de uma base de dados com ruído, ou seja, que contenha rótulos incorretos, pode confundir os algoritmos de aprendizado, comprometendo gravemente a taxa de acerto dos modelos, ou seja, a quantidade de elementos que os modelos irão fazer a classificação corretamente. Além disso, a presença de ruído aumenta a quantidade necessária de exemplos de treinamento e a complexidade dos modelos, que agora precisarão lidar não só com os dados corretos mas também precisarão separar os dados relevantes dos irrelevantes ou incorretos [6, 7], assim, é de grande importância tentar reduzir o ruído da base.

Definido o conjunto de dados que será utilizado para fazer a criação do modelo, recomenda-se a técnica de validação cruzada dos dados, onde um divide-se o conjunto de elementos entre treinamento e teste. Este passo se faz necessário para que se tenha um conjunto de dados para fazer o ajuste dos parâmetros livres do modelo, ou seja, os parâmetros que podem ser ajustados na função f, enquanto o conjunto de testes é utilizado para avaliar o desempenho do modelo em um conjunto novo de dados, ou seja, o quanto o modelo será capaz de generalizar o conhecimento adquirido para que quando encontre dados ainda não vistos, ele consiga identificar as classes, uma vez que cada elemento da população sempre terá suas especificidades, que poderão ser diferentes das características que apareceram no conjunto de treinamento [28].

Se um modelo só consegue bom desempenho no conjunto de treinamento, e não consegue ter taxa de acerto similar em um conjunto de testes, temos uma situação de sobreajuste (ou *overfitting*), onde o modelo se tornou mais complexo e não só aprendeu as

características determinantes para a tarefa de classificação mas também aprendeu com as especificidades e ruídos dos dados, não sendo adequado para a tarefa de classificação dos novos dados. Enquanto que se o modelo não se capta informações suficientes nos dados de treinamento e fica muito genérico, ele não aprende as características relevantes para a tarefa de classificação e também não consegue atuar em novos dados. Por isso é importante fazer a avaliação do modelo final no conjunto de testes, para que se possa medir seu desempenho em dados desconhecidos, que é onde o modelo será utilizado.

Dentre as formas mais comumente utilizadas de se fazer a validação cruzada, tem-se o *holdout* e o *k-fold* [27]. No *holdout*, um subconjunto de dados escolhidos aleatoriamente é separada para o conjunto de teste enquanto os demais dados são usados para treinamento. Este subconjunto de teste normalmente representa de 20,00% a 40,00% dos dados. A desvantagem deste método é que o desempenho apresentado varia muito em função dos dados que são escolhidos para o treinamento e o teste. Assim, uma forma de se trabalhar com a variância existente nos dados, é o *k-fold*, que trabalha de forma similar ao *holdout*, mas ao invés de separar um único conjunto de teste, separa o conjunto de dados em k subconjuntos, sendo k a quantidade de partes diferentes, onde uma parte será separada para o teste, e $k - 1$ partes serão usadas para compor o conjunto de treinamento. Esse processo é repetido k vezes, até que cada uma das partes tenha sido usada como conjunto de testes. Ao final, calcula-se a média das métricas obtidas em cada uma das iterações, trazendo uma métrica mais confiável. A desvantagem deste método é que ele precisará treinar k-modelos diferentes, exigindo mais tempo de processamento. Segundo [28], não há argumentos conclusivos sobre a quantidade de *folds* a ser utilizada, embora 10 *folds* tenha se tornado a quantidade padrão normalmente utilizada. Estes autores afirmam ainda que estudos empíricos demonstraram que a estratificação melhora significativamente os resultados.

Um problema de classificação de dados pode ser definido [2] como uma das seguintes abordagens: *single-label* ou *multi-label*. No caso *single-label*, o elemento a ser classificado só pode assumir uma única classificação. Quando existem apenas duas opções possíveis para a classificação, chama-se o problema de classificação binária, enquanto quando se tem vários possíveis valores para esta classificação, chama-se de classificação multiclasse. Já no caso *multi-label*, também chamado de multirrótulo, um dado pode ser categorizado em várias classes, e cada uma delas pode ter vários possíveis valore.

Uma outra caracterização que se faz quanto ao problema de classificação diz respeito à forma como as classes alvo estão estruturadas [30]. Tem-se casos onde os rótulos de classificação se apresentam organizados na forma de uma hierarquia de rótulos. Algumas formas diferentes podem ser usadas para classificar este tipo de dados. A primeira delas faz uso da hierarquia dos dados, fazendo com que as informações dos nós pais sejam

utilizadas ao se classificar um dado em algum nó filho. De maneira geral, define-se três formas de se fazer uso da hierarquia: a primeira cria um classificador para cada nó da hierarquia, caracterizando uma classificação multirrótulo; a segunda tem-se um classificador multiclasse para cada nó pai, fazendo com que o dado passe por uma sequência de classificadores até que se classifique em um nó filho; a terceira cria um classificador multiclasse por nível da árvore, sendo esta uma das opções mais utilizadas [30], apesar do risco de inconsistência da classificação ao permitir se escolher um nó pai que não é pai do nó filho escolhido.

A outra forma se se trabalhar com a hierarquia, e a mais simples, consiste em apenas descartar a hierarquia e tratar o problema de forma achatada, ou *flat*, ou seja, todas as classes alvos, sem a informação dos nós pais, irão compor o conjunto de classes alvo de classificação. Devido à simplicidade da desta abordagem, optou-se por trabalhar a hierarquia de assuntos neste trabalho desta forma.

Dados desbalanceados

Antes de criar os modelos de predição, é importante observar como os dados se encontram distribuídos em relação à cada classe alvo. Caso haja uma classe que contenha muito mais exemplares que outra, normalmente os modelos tendem a classificar a maior parte dos dados com a classe utilizada pela maioria dos demais dados. Assim, algumas são as possibilidades de manipulação dos dados para evitar que esse problema aconteça.

Existem três formas principais de lidar com esse problema. Uma delas é modificando o conjunto de dados para alterar a distribuição de dados de cada classe. Pode-se optar por gerar novos dados para as classes menos populadas (*oversampling*), remover dados das classes mais populadas (*downsampling* ou *undersampling*), ou ainda combinar as técnicas de *oversampling* e *downsampling*. Outra forma de lidar com esta configuração de dados é alterar os algoritmos para que levem em consideração o desbalanceamento de dados; ou pode-se combinar as duas alternativas. As técnicas que alteram o conjunto de dados tem sido mais utilizadas do que as técnicas que alteram a forma de funcionamento dos algoritmos [31, 32]. Além de se apresentar como uma solução de fácil implementação, é uma solução que será comum a qualquer algoritmo, ou seja, esse pré-processamento precisa acontecer apenas uma vez. Na maior parte das vezes, o *oversampling* traz resultados melhores que o *undersampling* uma vez que mantém a variabilidade de dados do conjunto original, entretanto, esse último, por reduzir a quantidade de dados, acaba por reduzir também o tempo de treinamento.

No que se refere ao tratamento do desbalanceamento para problemas de classificação multiclasse [33], temos que este se trata de um problema mais difícil de ser contornado, uma vez que a existência de muitas classes pode acabar por apresentar fronteiras de classes

que se sobrepõe no espaço vetorial, dificultando o processo de identificação da classe correta e reduzindo o desempenho dos algoritmos. Embora várias sejam as possibilidades de se atuar no balanceamento dos dados, no caso de problemas multiclasse, identificou-se uma queda de desempenho na aplicação de métodos de *undersampling*, uma vez pode-se acabar removendo elementos importantes para a demarcação das fronteiras de classe [33].

Uma técnica muito usada para os problemas multiclasse tem sido o uso da classificação binária [33], de forma a se contornar a sobreposição das fronteiras entre-classe, havendo duas possibilidades: pode-se se usar a tática um contra um (*one-versus-one* (OVO)) ou um contra todos (*one-versus-all* ou *one-versus-the-rest* (OVR)). Na primeira delas, cria-se um classificador binário para cada par de classes existente, montando-se um parque de $(n) * (n-1)$ classificadores binários, onde n representa o número de classes existentes. Cada elemento é classificado por cada um dos classificadores montados, e a classe que for mais frequentemente escolhida será a classe atribuída ao elemento. Este método é chamado de votação. Já na abordagem um contra todos, a classificação binária se atem a dizer se um elemento é de uma determinada classe ou não, tendo-se um parque de n classificadores, que irão dizer a probabilidade de um elemento pertencer à sua classe alvo. O elemento receberá a classe do modelo que apresentar maior probabilidade.

Normalmente, a abordagem OVO apresenta melhores resultados porque, por fazer uma classificação binária entre duas classes apenas, acaba lidando com uma quantidade menor de dados e apenas duas classes, e por isso, também normalmente trabalha com dados menos desbalanceados [33]. Por outro lado, esta abordagem precisa de mais classificadores do que na abordagem OVR, uma vez que precisa de um classificador para cada par de classes, enquanto na OVR, tem-se um classificador para cada classe. Neste trabalho, optou-se pela abordagem OVR, em função do elevado número de classes.

Outra abordagem que tem apresentado bons resultados na classificação de dados desbalanceados e com elevado ruído é o *ensemble*, onde se combina o resultado de vários classificadores para se obter um resultado mais preciso [34]. Cada classificador é treinando usando um subconjunto de dados diferente. Existem duas formas principais de se fazer a combinação dos modelos. A primeira é o *bagging*, onde modelos são treinados de forma independente e ao final faz-se uma combinação dos votos para a classificação. Já no segundo método, o *boosting*, os classificadores são treinados de forma encadeada, de forma a gerar aprendizados complementares, dando importância ponderada ao voto em função do desempenho de cada modelo. Comparando-se o *bagging* e o *boosting* quando aplicados a um conjunto de dados desbalanceado e com ruído [34], recomenda-se o uso de *bagging*, sem reposição de amostras.

Vistos os principais pontos referentes ao aprendizado de máquina e da tarefa de classificação, passa-se às particularidades da classificação de textos.

3.3.1 Classificação de textos

Dentre as várias áreas da mineração de dados, tem-se a Mineração de Textos (*Text Mining*), cujo objetivo é extrair o conhecimento de conjuntos de textos, onde os dados estão colocados de forma não estruturada, ou seja, não estão organizados em estrutura rígida pré definida, como tem-se em uma tabela por exemplo. A mineração de textos tenta atuar no contexto onde temos uma sobrecarga de informação pela elevada quantidade de textos, combinando técnicas de mineração de dados, aprendizado de máquina, Processamento de Linguagem Natural (*Natural Language Processing*) (NLP), recuperação da informação e gerenciamento do conhecimento [2]. Destaca-se aqui a área de NLP, que é um campo da computação que usa técnicas computacionais para aprender, entender e também produzir conteúdos na forma de linguagem humana [35].

Baseado nos trabalhos [36, 37, 38, 39],pode-se entender as principais etapas do processo de classificação de textos, que é uma das tarefas da mineração de textos de aprendizado supervisionado onde, dado um conjunto de textos previamente classificados, constrói-se um modelo capaz de classificar novos textos. A seguir serão apresentadas em detalhe cada uma das etapas que compõe esse processo.

Pré-processamento do texto

O pré-processamento de dados no contexto da mineração de texto se refere às técnicas que podem ser aplicadas para se transformar dados brutos não estruturados em um conjunto de dados que apresentem determinada estrutura, de forma torná-los aptos para serem utilizados pelos modelos de aprendizado de máquina, ou seja, representar os textos em uma forma de representação específica, com uma estrutura numérica que pode ser entendida pelo computador. A tarefa de mineração de textos é altamente dependente da forma como é feito o pré-processamento dos textos [2].

Um dos primeiros passos aplicados no pré-processamento, comum a quase todas as formas de representação textual, é o processo chamado de tokenização, onde transforma-se uma cadeia de textos em palavras, símbolos ou outros elementos relevantes. Cada unidade destas é chamada de *token*. Uma forma utilizada na tokenização é a extração de n-gramas, que são sequências de palavras que aparecem em uma mesma ordem, que serão interpretadas com um único *token* [36].

Em seguida, alguns outros passos podem ser aplicados. Um deles é a normalização do texto em caixa baixa (ou caixa alta), de forma a reduzir a variabilidade da escrita, possibilitando que palavras escritas de diferentes formas sejam reconhecidas como sendo a mesma palavra. Também é comum a remoção de pontuações e de marcações HTML (se houver), visando que se extraia apenas o conteúdo textual [36].

Outro passo que também pode ser utilizado é a radicalização do texto de forma a remover variações morfológicas das palavras, para que palavras no singular e plural sejam reconhecidas como a mesma palavra, bem como derivações de gênero ou conjugação verbal. Isso pode ser feito de duas formas diferentes. Uma é a lematização (*lemmatization*), onde busca-se transformar verbos conjugados para a forma infinitiva e palavras no plural para o singular. Para aplicar esta técnica, é preciso reconhecer a classe de cada palavra, o que envolve um passo a mais de reconhecimento de parte do discurso (*part-of-speech* (POS)), que é um processo demorado e muitas vezes suscetível a erros. [40, 36] A outra forma de radicalização é a técnica de *stemming*, que faz uma análise mais generalizada para todas as palavras (independente da classe gramatical), removendo os sufixos das palavras mantendo apenas a sua raiz, reduzindo-se a variabilidade das terminologias [41].

Feita a tokenização, tenta-se então reduzir o vocabulário de forma a manter-se apenas as palavras que possam ser mais relevantes para o contexto de análise. Uma técnica usada para este objetivo é a filtragem, onde busca-se remover do texto as palavras que são irrelevantes distinguir uma classe de outra. A forma mais frequente [2] de filtragem se refere à remoção do conjunto de palavras chamado *stopwords*, que é formado por artigos, preposições e outros, que não adicionam valor semântico ao texto. Além da remoção das *stopwords*, muitos sistemas partem para uma abordagem mais agressiva, chegando a remover 90 % a 99 % das palavras. Para tal, as palavras são classificadas em ordem de relevância, onde mantém-se apenas um seleto grupo das palavras mais relevantes [2].

O conjunto final de *tokens* únicos formará o vocabulário que compõe o dicionário de palavras da coleção existente, e cada documento será representado pelo conjunto de palavras que o formam.

Engenharia de *features*

Feito isso, parte-se para as diferentes formas de representação deste conjunto de *tokens* [2], etapa também conhecida por *feature engineering*. Cada *token* a ser representado nos algoritmos de classificação textual é uma característica do documento, termo comumente chamado de *features* na literatura. Recentemente fez-se uma revisão de 233 trabalhos relacionados à tarefa de classificação de textos publicados entre 2013 e 2018, tem-se que mais de 45,00% dos trabalhos trataram sobre as diferentes formas de representação e manipulação das *features*, mostrando que esta é uma área de grande importância dentro do processo de mineração de textos [37].

Uma das formas mais utilizadas [42] é a simples representação de palavras em vetores numéricos. Nesta representação, cada documento d pertencente à coleção de documentos D é representado em um espaço n-dimensional onde n representa a quantidade *features* x existentes na coleção. O vetor $w(d)$, que representa o documento d, é dado pela Equa-

ção 3.2, onde f representa a função que irá calcular a representatividade de cada *feature* x. O conjunto de documentos D é representado pela matriz composta pelo vetor w de cada documento d que compõe o conjunto D.

$$\mathrm{w(d)}=\{\mathrm{f(d,x_1)}, f(d, x_2), ..., f(d, x_i), ..., f(d, x_n)\} \tag{3.2}$$

Há vários métodos que podem ser utilizadas na função f. O mais simples deles é o *bag-of-words* (BOW), onde a função f é dada por $tf_{x,d}$ (*Term Frequency*), que é a quantidade de vezes que cada *feature* x aparece em cada documento d. Existe ainda uma função simplificada do BOW, chamada de modelo booleano [42], que ao invés de contar a quantidade de vezes que a palavra aparece, apenas informa se a apalavra existe no documento, assumindo o valor 1, ou se não existe, assumindo o valor 0.

Um segunda função utilizada para f é o TF-IDF (*Term Frequency - Inverse Document Frequency*)[43], que evolui a fórmula anterior para agora calcular a relevância de cada *feature* levando em consideração a sua frequência em cada documento d (*Term Frequency*), e considerando o quanto uma *feature* é rara ou comum considerando todos os documentos D disponíveis (*Document Frequency*). Na Equação 3.3 é possível encontrar a formula do TF-IDF, onde $idf_{x,D}$ (*Inverse Document Frequency*), dado pela Equação 3.4, é formado em função do número total de documentos, dado por $|D|$, divido pelo df_x, que conta a quantidade de documentos que contém a *feature* x.

$$\mathrm{tfidf}_{x,d,D} = tf_{x,d} \cdot idf_{x,D} \tag{3.3}$$

$$\mathrm{idf}_{x,D} = \log \frac{|D|}{df_x} \tag{3.4}$$

Uma outra função utilizada para f é o BM25, que adiciona ao cálculo utilizado no TF-IDF o tamanho de cada documento, sendo considerada a função ideal para tarefas de recuperação da informação [43]. Nesta função, que é baseada na mesma fórmula do TF-IDF, o componente $idf_{x,D}$ é mantido, e o $tf_{x,d}$ é substituído pela uma nova forma de cálculo $tf*_{x,d}$, dada pela equação Equação 3.6, onde d_l é o tamanho do documento em palavras, e $AvgD_l$ é a média do tamanho dos documentos. Os hiper-parâmetros k e b são adicionados, com a função de ajustar o impacto desejado das tamanho o documento e da frequência das *features* [44].

$$\mathrm{bm25}_{x,d,D} = tf *_{x,d} \cdot idf_{x,D} \tag{3.5}$$

$$\mathrm{tf} *_{x,d} = \frac{tf_{x,d}(k+1)}{k(1-b+\frac{b.d_l}{AvgD_l})+tf_{x,d}} \tag{3.6}$$

Para registro, apesar do amplo uso reportado nos diversos trabalhos citados anteriormente [36, 37, 38, 39] e sua efetividade, este tipo de representação apresenta alguns contrapontos [36]. O primeiro deles é que não se armazena a informação da ordem das palavras, o que traz perda de informação para estes métodos. Outro ponto a ser mencionado é que estes métodos que geram vetores de documentos de alta dimensionalidade, uma vez que cada vetor w terá como colunas a mesma quantidade *features* diferentes que existirem no conjunto de documentos D, podendo chegar a milhões de *features*, o que pode se colocar como uma limitação em função da quantidade de memória disponível para o processamento destas informações. Por fim, este método é incapaz de capturar e representar o significado semântico das palavras, ou seja, não identifica palavras que possam ter significados parecidos (sinônimos) ou os diferentes significados de uma palavra (polissemia).

Seleção de *features*

Trabalhar com o conjunto completo de todas estas *features* pode não ser beneficial para a tarefa de classificação [2]. Além de aumentar o espaço em memória necessário para a criação dos modelos e o tempo de processamento, muitas destas *features* podem não ser determinantes para a tarefa de classificação ou podem ainda ser ruídos , o que dificulta a tarefa de aprendizado, conforme explicado anteriormente. Assim, frequentemente busca-se formas fazer uma seleção de *features*, de forma a tentar encontrar um subconjunto de *features* que seja capaz de otimizar o resultado da tarefa de classificação em conjunto com o a otimização do tempo de execução da classificação. Uma das formas de selecionar estas *features* é no momento de gerar uma matriz TF-IDF por exemplo, onde pode-se usar parâmetros[9] para forçar a exclusão de palavras que aparecem em quase todos os documentos, ou palavras muito raras, que aparecem em pouquíssimos documentos.

Outra forma de se trabalhar com isso são as projeções do espaço vetorial, onde trabalha-se as informações existente de forma a se ter uma nova representação mais concisa do mesmo conjunto de dados [37]. Uma forma bastante utilizada de se reduzir o tamanho do espaço vetorial das *features* é o *Latent Semantic Analysis* (LSA) [45], que busca identificar os principais tópicos t encontrados no conjunto D, representando cada documento d em função de seus tópicos, e não de suas *features* originais, transformando a matriz de documentos D, de dimensão DXn, em uma matriz de dimensão DXk, onde k representa a quantidade de tópicos, que passam a ser as novas *features* de representação

[9]Parâmetros TF-IDF: `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

dos textos. Além de reduzir a dimensionalidade dos dados, esse método tenta capturar informações de sinonímia e polissemia das palavras nos textos.

O LSA atua por meio do Single Value Decomposition [46] (SVD), uma técnica de álgebra linear que faz a fatoração de matrizes e que consegue representar uma matriz de alta dimensionalidade em uma matriz de menor tamanho. O método recebe como entrada uma matriz de termos/documentos (como por exemplo a matriz TF-IDF) e tem como saída uma matriz de tópicos/documentos. Ao utilizar este método, é preciso informar a quantidade de tópicos k a serem extraídos. Este número varia em cada conjunto de dados, é em cada problema poderá assumir um valor diferente [47]. Assim, ao final, ao invés de se trabalhar com uma matriz cuja dimensionalidade das seja *features* seja dada em função da quantidade de *tokens*, será construído um conjunto de *features* que representa a quantidade de tópicos, reduzindo o tempo e a complexidade do processamento.

Seleção de instâncias

Uma outra forma de reduzir a quantidade de dados a serem processados trata-se da seleção de instâncias, ou seja, quais elementos farão parte do conjunto de dados. Conforme se pode ler em [48], a maior parte dos conjuntos de dados contém instâncias que não melhoram a qualidade da classificação, e ainda instâncias ruidosas que diminuem a taxa de acerto dos modelos de classificação. O trabalho analisa diversas abordagens utilizadas na remoção destas instâncias. Os autores informam que de uma maneira geral, não se recomenda a remoção de elementos próximos às bordas do espaço vetorial de uma classe, uma vez que esses elementos são relevantes para a tarefa de classificação, possibilitando diferenciar o limite de uma classe e outra.

Aponta-se ainda a escolha da forma que se irá fazer a remoção de instâncias é altamente dependente da estrutura como as instâncias estão espalhadas no espaço vetorial, podendo as classes serem linearmente separáveis, ou não. Além disso, conforme explicado, se há sobreposição das classes no espaço vetorial, a separação de uma classe de outra se torna mais difícil. Naturalmente, quanto mais classes houver, e quanto mais similares forem as classes, mais difícil é a tarefa de encontrar esta separação, e por consequência, mais difícil se torna a tarefa de fazer a seleção de instâncias de forma adequada, que não prejudique o aprendizado.

Indução de modelos de classificação

Uma vez processado o texto, faz-se uso dos algoritmos de aprendizado de máquina para a classificação dos documentos. A tarefa de classificação da informação é um tipo de aprendizado supervisionado, onde tem-se um conjunto de dados previamente rotulados,

e a máquina é capaz de aprender com essa informação como classificar novos dados não rotulados.

Os algoritmos de classificação podem ser divididos em seis tipos diferentes [28] . Os baseados em conhecimento operam por meio de um conjunto de regras utilizadas para atribuir a classe de um registro. Aqueles baseados em árvore trabalham com um conjunto de regras organizado em árvores onde o nó raiz e os nós intermediários testam atributos dos dados, os ramos representam os resultados e os nós folhas são os rótulos das classes. Os algoritmos conexionistas se estruturam em vários nós, onde os nós contêm diversos tipos de testes e são organizados em uma forma específica de grafos. Aqueles baseados em distância fazem que um elemento assuma a classe dos elementos rotulados que se encontram mais próximos. Os baseados em função possuem funções pré-definidas que tem seus parâmetros ajustados durante um processo de treinamento. Por fim, os probabilísticos encontram a probabilidade de um objeto pertencer à determinada classe analisando-se a distribuição de *features* de cada classe. São exemplos de algoritmos de classificação: Máquinas de Vetores de Suporte (SVM), Random Forest, Naïve Bayes e as Redes Neurais.

SVM

As Máquinas de Vetores de Suporte mapeiam os registros em um espaço n-dimensional, onde cada registro será representado por um vetor de n posições, onde n representa a quantidade de *features*. Depois, tenta-se encontrar um hiperplano neste espaço que seja capaz de separar as classes dos elementos, de forma que a distância do hiperplano à cada ponto de cada classe seja a maior possível. Os pontos que se encontram mais próximos do hiperplano de separação dos dados são chamados de vetores de suporte [49].

Na Figura 3.4, é possível encontrar um hiperplano bidimensional que separa um conjunto do outro. Os vetores suporte se encontram destacados. O SVM faz uso de um *kernel*, que é uma função de transformação do espaço dimensional em um novo formato que facilite a separação dos dados. Vários são os tipos de *kernel* disponíveis para uso no SVM. Neste trabalho, optou-se por trabalhar com o *kernel* Linear. Conforme se pode ler em [50], a maior parte dos problemas de classificação de texto apresentam classes linearmente separáveis.

Random Forest

O algoritmo *Random Forest* cria uma conjunto (também chamado de floresta ou *ensemble*) de Árvores de Decisão, cada uma com um subconjunto de dados, e depois utiliza *bagging* para combinar o resultado de todas as árvores geradas de forma a identificar a classe final por meio de votação [51]. Árvores de Decisão são um tipo de estrutura de dados onde

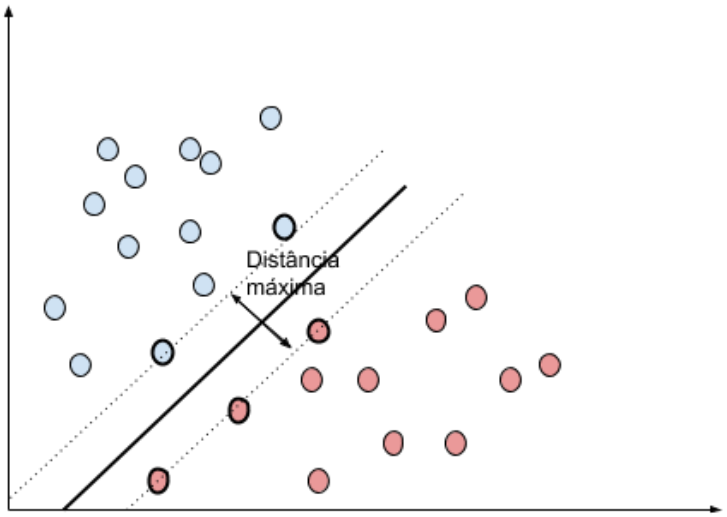

Figura 3.4: Máquinas de Vetores de Suporte.

cada nó interno faz a verificação de um dos atributos, cada ramo representa os possíveis resultados da verificação, e os nós folhas representam as classes a serem atribuídas [28].

Este algoritmo traz certa aleatoriedade ao modelo uma vez que, ao construir as árvores, os elementos que irão ser usados no treinamento de cada árvore são diferente para cada uma delas, o que cria um árvores com um conhecimento diferente. Além disso, cada árvores escolhe um subconjunto de *features* para analisar, o que gera árvores menores. Estas duas características fazem com que as chances de sobre-ajuste do modelo sejam menores do que as chances de sobre-ajuste de uma única árvore de decisão [51].

Naïve Bayes

O algoritmo Naïve Bayes [28] é do tipo probabilístico, e é um classificador estatístico baseado no Teorema de Bayes [52] com o objetivo de identificar a probabilidade de que um dado pertença a uma determinada classe. Esse algoritmo assume a premissa ingênua de que o valor de um atributo em uma determinada classe independe do valor dos demais atributos, de forma a simplificar os cálculos. Uma variação desse algoritmo é o Multinomial Naïve Bayes, que considera uma distribuição multinomial para a probabilidade de cada *feature* de um texto pertencer a uma classe x. Conforme se pode ler em [53], este modelo é mais adequado para problemas de textos com vocabulários de tamanho grande, e também textos de tamanhos variados.

Neste modelo, a probabilidade de um documento d pertencer a uma classe c é explicada pela Equação 3.7, onde x_k são as *features* de cada documento, $P(x_k|c)$ é a probabilidade condicional de uma *feature* x_k aparecer dado um documento de classe c. A probabilidade

$P(x_k|c)$ pode ser interpretada como uma medida do quanto de evidência x_k contribui para que c seja considerada a classe do documento. A probabilidade $P(c)$ é a probabilidade a priori de que um documento pertença a uma classe c, que no caso do Multinomial Naïve Bayes, parte de uma distribuição multinomial [54].

$$P(c|d) \propto P(c) \prod_{k=1}^{n_d} P(x_k|c) \tag{3.7}$$

Redes Neurais

As Redes Neurais [55] são um agregado de funções matemáticas organizados em forma de um grafo direcionado, que recebem como entrada um conjunto de valores numéricos e produz como saída outro conjunto de valores numéricos. Uma das formas mais comuns são as redes Multi-layer Perceptron (MLP). Nesta rede, cada nó no grafo é chamado de *perceptron*, e tem dentro de si uma função matemática de ativação, que transforma os dados de entrada em novos valores. A ligação entre cada *perceptron* recebe um peso, que multiplica os valores de saída de cada *perceptron*. Os *perceptrons* normalmente estão organizados com camadas. Nas redes neurais completamente conectadas, temos uma configuração onde cada os *perceptrons* de uma camada anterior faz uma ligação com cada um dos *perceptrons* da camada imediatamente posterior conforme se pode ver na Figura 3.5.

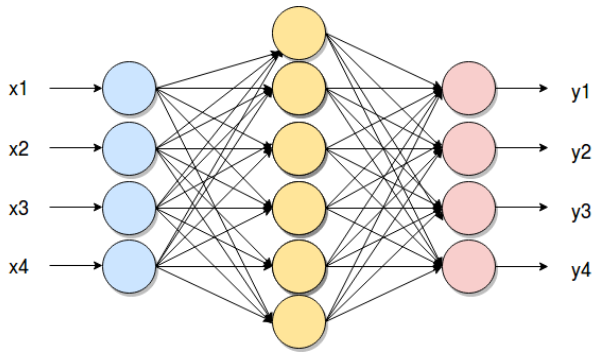

Figura 3.5: Rede Neural.

Uma vez que o dado percorre todos os nós da rede neural, compara-se o valor resultante com o valor esperado, e a diferença entre esses valores é o erro da rede. A ideia então é que se possa ajustar os parâmetros das funções de ativação e os pesos das conexões de forma a minimizar o erro. Uma das formas de se ajustar estes parâmetros é a retropropagação (*backpropagation*), onde o erro encontrado é propagado para os perceptrons das camadas anteriores como forma de fazer a calibragem correta dos valores dos pesos e funções.

As redes neurais podem ser utilizadas também para a tarefa de criar modelos de linguagem, que são modelos preditivos que tem por objetivo aprender a probabilidade conjunta

de uma sequência de palavras em uma linguagem a partir da análise das sequencias de palavras em conjuntos de textos [56, 57, 58]. Por ser uma técnica de aprendizado não supervisionado, estes modelos não necessitam de uma base de dados classificada, o que torna uma enorme quantidade de textos disponíveis aptos para treinamento, ainda que não estejam rotulados. Uma vez extraído o conhecimento da linguagem, este conhecimento pode ser aplicado em outros contextos, de forma a se utilizar o conhecimento adquirido em tarefas que contenham outros conjuntos de dados. Esta transferência de conhecimento entre contextos é conhecida como *transfer learning*, e possibilitou que muitas tarefas de NLP evoluissem, apresentado melhores resultados do que o estado da arte [59].

Os modelos de linguagem introduziram da ideia de *word embeddings* [43, 56], que são a representação interna das palavras dentro das redes neurais, e conseguem representar o significado das palavras com o conhecimento que foi aprendido. Os *word embeddings* são vetores n-dimensionais específicos, criados a partir da análise das relações entre as palavras. Estes vetores tentam representar a relação de uma palavra com as demais, baseado na proximidade e frequência de ocorrência de uma palavra com outra, o que acaba por trazer uma informação semântica para os vetores de palavra. Considerando um plano vetorial de alta dimensionalidade, os vetores de palavras são criados de forma que palavras similares ou palavras que apareçam em contextos similares fiquem posicionadas próximas umas das outras, e palavras de significados distintos com contextos distintos estejam posicionadas longe umas das outras. Assim, os vetores de palavra são bastante sensíveis ao contexto dos textos utilizados para treinamento.

Existem algumas formas diferentes de se gerar estes vetores. A primeira delas é o Word2Vector [60], que analisa janelas de contexto, ou seja, sequências de n palavras, onde n é a quantidade de palavras que irá fazer parte da janela de contexto analisada. A cada vez que uma palavra é encontrada próxima de outra, a relação entre elas é reforçada. Outro método conhecido é o GloVe [61], que não olha apenas as palavras que aparecem próximas em um contexto, mas olha uma matriz de co-ocorrência das palavras que compões todo o corpus de documento. Outra proposta é dada pelo FastText [62], que cria os *embeddings* não a partir das palavras, mas a partir de partes das palavras, sendo capaz de trabalhar melhor com as variações morfológicas das palavras. Mais detalhes sobre cada um destes métodos podem ser encontrados em [43].

Apesar de conseguir identificar palavras com significados similares, os vetores de palavra não conseguem captar os diferentes contextos que uma palavra pode aparecer, e por isso são altamente dependente do conjunto de dados utilizado para treinamento. Embora os vetores de palavras sejam uma das técnicas do estado da arte em NLP, uma das possíveis razões [19] de não ser tão amplamente usado no domínio jurídico é o fato de ser uma técnica melhor aplicada a textos de menor tamanho (como *tweets* ou de comentários

de clientes sobre serviços prestados) devido à quantidade de palavras, pois isso traz uma complexidade maior para o desafio de classificação uma vez que os algoritmos precisam ser capazes de extrair dados informativos de um conjunto muito maior de palavras. Além disso, tem também o fato de este contexto conter um vocabulário muito específico , e conforme mencionado, os vetores de palavra altamente sensíveis ao conjunto de textos e o contexto usado no treinamento dos vetores. Somente em 2017 teve-se a disponibilização de uma série de vetores de palavras da língua portuguesa pelo Núcleo Interinstitucional de Linguística Computacional (NILC)[10], da Universidade de São Paulo. De 17 fontes de dados diferentes usadas para o treinamento, apenas 2 continham textos de conteúdo relacionado a direito, ainda que não sejam treinados exclusivamente com corpus de domínio jurídico.

Com a evolução do poder computacional e a possibilidade de criar redes neurais mais complexas, as redes neurais evoluíram para redes neurais profundas, que são redes que possuem várias camadas internas, com muitos perceptrons, criando o que se chama de aprendizado profundo (*deep learning*), uma forma de aprendizado que consegue extrair novos níveis de entendimento dos dados. A criação de novas arquiteturas de redes neurais, como as redes recursivas, recorrentes e convolucionais, beneficiou estudos da área de NLP, que apresentaram grandes avanços [63]. Dentre eles, cita-se os modelos de linguagem, que passaram a ser capazes de criar *word embeddings* que conseguem entender os diferentes contextos semânticos e diferentes funções sintáticas que uma palavra pode ser usada [64, 65, 66, 67]. Dentre as atividades de NLP que foram beneficiadas com estes avanços, pode-se citar reconhecimento de parte do discurso (*POS tagging*), que busca fazer uma análise sintática das palavras nos textos; sumarização de textos; reconhecimento de entidades nomeadas; tarefas de classificação em geral, com destaque para a análise de sentimento; tradução de textos de uma linguagem para outra; *question-answering* ,usado para treinar respostas a perguntas); sistemas de diálogo (chat-bots); criação de linguagem natural por meio dos modelos de linguagem, entre outros. Maiores informações podem ser encontradas em [63]. Por serem redes neurais profundas, com várias camadas e vários neurônios, indica-se fortemente o uso de GPUs (Unidades de Processamento Gráfico) [68], que são unidades de hardware especializadas em multiplicações de matrizes, apresentando grande velocidade no processamento de uma elevada quantidade de dados.

Explicadas algumas das diferentes formas de se criar os modelos preditivos, passa-se à forma de avaliação destes modelos.

[10]Repositório NILC: `http://nilc.icmc.usp.br/embeddings`

Tabela 3.1: Matriz de Confusão

		Classe Predita	
		Positiva	Negativa
Classe original	Positiva	VP	FN
	Negativa	FP	VN

Avaliação dos modelos

Um modelo preditivo pode ser avaliado de diferentes perspectivas: quantidades de acertos, quantidade de erros, quantidade de elementos que não foram rotulados em determinando assunto, dentre outros. A escolha de qual métrica será utilizada depende do problema que se está trabalhando, devendo ser analisados a importância e o impacto dos resultados finais dos modelos.

Quanto às métricas que serão colhidas neste trabalho, tem-se a precisão, revocação (*recall*) e *F-Measure*. Essas métricas levam em consideração algumas medidas básicas relacionadas à avaliação dos modelos, como definidas a seguir:

VP (verdadeiro positivo), a quantidade de registros classificados como positivos que são da classe positiva.

VN (verdadeiro negativo), a quantidade de registros classificados como negativos que são da classe negativa.

FP (falso positivo), a quantidade de registros classificados como positivos que são da classe negativa.

FN (falso negativo), a quantidade de registros classificados como negativos que são da classe positiva.

Uma das formas de se ter uma ideia geral do desempenho do modelo é construindo uma matriz de confusão, onde mostra-se cada um destes valores [29]. A Tabela 3.1 apresenta um modelo desta matriz. Foi utilizada uma matriz de classificação binária de forma a facilitar o entendimento.

A partir destes valores, pode-se calcular métricas que dão visibilidade ao desempenho de um modelo [29] . Uma dessas métricas é a acurácia, que é a medida de tudo o que se acertou em relação ao total de elementos, dando uma visão geral do desempenho do modelo. Esta métrica é sensível a dados desbalanceados, uma vez que se houver uma classe majoritária que represente 95,00% dos dados, e o modelo classificar todos os dados como sendo desta classe, sua acurácia será de 95,00%, mas ele não terá acertado nenhum elemento da outra classe. A fórmula da acurácia está definida na Equação 3.8.

Outra métrica é a precisão, que considerando todos os itens que foram classificados como positivos (em uma classificação binária por exemplo), mede quantos realmente eram positivos, ou seja, o quão preciso (ou correto) o modelo é quando ele afirma que um elemento é de determinada classe. Já a revocação avalia o modelo de outra perspectiva: de todos os itens verdadeiramente positivos, mede o quantos o modelo disse que eram positivos, indicando se o modelo está conseguindo identificar a maior parte dos elementos de determinada classe ou não está conseguindo classificá-los corretamente. A fórmula destas duas métricas pode ser encontrada Equação 3.9 e na Equação 3.10.

Normalmente , a relação entre estas duas métricas tende a ser inversamente proporcional [29], pois ao fazer o ajuste de um modelo para que aumente sua precisão, ele irá ser mais cauteloso ao classificar um elemento como sendo positivo, se arriscando a fazer esta classificação apenas quando tiver mais certeza, aumentando portanto sua revocação, uma vez que provavelmente menos elementos positivos serão classificados como positivos. Por outro lado, ao tentar aumentar a revocação, o modelo será ajustado para que a maior parte dos elementos positivos sejam classificados como tal, ainda que isso signifique que o modelo generalize mais a regra de classificação e classifique também muitos elementos negativos como sendo positivos, o que irá diminuir sua precisão.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.8}$$

$$Precisão = \frac{VP}{FP + VP} \tag{3.9}$$

$$Revocação = \frac{VP}{FN + VP} \tag{3.10}$$

Assim, uma medida alternativa tenta equilibrar os valores de precisão e revocação, é a F-measure (também conhecida como F_1 score ou F-Score), que pode ser calculada pela média harmônica entre estas duas medidas, conforme mostrado na Equação 3.11.

$$Precisão = \frac{2 \times Precisão \times Revocação}{Precisão + Revocação} \tag{3.11}$$

Quando se trata de problemas multiclasse, estas métricas podem ser colhidas de duas formas diferentes [36]. A primeira delas é a *micro métrica*, que dá igual importância à cada instância, fazendo uma média ponderada de cada classe em função da quantidade de elementos contido em cada grupo. Já na *macro métrica*, é dada igual importância às classes, não levando em consideração a representatividade de cada uma delas.

Considere um conjunto de dados altamente desbalanceado, onde as classes de grande representatividade apresentam alta taxa de acerto, e classes de pouca representatividade apresentam baixas taxas de acerto. No caso da utilização de macro métricas, as medidas

serão puxadas para baixo, uma vez que todas as classes são igualmente consideradas. No caso da micro métrica, neste mesmo exemplo, as métricas serão puxadas para cima, uma vez as classes maiores terão maior influência no resultado final, uma vez que possuem mais instâncias e portanto terão mais peso no resultado final.

A escolha da utilização de macro métricas ou micro métricas depende do contexto negocial. Caso seja importante considerar a taxa de acerto separadamente em cada uma das classes, a macro métrica é a mais indicada. Caso seja importante considerar apenas a taxa de acerto geral da classificação dos elementos, independente da classe, a micro métrica é a mais indicada.

Neste trabalho, opta-se por utilizar a micro-métrica, uma vez que os dados estão altamente desbalanceados, e o maior interesse é que os modelos acertem o máximo de elementos possíveis, independente da classe, dado o caráter inovador da solução, sendo desejável que os modelos acertem o maior número de documentos possível, independente de sua classe.

Explicados os principais conceitos envolvendo os aspectos negociais e técnicos deste trabalho, explica-se a seguir o modelo de desenvolvimento utilizado para a condução dos experimentos.

3.4 Modelo CRISP-DM

Para a aplicação das técnicas de mineração de textos para a classificação dos processos judiciais trabalhistas, este trabalho seguirá as etapas do modelo *Cross Industry Standard Process for Data Mining* (CRISP-DM) [69], que foi estabelecido com o objetivo de definir as 6 etapas principais da mineração de dados, que podem ocorrer em ciclos de interação para o sucessivo aperfeiçoamento da solução. Na Figura 3.6 é possível identificar como estas etapas estão organizadas.

Na primeira etapa, chamada a de Entendimento do Negócio *(Business Understanding)*, busca-se entender o objetivo do trabalho, quais são os principais conceitos negociais e processos negociais envolvidos, entre outros. Esta etapa poderá ser revisitada sempre que se concluir um ciclo do CRISP-DM com o objetivo de validar o trabalho realizado.

A segunda etapa trata do Entendimento dos Dados *(Data Understanding)*, onde se estuda as diferentes formas de abordar o problema negocial com os dados disponíveis. Nesta etapa é comum faz uma análise exploratória dos dados, identificando suas distribuições, correlações entre os elementos, entre outros. A qualidade do dado é analisada com cuidado para que se possa verificar a confiabilidade das informações.

A terceira etapa trata da Preparação dos Dados *(Data Preparation)*, onde faz-se as manipulações necessárias para que os dados estejam em um formato apropriado para serem

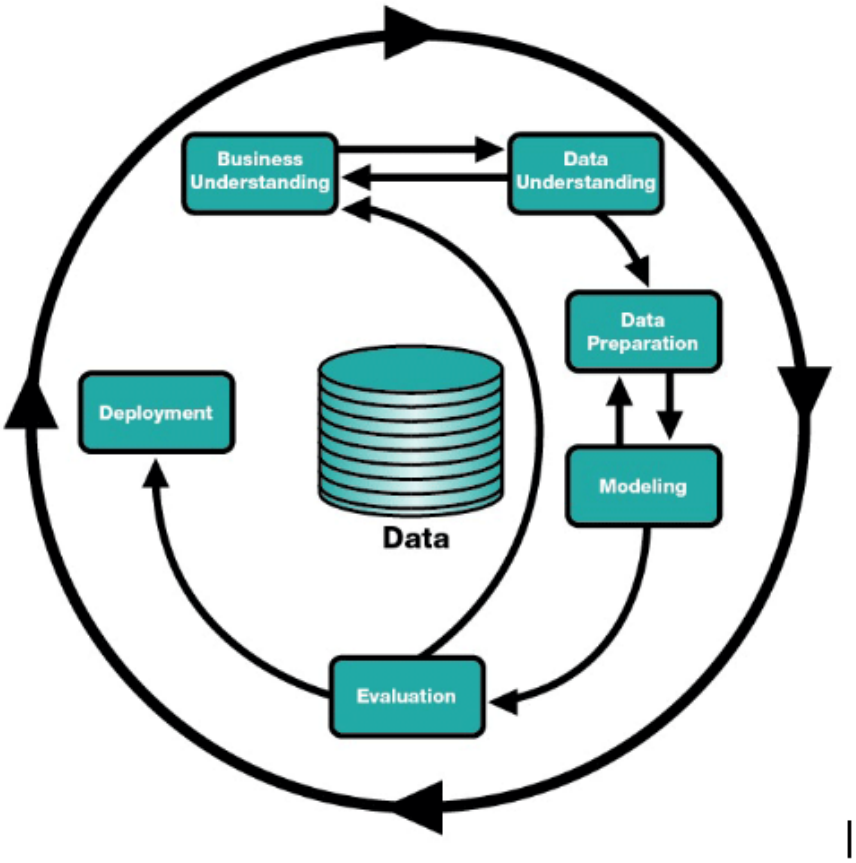

Figura 3.6: Modelo CRISP-DM (Fonte: [69]).

utilizado pelos algoritmos de classificação. Nesta etapa os valores faltantes são tratados, informações são normalizadas, pode-se aplicar técnicas de redução de ruído, balanceamento, redução de dimensionalidade, entre outras manipulações que podem beneficiar a criação dos modelos.

De posse do conjunto de dados trabalhados, passa-se à quarta etapa: *(Modeling)*, que constrói o modelo de mineração de dados. Nesta etapa escolhe-se os métodos a serem utilizados, ajusta-se os parâmetros da maneira mais adequada e cria-se um modelo a partir de um subconjunto de dados. Esse modelo é testado com conjuntos de dados desconhecidos de forma a medir seu desempenho. A métricas mais adequadas ao contexto são escolhidas e aferidas, e a partir desta análise, caso o desempenho não tenha se mostrado satisfatório, reavalia-se os as decisões tomadas ao longo do projeto, voltando à etapas anteriores e abordando o problema de formas diferentes que se tenha um modelo cujo desempenho seja satisfatório para o contexto em questão.

Uma vez criado o modelo, é preciso avaliá-lo do ponto de vista negocial. Assim, na quinta etapa, chamada de Avaliação *(Evaluation)*, os especialistas negociais fazem uma

análise dos resultados apresentados pelo modelo. Caso entenda-se que os resultados não atendem à demanda, retoma-se os passos anteriores de forma a construir um novo modelo que seja mais adequado.

Caso o modelo seja aprovado, passa-se à sexta e última fase, que á a Implantação *(Deployment)* do modelo. Uma vez que ele se encontra em um ambiente de produção, é importante que haja uma constante monitoração para verificar a necessidade de possíveis ajustes.

Capítulo 4

Desenvolvimento da Pesquisa e Resultados

Neste Capítulo será apresentada cada etapa do processo de desenvolvimento deste trabalho, desde o conhecimento construído na etapa de entendimento do negócio até a avaliação dos modelos criados. Como o trabalho seguiu o modelo de desenvolvimento CRISP-DM, apresentado na Seção 3.4, em alguns momentos foi necessário retomar etapas anteriores para buscar uma nova abordagem de lidar com o problema. Assim, cada iteração se encontra descrita nos próximos itens.

4.1 Entendimento do negócio

Nesta etapa, busca-se entender a relevância do assunto dentro do contexto jurídico trabalhista e como é a relação dos assuntos com o fluxo processual dentro do PJe. Parte deste entendimento já se encontra apresentado na Seção 1.1.

O assunto de um processo está diretamente ligado aos *pedidos* que são realizados em uma ação trabalhista. O *pedido* é o objeto da ação, é o que faz com que o autor do processo recorra à Justiça do Trabalho para ter o seu direito restaurado [70]. Os processos trabalhistas são caracterizados por conter, na maior parte das vezes, mais de um pedido. Raramente tem-se processos com um único pedido [71]. Isto se dá em razão da característica da relação de trabalho, onde costumeiramente tem-se múltiplas lesões de direito ao longo da relação empregatícia [72], que é amparada pela possibilidade da cumulação de pedidos [70], determinada pelo artigo 327 do CPC, estando alinhada com o princípio da economia processual. Evita-se, assim, a instauração de múltiplas ações trabalhistas para cada um dos pedidos [73]. Normalmente, para cada pedido, há um assunto distinto para o processo.

No PJe apenas a indicação do assunto principal é obrigatória no momento do protocolo - as demais marcações opcionais. No Manual de Utilização das TPUs, indica-se que o assunto principal do processo deve ser o primeiro assunto do processo, mas não há indicação de como identificá-lo nos processos da justiça do trabalho e não se encontrou outra literatura que indique de forma objetiva como fazer a distinção do assunto principal. Em alguns processos esta escolha é mais clara, mas em outros processos a escolha não é evidente, podendo inclusive não haver um assunto que seja mais importante do que o outro, tornando esta escolha subjetiva. Como até o momento se desconhece a publicação de outro trabalho que tenha tentado atuar especificamente neste problema, pelo caráter pioneiro deste trabalho neste domínio, optou-se por tentar classificar apenas o assunto principal, que atualmente, é obrigatório e único no sistema PJe.

Os pedidos de um processo são informados pelo advogado no momento da criação do processo em determinada instância e, a partir dos pedidos, pode-se obter os assuntos do processo. Conforme Manual de Utilização das TPUs, pode-se identificar o assunto do processo no documento de petição inicial, na parte que se refere aos fatos, logo após a citação das partes, ou na parte final. Já em graus recursais, como por exemplo o 2^o grau, os processos são editados pelos servidores para que sejam remetidos ao grau superior com as informações que serão julgadas naquele grau, e neste momento os servidores podem identificar os assuntos a serem remetidos nos documentos que contém o relatório da decisão recorrida.

Como no primeiro grau a classificação é feita por apenas um usuário, de perfil advogado, e no segundo grau essa informação é conferida e editada pelos servidores internos, optou-se por delimitar o escopo deste estudo aos processos que tramitam no 2^o grau, uma vez que a informação neste grau provavelmente já está mais correta do que a informação armazenada no 1^o grau.

Assim, é preciso entender quais são os documentos relevantes para análise nesta jurisdição. Para que um processo seja autuado neste nível, existem duas possibilidades. A primeira delas trata dos processos que são originários na segunda instância, ou seja, a competência do julgamento do processo em questão deve ser apreciada pela segunda instância, não cabendo julgamento pela primeira instância. Assim, os advogados protocolam o documento do tipo Petição Inicial diretamente na segunda instância.

A segunda possibilidade trata dos processos que já foram julgados em primeira instância, mas que uma das partes (ou ambas) discordam da decisão dos magistrados, de forma que recorrem à segunda instância para que julgue os itens em discordância. Os advogados podem recorrer elaborando um dos seguintes tipos documentos, que são protocolados ainda em primeira instância: Agravo de Instrumento em Agravo de Petição, Agravo de

Instrumento em Recurso Ordinário, Agravo de Petição, Recurso Adesivo, Recurso Ordinário.

Uma vez anexados um desses documentos ao processo e preenchidas as demais informações necessárias, o processo é remetido da primeira à segunda instância, e passa a tramitar neste grau. Os demais documentos que são juntados posteriormente à chegada dos processos são menos relevantes para a análise pois, nesse momento, os servidores já terão lido os documentos iniciais e identificado manualmente o assunto do processo. A maior parte dos processos que tramitam em segunda instância são processos provenientes da primeira instância.

Os documentos que chegam à segunda instância são:

Petição inicial: É a peça processual que dá início ao processo, abrangendo os fatos ocorridos, os fundamentos jurídicos e o pedido, apresentando ao juiz as informações necessárias para o julgamento da causa. Está descrita no artigo 319 do CPC e também no artigo 840 da CLT.

Recurso Ordinário: Trata-se de recurso de fundamentação livre cabível contra sentenças definitivas e terminativas proclamadas na primeira instância buscando uma reforma da decisão judicial que foi elaborada por um órgão hierarquicamente superior [74]. O Recurso Ordinário é regulamentado pelo artigo 895 da CLT.

Agravo de petição: Recurso utilizado para contestar decisões definitivas que aconteceram na fase de execução, visando rediscutir a penhora e os cálculos da liquidação [74, 75].

Recurso adesivo: Acontece quando as duas partes envolvidas no processo vencem e são vencidas em um processo. Assim, se uma das partes discordar da decisão e entrar com um recurso, a outra parte, ainda que inicialmente tenha optado por não recorrer à segunda instância dentro do prazo inicial, poderá protocolar recurso adesivo fora do prazo recursal original. É um recurso sub-ordinário, uma vez que está condicionado à existência do recurso principal protocolado pela parte contrária [74].

Agravo de instrumento: é uma forma de contestar decisão que tenha negado que um recurso já protocolado subisse à instância superior, ou seja, a parte entrou com um recurso, o recurso foi negado dentro da mesma instância, então a parte entra com agravo de instrumento para que ainda assim o processo seja apreciado pela instância superior. O agravo de instrumento pode ser sobre o Agravo de petição ou sobre o Recurso Ordinário, o que caracteriza os documentos de Agravo de Instrumento em Agravo de Petição e o Agravo de Instrumento em Recurso Ordinário, respectivamente. [76]

Portanto, é natural que nestes documentos estejam contidas as informações relacionadas pedidos do processo e, por consequência, aos assuntos que devem ser julgados pela 2^a instância. Assim, são documentos apropriados para a identificação dos assuntos processuais.

4.2 Entendimento dos dados

Nesta etapa, busca-se entender a distribuição dos dados que serão trabalhados por meio de uma análise exploratória.

4.2.1 Documentos

A partir de uma consulta nas bases de dados, fez-se uma análise da quantidade de documentos relevantes para o contexto da pesquisa, descritos na seção anterior. A Figura 4.1 mostra como estes documentos são distribuídos, a partir de dados foram colhidos das bases de 2^o grau de 22 TRTs[1]. Nota-se que a maior parte de documentos são Recursos Ordinários.

Considerando que os demais documentos nem sempre estão presentes em todos os processos e que a grande maioria dos processos encaminhados ao 2^o grau possui pelo menos um documento do tipo Recurso Ordinário, decidiu-se trabalhar com esse documento para fazer a classificação dos processos judiciais. Esta escolha se mostra viável uma vez que abrange a maior parte dos processos, resolvendo, portanto, o problema de classificação para a maior parte dos dados. O ponto negativo desta escolha é que os modelos não serão adequados para atuar em processos que não contenham o Recurso Ordinário.

Para exemplificar, as Figuras 4.2 a 4.4 apresentam o extrato de um Recurso Ordinário, cujas informações sensíveis foram escondidas. Nota-se que o documento é dirigido à Vara de Trabalho, na primeira instância, pois é ela quem julga se o Recurso Ordinário está apto para ser enviado à segunda instância.

Neste ponto, optou-se por fazer um recorte dos dados para que se trabalhasse apenas com processos que contenham um único Recurso Ordinário. Isto porque, de acordo com a organização dos dados no PJe, o assunto é vinculado ao processo, não havendo ligação direta de assuntos com documentos. Assim, se um processo tem mais de um Recurso Ordinário, cada um sendo juntado por uma parte diferente, é possível que um documento trate de um grupo de assuntos, e o outro documento de outro grupo de assuntos. Os dois grupos de assuntos juntos compõem o grupo de assuntos do processo, mas até o momento,

[1]O CSJT tem acesso à uma cópia de alguns dados da base dos 25 Tribunais, entretanto, nem todas estão disponíveis 24 horas por dia, podendo haver eventuais indisponibilidades. As bases do TRT da 15^a e 24^a região não estavam disponíveis no momento da recuperação destes dados

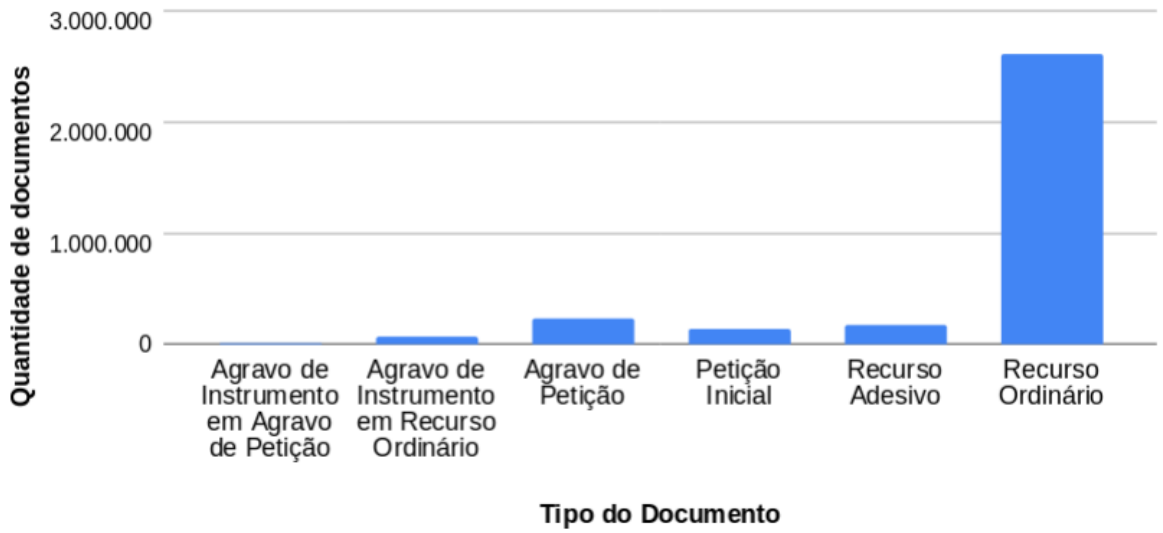

Figura 4.1: Quantitativo de documentos na segunda instância.

não há como saber qual assunto é tratado em qual sem que se tenha que fazer a leitura dos Recursos Ordinários.

4.2.2 Assuntos

O objeto de estudo deste trabalho está relacionado com a identificação do assunto processual. A TPU de Assuntos Processuais define a terminologia jurídica a ser utilizada para a categorização dos processos, e esta informação se refere ao conteúdo, à temática, à matéria do processo. Baseado na tabela disponibilizada pelo CNJ [2], o TST, dentro de sua competência, acrescentou assuntos específicos da JT que entendeu serem necessários à este ramo da justiça e removeu aqueles que não cabem neste contexto, transformando a TPU original em uma específica para a JT [3].

A tabela de assuntos está organizada de forma hierárquica, e quanto maior o nível do assunto, mais especificado estará. Atualmente, a tabela conta com 5 níveis de hierarquia, sendo 1131 assuntos ao todo, que estão distribuídos conforme a Tabela 4.1. Na Figura 4.5, tem-se um exemplo extraído da TPU de Assuntos Processuais do TST, onde pode-se entender como está organizada esta informação: no primeiro nível tem-se o assunto "Direito do Trabalho", abaixo dele tem-se o segundo nível com o assunto "Categoria Profissional Especial", em seguida, ocupando um 3^o nível, tem-se o assunto "Bancário", do qual mostra-se 5 assuntos filhos no nível 4, sendo um deles o "Enquadramento", dividido em

[2]Tabela do CNJ: `https://www.cnj.jus.br/sgt/consulta_publica_classes.php`
[3]Tabela do TST: `http://www.tst.jus.br/web/corregedoria/tabelas-processuais`

EXCELENTÍSSIMO SENHOR JUÍZ DA [redacted] VARA DO TRABALHO DE [redacted].

Processo nº [redacted]

[redacted], reclamada, já qualificada nos autos do processo em epígrafe, por meio do seu advogado, já habilitado, vem, perante Vossa Excelência, interpor **RECURSO ORDINÁRIO**, com fundamento no artigo 895, I, da CLT, comprovante do depósito recursal e das custas anexo, requerendo que, depois de cumpridas as demais formalidades legais, sejam as razões deste remetidas ao Egrégio Tribunal Regional do Trabalho da [redacted] Região, para apreciação e julgamento.

São os termos.

[redacted], 22 de janeiro de 2019.

[redacted]

OAB/ [redacted]

[redacted]

OABB/ [redacted]

RAZÕES DO RECURSO ORDINÁRIO DO RECORRENTE

Processo nº: [redacted]

Recorrente: [redacted]

Recorrido: [redacted]

Egrégio Tribunal,

Doutos Julgadores.

Data venia, a decisão que julgou a reclamatória aforada pelo recorrido parcialmente procedente merece ser reformada no que diz respeito aos pleitos de indenização pelo período da estabilidade correspondente a 12 meses de salários, além de 13º salário proporcional, férias proporcionais + 1/3 e FGTS + 40% (inclusive sua incidência sobre o 13º salário), todos referentes ao período da estabilidade;

- aviso prévio indenizado (33 dias), com reflexos em FGTS;

- férias integrais de 2016/2017 e proporcionais de 2017/2018 (7/12), ambas acrescidas do terço constitucional;

- FGTS mais a multa de 40% no período pleiteado pelo reclamante;

Figura 4.2: Exemplo de Recurso Ordinário (Página 1).

- indenização pela ausência de entrega das guias de seguro desemprego no valor correspondente ao que o reclamante teria direito a receber a este título;

- multa do art. 477, §8º da CLT, pois não está de acordo com o entendimento doutrinário, com entendimento jurisprudencial, com o ordenamento jurídico sobre a matéria e com as provas produzidas durante a instrução processual.

DA JUSTA CAUSA/ RENÚNCIA A ESTABILIDADE

Doutos julgadores a r. sentença de primeiro grau merece ser reformada, pois ao cessar o benefício do recorrido junto ao INSS, a recorrente entrou em contato com ele para que retornasse as atividades laborativas, sem lograr êxito, porém no dia 12-03-2018 o recorrido compareceu na recorrente e na ocasião o recorrido recebeu autorização para fazer exame médico para retorno ao trabalho, tendo recebido sua contrafé conforme anexo, porém ele não compareceu na clínica para fazer o exame.

Conforme certidão cartorial em anexo, no dia 04-04-2018 a recorrente protocolou junto ao Cartório [redacted], notificação extrajudicial para que o recorrido, retorna-se a exercer suas atividades laborativas ou apresentar alguma justificativa para as faltas que vinham ocorrendo, no prazo de 10 dias, sendo que no dia 09/04/2018 o recorrido foi notificado, via cartório, para retornar as suas atividades laborais, conforme notificação anexa, porém o mesmo não retornou.

Com fim do prazo de 10 dias para que o recorrido retornasse a recorrente o mesmo não retornou, sendo assim seu contrato de trabalho foi considerado extinto por justa causa, nos termos do art. 482, i, da CLT, em 20-04-2018, sendo que a recorrente entrou em contato com o recorrido para que ele comparecesse na sua sede para receber suas verbas resolutórias, porém o mesmo não compareceu para o recebimento das verbas.

Conforme dito acima, a recorrente tentou por diversas vezes entrar em contato com o recorrido, para que ele retornasse as suas atividades laborativas e, apesar de notificado para esse fim, ele não retornou ao seu emprego e por esta razão foi demitido por justa causa, por abandono de emprego, com fundamento no art. 482, "i", da CLT.

Portanto, devido a demissão por justa causa, por abandono de emprego, não há o que se falar em estabilidade do recorrido, pois ao não retornar ao seu emprego, presumiu-se que ele renunciou a sua estabilidade.

Portanto, diante do abandono do emprego praticado pelo recorrido, ficou claro que foi correta a justa causa aplicada pela recorrente, não se podendo falar em estabilidade, pois com o abandono de emprego ficou subentendido, que o recorrido abriu mão do seu direito de estabilidade provisória.

Assim, a r. sentença de primeiro grau merece ser reformada para que a ação seja julgada totalmente improcedente, tendo em vista a renúncia do suposto período de estabilidade do qual o recorrido renunciou.

Por outro lado, a recorrente informou ao juízo de primeiro grau que caso ele entendesse que o recorrido faz

Figura 4.3: Exemplo de Recurso Ordinário (Página 2).

jus ao retorno de seu emprego, a recorrente não se opõe que ele seja READMITIDO, bem como que seja encaminhado para PERÍCIA MÉDICA NO INSS para atestar sua suposta incapacidade e, se for o caso, sua readequação em uma nova função.

Sendo assim caso esta Egrégia Turma entenda que o recorrido tem direito ao período estabilitário, está recorrente não se opõem em readmitir o recorrido e resguardando o período de estabilidade de 12 meses.

Tendo em vista que a justa causa aplicada foi correta a r. sentença de primeiro grau de ser reformada para que julgue totalmente improcedente os pleitos de de aviso prévio, multa de 40% sobre o FGTS, guias ou indenização substitutiva do seguro desemprego, férias proporcionais 2017/208 com 1/3.

Outrossim, caso esta Egrégia Turma entenda por manter a r. sentença de primeiro grau, fato este que não se acredita que ocorrerá, uma vez que a recorrente agiu de forma correta ao aplicar a justa causa por abandono de emprego, uma vez que a mesma não pode ser penalizada pelo abandono de emprego que o recorrido causou, requer que a condenação em indenização substitutiva do seguro desemprego, seja revertida para entrega das guias para habilitação no seguro desemprego.

HONORÁRIOS SUCUBENCIAIS

Os honorários em epígrafe são devidos em caso de sucumbência, todavia, após as fundamentações acima, conclui-se que a recorrente logrará êxito na reforma da r. sentença de primeiro grau, sendo assim requer que a r. sentença de primeiro grau seja reformada para que julgue improcedente o pleito de honorários sucumbenciais.

DO PEDIDO

Ante o exposto, a recorrente requer que, seja recebido e conhecido o Recurso Ordinário, bem como, com o seu provimento, seja a r. decisão de primeiro grau reformada, para que a ação seja julgada totalmente improcedente.

São os termos.

[redacted], 22 de janeiro de 2019.

[redacted]

OAB/[redacted]

[redacted]

OABB/[redacted]

Figura 4.4: Exemplo de Recurso Ordinário (Página 3).

Tabela 4.1: Quantidades de Assuntos

Nível de Hierarquia	**Quantidade de Assuntos**
1	5
2	57
3	420
4	539
5	110
Total	*1131*

4 especializações de 5º nível. Na coluna mais à direita de cada um dos assuntos, pode-se encontrar o código único de cada um.

ASSUNTOS	Nível 1	Nível 2	Nível 3	Nível 4	Nível 5
DIREITO DO TRABALHO	864				
2 Categoria Profissional Especial	864	7644			
3 Bancários	864	7644	5280		
4 Cargo de Confiança	864	7644	5280	55312	
4 Chefia	864	7644	5280	55025	
4 Colocação ou Venda de Papéis / Valores Mobiliários	864	7644	5280	55316	
4 Divisor de Horas Extras	864	7644	5280	55317	
4 Enquadramento	864	7644	5280	55026	
5 Categoria Diferenciada	864	7644	5280	55026	55318
5 Financeiras / Equiparação Bancário	864	7644	5280	55026	55319
5 Empresa de Processamento de Dados	864	7644	5280	55026	55027
5 Isonomia/Diferença Salarial	864	7644	5280	55026	55028

Figura 4.5: Extrato da Tabela de Assuntos (Fonte: [77]).

Analisando a distribuição dos processos de acordo com os assuntos, considerando consulta executada nas bases de 2º grau dos mesmos 22 TRTs mencionados anteriormente, e o nível do assunto principal em relação à quantidade de processos, tem-se a imagem da Figura 4.6, onde nota-se que a maior parte dos processos está categorizada com assuntos do nível 4. Nem todos os assuntos possuem um nível mais detalhado (como 4 ou 5 por exemplo). Neste trabalho decidiu-se arbitrariamente fazer a classificação no nível 3 da tabela dado que nesta classificação os processos classificados em nível 4 e 5 também estarão rotulados, ainda que de forma mais genérica.

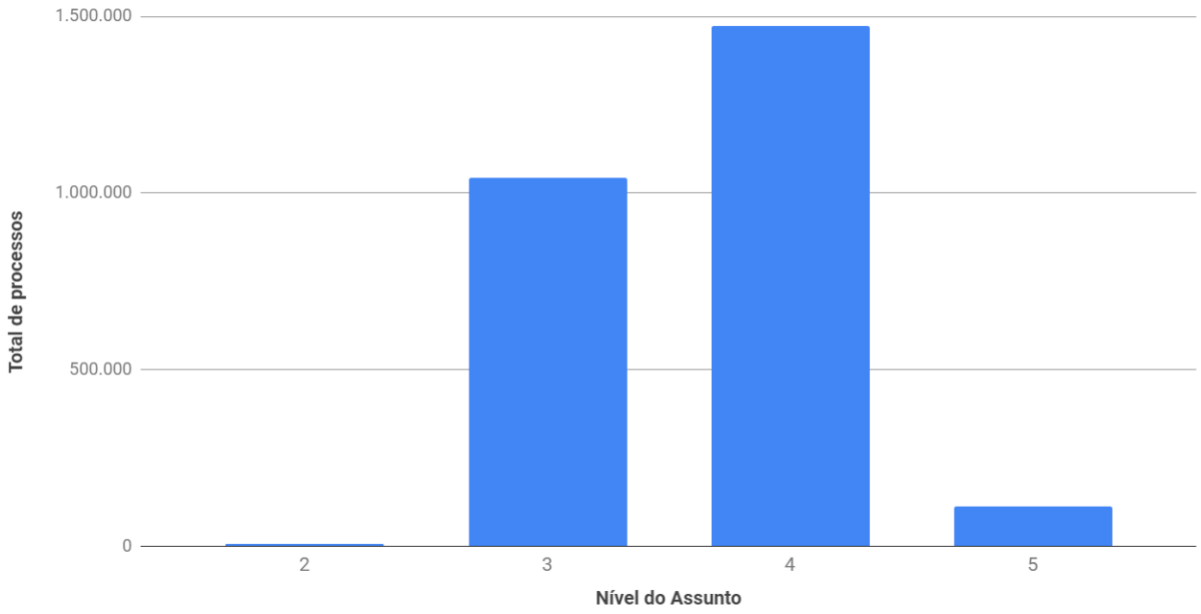

Figura 4.6: Distribuição da quantidade de processos de acordo com o nível do assunto principal.

No caso do processo cujo Recurso Ordinário foi apresentado como exemplo, encontrou-se os assuntos abaixo vinculados. Primeiramente tem-se o código de cada assunto utilizado na classificação, e em seguida tem-se a descrição do assunto completo, ou seja, toda a hierarquia que compõe este assunto. O nível 3 de cada assunto se encontra destacado em negrito. Nota-se que de 9 assuntos, 5 são do 3^o nível. O assunto principal é o de código 1907 "Justa Causa / Falta Grave ".

- **2210** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Verbas Rescisórias (2546)** / Multa do Artigo 467 da CLT (2210)
- **2212** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Verbas Rescisórias (2546)** / Multa do Artigo 477 da CLT (2212)
- **1998** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Verbas Rescisórias (2546)** / Multa de 40% do FGTS (1998)
- **2478** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Seguro Desemprego** (2478)
- **55566** - DIREITO PROCESSUAL CIVIL E DO TRABALHO (8826) / Ação Rescisória (55301) / **Honorários Advocatícios** (55566)
- **2546** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Verbas Rescisórias** (2546)

Tabela 4.2: Quantidades de documentos por assuntos no primeiro nível

Macro Assunto	**Quantidade de Processos**	**Percentual de Processos**	**Quantidade de assuntos no nível 3**
Direito do Trabalho	2.316.628	87.91%	205
Direito Administrativo e Outras Matérias de Direito Público	3.173	00.12%	71
Direito Civil	112	00.00%	8
Direito Internacional	140	00.00%	2
Direito Processual Civil e do Trabalho	315.105	11.96%	139

- **1907** - DIREITO DO TRABALHO (864) / Rescisão do Contrato de Trabalho (2620) / **Justa Causa / Falta Grave** (1907)
- **55257** - DIREITO PROCESSUAL CIVIL E DO TRABALHO (8826) / Jurisdição e Competência (8828) / **Competência (8829)** / Prevenção (55257)
- **2554** - DIREITO DO TRABALHO (864) / Contrato Individual de Trabalho (1654) / **Reconhecimento de Relação de Emprego** (2554)

Especialistas do Grupo Nacional de Negócio (GNN) do PJe (dois Juízes do Trabalho e uma Diretora de Secretaria), que é o grupo detentor do conhecimento negocial da equipe de desenvolvimento do PJe, sendo responsável por fazer solicitação, priorização e homologação das demandas negociais do PJe, foram consultados durante o desenvolvimento deste trabalho e disseram que o nível 3 contém informação detalhada o suficiente para trazer utilidade em nível negocial.

Em seguida, analisou-se distribuição dos processos em relação ao primeiro nível da árvore de assuntos, onde tem-se 5 assuntos que definem as grandes áreas do Direito. Neste trabalho, estes assuntos serão referenciados como *macro assuntos*. Na Tabela 4.2 tem-se a distribuição do assunto principal dos processos de acordo com cada um desses macro assuntos, bem como a distribuição da quantidade de assuntos existentes no nível 3 (alvo da classificação nesse projeto) de cada macro assunto. Ao todo, existem 420 assuntos de nível 3, entretanto 87.91% dos documentos se encontram concentrados no assunto Direito do Trabalho, que é a competência de julgamento da Justiça do Trabalho. Assim, de forma a delimitar um conjunto de dados que tenha informações suficientes para o treinamento do modelo, optou-se por trabalhar apenas com os assuntos descendentes dos assuntos Direito do Trabalho. Diminui-se, portanto, a quantidade de assuntos a serem usados na classificação para 205 assuntos.

O próximo passo foi a análise da distribuição de processos de acordo com a classificação do assunto principal no nível 3 do Direito do Trabalho,. A Figura 4.7, baseada na distribuição dos processos na base de treinamento, conforme tabela disponível no Apêndice A, apresenta uma versão gráfica destas informações. Nota-se que há uma grande quantidade de processos concentrada em poucos assuntos. Além disso, alguns têm quantidade muito pequena de exemplos e há um número considerável de categorias, fatos que dificultam o problema de classificação.

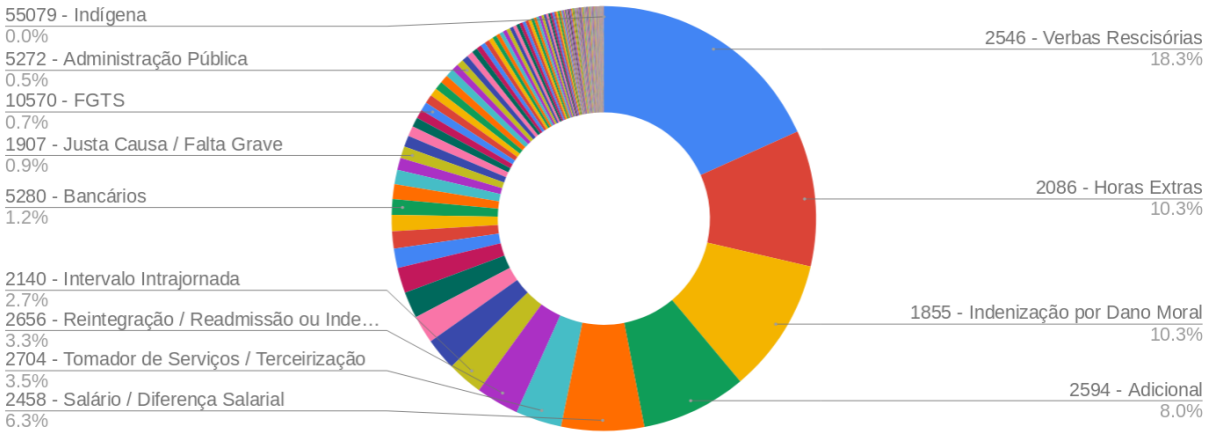

Figura 4.7: Distribuição de processos por assunto principal (Nível 3) no conjunto de treinamento.

Fez-se então a análise da distribuição cumulativa destes dados, de forma a encontrar a quantidade de assuntos que seriam utilizadas para treinamento, buscando fazer um corte que possibilitasse classificar a maior parte da base dentre aqueles classificados dentro do macro assunto do Direito do Trabalho, atendendo à premissa de haver exemplos suficientes para treinamento. Assim, foram escolhidos 36 assuntos, que explicam 90.15% da base, atuando então na maior parte do problema. Na Figura 4.8, que traz um gráfico do percentual acumulado dos processos em função do assunto, é possível acompanhar este corte.

Os assuntos escolhidos se encontram na Tabela 4.3,Tabela 4.4 e Tabela 4.5, onde é possível ver o assunto superior de nível 2, o assunto de nível 3, a quantidade de processos disponíveis e o percentual de representatividade acumulado. Nota-se que existem dois assuntos de Indenização por Dano Moral, um de código 1855, e um de código 55220. Conforme revisão dos especialistas, estes dois assuntos serão considerados como um só, sendo agrupados no assunto de código 1855. O assunto de nome FGTS, que também aparece duas vezes, já trata de questões diferentes, não sendo possível unificá-los. A distribuição acumulada de todos os processos se encontra disponível no Apêndice A.

Tabela 4.3: Assuntos escolhidos para a criação dos modelos de classificação (Parte I.

Assunto Nível 2	**Assunto Nível 3**	**Quantidade de Processos**	**Percentual Acumulado**
2620 - Rescisão do Contrato de Trabalho	2546 - Verbas Rescisórias	422894	18.28%
1658 - Duração do Trabalho	2086 - Horas Extras	239341	28.63%
2567 - Responsabilidade Civil do Empregador	1855 - Indenização por Dano Moral	237597	38.90%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2594 - Adicional	186050	46.94%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2458 - Salário / Diferença Salarial	145764	53.24%
1937 - Responsabilidade Solidária / Subsidiária	2704 - Tomador de Serviços / Terceirização	80903	56.74%
2620 - Rescisão do Contrato de Trabalho	2656 - Reintegração / Readmissão ou Indenização	75848	60.01%
1658 - Duração do Trabalho	2140 - Intervalo Intrajornada	63085	62.74%
2620 - Rescisão do Contrato de Trabalho	2435 - Rescisão Indireta	56775	65.20%
1654 - Contrato Individual de Trabalho	2029 - FGTS	48981	67.31%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2583 - Abono	46530	69.32%
1654 - Contrato Individual de Trabalho	2554 - Reconhecimento de Relação de Emprego	44634	71.25%
2567 - Responsabilidade Civil do Empregador	8808 - Indenização por Dano Material	34091	72.73%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2117 - Supressão / Redução de Horas Extras Habituais - Indenização	30467	74.04%
2662 – Férias	2021 - Indenização / Dobra / Terço Constitucional	28646	75.28%

Tabela 4.4: Assuntos escolhidos para a criação dos modelos de classificação (Parte II).

Assunto Nível 2	**Assunto Nível 3**	**Quantidade de Processos**	**Percentual Acumulado**
7644 - Categoria Profissional Especial	5280 - Bancários	27352	76.47%
2620 - Rescisão do Contrato de Trabalho	1904 - Despedida / Dispensa Imotivada	25977	77.59%
1654 - Contrato Individual de Trabalho	1844 - CTPS	25421	78.69%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2055 - Gratificação	21318	79.61%
2620 - Rescisão do Contrato de Trabalho	1907 - Justa Causa / Falta Grave	21001	80.52%
1654 - Contrato Individual de Trabalho	1806 - Alteração Contratual ou das Condições de Trabalho	19427	81.36%
55218 - Responsabilidade Civil em Outras Relações de Trabalho	55220 - Indenização por Dano Moral	18080	82.14%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2506 - Ajuda / Tíquete Alimentação	16668	82.86%
1695 - Direito Coletivo	4437 - Revisão de Sentença Normativa	15528	83.53%
10568 – Prescrição	10570 - FGTS	15127	84.18%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	1783 - Comissão	15041	84.83%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	1888 - Descontos Salariais - Devolução	14666	85.47%
2620 - Rescisão do Contrato de Trabalho	2478 - Seguro Desemprego	14606	86.10%
1937 - Responsabilidade Solidária / Subsidiária	5356 - Grupo Econômico	14604	86.73%
1695 - Direito Coletivo	1773 - Contribuição Sindical	14390	87.35%
1658 - Duração do Trabalho	1663 - Adicional Noturno	12040	87.87%
1654 - Contrato Individual de Trabalho	5272 - Administração Pública	12012	88.39%

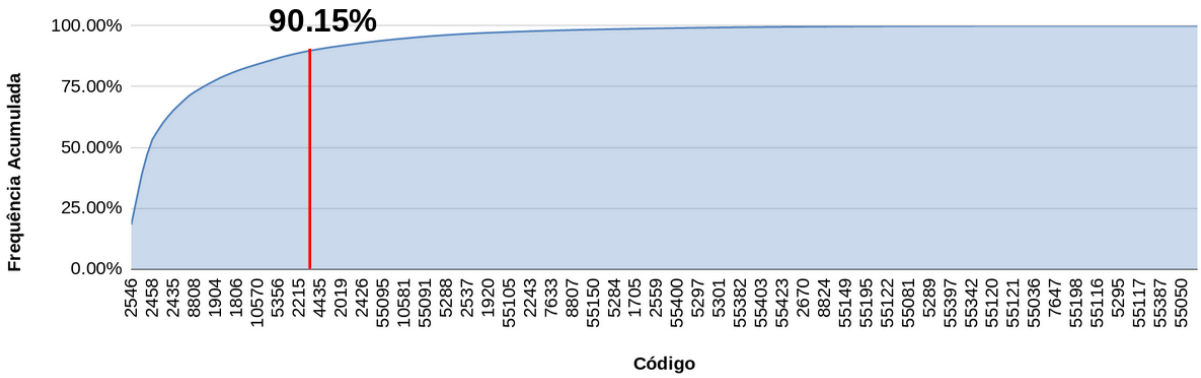

Figura 4.8: Distribuição acumulada de processos por assunto principal (Nível 3).

Tabela 4.5: Assuntos escolhidos para a criação dos modelos de classificação (Parte III).

Assunto Nível 2	**Assunto Nível 3**	**Quantidade de Processos**	**Percentual Acumulado**
2581 - Remuneração, Verbas Indenizatórias e Benefícios	2215 - Multa Prevista em Norma Coletiva	11292	88.88%
2581 - Remuneração, Verbas Indenizatórias e Benefícios	1767 - Cesta Básica	10672	89.34%
1658 - Duração do Trabalho	1661 - Horas in Itinere	9897	89.77%
1695 - Direito Coletivo	1690 - Contribuição / Taxa Assistencial	8848	90.15%

4.2.3 Seleção dos dados

Na Seção 4.2.1 mostrou-se um panorama geral da distribuição dos processos, considerando todos os Tribunais Regionais disponíveis para consulta no momento da análise. Conforme explicado, cada Tribunal possui suas bases de dados com uma infraestrutura independente. Assim, para trazer representatividade para este estudo, recuperou-se documentos das bases de 2^o grau de todos os Tribunais Regionais.

Além dos filtros explicados anteriormente, foi necessário aplicar ainda outras restrições. Uma delas se refere à confidencialidade da informação, não foi utilizado nenhum documento de processo em segredo de justiça e nenhum documento sigiloso. Outro ponto está relacionado com a dificuldade de se recuperar o conteúdo textual dos documentos PDFs. No momento do desenvolvimento deste trabalho, o módulo Pesquisa Textual do PJe, que faz a indexação do conteúdo textual dos documentos do PJe, PDFs e HTMLs, já estava disponível para instalação, mas a maior parte dos regionais ainda não haviam aderido à esta ferramenta, não sendo possível, portanto, recuperar diretamente o con-

teúdo textual destes documentos. Além disso, nem todos os regionais disponibilizam o conteúdo da base binária para o CSJT, o que inviabiliza uma extração representativa, motivo pelo qual foram escolhidos apenas documentos em formato HTML. Escolheu-se apenas documentos já assinados, uma vez que os documentos em elaboração podem não estar completos.

Outra preocupação referente a este estudo é quanto ao conjunto de dados utilizado para treinamento e teste, onde espera-se que os elementos estejam corretamente rotulados para um aprendizado supervisionado. Enquanto o projeto estava priorizado pela CNE do PJe, de forma a se obter dados corretamente classificados e mitigar este problema conhecido da base, uma analista judiciária do TRT da 3^a região foi alocada para fazer a reclassificação manual dos processos. Devido à necessidade de se investir esforços em outras demandas, ela precisou ser desalocada e não houve tempo hábil para a construção de uma massa de dados suficiente para o treinamento e foram classificados apenas 291 processos.

Em função do trabalho realizado pela servidora, tenta-se estimar qual seria custo aproximado de tempo e dinheiro empenhado para a classificação dos novos processos que chegaram à segunda instância. A servidora em questão, ocupante do cargo Analista Judiciário, foi alocada durante 2 semanas de trabalho, com dedicação exclusiva a esta tarefa, em um expediente de 7 horas diárias. A servidora foi alocada por duas semanas, sendo que em cada semana, trabalhou nesta atividade por 4 diárias e meia, conforme atos ATO CSJT.GP.SG Nº 49/2019 e ATO CSJT.GP.SG Nº 68/2019. Considerando uma diária completa de 7 horas de trabalho, descontando-se uma hora para almoço e descanso, considerou-se um valor aproximado de 5 horas por dia da semana. Neste período, foi possível obter 291 processos classificados manualmente, ou seja, aproximadamente 10 minutos por processo.

Projetando este dado, pode-se estimar que, caso um grupo de pessoas de perfil similar fosse alocado para esta tarefa no ano de 2018, que totaliza 1.150.552 de processos (Figura 3.3), cerca de 96.000 processos ao mês, considerando a jornada de trabalho de 7 horas por dia, 22 dias úteis no mês, e a classificação de 6 processos por hora (10 minutos pro processo), seriam necessárias aproximadamente 100 pessoas, por mês, para conseguir classificar todos os novos processos que chegam ao 2^o grau mensalmente. Considerando o salário base inicial de um Analista Judiciário de R$ 8.863,84[4], fazendo uma conta simplificada que multiplica apenas o salário mensal pela quantidade de pessoas alocadas nesta tarefa, tem-se um montante de R$883.684,00 mensais destinados à esta atividade, somando R$10.604.208,00 anuais.

[4]Valor extraído da tabela de valores de remuneração do TRT 3 de acordo com o edital do último concurso deste Tribunal, disponível em `http://www.concursosfcc.com.br/concursos/trt3r114/boletim_final_trt3r114.pdf`

Uma vez que a reclassificação manual de uma amostra representativa dos processos não pôde ser concluída, partiu-se para outra abordagem de redução de ruído na base. Tentou-se remover os processos que continham maiores chances de estarem incorretos. Conforme mencionado, sabe-se que esta é uma informação que nem sempre é preenchida corretamente ou completamente, ou seja, pode haver processos que estão categorizados errados, ou ainda que não tem todos os assuntos cadastrados como dado estruturado. Somado a este fato, tem-se a subjetividade da escolha do assunto principal, dentre os vários assuntos cadastrados [6, 7]. Assim, tentou-se remover do conjunto de processos analisados aqueles que tem mais probabilidade de estarem incorretos.

Considerando um processo que é remetido da primeira à segunda instância, este processo possivelmente estará recorrendo de pedidos específicos, ou seja, assuntos específicos (não necessariamente todos os pedidos que foram realizados em primeira instância). Considerando ainda a característica da justiça do trabalho de que estes processos normalmente apresentam cumulação de pedidos, e portanto, vários assuntos (conforme explicado na Seção 4.1), removeu-se do conjunto de treinamento os processos que contém apenas um assunto na segunda instância, para tentar evitar os casos em que o usuário marca o primeiro assunto disponível apenas para superar a obrigatoriedade desta indicação, sem que de fato faça uma análise do conteúdo do processo. Removeu-se ainda aqueles processos cujo grupo de assuntos era exatamente igual ao grupo de assuntos deste processo enquanto na primeira instância, indicando que possivelmente não foi dada atenção á tarefa de retificação dos assuntos para a remessa do processo.

Embora estas medidas tenham reduzido significativamente a quantidade de documentos passíveis de uso, elas trazem maior confiabilidade de que os dados usados para treinamento apresentam maior qualidade, ainda que não haja garantia disto. A seguir, tem-se consolidado todos os filtros utilizados nos processos do 2^o grau:

1. Processos que estejam no 2^o grau que tenham sido recebidos do 1^o grau
2. Processos que tenham sido recebidos da primeira instância.
3. Processos que tenham apenas um Recurso Ordinário.
4. Processos que não estão em segredo de justiça.
5. Processos que tem mais de um assunto.
6. Processos cujo grupo de assuntos do 1^o grau seja diferente do grupo de assuntos do 2^o grau.
7. Processos cujo nível 3 do assunto principal seja um dos assuntos listados na Tabela 4.3,Tabela 4.4 e Tabela 4.5.

8. O Recurso Ordinário não deve ser sigiloso.
9. O Recurso Ordinário deve estar assinado.
10. O Recurso Ordinário deve estar no formato HTML.

4.3 Preparação dos dados

Nesta seção serão abordados os aspectos relacionados ao processo utilizado para a preparação dos dados.

4.3.1 Recuperação dos dados

Os dados foram recuperados de bases de dados Postgres dos 24 TRTs, que nesse momento se encontravam disponíveis para consulta. Todo o trabalho foi realizado com Python, utilizando as ferramentas da biblioteca Scikit-Learn[5] e NLTK[6]. Todas as restrições de uso apontadas na Seção 4.2.3 foram aplicadas. Inicialmente, haviam 2.833.836 processos disponíveis em 2º grau, considerando todos os TRTs. Após a aplicação de todos os filtros, restaram apenas 242.492 processos elegíveis, que representa 8,50% da base disponível. A grande maior parte dos dados foram retirados em função da remoção de processos com apenas mais de um Recurso Ordinário, ou que tivessem apenas um assunto, ou cujo Recurso Ordinário estivesse em formato PDF. Ao final, gerou-se 24 arquivos CSV, um para cada Tribunal, contendo alguns metadados do documento para auxiliar na sua identificação, bem como o conteúdo completo de cada documento. Neste CSV, apenas o assunto principal de cada processo foi recuperado, que é o assunto de interesse neste estudo. Ao todo, foram recuperados 242.492 Recursos Ordinários.

4.3.2 Pré-processamento

De forma a reduzir a dimensionalidade dos textos, fez-se a limpeza e a normalização do conteúdo dos documentos recuperados. Inicialmente removeu-se todas as marcações HTML presentes. Em seguida, removeu-se os acentos e caracteres especiais, números e *stopwords* e aplicou-se a técnica de radicalização (*stemming*) para remover as variações de uma mesma palavra, e transformou-se tudo em caixa baixa. Por fim, removeu-se aqueles documentos cujo conteúdo final ficou vazio após o pré-processamento, chegando-se a um total de 241.101 Recursos Ordinários.

[5]Scikit-Learn: `https://scikit-learn.org/stable/`
[6]NLTK: `https://www.nltk.org/`

Antes de se aplicar o pré-processamento, removendo-se apenas as marcações HTML, tinha-se uma matriz de 242.492 linhas (quantidade de documentos) e 326.856 colunas (quantidade de *tokens* diferentes). Após a limpeza dos textos, chegou-se a um conjunto de 241.101 documentos com 217.493 *tokens*, havendo uma diminuição de 109.363 *tokens*.

4.4 Modelagem e avaliação

Nesta etapa, detalha-se a abordagem utilizada para a representação das *features* dos textos bem como as configurações utilizadas nos algoritmos dos modelos construídos. Para os experimentos realizados, foi utilizado um notebook ASUS 7th Gen Intel Core i7-7700HQ de 2.8 GHz, contendo 32 GB de memória RAM.

Considerando o caráter inovador deste trabalho no contexto do PJe, torna-se relevante que haja confiabilidade dos resultados apresentados, assim, a métrica de micro precisão será usada para comparar os classificadores e escolher o vencedor, uma vez que se trata de uma base de dados desbalanceada. Serão mostradas a acurácia, revocação e F-Measure, para dar visibilidade aos demais aspectos do modelo.

4.4.1 Modelagem

Uma vez que o texto foi pré-processado, conforme explicado antes, se faz necessário utilizar uma representação vetorial para que os algoritmos de classificação possam entender a informação contida em cada documento. Escolheu-se trabalhar com o TD-IDF, com a utilização dos parâmetros $min\_df = 5$ e $max\_df = 0,8$, o primeiro indicando que somente serão consideradas palavras que aparecerem em no mínimo 5 documentos, e o segundo indicando que serão desconsideradas palavras que aparecerem em mais do que 80,00% dos documentos, uma vez que palavras muito raras ou muito comuns acabam não sendo determinantes para diferenciar as classes. Os valores foram escolhidos arbitrariamente. Muitos nomes únicos das partes envolvidas são removidos ao se retirar palavras que aparecem em menos que 5 documentos. A matriz gerada passou de 217.493 tokens para 60.906.

Estes dados foram divididos em treinamento e teste com a técnica de *holdout*, separando-se de forma estratificada 20% dos dados para teste. Os dados de treinamento foram usados para a busca de hiper-parâmetros com validação cruzada de 5 *folds*, enquanto os dados de teste foram reservados para aferir as métricas finais dos modelos vencedores. Na Figura 4.9, mostra-se a distribuição de documentos para cada um dos assuntos elegidos para análise. Nota-se que é um problema altamente desbalanceado, variando-se de 207 documentos para o assunto de código 4437 a 36.937 documentos para o assunto de código

2546. Por ser um problema multiclasse, optou-se por utilizar a abordagem *one-versus-rest*, que treina um classificador para cada classe, escolhendo o assunto alvo como classe positiva, e considerando todos os outros assuntos como classe negativa. Os classificadores binários foram encapsulados no BalancedBaggingClassifier, que cria um ensemble de vários modelos com subconjuntos de dados balanceados (com *under sampling*), e faz uma votação ao final. Assim, cada classe foi treinada com o mesmo número de exemplos positivos e negativos, sendo limitados pela quantidade de exemplos na classe positiva de cada classificador.

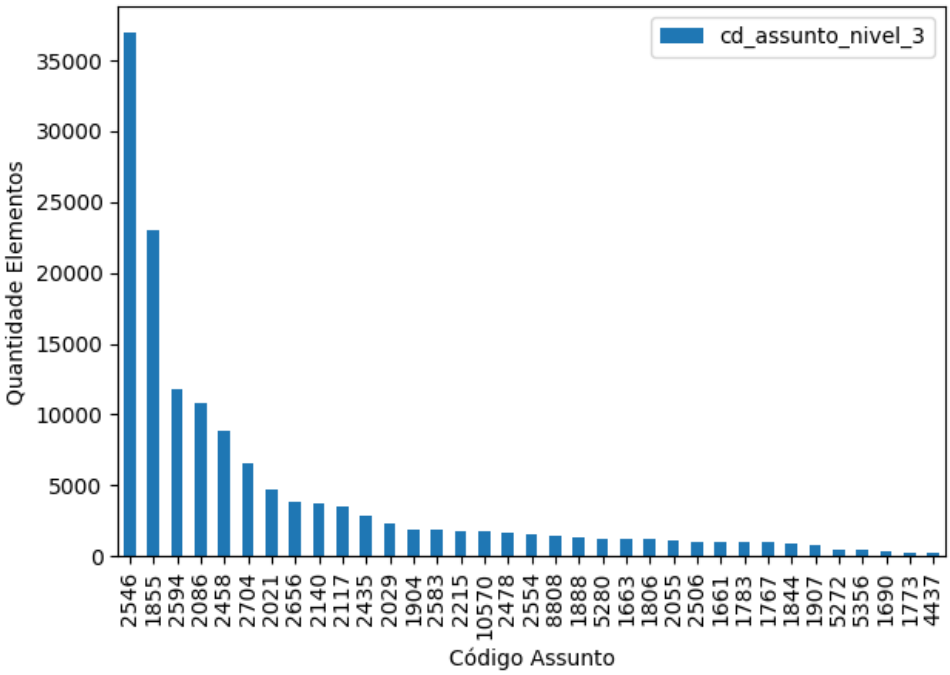

Figura 4.9: Distribuição de documentos por assunto nível 3 no conjunto de treinamento.

O próximo passo foi a modelagem. A partir da experiência demonstrada nos trabalhos relacionados listados no Capítulo 2, dentro das possibilidades deste trabalho, utilizou-se SVM, Random Forest (RF), Multinomial Naïve Bayes (MNB) e Multilayer Perceprtron (MLP). Optou-se por não utilizar os modelos de linguagem, dado que não havia GPU disponível para o treinamento das redes neurais profundas e que os dados não poderiam ser colocados na nuvem, uma vez que não estão disponíveis publicamente. Cada um desses algoritmos possui hiper-parâmetros de configuração, que podem melhorar ou piorar o desempenho de acordo com o problema. Assim, foi feita uma busca dos hiper-parâmetros que melhor se adequariam ao problema por meio do GridSearch (GS) para os algoritmos escolhidos para análise, testando um número limitado de combinações diferentes dentre o conjunto de valores possíveis informados. Variou-se a quantidade de estimadores e também a quantidade e exemplos. Os parâmetros que foram testados foram definidos

conforme a Tabela 4.6 enquanto os demais parâmetros foram utilizados conforme a configuração padrão do Scikit-Learn. Ao todo, rodou-se 10 combinações possíveis para cada algoritmo, totalizando 56 horas e 25 minutos de execução. Todas as execuções foram realizadas com validação cruzada de 5-*folds*, totalizando 200 modelos. Escolheu-se cinco em função do elevado tempo de processamento de todos os modelos, uma vez que um modelo novo é criado para cada *fold*. O resultado da micro precisão de cada execução pode ser encontrado no Apêndice A.

Ao final da execução do GS, os melhores parâmetros para cada modelo foram selecionados, e esta foi a configuração utilizada nos testes que se seguiram. Os resultados dos modelos vencedores podem ser encontrados na Tabela 4.7, onde nota-se que a melhor micro precisão é do MLP, apresentando o valor de 44,26%, ou seja, de todas as classificações que ele disse ser de determinado assunto, apenas 44,26% realmente eram. Embora este trabalho não possa ser diretamente comparado aos trabalhos relacionados, uma vez que contem dados diferentes e classes diferentes, apresentando portanto um desafio diferente, buscou-se a micro-precisão média daqueles trabalhos apenas para se ter um referencial, que foi aproximadamente 85,00%, um valor bastante superior aos 44,26% apresentados pelo MLP. Este valor de 85,00% foi calculado manualmente, uma vez que a maior parte dos trabalhos relacionados não traz a micro precisão ou mesmo a diferenciação entre micro e macro métricas, mesmo sendo trabalhos com problemas multiclasse desbalanceados. Foi identificada a micro precisão de 83,40% em [19] e foi calculada manualmente a micro precisão de 90,96% em [16] e 83,09% em [23]. Assim, dada a grande diferença apresentada, considerou-se ruins os resultados obtidos nesta primeira tentativa.

Para tentar melhorar estes valores, retornou-se às etapas de entendimento e preparação dos dados para analisar a quantidade de palavras contida em cada texto. Inicialmente, fez-se uma análise geral da distribuição da quantidade de palavras por texto, conforme se pode ver na imagem apresentada na Figura 4.10. Nota-se que a grande maior parte dos textos está concentrada em até 9.000 palavras aproximadamente, havendo alguns *outliers* com textos muito grandes, chegando a 60.000. Naturalmente, a classificação de um único assunto principal em um texto com 20.000 palavras ou mais é um problema de maior dificuldade comparado com os textos menores. Para registro, de forma a se ter um parâmetro de comparação, no Recurso Ordinário exposto na 4.2.1, tem-se um total de 1028 palavras, um texto considerado pequeno dentro do tamanho dos textos apresentado na Figura 4.10. Por esse motivo, resolveu-se remover a maior parte dos textos considerados *outliers*, sendo considerado o limite superior de 10.000 palavras, o que ocasionou na remoção de 6.216 documentos (2.57% dos documentos).

Outra análise feita foi quanto ao limite inferior da quantidade de palavras. Muitos usuários colocam um curto trecho de texto no campo de redação do Recurso Ordinário no

Tabela 4.6: Parâmetros utilizados no GridSearch.

Multinomial Naive Bayes		
Parâmetros	Valores	Melhor valor
n_estimators	3; 5	5
max_samples	0,8; 0,5	0,8
base_alpha	0,0001; 0,001; 0,01; 0,1; 0,5; 1	0,5
SVM		
Parâmetros	Valores	Melhor valor
n_estimators	3; 5	5
max_samples	0,8; 0,5	0,8
base_C	0,01; 0,1; 1;10	1
Random Forest		
Parâmetros	Valores	Melhor valor
n_estimators	3; 5	3
max_samples	0,8; 0,5	0,5
base_max_depth	30; 50; 100	100
base_n_estimators	100; 200; 300	200
base_ _min_samples_leaf	0,05; 0,1; 0,5	0,05
base_ _min_samples_split	0,05; 0,1; 0,5	0,1
base_max_features	0,3; 0,5; 0,8	0,3
Multi-Layer Perceptron		
Parâmetros	Valores	Melhor valor
n_estimators	3; 5	3
max_samples	0,8; 0,5	0,8
base_hidden _layer_sizes	(10,10); (10,5,10)	(10 , 10)
base_activation	identity; logistic; tanh; relu	logistic
base_solver	sgd; adam; lbfgs	lbfgs
base_alpha	0,001; 0,01; 0,05; 0,1	0,05
base_learning_rate	constant; adaptive; invscaling	constant
base_max_iter	200; 300; 400	400

Tabela 4.7: Resultados da modelagem inicial **(TF-IDF (GS))**.

Modelo	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
MNB	29,35%	38,75%	22,43%	23,02%
SVM	**33,74%**	39,50%	**24,73%**	**25,61%**
RF	28,77%	40,73%	19,44%	19,68%
MLP	27,44%	**44,26%**	18,21%	15,72%
	Macro Acurácia	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
MNB	22,43%	19,69%	29,35%	18,65%
SVM	**24,73%**	**20,83%**	**33,74%**	**21,44%**
RF	19,44%	20,24%	28,77%	18,09%
MLP	17,74%	18,10%	29,10%	17,06%

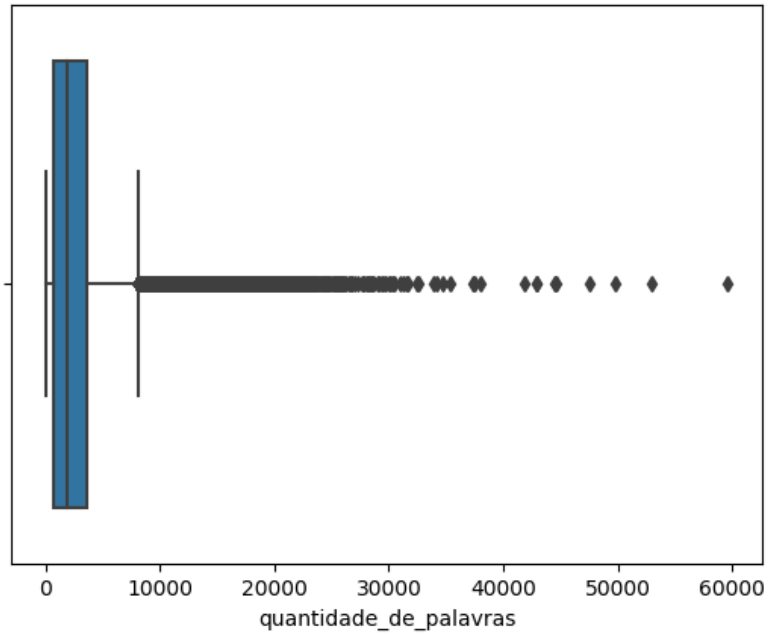

Figura 4.10: Boxplot da quantidade inicial de palavras por texto.

PJe, o que gera o texto HTML recuperado, e anexam a peça real do Recurso Ordinário em PDF, que não foi recuperado neste trabalho. Assim, analisou-se amostras dos textos com até 100, 200, 300, 400, 500 e 600 palavras em busca de qual seria o limite de corte inferior, de forma a excluir da análise os textos que apenas faziam referência ao real Recurso Ordinário anexado em PDF. Notou-se que existem muitos textos com os conteúdos "PDF em Anexo", "Segue", "Recurso em anexo", "Anexo", sendo 15.671 documentos com até 5 palavras. Buscando outras variações, somaram-se 38.003 textos com até no máximo

100 palavras. Nos textos de 100 até 400 palavras, notou-se que raramente se fala sobre os assuntos, havendo uma introdução da interposição do Recurso, que se encontrará em anexo. Na Figura 4.11 é possível ver alguns exemplos destes textos, onde vê-se que não há possibilidade da inferência do assunto nestes exemplos, conforme se pode ler.

Analisando os textos com mais de 400 palavras, começa-se a ter muitos exemplos onde aborda-se de fato o assunto, como se pode ver exemplo extraído na Figura 4.12. Alguns textos relevantes podem ter sido excluídos com este corte, e ainda podem haver textos pouco informativos quanto ao assunto, mas para não se perder textos de teor mais simples, optou-se por utilizar o limite inferior de 400 palavras. Conforme explicado na Seção 3.3.1, a não exclusão destes documentos além de não trazer informações relevantes aos algoritmos, que precisarão processar mais dados sem adquirir conhecimento, leva à confusão dos modelos, uma vezes que estes textos não informativos podem constar como exemplos nas classes positivas e também nas classes negativas, dificultando o aprendizado. A nova distribuição dos dados pode ser vista na Figura 4.13. Ao todo, somando-se com os textos removidos por ultrapassarem o limite superior, foram removidos 60.429 documentos, 25,06% dos documentos recuperados, totalizando 180.672 para a criação dos modelos. Após esta remoção, a matriz TF-IDF passou a contar com 56.691 *features*.

O resultado dos modelos, treinados após a limpeza do conjunto de dados, se encontram na Tabela 4.8, com os de melhor desempenho em negrito. Nota-se que quase todos os valores se apresentam superiores àqueles apresentados na Tabela 4.7, ou seja, de fato, a maior parte dos modelos pode separar melhor qual documento era de cada classe após remoção de um subconjunto de documentos. De maneira geral, a macro acurácia foi a métrica mais impactada, aumentando em média 6.22%, enquanto a micro precisão aumentou para 3 algoritmos, e caiu significativamente para o MLP. O SVM foi o melhor modelo nesta análise, com 42,59% de micro precisão. Pode-se concluir que, apesar do ruído, o MLP teve melhor desempenho na micro precisão com todo o conjunto de dados, enquanto os demais modelos se beneficiaram da limpeza. Portanto, buscando melhorar ainda mais os resultados, os experimentos seguintes utilizam os dados limpos.

Mesmo com a limpeza dos textos, ainda não se alcançou resultados razoáveis. Retornou-se então ao início da modelagem, e uma terceira abordagem envolveu o uso do BM25, ao invés do TF-IDF. Conforme explicado na Seção 3.3.1, este tipo de modelagem leva em consideração o tamanho dos textos, mas há uma grande variabilidade deste parâmetro. Não há implementação do BM25 no Scikit-learn, portanto utilizou-se a implementação disponibilizada pela comunidade[7].

Os resultados apresentados pelos modelos vencedores se encontram na Tabela 4.9. A maior parte dos resultados flutuou menos de 1,00% dos valores anteriores (Tabela 4.8),

[7]BM25: `https://github.com/arosh/BM25Transformer/blob/master/bm25.py`

Texto	Qtd. Palavras
EXMO. SR. DR. JUIZ FEDERAL DA PRIMEIRA VARA DO TRABALHO . Processo nº , já qualificada nos autos da reclamação trabalhista acima em epígrafe que lhe move , também já qualificado, feito que tramita por essa M. Justiça, por seu procurador que esta subscreve vem, respeitosamente a presença de V. Exa., requerer juntada nos autos do processo da Petição Recurso Ordinário e guias devidamente recolhidas de depósito recursal e custas processuais. Termos em que, pede deferimento. , 22 de fevereiro de 2017. OAB/	101
EXCELENTÍSSIMO SENHOR DOUTOR JUIZ DA 1ª VARA DO TRABALHO DE - ESTADO DE . PROCESSO Nº e , empresas devidamente qualificadas nos autos do processo de numero em epigrafe, em tramite perante este r. Juízo, que lhe move , através de seu advogado e procurador que a presente subscreve, vem, mui respeitosamente, à presença de Vossa Excelência, n os termos do artigo 1º do Ato número 423/CSJT/GP/SG, de 12 de novembro de 2013, requerer a juntada do incluso RECURSO ORDINÁRIO , em arquivo eletrônico, tipo "Portable Document Format" (.pdf), de qualidade padrão "PDF-A", nos termos do artigo 1º, § 2º, inciso II, da Lei nº 11.419, de 19 de dezembro de 2006, e em conformidade com o parágrafo único do artigo 1º. do Ato acima mencionado . Termos em que, Pede deferimento. , 13 de fevereiro de 2015. OAB/	201
E XCELENTÍSSIMO SENHOR DOUTOR JUIZ FEDERAL DA 1ª VARA DO TRABALHO DE . AUTOS DO PROCESSO Nº , devidamente qualificada nos autos do processo supra referenciado, RECLAMAÇÃO TRABALHISTA , que lhe promove , por seu advogado e procurador, ao final assinado, vem, respeitosamente, perante Vossa Excelência, não concordando com a r. sentença (ID.) , interpor o presente RECURSO ORDINÁRIO , com amparo nas razões anexas em PDF, requerendo juntada das mesmas nos autos para os devidos fins legais. Anexa á presente Guia de Recolhimento para fins de Depósito Recursal junto a Justiça do Trabalho - SEFIP 8.40 no valor de R$ 8.960,00 (oito mil novecentos e sessenta reais) e Guia de Recolhimento da União - GRU Judicial no valor de R$ 200,00 (duzentos reais), referente às custas. A Recorrente informa, que a Guia de Recolhimento para fins de Depósito Recursal junto a Justiça do Trabalho anexada a presente e a SEFIP 8.40 que corresponde a GFIP emitida eletronicamente, conforme modelo do Anexo I da Instrução Normativa nº 26, decorrente da Resolução nº 124, de 02/09/2004 e Circular da CEF nº 321, de 01/09/2014. Declara, ainda, que as Guia de Recolhimento para fins de Depósito Recursal junto a Justiça do Trabalho - SEFIP 8.40, no valor de R$ 8.960,00 e a Guia de Recolhimento da União - GRU Judicial no valor de R$ 200,00 referente à custa, anexadas a presente via peticionamento eletrônico vai PJE, são as originais devidamente scaneadas e salvas no formado pdf. Requer ainda que seja o presente recurso recebido e processado, determinando-se seu encaminhamento ao Egrégio Tribunal Regional do trabalho da Região para o reexame da questão. Termos em que, Pede deferimento. , 21 de junho de 2017. Advogado OAB/ Nº	306

Figura 4.11: Exemplos de textos com até 400 palavras.

Texto	Qtd. Palavras
Exmo. Sr. Dr. Juiz da MM. Vara do Trabalho de Ref.: Proc. nº . , nos autos da reclamação trabalhista ajuizada por , vem, por seus advogados, interpor RECURSO ORDINÁRIO , o que faz na forma dos seguintes fundamentos, requerendo, cumpridas as formalidades legais, seu encaminhamento à superior instância. P. Deferimento , 08 de setembro 2015. OAB/ OAB/ EGRÉGIO TRIBUNAL REGIONAL DO TRABALHO DA REGIÃO. Ref.: Proc. n º Pela Recorrente: . EGRÉGIA TURMA D.m.v , mas merece reforma a r. sentença de fls. que julgou procedente em parte os pedidos, como ficará demonstrado a seguir. DO CONHECIMENTO O presente recurso merece ser conhecido, uma vez que presentes todos os requisitos intrínsecos e extrínsecos de admissibilidade, notadamente o preparo, conforme guia de custas judiciais e depósito recursal em anexo. O recurso é tempestivo. Dessa forma, deve ser conhecido o presente recurso. DO MÉRITO DAS HORAS EXTRAS A r. sentença de primeiro grau julgou procedente o pedido de horas extras, com base na Súmula 338 do C. TST. D.m.v , mas equivocado o entendimento do ilustríssimo MM Juiz de piso. A r. sentença fixou a jornada da trabalhadora da seguinte forma: 12 às 20h, com uma hora de intervalo intrajornada. Ocorre que de acordo com a jornada indicada pelo próprio autor na inicial, não há que se falar em pagamento horas extras e reflexos, tendo em vista que não ultrapassava a jornada diária de 8h e da 44ª semanal. Assim, que a autora não desincumbiu o ônus de provar fato constitutivo do seu direito, razão pela qual merece reforma a r. sentença de piso para julgar improcedente o pedido. Por fim, devem ser compensados todos os pagamentos feitos a autora idênticos títulos, merecendo reforma a r. sentença nesse particular. CONCLUSÃO Assim, por todo o exposto, requer seja conhecido o recurso ordinário, por tempestivo, conforme explicitado na preliminar de tempestividade. Por fim, requer seja dado provimento ao presente Recurso Ordinário, a fim de que a r. sentença seja reformada para julgar totalmente improcedente a ação, nos termos da fundamentação, por ser medida que se impõe! P. Deferimento. , 08 de setembro de 2015. OAB/ OAB/	402

Figura 4.12: Exemplos de texto com mais de 400 palavras.

sendo que metade das métricas foram alteradas positivamente, e a outra metade negativamente. Como não há impacto significativo, este tipo de modelagem será descartado para futuros testes. A melhor micro precisão foi apresentada pelo SVM, que subiu 0,23%, passando para 41,84%.

A seguir, testou-se o uso de uma nova forma de modelar os textos utilizando o LSA, um modelo de extração de tópicos. O LSA foi testado com 100 e com 250 tópicos, conforme um dos melhores resultados apresentados em [19]. Neste modelo, a matriz de *features* é reduzida a uma matriz limitada à quantidade de tópicos, capazes de representar cada um dos documentos, assim, o espaço vetorial de análise fica bastante reduzido, conforme explicado na Seção 3.3.1[8]. Os resultados apresentados pelos modelos vencedores se encon-

[8]Importante mencionar que o modelo MNB não foi testado com esta modelagem por não aceitar valores negativos, presentes em matrizes LSA. Optou-se pela não utilização do algoritmo NMF, que produz uma matriz sem valores negativos, apenas para se manter a mesma modelagem utilizada no trabalho [19]

Tabela 4.8: Resultados da modelagem após remoção de documentos **(TF-IDF)**.

Modelo	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
MNB	35,30%	40,77%	27,89%	27,76%
SVM	**39,67%**	**42,59%**	**29,30%**	**29,60%**
RF	35,69%	41,61%	24,95%	23,88%
MLP	32,10%	38,97%	27,08%	25,14%
	Macro Acurácia	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
MNB	27,89%	**24,32%**	35,30%	24,06%
SVM	**29,30%**	23,68%	**39,67%**	**25,64%**
RF	24,95%	22,73%	35,69%	22,43%
MLP	27,08%	23,07%	32,10%	23,48%

Tabela 4.9: Resultados da modelagem com **BM25**.

Modelo	**Micro Acurácia**	**Acurácia**	**Micro Revocação**	**Micro F-Measure**
MNB	35,44%	40,85%	27,94%	**28,35%**
SVM	**39,14%**	40,88%	24,56%	23,90%
RF	35,75%	**41,84%**	24,93%	23,80%
MLP	32,14%	38,90%	**28,03%**	27,70%
	Macro Acurácia	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
MNB	27,94%	**23,47%**	35,44%	23,53%
SVM	24,56%	21,49%	**39,14%**	22,74%
RF	24,93%	22,55%	35,75%	22,25%
MLP	**28,03%**	23,11%	32,14%	**24,94%**

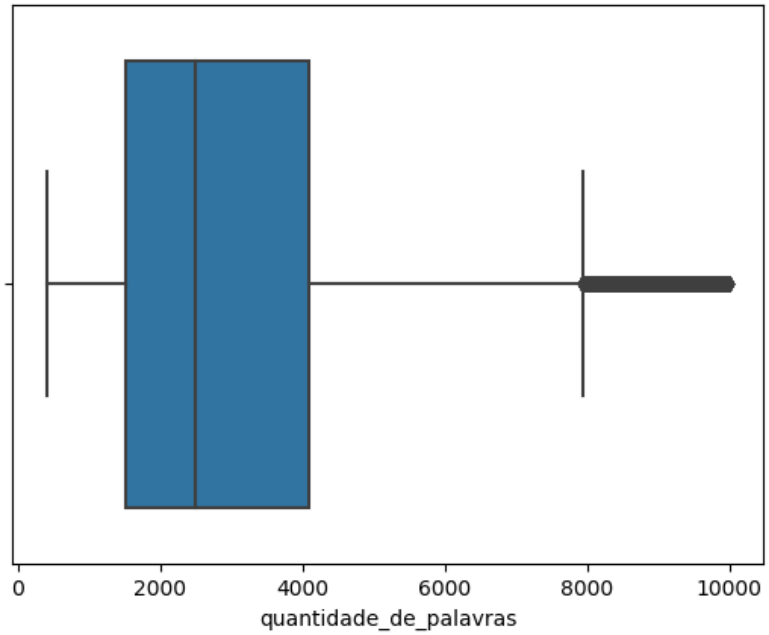

Figura 4.13: Boxplot da quantidade final de palavras por texto .

tram na Tabela 4.10, onde é possível ver que todos os melhores resultados, destacados em negrito, se encontram com a modelagem de 250 tópicos. O MLP apresenta o melhor resultado, chegando a 46,03%, o valor mais alto alcançado até o momento para esta métrica, sendo esta portanto a configuração vencedora para o abordagem multiclasse escolhida.

Os modelos foram testados ainda com vetores de palavra GloVe, utilizando os *word embeddings* no repositório do NILC, entretanto, os resultados apresentados foram inferiores a todos os demais resultados, e por esse motivo, não foram reportados. Assim, conclui-se os experimentos com a análise do acerto do assunto principal dos processos. Na tabela Tabela 4.11 encontra-se um resumo das melhores configurações testadas para cada um dos algoritmos, e no Apêndice B tem-se a matriz de confusão normalizada de cada um destes modelos .

A melhor micro precisão apresentada foi 46,04%, obtida pelo MLP, utilizando LSA com 250 tópicos. A segunda melhor micro precisão foi do SVM com os vetores TF-IDF, chegando a 42,59%. Em seguida, o RF com BM25 apresentou 41,84%, e por último o MNB obteve 40,85% de micro precisão. Fazendo-se uma análise das demais métricas, nota-se que o SVM apresentou os maiores valores em todas as métricas, com exceção da micro precisão, se colocando como um forte segundo candidato, uma vez que apresentou melhores métricas que o MLP nas demais avaliações. Fazendo uma análise da acurácia e precisão, nota-se que as macro métricas foram menores do que as micro métricas, indi-

Tabela 4.10: Resultados da modelagem **LSA**.

Features	**Modelo**	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
LSA - 100	SVM	34,68%	39,00%	26,65%	26,38%
LSA - 100	RF	32,83%	38,45%	25,96%	25,72%
LSA - 100	MLP	29,38%	44,05%	17,95%	19,00%
LSA - 250	SVM	**36,19%**	39,44%	**28,06%**	**27,99%**
LSA - 250	RF	33,27%	38,36%	26,50%	26,22%
LSA - 250	MLP	27,55%	**46,03%**	17,48%	17,74%
Features	**Modelo**	**Macro Acurácia**	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
LSA - 100	SVM	26,65%	20,62%	34,68%	22,11%
LSA - 100	RF	25,96%	22,54%	32,83%	22,72%
LSA - 100	MLP	17,95%	19,91%	29,38%	16,88%
LSA - 250	SVM	**28,06%**	22,23%	**36,19%**	**23,95%**
LSA - 250	RF	26,50%	**23,15%**	33,27%	23,28%
LSA - 250	MLP	17,48%	19,83%	27,55%	15,54%

cando que houveram resultados positivos para classes de maior quantidade de elementos, influenciando positivamente as micro métricas, mas que muitas classes com menos instâncias obtiveram valores mais baixos, influenciando negativamente na macro métrica. Naturalmente, ao dar preferência para os modelos que apresentaram melhor micro precisão, que é a métrica escolhida para a análise deste estudo, a micro revocação se mostrou mais baixa de maneira geral, uma vez que os modelos se preocuparam mais em estarem certos quando afirmassem que um processo é de determinado assunto, do que em identificar o máximo de processos de cada assunto. Considerando o desbalanceamento dos dados, na micro revocação a métrica foi puxada pra baixo, pois houveram classes com alta representatividade que poucos processos foram identificados, e na macro revocação, a métrica é puxada pra cima uma vez que algumas classes com poucos elementos acabaram por ter alta revocação. A F-Measure, por ser baseada nas demais métricas, acompanhou os resultados apresentados pela precisão e revocação em cada uma das abordagens, apresentando micro F-Measure maior que a macro F-Measure para todos os modelos.

Na Figura 4.14 encontra-se a matriz de confusão normalizada do modelo MLP, com LSA de 250 tópicos. De forma a analisar o conteúdo de cada um desses conjuntos de documentos, elaborou-se nuvens de palavras, criadas a partir de dos documentos cada assunto, limitando-se a uma amostra de no máximo 2.000 documentos escolhidos aleatoriamente, que mostra em destaque as palavras mais recorrentes [9] em cada tema. No Apêndice C

[9]Uma vez que no processamento ads matrizes TF-IDF removeram-se palavras que apareciam em mais de 80% dos textos, foram removidas as stopwords negociais 'trabalho', 'juiz', 'vara', 'recurso', 'ordinario', 'reclamada', 'reclamante', 'reclamado', 'recorrente', 'trabalho', 'empregado', 'empregada', 'art',

Tabela 4.11: Melhores resultados analisando o assunto principal.

Modelo	**Features**	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
MNB	BM25	35,44%	40,85%	27,94%	28,35%
SVM	TF-IDF	**39,67%**	42,59%	**29,30%**	**29,60%**
RF	BM25	35,75%	41,84%	24,93%	23,80%
MLP	LSA - 250	27,55%	**46,03%**	17,48%	17,74%
	Features	**Macro Acurácia**	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
MNB	BM25	27,94%	23,47%	35,44%	23,53%
SVM	TF-IDF	**29,30%**	**23,68%**	**39,67%**	**25,64%**
RF	BM25	24,93%	22,55%	35,75%	22,25%
MLP	LSA - 250	17,48%	19,83%	27,55%	15,54%

pode-se encontrar as nuvens de palavras de todos os assuntos.

Nota-se que algumas classes houve acerto de mais 50,00% dos elementos, chegando em 77,00% para o assunto de código 2086, "Horas Extras", 66,00% para o assunto de código 4437, "Revisão de Sentença Normativa" e 59,00% para o assunto de código 2458, "Salário / Diferença Salarial", sendo estas as três classes de maior taxa de acerto. Nas Figuras 4.15 a 4.17 é possível encontrar a núvem de palavras destes três assuntos.

No assunto "Horas Extras" (código 2086), as palavras "horas", "extras", "jornada", "intervalo", "adicional" se mostram em grande destaque, se mostrando discriminativas para este assunto, indicando documentos que tenham se referido a pagamento de horas extras, jornada extraordinária trabalho, grandes intervalos de trabalho ou mesmo a ausência de intervalos na jornada de trabalho, todos relacionados horas extras trabalhadas. Este assunto continha 10.797 documentos no conjunto de treinamento, sendo utilizado 80,00% deles em cada modelo do *ensemble*, onde eles foram testados contra a mesma quantidade de documentos de outro assunto principal, escolhidos aleatoriamente. Como é uma quantia elevada de documentos, tem-se maior representatividade e maior variância das informações, o que beneficia os modelos de aprendizado.

'processo', 'trabalhista', 'tribunal', 'regional', 'regiao', 'autor', 'reu', 'turma', 'trt', 'advogado', 'advogada', 'oab', 'ser', 'tst','sentenca', 'pagamento', 'direito', 'trabalhista', 'assim', 'autos', 'justica', 'lei', 'clt', 'decisao', 'empresa', 'contratante', 'contratada', 'contratado', 'recorrido', 'recorrida' e 'conforme', que apareciam em destaque na maior parte das nuvens geradas, sendo, portanto, consideradas não discriminativas. Este não é precisamente o mesmo pré-processamento que foi feito no pipeline deste trabalho, uma vez que este envolveu a radicalização das palavras, o que tornaria a nuvem de palavras de difícil entendimento. Assim, optou-se por uma versão simplificada do pré-processamento, que removeu acentos, números, pontuações e transformou tudo em letras minúsculas.

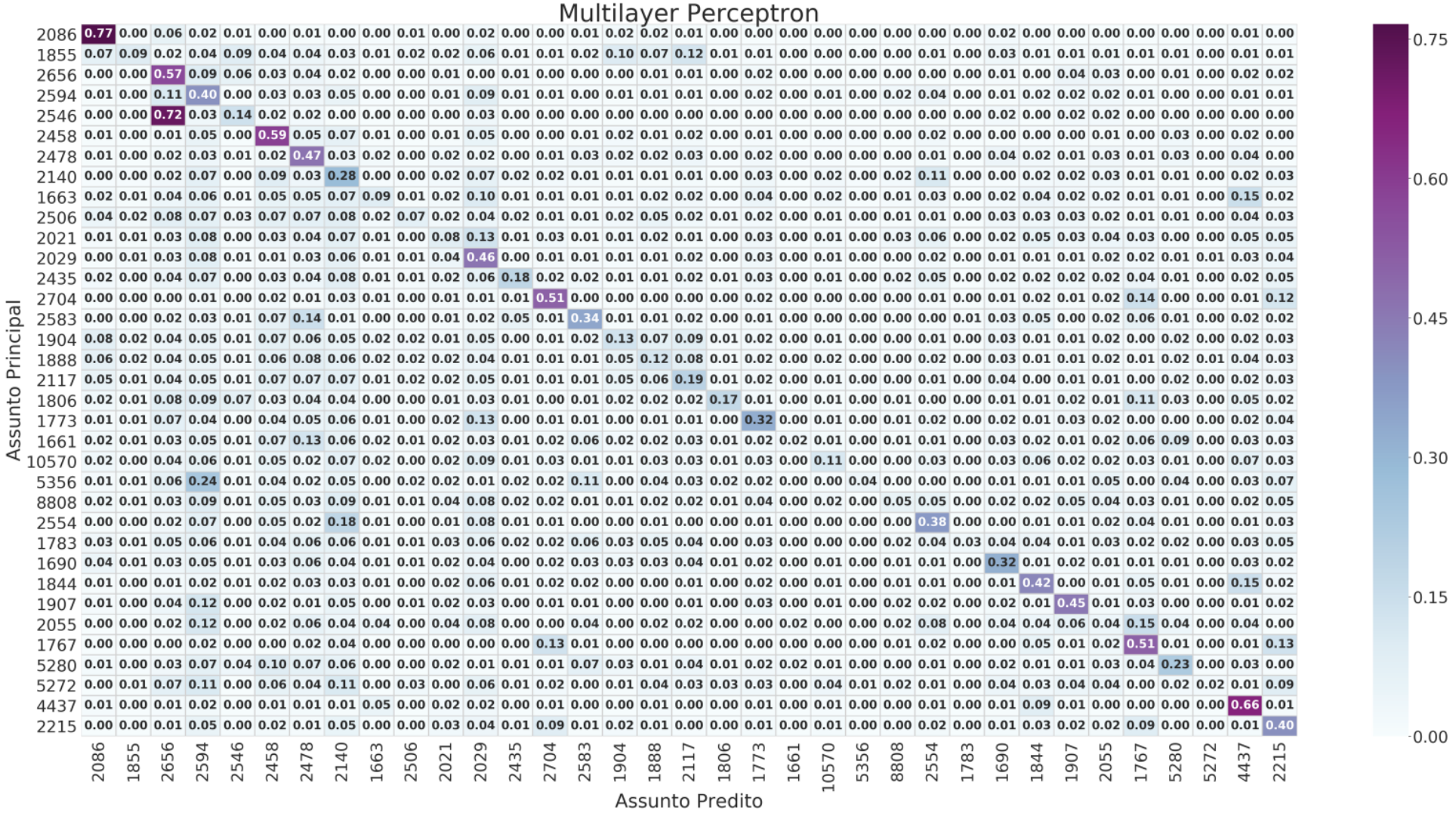

Figura 4.14: Matriz de confusão do MLP com LSA250.

Para o assunto “Salário / Diferença Salarial” (código 2458), as palavras “salario”, “salarial”, “funcao”, “horas” aparecem em destaque, podendo indicar casos onde foi requerido pagamento de salário ou alegou-se a existência de diferença salarial em função de horas de trabalho ou funções diferentes que pudessem estar sendo exercidas pelo trabalhador. O conjunto de treinamento continha 8.842 documentos desta classe, havendo alta representatividade também.

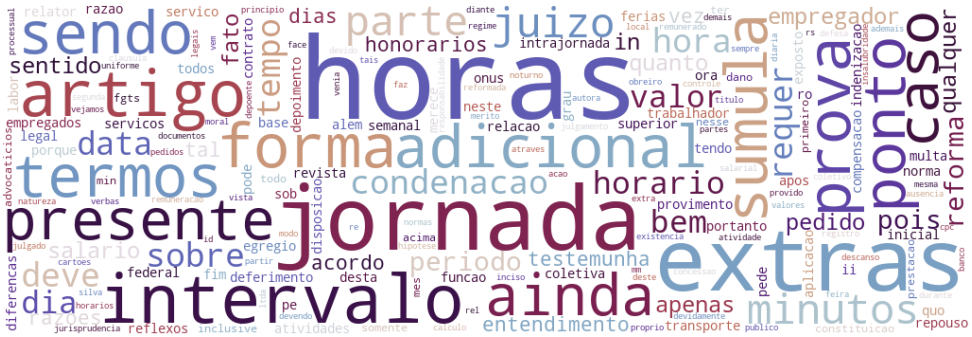

Figura 4.15: Nuvem de palavras do assunto 2086 - Horas Extras.

Figura 4.16: Nuvem de palavras do assunto 2458 - Salário / Diferença Salarial.

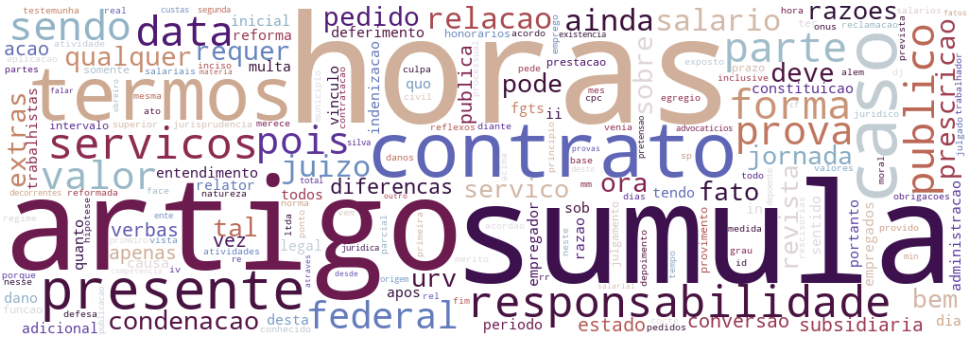

Figura 4.17: Nuvem de palavras do assunto 4437 - Sentença Normativa.

Partindo para a análise dos assuntos onde quase não se obteve exemplos, temos os assuntos de código 1661, “Horas In Itinere” , 1783, “Comissão” e 5272, “Adminsitração Pública”, que apresentaram no máximo 0,03% de acerto. Analisando-se o primeiro deles, “Horas In Itinere” (código 1661), nota-se o destaque das palavras “horas”, “transporte”, “itinere”, “jornada”, “tempo”, que de fato estão relacionadas com este tema, que trata do tempo que o trabalhado passa no itinerário de casa para o trabalho ou vice-versa. As palavras chave deste tema se assemelham com aquelas do tema “Horas Extras”, embora os assuntos mais votados para os documentos que eram desta classe tenham sido “Seguro Desemprego” (Código 2478) e “Bancários” (Código 5280) (cujas palavras de destaque são similares.). Este assunto continha 1.043 documentos no conjunto de treinamento, sendo um valor intermediário de amostras comparado com o tamanho das amostras dos demais assuntos.

Para o assunto “Comissão” (código 1783), tem-se em destaque as palavras “comissoes”, “horas”, “extras”, “forma”, “prova” entre outras, sendo apenas a palavra “comissoes” fortemente ligada ao contexto negocial do tema. As palavras “horas” e “extras” estão bastante relacionadas com o assunto “Horas Extras” (código 2086), que recebeu a mesma quantidade de votos que o assunto “Comissão”. Este assunto também continha 1.043 documentos no conjunto de treinamento.

Para o assunto “Adminsitração Pública” (código 5272), tem-se em destaque as palavras “publico”, “concurso”, “relacao”, “contrato”, “federal”, entre outras. As palavras “publico”, “concurso” e “federal” são de fato ligadas ao contexto negocial deste tema, sendo “publico” parte do próprio nome do assunto, “concurso” uma das formas de se ingressar no serviço público, e federal uma das esferas da administração pública. Os assuntos que foram mais votados pelos modelos para os documentos desta classe foram “Adicional” (código 2594), “Intervalo Intrajornada” (código 2140) e “Multa Prevista em Norma Coletiva” (código 2215). As palavras em destaque destes assuntos não se confundem com as palavras do assunto “Adminsitração Pública”, que continha 486 documentos no conjunto de treinamento.

Os dois assuntos que apresentaram maior taxa de acerto foram assuntos que continham elevado número de documentos, comparando-se com os demais, embora o terceiro assunto com maior taxa de acerto tenha sido o oposto, pois era o assunto com a menor quantidade de documentos disponíveis. Por outro lado, outros assuntos de elevada representatividade, como por exemplo “Verbas Recisórias” (código 2546), que continha 36.937 ou “Indenização por Dano Moral” (código 1855), que continha 22.976, apresentaram apenas 14,00% e 9,00% de acerto.

Partiu-se então para uma análise do problema de classificação, retomando-se um passo anterior, no entendimento do negócio. Neste trabalho, optou-se por classificar apenas o

Figura 4.18: Nuvem de palavras do assunto 1661 - Horas In Itinere.

Figura 4.19: Nuvem de palavras do assunto 1783 - Comissao.

Figura 4.20: Nuvem de palavras do assunto 5272 - Adminsitração Pública.

assunto principal de um documento, que de acordo com o sistema, recebe uma marcação especial como "Principal". Entretanto, conforme explicado na Seção 4.1, é comum, na justiça do trabalho, que se tenha mais de um assunto por processo. Além disso, a escolha do assunto principal é muitas vezes subjetiva, ou seja, a depender de quem irá analisar o documento, pode-se chegar ao entendimento que o assunto principal é A, enquanto se outra pessoa analisar o mesmo documento, pode-se chegar ao entendimento que o assunto principal é B. Como esta subjetividade é inerente ao contexto negocial, ela também

Tabela 4.12: Resultados da análise de acerto analisando-se qualquer assunto do processo.

Modelo	**Features**	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**
MNB	BM25	56,81%	74,49%	50,88%	54,87%
SVM	TF-IDF	**60,42%**	73,63%	**52,51%**	**55,72%**
RF	BM25	56,95%	**75,21%**	49,24%	52,81%
MLP	LSA - 250	40,75%	73,89%	33,04%	38,46%
	Features	**Macro Acurácia**	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
MNB	BM25	50,88%	43,00%	56,81%	41,67%
SVM	TF-IDF	**52,51%**	40,33%	**60,42%**	**42,02%**
RF	BM25	49,24%	**43,56%**	56,95%	40,48%
MLP	LSA - 250	33,04%	35,04%	40,75%	26,97%

acontece em uma decisão por máquina. Além disso, fazendo-se uma análise manual dos textos, nota-se que a quantidade de palavras relacionadas ao assunto principal não é necessariamente maior do que a quantidade de palavras relacionada aos demais assuntos, ou seja, não há uma relação de predominância do assunto principal. Ainda que em alguns casos possa haver uma predominância de palavras que estejam relacionadas ao assunto principal em específico, não é o que acontece na maior parte dos casos, o que, naturalmente, torna o aprendizado mais difícil.

Considerando esta perspectiva da análise, decidiu-se analisar qual seria a taxa de acerto dos modelos fazendo-se a seguinte pergunta“ *O processo está classificado com o assunto predito pelo modelo, ainda que não seja como assunto principal?*” Para responder a esta pergunta, para cada predição feita, analisou-se o quanto seria a taxa de acerto dos modelos caso o assunto predito fosse um dos assuntos relacionados do processo. Os resultados colhidos seguem na Tabela 4.12, onde nota-se uma melhora bastante significativa em todas as métricas, com destaque para a micro precisão, que apresentou em média 31,48% de melhora, chegando a 75,21% para o modelo Random Forest e 74,49% para o MNB. Analisando-se os outros modelos, nota-se que o SVM e o MLP apresentaram micro precisão bastante similares, 73,63% e 73,89% respectivamente. Estes resultados já podem ser considerados aceitáveis uma vez que não estão tão distantes dos resultados apresentados nos trabalhos relacionados.

Ao aumentar a taxa de acerto quando se desconsidera a escolha do assunto principal e se analisa apenas se o assunto está contido no documento, corrobora-se a subjetividade do assunto principal. Nota-se que os algoritmos, com uma grande quantidade de massa de dados para treinamento, conseguiram distinguir um pouco do ruído e de fato construir algum aprendizado para fazer a identificação de um assunto. Apesar de acertarem um dos assuntos, não necessariamente acertaram o assunto principal, ou seja, não foi possí-

Tabela 4.13: Resultados da análise multirrótulo.

Modelo	**Precisão com 5 Opções**	**Precisão com 10 Opções**
MNB	52,56%	65,38%
SVM	55,77%	70,84%
RF	**55,79**%	**70,89**%
MLP	48,60%	68,22%

vel extrair este conhecimento em específico da base. Uma tabela resumo com todos os resultados obtidos se encontra disponível no Apêndice A.

Uma última análise considera a natureza do problema, em que os documentos possuem mais de um assunto. Faz-se adequada a análise de uma avaliação multirrótulo, onde será analisada a quantidade de acertos que os modelos fariam caso propusessem mais de uma resposta. Retomando a etapa de entendimento dos dados, verificou-se que, excluídos os processos que contêm apenas um assunto, uma vez que se está partindo da premissa que há alta probabilidade de erro nestes documentos dada a natureza da cumulação de pedidos na justiça trabalhista, a média de assuntos de um processo varia entre 5 e 6 assuntos. Assim, foi feita uma análise de qual seria o percentual de acerto dos modelos caso se propusessem a identificar 5 assuntos mais prováveis, e depois, 10 assuntos. Para escolher os 5 ou 10 assuntos, ordenou-se a probabilidade da predição de cada um dos assuntos, e escolheu-se as classes mais prováveis, tomando 5 e 10 assuntos para a predição. Como assuntos alvo, foram considerados apenas os assuntos passíveis de acerto, ou seja, os assuntos dentro do escopo de análise deste trabalho, de forma a se ter uma comparação justa, uma vez que os modelos não poderiam acertar modelos para assuntos que não foram treinados. Assuntos de nível 3 que se encontravam duplicados foram removidos[10].

Na Tabela 4.13, tem-se o percentual de acerto de cada modelo, ou seja, qual foi a porcentagem de assuntos que cada modelo acertou com 5 e 10 opções. Se cada modelo fornecesse 5 opções de assuntos, temos que quase todos os modelos acertariam pelo menos metade dos assuntos preditos, com exceção do MLP, que chegou a apenas 48,60% de acertos. O modelo vencedor para 5 opções de assuntos foi o RF, com precisão de 55,79%. Já para o caso de cada modelo informar 10 opções, cada modelo teve a precisão aumentada em pelo menos 12%, sendo novamente o RF o modelo vencedor, apresentando 70,89% de precisão.

A abordagem multirrótulo seria de grande utilidade enquanto ferramenta para o PJe. Uma possível alteração ao sistema é a integração dos modelos na tela que faz a remessa dos processos, onde os servidores já teriam os Recursos Ordinários juntados ao processo. Neste ponto, seria possível colher o resultado dos modelos, que já teriam lido o documento

[10]A duplicidade de assuntos de nível 3 pode ser causada pela existência de uma classificação com diferentes assuntos de nível 4 ou nível 5, filhos do mesmo assunto de nível 3

no momento em que foram assinados, e então fariam uma sugestão ordenada dos assuntos mais prováveis. Cada assunto de nível 3 poderia ser expandido para os níveis 4 e 5 quando houver opções disponíveis para facilitar a escolha do usuário, que atualmente precisa escolher manualmente entre 877 assuntos.

Importante mencionar ainda um outro impacto da decisão de modelagem multiclasse do problema, que se dá pelo processo de extração e transformação dos dados utilizados. Com a escolha da abordagem multiclasse, que, conforme explicado na Seção 3.3, faz a predição de apenas uma única classe considerando diferentes valores possíveis para esta classificação, faz-se necessário trabalhar com apenas um assunto, o assunto principal. Assim, apenas os processos com este assunto como principal foram coletados, e estes dados foram passados para os algoritmos. Não foi analisado quais eram os demais assuntos destes processos. Ao fazer isso, considerando a forma binária de tratar o problema que foi adotada, é fato que existiram processos utilizados como exemplos negativos que não eram tinham como assunto principal o assunto de interesse, mas que continham este assunto como um dos demais assuntos. Conforme foi explicado na Seção 3.3, onde se abordou o impacto do ruído na base, isso acaba por tornar a tarefa de identificar o que difere um grupo de assuntos do outro mais difícil, uma vez que ambos os conjuntos podem conter palavras relacionadas ao assunto em questão. Como resultado, os modelos acabam não conseguindo captar tantas especificidades que diferenciam o assunto, influenciando negativamente as métricas de avaliação.

4.5 Avaliação

O escopo deste trabalho está delimitado até a avaliação do modelo de aprendizagem de máquina, por meio das métricas descritas na Seção 3.3.1. Uma vez atingidos valores aceitáveis para a precisão, os resultados poderão ser encaminhados para uma equipe especialista, de forma que possam avaliar o resultado pela ótica do negócio. Uma possível forma de fazer esta avaliação seria a partir de uma amostra estratificada dos processos que foram classificados, de forma a se aferir a precisão manualmente, identificando qual foi o percentual de processos que realmente eram do assunto que foi predito pelos modelos. Uma vez aprovados os modelos, pode-se seguir à incorporação dos modelos no PJe, partindo à fase de implantação.

4.6 Implantação

O código utilizado neste estudo se encontra disponível no Apêndice D ,e também no repositório do GitHub do projeto[11] . Este artefato de software foi estruturado de forma que, para um determinado conjunto de dados, possa-se avaliar e eleger o melhor modelo, tornando o trabalho reutilizável para implantação futura, feitos os eventuais ajustes que possam ser necessários.

[11] GitHub: `https://github.com/anacarolinarocha/classificadorDeAssuntos`

Capítulo 5

Conclusões e trabalhos futuros

Nesta seção, são apresentadas as conclusões gerais do trabalho bem como os aspectos que podem ser explorados em estudos futuros.

5.1 Conclusões

Com o elevado número de processos na justiça do trabalho, quer-se criar formas de extrair informações de forma automática das peças processuais que são juntadas aos processos. Assim, este trabalho tem por objetivo avaliar a aplicação de algoritmos de aprendizado de máquina para verificar seu desempenho na atividade de identificar o assunto principal de processos com Recurso Ordinário que chegam ao 2^o grau do PJe.

Dentre as dificuldades encontradas no presente trabalho, tem-se a existência de uma base de dados com elevada taxa de ruído, a dificuldade de se distinguir o assunto principal dos demais assuntos combinada com a sua não predominância textual, e a existência de textos de grandes extensões na maior parte dos dados, com um vocabulário jurídico muito específico.

Inicialmente, foi dada uma explicação sobre a organização da Justiça do Trabalho, e como se da a relação desta Justiça com o sistema PJe. Em seguida, foi feita uma análise exploratória dos dados, onde se analisou a distribuição processos, dos documentos passíveis de análise, o teor dos documentos, a distribuição dos assuntos quanto aos níveis e a distribuição dos processos quanto aos assuntos no nível 3. Estas informações agora se encontram disponíveis para apoio de outras pesquisas que envolvam estes dados.

Conforme revisão da literatura, escolheu-se a aplicação dos algoritmos Multinomial Naïve Bayes, SVM, Multi-layer Perceptron e Random Forest para fazer a predição dos assuntos, combinados com três tipos de representação de textos, os TF-IDF, BM25 e LSA com 100 e 250 tópicos. Foi realizada ainda uma busca de hiper-parâmetros para encontrar a melhor configuração dentre as várias possibilidades testadas em cada algoritmo.

A métrica escolhida para a ordenação dos modelos foi a micro precisão, sendo o melhor resultado apresentado pelo MLP, com 46,01% de micro precisão. O SVM, embora tenha apresentado uma micro precisão menor, de 42,58%, foi o modelo que apresentou os melhores resultados em todas as outras métricas aferidas. O valor de 46,01% de micro precisão foi considerado muito baixo comparado com os demais trabalhos de áreas similares, que apresentaram em média 85,00% de micro precisão, de forma que novas análises foram feitas com os modelos vencedores de cada algoritmo.

Dada a subjetividade da escolha do assunto principal, fez-se a análise das métricas observando-se se o modelo teria predito um dos assuntos que existe no processo, ainda que não fosse o principal. Ao mudar esta perspectiva, as métricas de micro precisão subiram em média 31,48%, sendo que o modelo RF apresentou o valor de 75,21% na micro precisão. Este valor já é considerado razoável dada a dificuldade do trabalho e o a modelagem multiclasse que foi utilizada, que, conforme explicado, acabou por inserir mais ruído na base de treinamento.

Uma última análise foi feita, trabalhando-se com o problema a partir de um ponto de vista multirrótulo. Neste caso, analisou-se qual seria o percentual de assuntos corretos que seriam preditos pelos modelos ao permitir que o modelo tentasse acertar 5 assuntos ou ainda 10 assuntos. Nos dois casos o modelo vencedor foi o SVM. Para o caso de 5 tentativas, o modelo conseguiu acertar 55,77% dos assuntos do processo, e para o caso de 10 tentativas, o modelo acertou 70,84%, ou seja, considerando a quantidade média de 5 assuntos por processo, o modelo acertaria 3 assuntos.

O aumento significativo na métrica de precisão, quando se analisa se houve acerto de qualquer um dos assuntos, é um forte indício da subjetividade desta escolha, deixando a indagação para os especialistas sobre a relevância de se ter um dos assuntos denominados como principal como uma escolha obrigatória no PJe.

Por fim, como último objetivo colocado, foi disponibilizado um artefato de software capaz de receber um arquivo CSV, processá-lo em um pipeline especifico, fazer a criação de modelos com uma busca limitada de hiper-parâmetros, utilizando o texto em diferentes formatos de *features*, e escolher o modelo vencedor baseado na micro precisão.

5.2 Trabalhos futuros

A partir do estudo feito, tem-se alguns espaços que que ainda podem ser explorados para resolver este problema. Acredita-se que o maior desafio encontrado neste trabalho foi a elevada quantidade de ruído na base, indicando que mais esforços devem ser feitos com o objetivo de atenuar este problema, uma vez que se sabe que a elevada quantidade de ruído na base é altamente prejudicial aos algoritmos de classificação. Este problema

pode ser melhor explorado investindo-se mais em classificação manual; aplicando técnicas computacionais específicas para identificar o ruído da base; e ainda trabalhando-se com abordagens semi-supervisionadas, onde apenas parte dos dados precisaria estar corretamente classificada. Ainda sobre a qualidade dos dados, o processo de extração de dados utilizado pode ser reformulado, bem como o pipeline de processamento para a montagem dos modelos binários, para que cada modelo binário utilizasse como contraexemplos apenas processos que não contivessem o assunto alvo como assunto.

Dentre outra possibilidades de trabalhos futuros, pode-se se citar algumas formas de manipulação do texto que não foram exploradas, dentre elas o TF-IDF com n-gramas; o uso de vetores de palavras que tenham sido treinados em um conjunto de textos da justiça do trabalho; a extração de features que representem aspectos linguísticos do texto; a agregação de outras informações como entidades nomeadas, entre outros. Novos avanços também tem sido apresentados fazendo outros tipos de tokenização, como por exemplo, a quebra das palavras em sub-palavras[1].

Outro ponto a ser explorado se refere à hierarquia dos assuntos. Neste trabalho, a hierarquia foi tratada de forma *flat*, ou seja, desconsiderou-se os nós pais para fazer a classificação no nível 3. Estas informações podem ser utilizadas para fazer uso de uma abordagem que considere toda a árvore de assuntos.

Optando-se por um outro ponto de partida, considerando a subjetividade da escolha do assunto principal apresentada, pode-se fazer a modelagem do problema como um problema multirrótulo, utilizando algoritmos específicos para este tipo de abordagem.

Por fim, podem ser testados novos modelos de classificação e outras formas de ensamble não testadas aqui. Em especial, dentro dos modelos de redes neurais, grandes avanços têm sido feitos com as redes recorrentes e vetores de palavras, uma vez que estas conseguem levar em consideração a sequência das palavras, e muito disso se perde ao fazer uso de modelagens como TF-IDF, BM25 ou LSA.

Devido a limitações de recurso disponível para este trabalho, algumas técnicas mais atuais ficaram fora do escopo de análise deste estudo. Dentro do estado da arte que se apresenta atualmente, um dos modelos mais promissores são os modelos de linguagem, que fazem uso da transferência de conhecimento, permitindo extrair conhecimento de um enorme conjunto de textos, tais como ULMFit, Bert e GPT-2. Acredita-se que com a disseminação de modelos pré-treinados em português o uso destes modelos estará cada dia mais viável.

[1]SentencePiece: `https://www.aclweb.org/anthology/D18-2012/`

Referências

[1] Souza, Ellen, Danilo Costa, Dayvid W Castro, Douglas Vitório, Ingryd Teles, Rafaela Almeida, Tiago Alves, Adriano LI Oliveira e Cristine Gusmão: *Characterising text mining: a systematic mapping review of the Portuguese language*. IET Software, 12(2):49–75, jul 2017, ISSN 1751-8814. 1

[2] Feldman, Ronen e James Sanger: *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge university press, 2006, ISBN 978-0-521-83657-9. 1, 23, 26, 27, 29

[3] Cao, Longbing: *Data science: challenges and directions*. Communications of the ACM, 60(8):59–68, julho 2017, ISSN 00010782. 1, 20

[4] Cintra, Antônio C. de Araújo, Ada Pellegrini Grinover e Cândido Rangel Dinamarco: *Teoria Geral do Processo*. Malheiros Editores LTDA., 30ª edição, 1988. 1

[5] Oulasvirta, Antti, Janne P. Hukkinen e Barry Schwartz: *When more is less: The paradox of choice in search engine use*. Em *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, páginas 516–523, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-483-6. `http://doi.acm.org/10.1145/1571941.1572030`. 3

[6] Guan, Donghai e Weiwei Yuan: *A Survey of mislabeled training data detection techniques for pattern classification*. IETE Technical Review, 30:524–530, novembro 2013. 3, 14, 22, 57

[7] Frénay, Benoît e Michel Verleysen: *Classification in the presence of label noise: a survey*. IEEE transactions on neural networks and learning systems, 25(5):845–869, 2013. 3, 22, 57

[8] Loevinger, Lee: *Jurimetrics–the next step forward*. Minn. L. Rev., 33:455, 1948. 3

[9] Nunes, Marcelo Guedes: *Jurimetria: como a estatística pode reinventar o direito*. São Paulo: Editora Revista dos Tribunais, 2016. 4

[10] Armonas Colombo, Bruna, Pedro Buck e Vinicius Miana Bezerra: *Challenges When Using Jurimetrics in Brazil—A Survey of Courts*. Future Internet, 9(4):68, outubro 2017. 4

[11] Daniel Francisco Nagao Menezes, Felipe Chiarello de Souza Pinto e: *Jurimetria: construindo a teoria*. Em *Teoria da decisão e realismo jurídico: XXIII Congresso Nacional do CONPEDI UFPB*, João Pessoa, 2014. CONPEDI, ISBN 978-85-5505-011-4. 4

[12] BRASIL, Conselho Nacional de Justiça: *Justiça em Números 2019*, 2019. `https://www.cnj.jus.br/wp-content/uploads/conteudo/arquivo/2019/08/justica_em_numeros20190919.pdf`. 4, 5, 16, 17, 19

[13] Seger, Giovana Abreu da Silva e Marcelo Seger: *Princípio da segurança jurídica*. Revista Eletrônica Direito e Política, 8(3):2445–2458, 2013, ISSN 1980-7791. 5

[14] Carvalho, Maximiliano Pereira de: *O princípio da automatização do processo eletrônico como catalizador da observância aos precedentes do tst*. Revista Fórum Trabalhista - RFT, 24:89–100, 2017, ISSN 2238-6815. `http://biblioteca2.senado.gov.br:8991/F/UM3351MGC2LVSYD8UPUEAL7AA9BD6YYS3HBYUTN8RCP5QAGPPG-00509?func=full-set-set&set_number=000149&set_entry=000002&format=999`. 5, 18

[15] Ferauche, Thiago: *Aplicação de Técnicas de Mineração de Textos para Classificação de Ementas da Jurisprudência de Justiça do Trabalho de São Paulo*. Tese de Mestrado, Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2011. 9, 15

[16] Ticom, Antonio Alexandre Mello: *Aplicação das técnicas de mineração de textos e sistemas especialistas na liquidação de processos trabalhistas*. Tese de Mestrado, UFRJ, 2007. 10, 15, 61

[17] Ferreira, Marcelo Herton Pereira: *Classificação de peças processuais jurídicas: Inteligência artificial no direito*. Monografia (Bacharel em Engenharia de Software), UnB (Universidade de Brasília), Brasília, Brazil. 10

[18] Pinto, Luis e Andrés Melgar: *A classification model for portuguese documents in the juridical domain*. Em *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–4. IEEE, 2016. 11

[19] Soh, Jerrold, How Khang Lim e Ian Ernst Chai: *Legal area classification: A comparative study of text classifiers on singapore supreme court judgments*. Em *Proceedings of the Natural Legal Language Processing Workshop 2019*, páginas 67–77, 2019. 11, 34, 61, 66

[20] Szymański, Piotr e Tomasz Kajdanowicz: *A network perspective on stratification of multi-label data*. Em Torgo, Luís, Bartosz Krawczyk, Paula Branco e Nuno Moniz (editores): *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 74 de *Proceedings of Machine Learning Research*, páginas 22–35, ECML-PKDD, Skopje, Macedonia, 22 Sep 2017. PMLR. 12

[21] Lei, Miaomiao, Jidong Ge, Zhongjin Li, Chuanyi Li, Yemao Zhou, Xiaoyu Zhou e Bin Luo: *Automatically classify chinese judgment documents utilizing machine learning algorithms*. Em *International Conference on Database Systems for Advanced Applications*, páginas 3–17. Springer, 2017. 13

[22] Aletras, Nikolaos, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro e Vasileios Lampos: *Predicting judicial decisions of the European Court of Human Rights: a Natural Language Processing perspective*. PeerJ Computer Science, 2:e93, outubro 2016, ISSN 2376-5992. 13

[23] Xiang, Zheng, Qianzhou Du, Yufeng Ma e Weiguo Fan: *Assessing Reliability of Social Media Data: Lessons from Mining TripAdvisor Hotel Reviews*. Em Schegg, Roland e Brigitte Stangl (editores): *Information and Communication Technologies in Tourism 2017*, páginas 625–638. Springer International Publishing, Cham, 2017, ISBN 978-3-319-51167-2 978-3-319-51168-9. 13, 61

[24] Liu, X., Y. Dai, Y. Zhang, Q. Yuan e L. Zhao: *A preprocessing method of AdaBoost for mislabeled data classification*. Em *2017 29th Chinese Control And Decision Conference (CCDC)*, páginas 2738–2742, maio 2017. 14

[25] Shi, Yong, Peijia Li e Lingfeng Niu: *Augmented SVM with ordinal partitioning for text classification*. Em *Proceedings of the International Conference on Web Intelligence*, páginas 959–962. ACM Press, 2017. 14

[26] Andrade, Patrícia Helena Maia Alves: *Aplicação de Técnicas de Mineração de Textos para Classificação de Documentos: um Estudo da Automatização da Triagem de Denúncias na CGU*. Tese de Mestrado, Universidade de Brasília, 2015. 14

[27] Brink, Henrik, Joseph Richards e Mark Fetherolf: *Real-world machine learning*. Manning Publications Co., 2016. 21, 22, 23

[28] Silva, Leandro Nunes de Castro e Daniel Gomes Ferrar: *Introdução à Mineração de Dados. Conceitos Básicos, Algoritmos e Aplicações*. Saraiva, edição: 1ª edição, fevereiro 2016, ISBN 978-85-472-0098-5. 21, 22, 23, 31, 32

[29] Han, Jiawei, Jian Pei e Micheline Kamber: *Data mining: concepts and techniques*. Elsevier, 2011. 21, 36, 37

[30] Silla, Carlos N e Alex A Freitas: *A survey of hierarchical classification across different application domains*. Data Mining and Knowledge Discovery, 22(1-2):31–72, 2011. 23, 24

[31] Krawczyk, Bartosz: *Learning from imbalanced data: open challenges and future directions*. 2016. 24

[32] Haixiang, Guo, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue e Gong Bing: *Learning from class-imbalanced data: Review of methods and applications*. Expert Systems with Applications, 73:220–239, maio 2017, ISSN 09574174. `https://linkinghub.elsevier.com/retrieve/pii/S0957417416307175`, acesso em 2018-10-28. 24

[33] *Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches - ScienceDirect.* https://www-sciencedirect.ez54.periodicos.capes.gov.br/science/article/pii/S0950705113000300, acesso em 2018-10-28. 24, 25

[34] Khoshgoftaar, Taghi M, Jason Van Hulse e Amri Napolitano: *Comparing boosting and bagging techniques with noisy and imbalanced data.* IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 41(3):552–568, 2010. 25

[35] Hirschberg, Julia e Christopher D Manning: *Advances in natural language processing.* Science, 349(6245):261–266, 2015. 26

[36] Kowsari, Kamran, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes e Donald Brown: *Text classification algorithms: A survey.* Information, 10(4):150, 2019. 26, 27, 29, 37

[37] Mirończuk, Marcin Michał e Jarosław Protasiewicz: *A recent overview of the state-of-the-art elements of text classification.* Expert Systems with Applications, 106:36–54, setembro 2018, ISSN 0957-4174. 26, 27, 29

[38] Baharudin, Baharum, Lam Hong Lee e Khairullah Khan: *A Review of Machine Learning Algorithms for Text-Documents Classification.* Journal of Advances in Information Technology, 1(1), fevereiro 2010, ISSN 1798-2340. 26, 29

[39] Laender, Alberto H. F., Berthier A. Ribeiro-Neto, Altigran S. da Silva e Juliana S. Teixeira: *A Brief Survey of Web Data Extraction Tools.* SIGMOD Rec., 31(2):84–93, junho 2002, ISSN 0163-5808. 26, 29

[40] Souza, Ellen, Danilo Costa, Dayvid W Castro, Douglas Vitório, Ingryd Teles, Rafaela Almeida, Tiago Alves, Adriano LI Oliveira e Cristine Gusmão: *Characterising text mining: a systematic mapping review of the portuguese language.* IET Software, 12(2):49–75, 2017. 27

[41] Singh, Jasmeet e Vishal Gupta: *Text stemming: Approaches, applications, and challenges.* ACM Comput. Surv., 49(3):45:1–45:46, setembro 2016, ISSN 0360-0300. http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/2975608. 27

[42] *Feature selection for text classification with Naïve Bayes.* Expert Systems with Applications, 36(3):5432–5435, abril 2009, ISSN 0957-4174. https://www-sciencedirect.ez54.periodicos.capes.gov.br/science/article/pii/S0957417408003564, acesso em 2019-01-12. 27, 28

[43] Alvarez, Jon Ezeiza e Hannah Bast: *A review of word embedding and document similarity algorithms applied to academic text*, 2017. 28, 34

[44] HE, Ben e Iadh Ounis: *A study of parameter tuning for term frequency normalization.* Em *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, páginas 10–16, New York, NY, USA, 2003. ACM, ISBN 1-58113-723-0. 28

[45] Landauer, Thomas K e Susan T Dumais: *A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.* Psychological review, 104(2):211, 1997. 29

[46] Stewart, G. W.: *On the Early History of the Singular Value Decomposition.* SIAM Review, 35(4):551–566, 1993, ISSN 0036-1445. 30

[47] Bradford, Roger B.: *An empirical study of required dimensionality for large-scale latent semantic indexing applications.* Em *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, página 153, Napa Valley, California, USA, 2008. ACM Press, ISBN 978-1-59593-991-3. `http://portal.acm.org/citation.cfm?doid=1458082.1458105`, acesso em 2018-11-14. 30

[48] Brighton, Henry e Chris Mellish: *Advances in Instance Selection for Instance-Based Learning Algorithms.* Data Mining and Knowledge Discovery, 6(2):153–172, abril 2002, ISSN 1573-756X. `https://doi.org/10.1023/A:1014043630878`, acesso em 2018-12-27. 30

[49] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks.* Machine Learning, 20(3):273–297, setembro 1995, ISSN 0885-6125, 1573-0565. 31

[50] Joachims, Thorsten: *Text categorization with Support Vector Machines: Learning with many relevant features.* Em Carbonell, Jaime G., Jörg Siekmann, G. Goos, J. Hartmanis, J. van Leeuwen, Claire Nédellec e Céline Rouveirol (editores): *Machine Learning: ECML-98*, volume 1398, páginas 137–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, ISBN 978-3-540-64417-0 978-3-540-69781-7. `http://link.springer.com/10.1007/BFb0026683`, acesso em 2018-11-18. 31

[51] Breiman, Leo: *Random Forests.* Machine Learning, 45(1):5–32, 2001, ISSN 08856125. 31, 32

[52] Bussab, Wilton O. e Pedro A. Morettin: *Estatística Básica - 9ª Ed. 2017*, 2017. 32

[53] McCallum, Andrew, Kamal Nigam *et al.*: *A comparison of event models for naive bayes text classification.* Em *AAAI-98 workshop on learning for text categorization*, volume 752, páginas 41–48. Citeseer, 1998. 32

[54] Sanderson, Mark, D Christopher, Hinrich Manning *et al.*: *Introduction to Information Retrieval.* Cambridge University Press, 2018. `https://nlp.stanford.edu/IR-book/information-retrieval-book.html`. 33

[55] Haykin, Simon: *Redes Neurais - Principios e Praticas.* Bookman, 2ª edição, 2003, ISBN 85-7307-718-2. 33

[56] Li, Yang e Tao Yang: *Word embedding for understanding natural language: a survey.* Em *Guide to Big Data Applications*, páginas 83–104. Springer, 2018. 34

[57] Bengio, Yoshua, Réjean Ducharme, Pascal Vincent e Christian Jauvin: *A neural probabilistic language model.* Journal of machine learning research, 3(Feb):1137–1155, 2003. 34

[58] Xu, Wei e Alex Rudnicky: *Can artificial neural networks learn language models?* Em *Sixth International Conference on Spoken Language Processing*, 2000. 34

[59] Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu e Pavel Kuksa: *Natural language processing (almost) from scratch.* Journal of machine learning research, 12(Aug):2493–2537, 2011. 34

[60] Mikolov, Tomas, Kai Chen, Greg Corrado e Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space.* arXiv:1301.3781 [cs], janeiro 2013. arXiv: 1301.3781. 34

[61] Pennington, Jeffrey, Richard Socher e Christopher Manning: *Glove: Global vectors for word representation.* Em *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, páginas 1532–1543, 2014. 34

[62] Bojanowski, Piotr, Edouard Grave, Armand Joulin e Tomas Mikolov: *Enriching word vectors with subword information.* Transactions of the Association for Computational Linguistics, 5:135–146, 2017. 34

[63] Young, Tom, Devamanyu Hazarika, Soujanya Poria e Erik Cambria: *Recent trends in deep learning based natural language processing.* ieee Computational intelligenCe magazine, 13(3):55–75, 2018. 35

[64] Peters, Matthew E, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee e Luke Zettlemoyer: *Deep contextualized word representations.* Em *Proceedings of NAACL-HLT*, páginas 2227–2237, 2018. 35

[65] Howard, Jeremy e Sebastian Ruder: *Universal language model fine-tuning for text classification.* Em *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 328–339, 2018. 35

[66] Devlin, Jacob, Ming Wei Chang, Kenton Lee e Kristina Toutanova: *BERT: Pre-training of deep bidirectional transformers for language understanding.* Em *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 4171–4186, Minneapolis, Minnesota, junho 2019. Association for Computational Linguistics. `https://www.aclweb.org/anthology/N19-1423`. 35

[67] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei e Ilya Sutskever: *Language models are unsupervised multitask learners.* OpenAI Blog, 1(8), 2019. 35

[68] Shi, Shaohuai, Qiang Wang, Pengfei Xu e Xiaowen Chu: *Benchmarking state-of-the-art deep learning software tools.* Em *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, páginas 99–104. IEEE, 2016. 35

[69] *CRISP-DM by Smart Vision Europe.* `http://crisp-dm.eu/`, acesso em 2018-06-04. 38, 39

[70] Bezerra Leite, Carlos Henrique: *Curso de Direito Processual do Trabalho.* Saraiva Jur, 2019, ISBN 9788553602674. 41

[71] Eça, Vitor Salino de Moura: *Direito Processual do Trabalho.* LTr, 2019, ISBN 9788530100261. 41

[72] Dias, Carlos Eduardo de Oliveira, Guilherme Guimarães Feliciano e Manoel Carlos Toledo Filho: *Comentários ao novo cpc e sua aplicação ao processo do trabalho.* São Paulo: LTr, 2, 2017. 41

[73] Lima, Leonardo Tibo Barbosa: *Lições de Direito Processual do Trabalho: Teoria e Prática.* LTr, 2019, ISBN 9788530100452. 41

[74] Oliveira, Fernando José Vianna: *Os recursos na Justiça do Trabalho - Artigos - Conteúdo Jurídico.* http://www.conteudojuridico.com.br/artigo,os-recursos-na-justica-do-trabalho,32695.html, acesso em 2018-05-31. 43

[75] Castro, Marco Antonio de, Nayara Regina Giroldo Bacanof, Valdeleni Aparecida Mendes Alquier e Renato Luiz de Avelar Bandini: *Agravo de Petição no Âmbito Trabalhista.* JICEX, 3(3), março 2015, ISSN 2357-867X. 43

[76] *Agravo de instrumento em matéria trabalhista.* https://www.direitonet.com.br/artigos/exibir/10482/Agravo-de-instrumento-em-materia-trabalhista. 43

[77] *Tabelas Processuais Unificadas - TST.* http://www.tst.jus.br/web/corregedoria/tabelas-processuais, acesso em 2018-05-28. 49

Apêndice A

Tabelas complementares

Tabela A.1: Distribuição de processos por assunto nível 3 (Parte I).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
2546 - Verbas Rescisórias	422894	18.28%	18.28%
2086 - Horas Extras	239341	10.35%	28.63%
1855 - Indenização por Dano Moral	237597	10.27%	38.90%
2594 - Adicional	186050	8.04%	46.94%
2458 - Salário / Diferença Salarial	145764	6.30%	53.24%
2704 - Tomador de Serviços / Terceirização	80903	3.50%	56.74%
2656 - Reintegração / Readmissão ou Indenização	75848	3.28%	60.01%
2140 - Intervalo Intrajornada	63085	2.73%	62.74%
2435 - Rescisão Indireta	56775	2.45%	65.20%
2029 - FGTS	48981	2.12%	67.31%
2583 - Abono	46530	2.01%	69.32%
2554 - Reconhecimento de Relação de Emprego	44634	1.93%	71.25%
8808 - Indenização por Dano Material	34091	1.47%	72.73%
2117 - Supressão / Redução de Horas Extras Habituais - Indenização	30467	1.32%	74.04%
2021 - Indenização / Dobra / Terço Constitucional	28646	1.24%	75.28%
5280 - Bancários	27352	1.18%	76.47%
1904 - Despedida / Dispensa Imotivada	25977	1.12%	77.59%
1844 - CTPS	25421	1.10%	78.69%
2055 - Gratificação	21318	0.92%	79.61%
1907 - Justa Causa / Falta Grave	21001	0.91%	80.52%
1806 - Alteração Contratual ou das Condições de Trabalho	19427	0.84%	81.36%
55220 - Indenização por Dano Moral	18080	0.78%	82.14%
2506 - Ajuda / Tíquete Alimentação	16668	0.72%	82.86%
4437 - Revisão de Sentença Normativa	15528	0.67%	83.53%
10570 - FGTS	15127	0.65%	84.18%
1783 - Comissão	15041	0.65%	84.83%
1888 - Descontos Salariais - Devolução	14666	0.63%	85.47%
2478 - Seguro Desemprego	14606	0.63%	86.10%
5356 - Grupo Econômico	14604	0.63%	86.73%
1773 - Contribuição Sindical	14390	0.62%	87.35%
1663 - Adicional Noturno	12040	0.52%	87.87%
5272 - Administração Pública	12012	0.52%	88.39%
2215 - Multa Prevista em Norma Coletiva	11292	0.49%	88.88%
1767 - Cesta Básica	10672	0.46%	89.34%
1661 - Horas in Itinere	9897	0.43%	89.77%
1690 - Contribuição / Taxa Assistencial	**8848**	**0.38%**	**90.15%**

Tabela A.2: Distribuição de processos por assunto nível 3 (Parte II).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
4435 - Norma Coletiva - Aplicabilidade / Cumprimento	8572	0.37%	90.52%
55348 - Direito de Greve	8002	0.35%	90.87%
2624 - Complementação de Aposentadoria / Pensão	7972	0.34%	91.21%
2139 - Intervalo Interjornadas	7058	0.31%	91.52%
2019 - Fruição / Gozo	6791	0.29%	91.81%
55345 - Acordo e Convenção Coletivos de Trabalho	6782	0.29%	92.10%
2364 - Plano de Saúde	6723	0.29%	92.39%
55405 - Quitação	6692	0.29%	92.68%
2426 - Repouso Semanal Remunerado e Feriado	6256	0.27%	92.95%
55056 - Judicial	6100	0.26%	93.22%
55170 - Participação nos Lucros ou Resultados - PLR	6001	0.26%	93.48%
2663 - Abono Pecuniário	5851	0.25%	93.73%
55095 - Compensação de Jornada	5414	0.23%	93.96%
2116 - Sobreaviso / Prontidão / Tempo à Disposição	5347	0.23%	94.20%
55172 - Quebra de Caixa	5118	0.22%	94.42%
55108 - Alteração da Jornada	4893	0.21%	94.63%
10581 - Turno Ininterrupto de Revezamento	4693	0.20%	94.83%
10571 - Acidente de Trabalho	4550	0.20%	95.03%
1816 - Contrato por Prazo Determinado	4546	0.20%	95.22%
5294 - Professores	4349	0.19%	95.41%
55091 - Enquadramento Sindical	4019	0.17%	95.59%
2540 - Vale Transporte	3959	0.17%	95.76%
2666 - Décimo Terceiro Salário	3605	0.16%	95.91%
7631 - Honorários Profissionais	3569	0.15%	96.07%
5288 - Digitadores / Mecanógrafos / Datilógrafos	3564	0.15%	96.22%
55355 - Sentença Normativa	3189	0.14%	96.36%
2331 - Prêmio	3105	0.13%	96.49%
8805 - Sucessão de Empregadores	3070	0.13%	96.63%
2537 - Unicidade Contratual	2859	0.12%	96.75%
55219 - Indenização por Dano Material	2575	0.11%	96.86%
10564 - Contribuição Sindical Rural	2396	0.10%	96.96%
8806 - Subempreitada	2143	0.09%	97.06%
1920 - Diárias	2074	0.09%	97.15%
55204 - Pedido de Demissão	2018	0.09%	97.23%

Tabela A.3: Distribuição de processos por assunto nível 3 (Parte III).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
1691 - Contribuição Confederativa	1937	0.08%	97.32%
4442 - Restituição / Indenização de Despesa	1935	0.08%	97.40%
55105 - Advogados	1817	0.08%	97.48%
5276 - Controle de Jornada	1817	0.08%	97.56%
4452 - Representação Sindical	1806	0.08%	97.64%
1703 - Eleição de Dirigente Sindical	1768	0.08%	97.71%
2243 - Plano de Demissão Voluntária / Incentivada	1707	0.07%	97.79%
55216 - Indenização por Dano Moral Coletivo	1559	0.07%	97.85%
4438 - Norma Coletiva - Anulação	1518	0.07%	97.92%
55386 - Desvio de Função e Reenquadramento	1489	0.06%	97.98%
7633 - Trabalhador Avulso	1432	0.06%	98.05%
2477 - Seguro de Vida	1413	0.06%	98.11%
55055 - Extrajudicial	1350	0.06%	98.17%
2409 - Enquadramento / Classificação	1346	0.06%	98.22%
8807 - Sócio / Acionista	1334	0.06%	98.28%
7629 - Empreitada	1152	0.05%	98.33%
5273 - Suspensão / Interrupção do Contrato de Trabalho	1137	0.05%	98.38%
55340 - Responsabilidade	1117	0.05%	98.43%
55150 - Complementação de Benefício Previdenciário	1099	0.05%	98.48%
55088 - Comprovação de Repasse da Contribuição Sindical	1078	0.05%	98.52%
55322 - Empregados Portuários	1014	0.04%	98.57%
1789 - Complemento Temporário Variável de Ajuste ao Piso de Mercado	994	0.04%	98.61%
5284 - Engenheiro, Arquiteto e Engenheiro Agrônomo	982	0.04%	98.65%
2557 - Contrato de Aprendizagem	924	0.04%	98.69%
55385 - Comissões	911	0.04%	98.73%
2606 - Ajuda de Custo	904	0.04%	98.77%
1705 - Registro de Entidade Sindical	895	0.04%	98.81%
55053 - Enquadramento	893	0.04%	98.85%
55113 - Base de Cálculo	852	0.04%	98.88%
7646 - Rural	836	0.04%	98.92%
2559 - Contrato de Estágio	825	0.04%	98.96%
55192 - Aposentadoria	823	0.04%	98.99%
55202 - Morte	806	0.03%	99.03%
2273 - PIS - Indenização	734	0.03%	99.06%

Tabela A.4: Distribuição de processos por assunto nível 3 (Parte IV).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
55400 - Sexta Parte	729	0.03%	99.09%
2450 - Gorjeta	720	0.03%	99.12%
5296 - Radialistas	718	0.03%	99.15%
55209 - Indenização por Dano Estético	699	0.03%	99.18%
5297 - Serviços de Telefonia ouTelegrafia	683	0.03%	99.21%
5277 - Aeronautas	661	0.03%	99.24%
5282 - Domésticos	655	0.03%	99.27%
2349 - Contribuição de Previdência Privada - Resgate	653	0.03%	99.30%
5301 - Vigia e Vigilantes	581	0.03%	99.32%
55087 - Multa por Atraso de Contribuição Sindical	553	0.02%	99.35%
5279 - Atleta Profissional	543	0.02%	99.37%
5299 - Trabalhadores em Petróleo	538	0.02%	99.39%
55382 - Ação Trabalhista Arquivada - Interrupção	521	0.02%	99.41%
55043 - Ferroviários	503	0.02%	99.44%
8813 - Licenças e Folgas - Conversão em Pecúnia	473	0.02%	99.46%
2233 - Contratação de Reabilitados e Deficientes Habilitados	456	0.02%	99.48%
55403 - Fraude	449	0.02%	99.50%
55008 - Prorrogação de Sentença Normativa	435	0.02%	99.51%
7630 - Representante Comercial Autônomo	430	0.02%	99.53%
55197 - Falência	425	0.02%	99.55%
55423 - Concessão de Serviço Público	423	0.02%	99.57%
55225 - Regime Jurídico - Mudança	394	0.02%	99.59%
7645 - Aeroviários	371	0.02%	99.60%
55343 - Responsabilidade	339	0.01%	99.62%
2670 - Cooperativa de Trabalho	338	0.01%	99.63%
2558 - Advertência / Suspensão	338	0.01%	99.65%
55034 - Outras Categorias Profissionais	336	0.01%	99.66%
55383 - Alteração Contratual	316	0.01%	99.68%
8824 - Indenização por Tempo de Serviço	310	0.01%	99.69%
55344 - CTPS	307	0.01%	99.70%
5286 - Jornalistas	291	0.01%	99.71%
55347 - Anulação de Constituição de Sindicato	290	0.01%	99.73%
55149 - Ajuda Quilometragem	285	0.01%	99.74%
55199 - Indenização por Rescisão Antecipada de Contrato a Termo	270	0.01%	99.75%

Tabela A.5: Distribuição de processos por assunto nível 3 (Parte V).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
1957 - PIS / RAIS - Cadastramento	268	0.01%	99.76%
55339 - Extinção Normal do Contrato a Termo	258	0.01%	99.77%
55195 - Juros de Mora	258	0.01%	99.79%
55196 - Extinção do Estabelecimento / Empresa	256	0.01%	99.80%
5287 - Marítimos	243	0.01%	99.81%
55148 - Ajuda Combustível	240	0.01%	99.82%
55122 - Trabalhador Autônomo Não Especificificado	234	0.01%	99.83%
55082 - Reintegração de Posse - Despejo	214	0.01%	99.84%
2421 - Mulher	216	0.01%	99.85%
55009 - Extensão de Sentença Normativa	208	0.01%	99.85%
55081 - Menor	199	0.01%	99.86%
1849 - Culpa Recíproca	192	0.01%	99.87%
55338 - Incidência em Indenização PDV / PDI	187	0.01%	99.88%
55354 - Prazo de Vigência - Norma Coletiva	178	0.01%	99.89%
5289 - Médicos	169	0.01%	99.89%
55104 - Trabalho Externo	163	0.01%	99.90%
55070 - Exame Médico	162	0.01%	99.91%
55115 - Férias Coletivas	147	0.01%	99.92%
55397 - Auxílio Creche	130	0.01%	99.92%
7632 - Trabalhador Eventual	119	0.01%	99.93%
2493 - Tarefa	115	0.01%	99.93%
55080 - Deficiente Físico	102	0.00%	99.94%
55342 - Forma de Cálculo	95	0.00%	99.94%
5290 - Mineiros de Subsolos	89	0.00%	99.94%
2133 - Inquérito Administrativo - Validade	87	0.00%	99.95%
55041 - Enfermeiros	86	0.00%	99.95%
55120 - Mandato	82	0.00%	99.95%
55052 - Rural	69	0.00%	99.96%
10569 - Técnico em Radiologia	69	0.00%	99.96%
55384 - Complementação de Aposentadoria / Pensão	67	0.00%	99.96%
55121 - Parceria	66	0.00%	99.97%
55159 - Gueltas	63	0.00%	99.97%
55337 - Forma de Cálculo	60	0.00%	99.97%
55040 - Enfermagem	52	0.00%	99.97%

Tabela A.6: Distribuição de processos por assunto nível 3 (Parte VI).

Assunto	**Quantidade de Processos**	**Percentual**	**Frequência Cumulativa**
55036 - Corretores de Imóveis	51	0.00%	99.98%
5291 - Músicos Profissionais	50	0.00%	99.98%
55010 - Espontânea	45	0.00%	99.98%
55118 - Contrato em Regime de Tempo Parcial	41	0.00%	99.98%
7647 - Diarista	42	0.00%	99.98%
55021 - Administradores	39	0.00%	99.99%
5292 - Operadores de Carga e Descarga (Estiva e Capatazia)	37	0.00%	99.99%
5278 - Artistas	32	0.00%	99.99%
55198 - Força Maior / Factum Principis	31	0.00%	99.99%
55169 - Retribuição por Invenção e Patente	27	0.00%	99.99%
55037 - Corretores de Seguros	22	0.00%	99.99%
55044 - Fisioterapeutas / Terapeutas Ocupacionais	19	0.00%	99.99%
55116 - Assistentes Sociais	16	0.00%	99.99%
55065 - Teletrabalho / Trabalho à Distância / Trabalho em Domicílio	17	0.00%	99.99%
55024 - Contrato de Equipe	17	0.00%	99.99%
55123 - Químicos	13	0.00%	100.00%
5295 - Trabalhador Voluntário	15	0.00%	100.00%
55049 - Relações Públicas	12	0.00%	100.00%
55035 - Corretagem	11	0.00%	100.00%
5293 - Operadores Cinematográficos	11	0.00%	100.00%
55117 - Contabilistas	12	0.00%	100.00%
55045 - Nutricionistas	10	0.00%	100.00%
55048 - Cabineiros de Elevador	6	0.00%	100.00%
5281 - Biólogos	6	0.00%	100.00%
55387 - Publicitários	6	0.00%	100.00%
55032 - Expurgos Inflacionários	6	0.00%	100.00%
55119 - Mãe Social	5	0.00%	100.00%
55031 - Bibliotecários	4	0.00%	100.00%
55050 - Economistas	3	0.00%	100.00%
55047 - Psicólogos	3	0.00%	100.00%
55039 - Secretários	3	0.00%	100.00%
55079 - Indígena	1	0.00%	100.00%

Tabela A.7: Distribuição de processos por assunto no conjunto de treinamento .

Código do Assunto	**Quantidade de Documentos**
2546	36937
1855	22976
2594	11815
2086	10797
2458	8842
2704	6557
2021	4752
2656	3874
2140	3680
2117	3560
2435	2817
2029	2349
1904	1858
2583	1854
2215	1781
10570	1736
2478	1644
2554	1559
8808	1445
1888	1290
5280	1257
1663	1226
1806	1185
2055	1166
2506	1054
1661	1043
1783	1043
1767	976
1844	919
1907	744
5272	486
5356	450
1690	399
1773	259
4437	207

Tabela A.8: Resultados do GridSearch para o MNB.

0.388 (+/-0.010) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.5}
0.390 (+/-0.009) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___alpha': 0.001}
0.391 (+/-0.008) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___alpha': 0.01}
0.385 (+/-0.006) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.001}
0.387 (+/-0.010) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.1}
0.387 (+/-0.004) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.1}
0.390 (+/-0.008) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___alpha': 1}
0.389 (+/-0.008) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.01}
0.392 (+/-0.008) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___alpha': 0.5}
0.388 (+/-0.005) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___alpha': 0.001}

Tabela A.9: Resultados do GridSearch para o SVM.

0.372 (+/-0.009) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___base_estimator___C': 0.01}
0.380 (+/-0.010) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 0.01}
0.374 (+/-0.004) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 10}
0.392 (+/-0.004) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 1}
0.389 (+/-0.006) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 0.1}
0.367 (+/-0.005) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___base_estimator___C': 10}
0.389 (+/-0.005) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 0.1}
0.389 (+/-0.004) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___base_estimator___C': 1}
0.382 (+/-0.005) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___base_estimator___C': 1}
0.385 (+/-0.004) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___base_estimator___C': 1}

Tabela A.10: Resultados do GridSearch para o RF.

0.395 (+/-0.006) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 200, 'estimator___base_estimator___min_samples_split': 0.05, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.8, 'estimator___base_estimator___max_depth': 50}
0.000 (+/-0.000) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 100, 'estimator___base_estimator___min_samples_split': 0.05, 'estimator___base_estimator___min_samples_leaf': 0.5, 'estimator___base_estimator___max_features': 0.3, 'estimator___base_estimator___max_depth': 100}
0.404 (+/-0.006) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 200, 'estimator___base_estimator___min_samples_split': 0.05, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.5, 'estimator___base_estimator___max_depth': 30}
0.000 (+/-0.000) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 100, 'estimator___base_estimator___min_samples_split': 0.1, 'estimator___base_estimator___min_samples_leaf': 0.5, 'estimator___base_estimator___max_features': 0.5, 'estimator___base_estimator___max_depth': 50}
0.395 (+/-0.003) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 300, 'estimator___base_estimator___min_samples_split': 0.1, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.8, 'estimator___base_estimator___max_depth': 100}
0.390 (+/-0.009) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___n_estimators': 300, 'estimator___base_estimator___min_samples_split': 0.1, 'estimator___base_estimator___min_samples_leaf': 0.1, 'estimator___base_estimator___max_features': 0.5, 'estimator___base_estimator___max_depth': 100}
0.404 (+/-0.006) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___n_estimators': 200, 'estimator___base_estimator___min_samples_split': 0.1, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.3, 'estimator___base_estimator___max_depth': 100}
0.398 (+/-0.009) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___n_estimators': 100, 'estimator___base_estimator___min_samples_split': 0.05, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.8, 'estimator___base_estimator___max_depth': 50}
0.392 (+/-0.015) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___n_estimators': 200, 'estimator___base_estimator___min_samples_split': 0.5, 'estimator___base_estimator___min_samples_leaf': 0.05, 'estimator___base_estimator___max_features': 0.3, 'estimator___base_estimator___max_depth': 30}
0.000 (+/-0.000) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___n_estimators': 300, 'estimator___base_estimator___min_samples_split': 0.05, 'estimator___base_estimator___min_samples_leaf': 0.5, 'estimator___base_estimator___max_features': 0.8, 'estimator___base_estimator___max_depth': 50}

Tabela A.11: Resultados do GridSearch para o MLP (Parte I).

0.000 (+/-0.000) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'sgd', 'estimator___base_estimator___max_iter': 300, 'estimator___base_estimator___learning_rate': 'invscaling', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'relu'}
0.033 (+/-0.052) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'sgd', 'estimator___base_estimator___max_iter': 300, 'estimator___base_estimator___learning_rate': 'constant', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.001, 'estimator___base_estimator___activation': 'logistic'}
0.351 (+/-0.011) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'adam', 'estimator___base_estimator___max_iter': 200, 'estimator___base_estimator___learning_rate': 'adaptive', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.1, 'estimator___base_estimator___activation': 'relu'}
0.425 (+/-0.013) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___solver': 'lbfgs', 'estimator___base_estimator___max_iter': 400, 'estimator___base_estimator___learning_rate': 'constant', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'logistic'}
0.009 (+/-0.001) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'adam', 'estimator___base_estimator___max_iter': 400, 'estimator___base_estimator___learning_rate': 'invscaling', 'estimator___base_estimator___hidden_layer_sizes': (10, 5, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'logistic'}
0.366 (+/-0.005) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.8, 'estimator___base_estimator___solver': 'lbfgs', 'estimator___base_estimator___max_iter': 400, 'estimator___base_estimator___learning_rate': 'constant', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'tanh'}
0.322 (+/-0.043) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'sgd', 'estimator___base_estimator___max_iter': 400, 'estimator___base_estimator___learning_rate': 'constant', 'estimator___base_estimator___hidden_layer_sizes': (10, 10), 'estimator___base_estimator___alpha': 0.001, 'estimator___base_estimator___activation': 'relu'}

Tabela A.12: Resultados do GridSearch para o MLP (Parte II).

0.063 (+/-0.000) for {'estimator___n_estimators': 3, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'sgd', 'estimator___base_estimator___max_iter': 300, 'estimator___base_estimator___learning_rate': 'invscaling', 'estimator___base_estimator___hidden_layer_sizes': (10, 5, 10), 'estimator___base_estimator___alpha': 0.1, 'estimator___base_estimator___activation': 'identity'}
0.198 (+/-0.012) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.5, 'estimator___base_estimator___solver': 'sgd', 'estimator___base_estimator___max_iter': 400, 'estimator___base_estimator___learning_rate': 'constant', 'estimator___base_estimator___hidden_layer_sizes': (10, 5, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'relu'}
0.401 (+/-0.017) for {'estimator___n_estimators': 5, 'estimator___max_samples': 0.8, 'estimator___base_estimator___solver': 'lbfgs', 'estimator___base_estimator___max_iter': 300, 'estimator___base_estimator___learning_rate': 'invscaling', 'estimator___base_estimator___hidden_layer_sizes': (10, 5, 10), 'estimator___base_estimator___alpha': 0.05, 'estimator___base_estimator___activation': 'tanh'}

Tabela A.13: Resultados consolidados

Análise do acerto do assunto principal									
Features	**Modelo**	**Micro Acurácia**	**Micro Precisão**	**Micro Revocação**	**Micro F-Measure**	**Macro Acurácia**	**Macro Precisão**	**Macro Revocação**	**Macro F-Measure**
TF-IDF (GS)	MNB	29,35%	38,75%	22,43%	23,02%	22,43%	19,69%	29,35%	18,65%
TF-IDF (GS)	SVM	33,74%	39,50%	24,73%	25,61%	24,73%	20,83%	33,74%	21,44%
TF-IDF (GS)	RF	28,77%	40,73%	19,44%	19,68%	19,44%	20,24%	28,77%	18,09%
TF-IDF (GS)	MLP	27,44%	44,26%	18,21%	15,72%	17,74%	18,10%	29,10%	17,06%
TF-IDF	MNB	35,30%	40,77%	27,89%	27,76%	27,89%	**24,32%**	35,30%	24,06%
TF-IDF	**SVM**	**39,67%**	42,59%	**29,30%**	**29,60%**	**29,30%**	23,68%	**39,67%**	**25,64%**
TF-IDF	RF	35,69%	41,61%	24,95%	23,88%	24,95%	22,73%	35,69%	22,43%
TF-IDF	MLP	32,10%	38,97%	27,08%	25,14%	27,08%	23,07%	32,10%	23,48%
BM25	**MNB**	35,44%	40,85%	27,94%	28,35%	27,94%	23,47%	35,44%	23,53%
BM25	SVM	39,14%	40,88%	24,56%	23,90%	24,56%	21,49%	39,14%	22,74%
BM25	**RF**	35,75%	41,84%	24,93%	23,80%	24,93%	22,55%	35,75%	22,25%
BM25	MLP	32,14%	38,90%	28,03%	27,70%	28,03%	23,11%	32,14%	24,94%
LSA - 100	SVM	34,68%	39,00%	26,65%	26,38%	26,65%	20,62%	34,68%	22,11%
LSA - 100	RF	32,83%	38,45%	25,96%	25,72%	25,96%	22,54%	32,83%	22,72%
LSA - 100	MLP	29,38%	44,05%	17,95%	19,00%	17,95%	19,91%	29,38%	16,88%
LSA - 250	SVM	36,19%	39,44%	28,06%	27,99%	28,06%	22,23%	36,19%	23,95%
LSA - 250	RF	33,27%	38,36%	26,50%	26,22%	26,50%	23,15%	33,27%	23,28%
LSA - 250	**MLP**	27,55%	**46,03%**	17,48%	17,74%	17,48%	19,83%	27,55%	15,54%
Glove-300	SVM	19,03%	28,06%	16,38%	16,04%	16,38%	11,41%	19,03%	10,67%
Glove-300	RF	21,70%	29,54%	12,96%	11,66%	12,96%	13,89%	21,70%	11,74%
Glove-300	MLP	8,51%	15,74%	79,87%	25,27%	4,81%	6,41%	74,48%	10,72%
Melhores modelos avaliados quanto ao acerto de qualquer assunto									
BM25	MNB	35,44%	40,85%	27,94%	28,35%	27,94%	23,47%	35,44%	23,53%
TF-IDF	SVM	**39,67%**	42,59%	**29,30%**	**29,60%**	**29,30%**	**23,68%**	**39,67%**	**25,64%**
BM25	RF	35,75%	**41,84%**	24,93%	23,80%	24,93%	22,55%	35,75%	
LSA - 250	MLP	27,55%	46,03%	17,48%	17,74%	17,48%	19,83%	27,55%	15,54%

Apêndice B

Matrizes de confusão

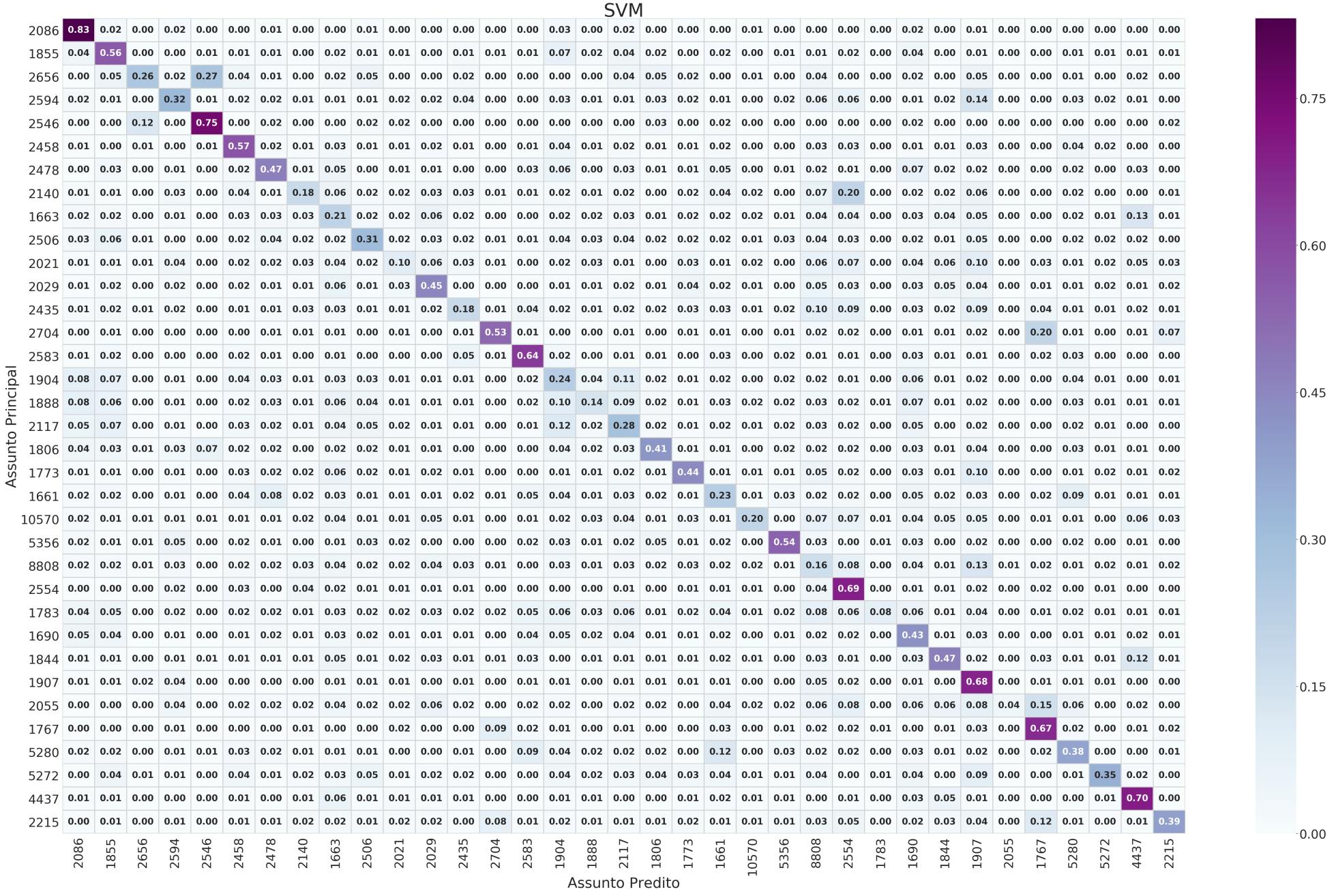

Figura B.1: Matriz de confusão do SVM com TF-IDF.

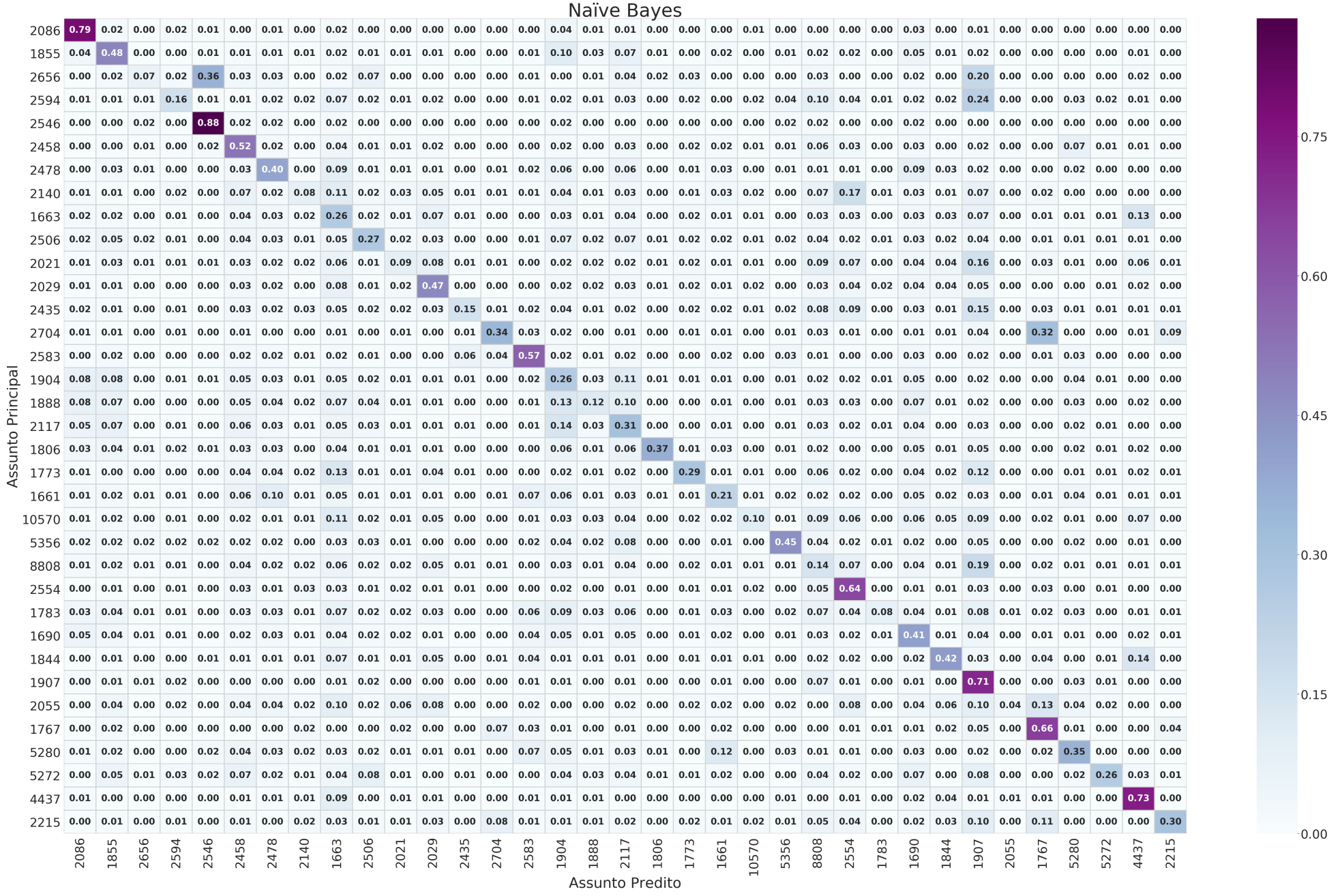

Figura B.2: Matriz de confusão do MNB com BM25.

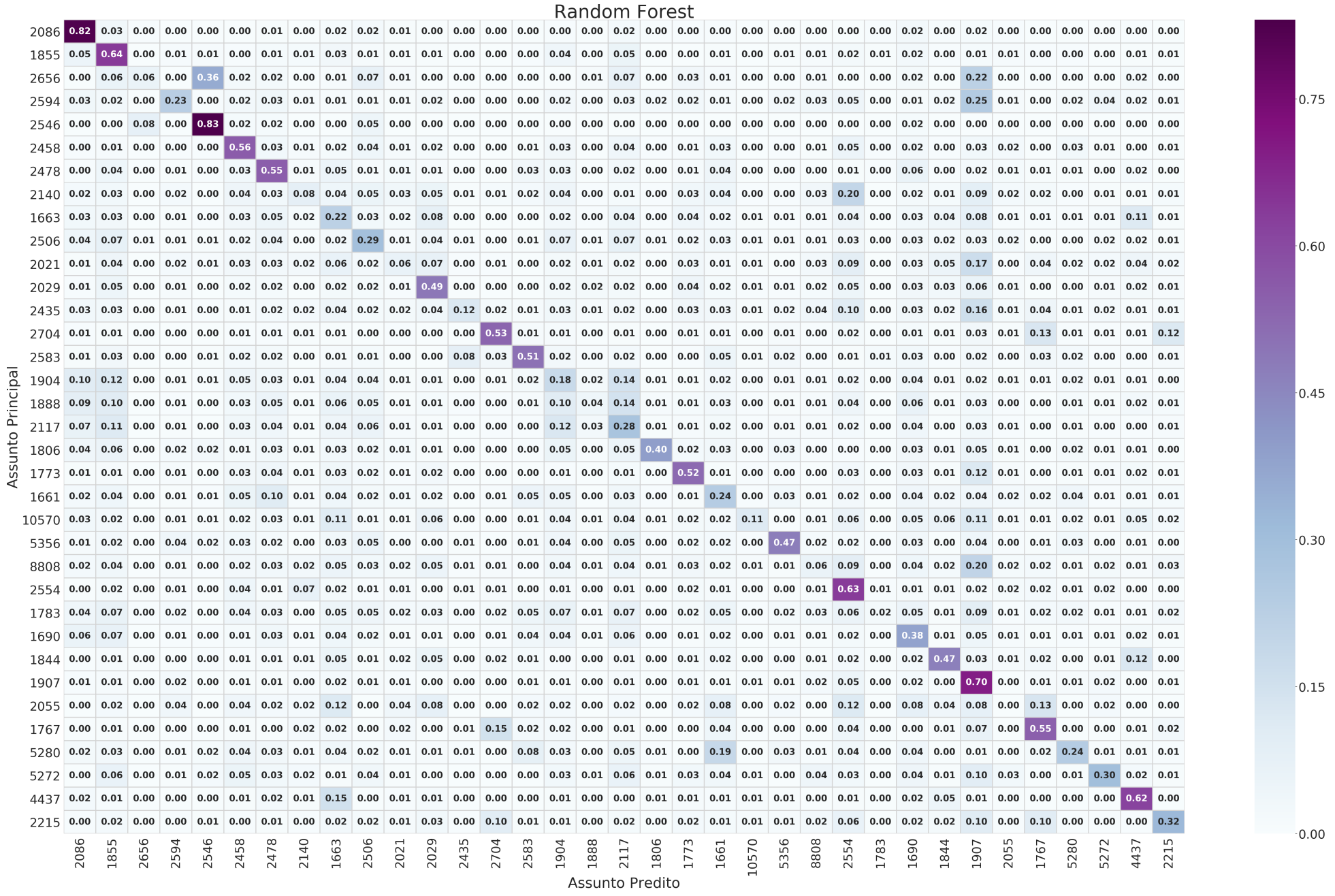

Figura B.3: Matriz de confusão do RF com BM25.

Apêndice C

Nuvens de palavras

Figura C.1: Nuvem de palavras do assunto 2546 - Verbas Rescisórias.

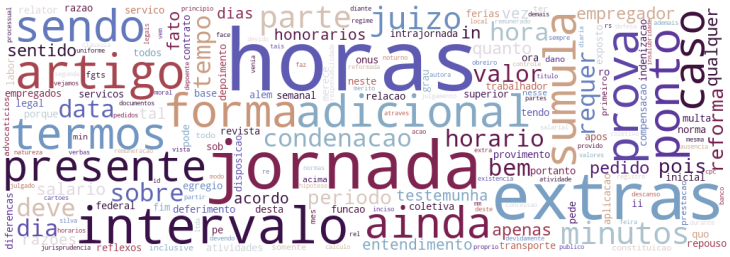

Figura C.2: Nuvem de palavras do assunto 2086 - Horas Extras.

Figura C.3: Nuvem de palavras do assunto 1855 - Indenização por Dano Moral.

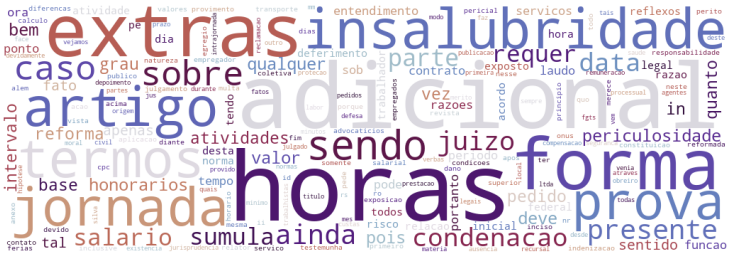

Figura C.4: Nuvem de palavras do assunto 2594 - Adicional.

Figura C.5: Nuvem de palavras do assunto 2458 - Salário / Diferença Salarial.

Figura C.6: Nuvem de palavras do assunto 2704 - Tomador de Serviços / Terceirização.

Figura C.7: Nuvem de palavras do assunto 2656 - Reintegração / Readmissão ou Indenização.

Figura C.8: Nuvem de palavras do assunto 2140 - Intervalo Intrajornada.

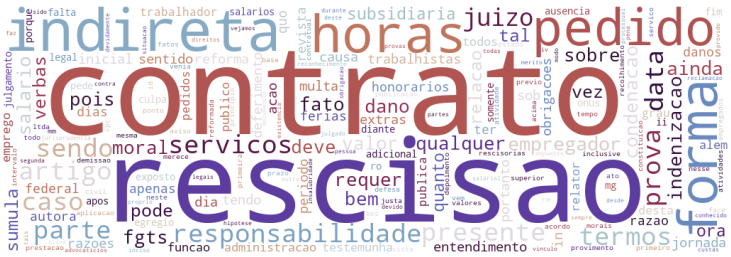

Figura C.9: Nuvem de palavras do assunto 2435 - Rescisão Indireta.

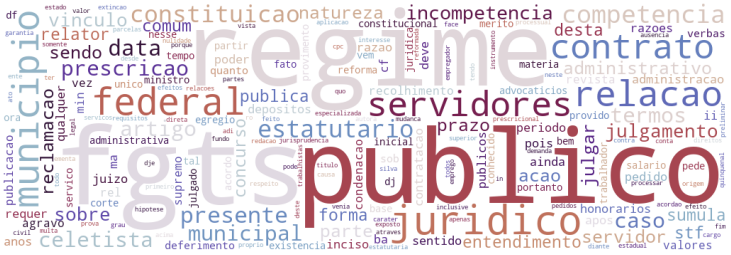

Figura C.10: Nuvem de palavras do assunto 2029 - FGTS.

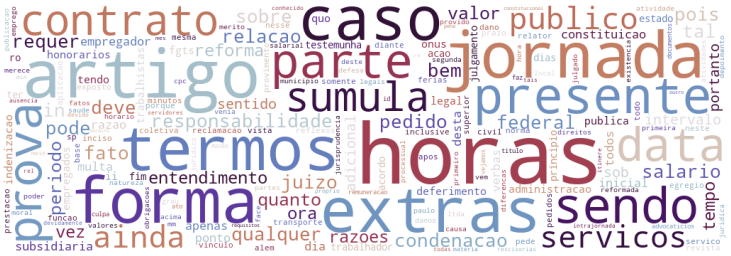

Figura C.11: Nuvem de palavras do assunto 2583 - Abono.

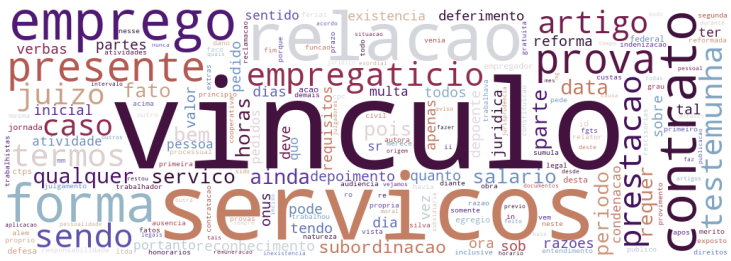

Figura C.12: Nuvem de palavras do assunto 2554 - Reconhecimento de Relação de Emprego.

Figura C.13: Nuvem de palavras do assunto 8808 - Indenização por Dano Material.

Figura C.14: Nuvem de palavras do assunto 2117 - Supressão / Redução de Horas Extras Habituais - Indenização.

Figura C.15: Nuvem de palavras do assunto 2021 - Indenização / Dobra / Terço Constitucional.

Figura C.16: Nuvem de palavras do assunto 5280 - Bancários.

Figura C.17: Nuvem de palavras do assunto 1904 - Despedida / Dispensa Imotivada.

Figura C.18: Nuvem de palavras do assunto 1844 - CTPS.

Figura C.19: Nuvem de palavras do assunto 2055 - Gratificação.

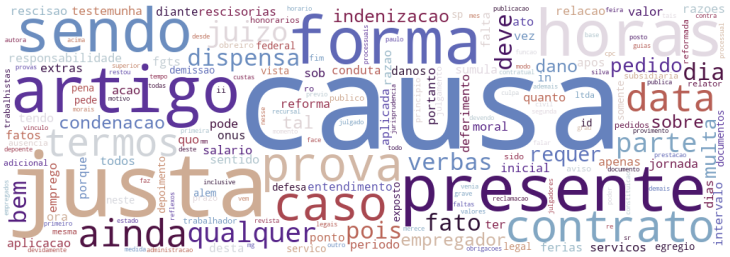

Figura C.20: Nuvem de palavras do assunto 1907 - Justa Causa / Falta Grave.

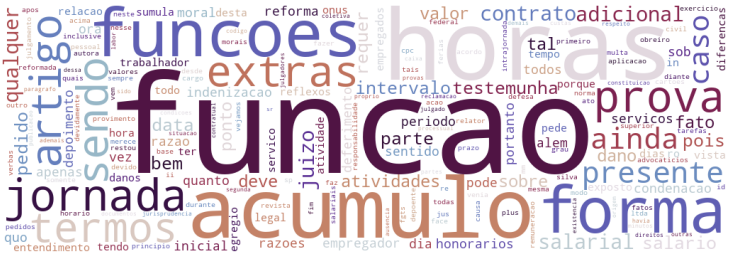

Figura C.21: Nuvem de palavras do assunto 1806 - Alteração Contratual ou das Condições de Trabalho.

Figura C.22: Nuvem de palavras do assunto 55220 - Indenização por Dano Moral.

Figura C.23: Nuvem de palavras do assunto 2506 - Ajuda / Tíquete Alimentação.

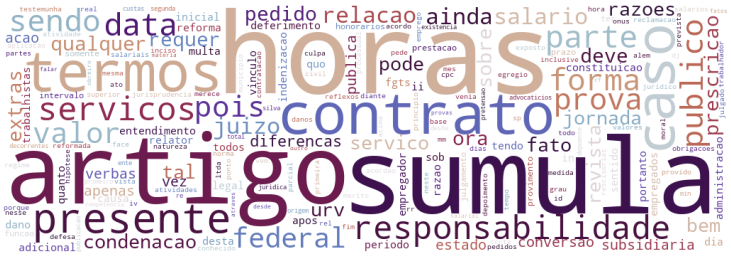

Figura C.24: Nuvem de palavras do assunto 4437 - Revisão de Sentença Normativa.

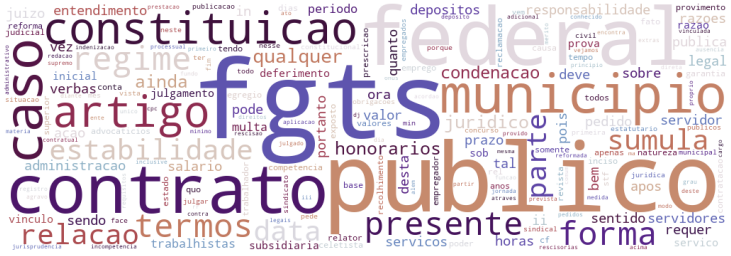

Figura C.25: Nuvem de palavras do assunto 10570 - FGTS.

Figura C.26: Nuvem de palavras do assunto 1783 - Comissão.

Figura C.27: Nuvem de palavras do assunto 1888 - Descontos Salariais - Devolução.

Figura C.28: Nuvem de palavras do assunto 2478 - Seguro Desemprego.

Figura C.29: Nuvem de palavras do assunto 5356 - Grupo Econômico.

Figura C.30: Nuvem de palavras do assunto 1773 - Contribuição Sindical.

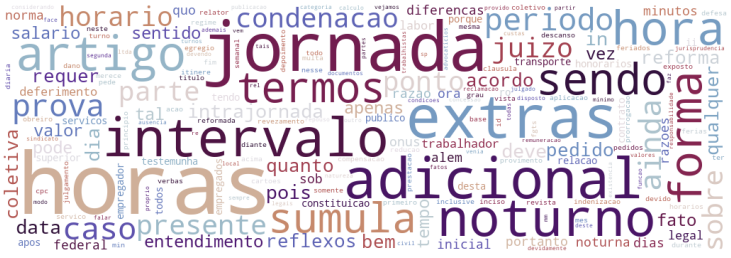

Figura C.31: Nuvem de palavras do assunto 1663 - Adicional Noturno.

Figura C.32: Nuvem de palavras do assunto 5272 - Administração Pública.

Figura C.33: Nuvem de palavras do assunto 2215 - Multa Prevista em Norma Coletiva.

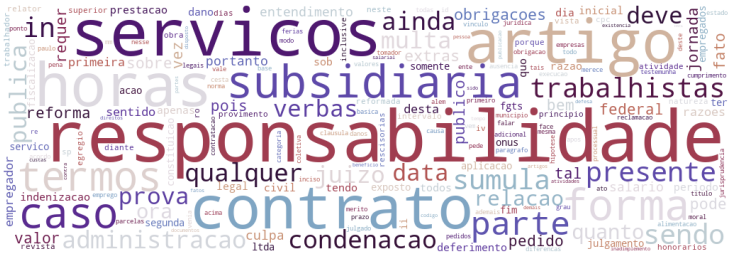

Figura C.34: Nuvem de palavras do assunto 1767 - Cesta Básica.

Figura C.35: Nuvem de palavras do assunto 1661 - Horas in Itinere.

Apêndice D

Código

Listing D.1: Codigo principal

```
#!/usr/bin/env python
# coding: utf-8

## Classificador de Assuntos
#
# Por Ana Carolina Pereira Rocha

from docutils.nodes import header
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC, LinearSVC
from datetime import timedelta
import time
import sys
from datetime import datetime
import seaborn as sns
import matplotlib.pyplot as plt
import uuid
import os
from sklearn.calibration import CalibratedClassifierCV
import argparse
import multiprocessing as mp
import numpy as np
import pandas as pd

# Verificando o ambiente de execucao do conda

import os
print(os.environ['CONDA_DEFAULT_ENV'])
```

```
import funcoes as func
from modelo import *

n_cores = mp.cpu_count()
n_cores_grande = round(n_cores * 0.8)
n_cores_pequeno = round(n_cores * 0.35)

# #### ATENCAO:
#
# A celula abaixo deve ser editada para conter o caminho correto para a pasta onde
# os dados serao buscados, e a pasta onde serao gravadas as saidas do processamento
# deste codigo. O caminho de cada pasta deve ser terminado com a '/' no final.

path_fonte_de_dados = '/home//DocumentosClassificadorAssuntos/'
path_resultados = '/home/DocumentosClassificadorAssuntos/DocsProcessados/'

if not os.path.exists(path_resultados):
    os.makedirs(path_resultados)

float_formatter = lambda x: "%.4f" % x
np.set_printoptions(formatter={'float_kind':float_formatter})

columnsResultados=['id_execucao', 'data', 'nome','feature_type','tempo_processamento',
                   'tamanho_conjunto_treinamento', 'accuracy','balanced_accuracy',
                   'micro_precision','micro_recall','micro_fscore','macro_precision',
                   'macro_recall','macro_fscore','best_params_','best_estimator_',
                   'grid_scores_','grid_cv_results','confusion_matrix',
                   'classification_report','num_estimators','max_samples']
df_resultados = pd.DataFrame(columns = columnsResultados)
nome_arquivo_destino = path_resultados + "Metricas.csv"
if  not (os.path.isfile(nome_arquivo_destino)):
    with open(nome_arquivo_destino, 'a') as f:
        df_resultados.to_csv(f, header=True)
nome_classification_reports = path_resultados + 'ClassificationReport'

id_execucao = str(uuid.uuid1())[:7]
data = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

modelos = []

listaAssuntos=[2546,2086,1855,2594,2458,2704,2656,2140,2435,2029,2583,2554,8808,
               2117,2021,5280,1904,1844,2055,1907, 1806,55220,2506, 4437,10570,
               1783,1888,2478,5356,1773,1663,5272,2215,1767,1661,1690]
```

```
# Definindo modelos que serao usados

classificadorNB = MultinomialNB()
classificadorRF = RandomForestClassifier(random_state=42)
classificadorSVM = CalibratedClassifierCV(LinearSVC(class_weight='balanced',
                                                    max_iter=10000,random_state=42),
                                                  method='sigmoid', cv=5)
classificadorMLP = MLPClassifier(early_stopping= True,random_state=42)

nomeAlgoritmoNB='Multinomial Naive Bayes'
nomeAlgoritmoRF='Random Forest'
nomeAlgoritmoSVM='SVM'
nomeAlgoritmoMLP="Multi-Layer Perceptron"

# Pre-processamento dos documentos

path_destino_de_dados = path_fonte_de_dados + 'DocumentosProcessados/'
if not os.path.exists(path_destino_de_dados):
        os.makedirs(path_destino_de_dados)

#func.processaDocumentos(path_fonte_de_dados,path_destino_de_dados)
print("Todos os documentos disponiveis foram processados")


# Recuperando textos

qtdElementosPorAssunto=1000000
df_amostra = func.recupera_amostras_de_todos_regionais(listaAssuntos,
                                qtdElementosPorAssunto, path_destino_de_dados)


# Juntando os assuntos 55220 e 1855, ambos Indenizacao por Dano Moral

df_amostra.loc[df_amostra['cd_assunto_nivel_3'] == 55220, 'cd_assunto_nivel_3'] = 1855
df_amostra.loc[df_amostra['cd_assunto_nivel_2'] == 55218, 'cd_assunto_nivel_3'] = 2567

print('Total de textos recuperados: ' + str(len(df_amostra)))
df_amostra = df_amostra.dropna(subset=['texto_stemizado'])
print('Total de textos recuperados com conte do: ' + str(len(df_amostra)))


# Analisando tamanho dos textos

df_amostra['quantidade_de_palavras'] = \
    [len(x.split()) for x in df_amostra['texto_processado'].tolist()]
```

```
sns.boxplot(df_amostra['quantidade_de_palavras'])
plt.savefig("{0}{1}.png".format(path_resultados, "Distribuicao_Tamanho_Textos_Original")

df_amostra_f = df_amostra[((df_amostra.quantidade_de_palavras < 400) &
                           (df_amostra.quantidade_de_palavras > 0))]
print('Quantidade de textos entre 0 e 400 palavras: ' + str(len(df_amostra_f)))
df_amostra_f = df_amostra[(df_amostra.quantidade_de_palavras > 10000)]
print('Quantidade de textos com mais de 10.000 palavras: ' + str(len(df_amostra_f)))
df_amostra.shape
df_amostra_f = df_amostra[((df_amostra.quantidade_de_palavras < 10000) &
                           (df_amostra.quantidade_de_palavras > 400))]
df_amostra_f= df_amostra_f.sort_values(by='quantidade_de_palavras', ascending=True)
df_amostra_f.shape
df_amostra = df_amostra_f
plt.clf()
plt.cla()
plt.close()
sns.boxplot(df_amostra['quantidade_de_palavras'])
plt.savefig("{0}{1}.png".format(path_resultados, "Distribuicao_Tamanho_Textos Final"))

print('Total de textos utilizados: ' + str(len(df_amostra)))
X_train, X_test, y_train, y_test = func.splitTrainTest(df_amostra)
print("Amostra de teste de " + str(X_test.shape[0]) + " elementos")
print("Amostra de treinamento de " + str(X_train.shape[0]) + " elementos")

title = "Balanceamento de assuntos na amostra de " + str(X_train.shape[0])
func.mostra_balanceamento_assunto(y_train.value_counts(), title,
                                  "Quantidade Elementos", "Codigo Assunto",
                                  path_resultados, y_train.shape[0])


## Criando matrizes

#### TF-IDF

start_time = time.time()
tfidf_transformer,x_tfidf_train, x_tfidf_test = \
            func.extraiFeaturesTFIDF_train_test(df_amostra,
            X_train['texto_stemizado'], X_test['texto_stemizado'], path_resultados)
total_time = time.time() - start_time
print("Tempo para montar matrizes TF-IDF (features: "
      + str(x_tfidf_train.shape[1]) + ") :"
      + str(timedelta(seconds=total_time)))

#### BM25
```

```
bm25_transformer,x_bm25_train, x_bm25_test = func.extraiFeaturesBM25(df_amostra,
                        tfidf_transformer, x_tfidf_train, x_tfidf_test, path_resultados)

# #### LSI

lsi100_transformer,x_lsi100_train, x_lsi100_test = func.extraiFeaturesLSI(df_amostra,
                                X_train['texto_stemizado'], X_test['texto_stemizado'],
                                100, path_resultados)
lsi250_transformer,x_lsi250_train, x_lsi250_test = func.extraiFeaturesLSI(df_amostra,
                                X_train['texto_stemizado'], X_test['texto_stemizado'],
                                 250, path_resultados)

# ## Grid Search
# #### Com TF-IDF
#
# Coloque aqui a quantidade de configuracoes diferentes a serem testadas
# no GridSearch para cada modelo.

numero_de_configuracoes_por_modelo=2

# #### Multinomial Naïve-Bayes (NB)

param_grid_NB = {
    'estimator__n_estimators': [3,5],
    'estimator__max_samples': [0.8,0.5],
    'estimator__base_estimator__alpha': [0.0001, 0.001, 0.01, 0.1, 0.5, 1]
}
modeloNB = func.chama_treinamento_modelo(x_tfidf_train, y_train, x_tfidf_test,
                  y_test, classificadorNB,
                  nomeAlgoritmoNB , 'TFIDF',param_grid_NB,
                  numero_de_configuracoes_por_modelo,
                  n_cores_grande,id_execucao ,data,path_resultados,df_resultados,
                  nome_arquivo_destino,X_test)
modelos.append([modeloNB.getNome(),modeloNB.getFeatureType(),
                 modeloNB.getMicroPrecision(),modeloNB])

# #### SVM

param_grid_SVM = {
    'estimator__n_estimators': [3, 5],
    'estimator__max_samples': [0.8, 0.5],
    'estimator__base_estimator__base_estimator__C': [0.01, 0.1, 1, 10]
}
modeloSVM = func.chama_treinamento_modelo(x_tfidf_train,y_train, x_tfidf_test,y_test,
```

```
                    classificadorSVM, nomeAlgoritmoSVM, 'TFIDF',
                    param_grid_SVM, numero_de_configuracoes_por_modelo,
                    n_cores_grande, id_execucao ,data, path_resultados, df_resultados,
                    nome_arquivo_destino, X_test)
modelos.append([modeloSVM.getNome(), modeloSVM.getFeatureType(),
                modeloSVM.getMicroPrecision(),modeloSVM])

#### Random Forest (RF)

param_grid_RF = {
    'estimator__n_estimators': [3,5],
    'estimator__max_samples': [0.8,0.5],
    'estimator__base_estimator__max_depth': [30,50,100],
    'estimator__base_estimator__n_estimators': [100,200,300],
    'estimator__base_estimator__min_samples_leaf': [0.05, 0.1, 0.5],
    'estimator__base_estimator__min_samples_split': [0.05, 0.1, 0.5],
    'estimator__base_estimator__max_features': [0.3, 0.5, 0.8]
}
modeloRF = func.chama_treinamento_modelo(x_tfidf_train,y_train,
                    x_tfidf_test,y_test, classificadorRF,
                    nomeAlgoritmoRF, 'TFIDF',param_grid_RF,
                    numero_de_configuracoes_por_modelo,
                    n_cores_grande, id_execucao ,data, path_resultados, df_resultados,
                    nome_arquivo_destino, X_test)
modelos.append([modeloRF.getNome(),modeloRF.getFeatureType(),
                modeloRF.getMicroPrecision(),modeloRF])

#### Multi-layer Perceptron

param_grid_MLP = {
    'estimator__n_estimators': [3,5],
    'estimator__max_samples': [0.8,0.5],
    'estimator__base_estimator__hidden_layer_sizes': [(10,10),(10,5,10)],
    'estimator__base_estimator__activation': ['identity', 'logistic', 'tanh', 'relu'],
    'estimator__base_estimator__solver': ['sgd', 'adam','lbfgs'],
    'estimator__base_estimator__alpha': [0.001, 0.01, 0.05, 0.1],
    'estimator__base_estimator__learning_rate': ['constant','adaptive','invscaling'],
    'estimator__base_estimator__max_iter': [200,300,400]
}
modeloMLP = func.chama_treinamento_modelo(x_tfidf_train,y_train, x_tfidf_test,y_test,
                    classificadorMLP, nomeAlgoritmoMLP, 'TFIDF',
                    param_grid_MLP, numero_de_configuracoes_por_modelo,
                    n_cores_pequeno, id_execucao ,data, path_resultados,
                    df_resultados, nome_arquivo_destino, X_test)
modelos.append([modeloMLP.getNome(),modeloMLP.getFeatureType(),
```

```
                modeloMLP.getMicroPrecision(),modeloMLP])

#### Criando dicionarios com a melhor configuracao de cada modelo

#MNB
param_grid_melhor_NB = {
    'estimator__n_estimators':
        [modeloNB.getBestParams().get('estimator__n_estimators')],
    'estimator__max_samples':
        [modeloNB.getBestParams().get('estimator__max_samples')],
    'estimator__base_estimator__alpha':
        [modeloNB.getBestParams().get('estimator__base_estimator__alpha')]
}

# SVM
param_grid_melhor_SVM = {
    'estimator__n_estimators':
        [modeloSVM.getBestParams().get('estimator__n_estimators')],
    'estimator__max_samples':
        [modeloSVM.getBestParams().get('estimator__max_samples')],
    'estimator__base_estimator__base_estimator__C':
        [modeloSVM.getBestParams().
            get('estimator__base_estimator__base_estimator__C')]
}

# RF
param_grid_melhor_RF = {
    'estimator__n_estimators':
        [modeloRF.getBestParams().get('estimator__n_estimators')],
    'estimator__max_samples':
        [modeloRF.getBestParams().get('estimator__max_samples')],
    'estimator__base_estimator__max_depth':
        [modeloRF.getBestParams().get('estimator__base_estimator__max_depth')],
    'estimator__base_estimator__n_estimators':
        [modeloRF.getBestParams().get('estimator__base_estimator__n_estimators')],
    'estimator__base_estimator__min_samples_leaf':
        [modeloRF.getBestParams().get('estimator__base_estimator__min_samples_leaf')],
    'estimator__base_estimator__min_samples_split':
        [modeloRF.getBestParams().get('estimator__base_estimator__min_samples_split')],
    'estimator__base_estimator__max_features':
        [modeloRF.getBestParams().get('estimator__base_estimator__max_features')]
}

# MLP
param_grid_melhor_MLP = {
```

```
    'estimator__n_estimators':
        [modeloMLP.getBestParams().get('estimator__n_estimators')],
    'estimator__max_samples':
        [modeloMLP.getBestParams().get('estimator__max_samples')],
    'estimator__base_estimator__hidden_layer_sizes':
        [modeloMLP.getBestParams().get('estimator__base_estimator__hidden_layer_sizes')]
    'estimator__base_estimator__activation':
        [modeloMLP.getBestParams().get('estimator__base_estimator__activation')],
    'estimator__base_estimator__solver':
        [modeloMLP.getBestParams().get('estimator__base_estimator__solver')],
    'estimator__base_estimator__alpha':
        [modeloMLP.getBestParams().get('estimator__base_estimator__alpha')],
    'estimator__base_estimator__learning_rate':
        [modeloMLP.getBestParams().get('estimator__base_estimator__learning_rate')],
    'estimator__base_estimator__max_iter':
        [modeloMLP.getBestParams().get('estimator__base_estimator__max_iter')]
}

# #### BM25

modeloNB_BM25 = func.chama_treinamento_modelo(x_bm25_train,y_train, x_bm25_test,
                    y_test, classificadorNB,
                    nomeAlgoritmoNB,'BM25',param_grid_melhor_NB, 1,n_cores_grande,
                    id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                    X_test)
modelos.append([modeloNB_BM25.getNome(),modeloNB_BM25.getFeatureType(),
                  modeloNB_BM25.getMicroPrecision(),modeloNB_BM25])

modeloSVM_BM25 = func.chama_treinamento_modelo(x_bm25_train,y_train, x_bm25_test,
                     y_test, classificadorSVM,
                     nomeAlgoritmoSVM,'BM25',param_grid_melhor_SVM, 1,n_cores_grande,
                     id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                     X_test)
modelos.append([modeloSVM_BM25.getNome(),modeloSVM_BM25.getFeatureType(),
                  modeloSVM_BM25.getMicroPrecision(),modeloSVM_BM25])

modeloRF_BM25 = func.chama_treinamento_modelo(x_bm25_train,y_train, x_bm25_test,
                    y_test, classificadorRF,nomeAlgoritmoRF,
                    'BM25',param_grid_melhor_RF, 1,n_cores_grande,id_execucao,
                    data,path_resultados,df_resultados,nome_arquivo_destino,X_test)
modelos.append([modeloRF_BM25.getNome(),modeloRF_BM25.getFeatureType(),
                  modeloRF_BM25.getMicroPrecision(),modeloRF_BM25])

modeloMLP_BM25 = func.chama_treinamento_modelo(x_bm25_train,y_train, x_bm25_test,
                     y_test, classificadorMLP,
```

```
                    nomeAlgoritmoMLP, 'BM25',param_grid_melhor_MLP, 1,n_cores_pequeno,
                    id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                    X_test)
modelos.append([modeloMLP_BM25.getNome(),modeloMLP_BM25.getFeatureType(),
                modeloMLP_BM25.getMicroPrecision(), modeloMLP_BM25])

# #### LSI 100

modeloSVM_LSI100 = func.chama_treinamento_modelo(x_lsi100_train,y_train, x_lsi100_test ,
                y_test, classificadorSVM,
                nomeAlgoritmoSVM, 'LSI100',param_grid_melhor_SVM, 1,n_cores_grande,
                id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                X_test)
modelos.append([modeloSVM_LSI100.getNome(),modeloSVM_LSI100.getFeatureType(),
                modeloSVM_LSI100.getMicroPrecision(),
                modeloSVM_LSI100])

modeloRF_LSI100 = func.chama_treinamento_modelo(x_lsi100_train, y_train,x_lsi100_test ,
                y_test, classificadorRF,
                nomeAlgoritmoRF, 'LSI100',param_grid_melhor_RF, 1,n_cores_grande,
                id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                X_test)
modelos.append([modeloRF_LSI100.getNome(),modeloRF_LSI100.getFeatureType(),
                modeloRF_LSI100.getMicroPrecision(), modeloRF_LSI100])

modeloMLP_LSI100 = func.chama_treinamento_modelo(x_lsi100_train,y_train, x_lsi100_test ,
                y_test, classificadorMLP,
                nomeAlgoritmoMLP,'LSI100',param_grid_melhor_MLP, 1,n_cores_pequeno,
                id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                X_test)
modelos.append([modeloMLP_LSI100.getNome(),modeloMLP_LSI100.getFeatureType(),
                modeloMLP_LSI100.getMicroPrecision(), modeloMLP_LSI100])

# #### LSI 250

modeloSVM_LSI250 = func.chama_treinamento_modelo(x_lsi250_train,y_train, x_lsi250_test ,
                y_test, classificadorSVM,
                nomeAlgoritmoSVM, 'LSI250',param_grid_melhor_SVM, 1,n_cores_grande,
                id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                X_test)
modelos.append([modeloSVM_LSI250.getNome(),modeloSVM_LSI250.getFeatureType(),
                modeloSVM_LSI250.getMicroPrecision(), modeloSVM_LSI250])

modeloRF_LSI250 = func.chama_treinamento_modelo(x_lsi250_train, y_train,x_lsi250_test ,
                y_test, classificadorRF,
```

```
                    nomeAlgoritmoRF, 'LSI250',param_grid_melhor_RF, 1,n_cores_grande,
                    id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                    X_test)
modelos.append([modeloRF_LSI250.getNome(),modeloRF_LSI250.getFeatureType(),
                    modeloRF_LSI250.getMicroPrecision(), modeloRF_LSI250])

modeloMLP_LSI250 = func.chama_treinamento_modelo(x_lsi250_train,y_train, x_lsi250_test ,
                    y_test, classificadorMLP,
                    nomeAlgoritmoMLP,'LSI250',param_grid_melhor_MLP, 1,n_cores_pequeno,
                    id_execucao ,data,path_resultados,df_resultados,nome_arquivo_destino,
                    X_test)
modelos.append([modeloMLP_LSI250.getNome(),modeloMLP_LSI250.getFeatureType(),
                    modeloMLP_LSI250.getMicroPrecision(), modeloMLP_LSI250])

# Encontrando o modelo vencedor

modelos_df = pd.DataFrame(modelos,
                      columns=['Nome Modelo','Feature Type','micro_precision','Modelo'])
modelos_df= modelos_df.sort_values(by='micro_precision', ascending=False)
print("O modelo vencedor foi o " + modelos_df.iloc[0]['Nome Modelo'] + ", com " +
      (str("%.2f" % modelos_df.iloc[0]['micro_precision'])) + " de micro precisao")

modelo_vencedor = modelos_df.iloc[0]['Modelo']

arquivoPickle = open(path_resultados + "MelhorModelo.p", 'wb')
pickle.dump(modelo_vencedor.getBestEstimator(), arquivoPickle)
arquivoPickle.close()

if modelo_vencedor.getFeatureType() == 'LSI100':
    feature_vencedora = open(path_resultados + "MelhorModeloFeature.p", 'wb')
    pickle.dump(lsi100_transformer, feature_vencedora)
    feature_vencedora.close()
elif modelo_vencedor.getFeatureType() == 'LSI250':
    feature_vencedora = open(path_resultados + "MelhorModeloFeature.p", 'wb')
    pickle.dump(lsi250_transformer, feature_vencedora)
    feature_vencedora.close()
elif modelo_vencedor.getFeatureType() == 'TFIDF':
    feature_vencedora = open(path_resultados + "MelhorModeloFeature.p", 'wb')
    pickle.dump(tfidf_transformer, feature_vencedora)
    feature_vencedora.close()
elif modelo_vencedor.getFeatureType() == 'BM25':
    feature_vencedora = open(path_resultados + "MelhorModeloFeature.p", 'wb')
    pickle.dump(bm25_transformer, feature_vencedora)
    feature_vencedora.close()
```

```
print("O modelo para transformacao dos textos pre-processados se encontra no arquivo "
      + path_resultados +
      "MelhorModeloFeature.p" + " e o modelo de classificacao no arquivo "
      + path_resultados + "MelhorModelo.p")
```

Listing D.2: Funcoes Auxiliares

```
##############################################
# Script com funcoes auxiliares utilizadas no projeto Classificador de Assuntos
# Por Ana Carolina Pereira Rocha
# Data: 10/12/2019
##############################################
from modelo import *

##############################################
##############################################
# FUNCOES DE PRE-PROCESSAMENTO DE TEXTOS
##############################################
##############################################
import ssl
import nltk

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context
# nltk.download('stopwords')
# nltk.download('rslp')


# -----------------------------------------
# Funcao que remove marcacoes HTML de um texto
# -----------------------------------------
from bs4 import BeautifulSoup


def removeHTML(texto):
    """
    Funcao para remover HTML
    :param texto:
    :return: texto sem tags HTML
    """
```

```
    texto = texto.replace('\n', ' ')
    texto = texto.replace('\t', ' ')
    return BeautifulSoup(texto, 'lxml').get_text(" ", strip=True)


# ------------------------------------------------
# Funcao que remove acentos, numeros, palavras menores que 3 caracteres,
# caracteres especiais e tranforma em minusculo
# ------------------------------------------------
import re

stemmer = nltk.stem.RSLPStemmer()


def processa_stemiza_texto(texto):
    """
    Funcao para remover caracteres especiais, acentos, pontuacoes, n meros,
    stopwords e  palavras menores que 3 caracteres.
    :param texto:
    :return: texto processado
    """
    global stopwords_processadas
    textoProcessado = normalize('NFKD', texto).encode('ASCII', 'ignore').decode(
        'ASCII')
    textoProcessado = re.sub('[^a-zA-Z]', ' ', textoProcessado)
    textoProcessado = textoProcessado.lower()
    textoProcessado = textoProcessado.split()
    textoProcessado = [palavra for palavra in
                       textoProcessnome_arquivo_destinoado if
                       not palavra in stopwords_processadas]
    textoProcessado = [palavra for palavra in textoProcessado if
                       len(palavra) > 3]
    textoProcessado = [stemmer.stem(palavra) for palavra in textoProcessado]
    return ' '.join(word for word in textoProcessado)


# ------------------------------------------------
# Funcao que faz o processamento dos textos para cada regional
# ------------------------------------------------
from unicodedata import normalize
import pandas as pd
import os
import csv
import time
import multiprocessing as mp
```

```
from datetime import timedelta

# [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]
stopwords_processadas = []


def processaDocumentos(path_fonte_de_dados, path_destino_de_dados,
                       regionais=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
                                  15, 16, 17, 18, 19, 20, 21, 22, 23,
                                  24]):
    """
    Funcao para processar os documentos de todos os regionais. Ao final do
    processamento, serao armazenados arquivos csv com o conte do dos documentos
    processados e seus metadados.
    :param path_fonte_de_dados: Diretorio onde serao buscados os arquivos csv
    com o conte do dos documentos a serem processados. E esperado que os documentos
    contenham o nome no seguinte padrao: 'TRT_XX_documentosSelecionados.csv',
         onde XX se refere    sigla do Tribunal Regional, sempre com dois
         digitos (01,02,10,20...)
    :param path_destino_de_dados: Diretorio onde serao gravados os documentos
    que foram processados. Os documentos serao armazenados com o seguinte nome:
    'TRT_XX_documentosSelecionadosProcessados.csv'
    :param regionais: lista dos Tribunais Regionais a serem buscados (apenas um
     digito). Por padrao, ira buscar todos
        os regionais, da 1      24   regiao
    """
    global stopwords_processadas
    print("Passando stopwords pelo pre processamento....")
    stopwords = nltk.corpus.stopwords.words('portuguese')
    for row in stopwords:
        palavraProcessada = normalize('NFKD', row).encode('ASCII',
                                                          'ignore').decode(
            'ASCII')
        stopwords_processadas.append(palavraProcessada)

    for regional in regionais:
        sigla_trt = "{:02d}".format(regional)
        print("-----------------------------------------------------------")
        print('Processando texto dos documentos do TRT ' + sigla_trt)
        nome_arquivo_origem = path_fonte_de_dados + 'TRT_' + sigla_trt\
                              + '_documentosSelecionados.csv'
        nome_arquivo_destino = path_destino_de_dados + 'TRT_' \
                               + sigla_trt + '_documentosSelecionadosProcessados.csv'

        if not os.path.exists(nome_arquivo_origem):
```

```
        print(
            "Nao foi encontrado o arquivo de documentos do TRT "
            + sigla_trt + ". Buscou-se pelo arquivo " + nome_arquivo_origem)
        continue

colnames = ['index', 'nr_processo', 'id_processo_documento',
            'cd_assunto_nivel_5', 'cd_assunto_nivel_4',
            'cd_assunto_nivel_3', 'cd_assunto_nivel_2',
            'cd_assunto_nivel_1', 'tx_conteudo_documento',
            'ds_identificador_unico',
            'ds_identificador_unico_simplificado', 'ds_orgao_julgador',
            'ds_orgao_julgador_colegiado', 'dt_juntada']
df_trt = pd.read_csv(nome_arquivo_origem, sep=',', names=colnames,
                     index_col=0, header=None,
                     quoting=csv.QUOTE_ALL)

# remove as tags HTML
start_time = time.time()
pool = mp.Pool(7)
df_trt = df_trt.dropna(subset=['tx_conteudo_documento'])
df_trt['texto_processado'] = pool.map(removeHTML, [row for row in
                                                   df_trt[
                                                       'tx_conteudo_documento']])
pool.close()
total_time = time.time() - start_time
print('Tempo para processamento do texto:' + str(
    timedelta(seconds=total_time)))

# faz a stemizacao
start_time = time.time()
pool = mp.Pool(7)
df_trt = df_trt.dropna(subset=['texto_processado'])
df_trt['texto_stemizado'] = pool.map(processa_stemiza_texto,
                                     [row for row in
                                      df_trt['texto_processado']])
pool.close()
total_time = time.time() - start_time
print('Tempo para stemizacao do texto:' + str(
    timedelta(seconds=total_time)))

# ------------------------------------------------------------
# VERIFICA O CONTEUDO DE UM DOCUMENTO
# f = open("./teste.html", "w")
# f.write(df_trt.iloc[0]['tx_conteudo_documento'])
# f.close()
```

```
        # import webbrowser
        # webbrowser.get('firefox').open_new_tab('./teste.html')
        # ----------------------------------------------------------------------

        df_trt = df_trt.drop(columns=['tx_conteudo_documento'])
        print("Encontrados " + str(
            df_trt.shape[0]) + " documentos para o TRT " + sigla_trt)

        if os.path.isfile(nome_arquivo_destino):
            os.remove(nome_arquivo_destino)

        df_trt.to_csv(nome_arquivo_destino, sep='#', quoting=csv.QUOTE_ALL)


################################################
################################################
# FUNCOES DE RECUPERACAO DOS DADOS
################################################
################################################


# ----------------------------------------------------
# Funcao que recuperar amostra estratificada pelo codigo de assunto dentre
# todos os codigos existentes no dataset. Nao faz bootstrapping..
# ----------------------------------------------------
def stratified_sample_df(df, col, n_samples):
    """
    Funcao que recupera n_samples de cada documento baseado.
    :param df: data frame que contem os dados
    :param col: nome da coluna para fazer a stratificacao
    :param n_samples: quantidade de elementos a ser recuperado de cada classe.
    O codigo abaixo pode funcionar com ooversampling, recuperando sempre n_samples
    de um assunto, ou sem oversampling, recuperando n_samples ou a
    quantidade disponivel de exemplos, caso nao exista n_samples exemplos.
    :return: dataframe estratificado pelos valores apresentados na coluna col
    """
    # ----------------------------------
    # COM OVER SAMPLING
    # min_accepted = 50
    # df_ = df.groupby(col).apply(lambda x: x.sample(calcularValorMinimo(x.shape[0],
    # n_samples,min_accepted), random_state=42, replace = isResampling(x.shape[0],
    # min_accepted)))

    # ----------------------------------
    # SEM OVER SAMPLING
```

```
    df_ = df.groupby(col).apply(
        lambda x: x.sample(min(x.shape[0], n_samples), random_state=42))

    # -----------------------------
    df_.index = df_.index.droplevel(0)
    return df_


def isResampling(value, min_accepted):
    if (value > min_accepted):
        return False
    else:
        return True


def calcularValorMinimo(value, n_samples, min_accepted):
    minimoEncontrado = min(value, n_samples)
    if (minimoEncontrado < min_accepted):
        return min_accepted
    else:
        return minimoEncontrado


# ------------------------------------------------
# Funcao que recupera os documentos de cada csv
# ------------------------------------------------

def collect_results(result):
    """Uses apply_async's callback to setup up a separate Queue for each process"""
    results.extend(result)


def recupera_n_amostras_por_assunto_por_regional(sigla_trt, assuntos,
                                                 nroElementos, path, sufixo):
    """
    Funcao que, dado um regional, uma lista de assuntos, e definida a a quantidade
    de amostras de cada item, busca o arquivo com os documentos do regional informado
    e retira o n mero de elementos de dado assunto deste regional
    :param regional: sigla do regional onde se deve buscar os dados
    :param assuntos: lista de assuntos a se buscar
    :param quantidadeAmostras: quantidade de elementos de cada assunto. Se nao
    existir a quantidade demandada, ira limitar a quantidade retornada em cada classe
    ao minimo existente
    :return:
    """
```

```
    nome_arquivo = path + 'TRT_' + sigla_trt + '_documentosSelecionadosProcessados' \
                   + sufixo + '.csv'
    if not os.path.exists(nome_arquivo):
        print(
            "Nao foi encontrado o arquivo de documentos do TRT " + sigla_trt
            + ". Buscou-se pelo arquivo " + nome_arquivo)
        return []
    df_trt_csv = pd.read_csv(nome_arquivo, sep='#', quoting=csv.QUOTE_ALL)
    df_trt_csv.loc[:, 'sigla_trt'] = "TRT" + sigla_trt;

    # Removendo dados que nao serao necessarios nessa iteracao
    df_trt_csv.cd_assunto_nivel_3 = pd.to_numeric(df_trt_csv.cd_assunto_nivel_3)
    df_trt_filtrado = df_trt_csv[df_trt_csv.cd_assunto_nivel_3.isin(assuntos)]

    del (df_trt_csv)
    # Estratificando
    df_amostra = stratified_sample_df(df_trt_filtrado, 'cd_assunto_nivel_3',
                                      nroElementos)

    # df_amostra['cd_assunto_nivel_3'].value_counts()
    print(
        "Quantidade de documentos recuperados no TRT " + sigla_trt + ": " + str(
            df_amostra.shape[0]))

    return df_amostra.values.tolist()


def recupera_amostras_de_todos_regionais(listaAssuntos, nroElementos, path,
                                         sufixo='',
                                         regionais=[1, 2, 3, 4, 5, 6, 7, 8, 9,
                                                    10, 11, 12, 13, 14, 15, 16,
                                                    17, 18, 19,
                                                    20, 21, 22, 23, 24]):
    """
    Funcao que busca n (nroElementos) documentos (ou tantos quanto disponiveis)
    em arquivos CSVs para os 24 Tribunais Regionais
    :param listaAssuntos: assuntos a serem buscados
    :param nroElementos: quantidade de elementos a ser recuperada
    :param regionais: lista dos regionais nos quais se vai buscar os documentos.
    Por padrao, ira buscar todos
        os regionais, da 1      24   regiao
    :param path: local onde recuperar os documentos dos regionais
    :return: data frame com o conte do de documentos e os metadadaos
    correpondentes de todos os regionais
    """
```

```
    global results
    results = []
    print("Buscando " + str(
        nroElementos) + " elementos de cada assunto em cada regional")
    start_time = time.time()

    pool = mp.Pool(processes=mp.cpu_count())
    # for i in range (1,25):
    for regional in regionais:
        pool.apply_async(recupera_n_amostras_por_assunto_por_regional,
                         args=(
                         "{:02d}".format(regional), listaAssuntos, nroElementos,
                         path, sufixo),
                         callback=collect_results)
    pool.close()
    pool.join()

    df = pd.DataFrame(results,
                      columns=['index', 'nr_processo', 'id_processo_documento',
                               'cd_assunto_nivel_1',
                               'cd_assunto_nivel_2', 'cd_assunto_nivel_3',
                               'cd_assunto_nivel_4',
                               'cd_assunto_nivel_5', 'ds_identificador_unico',
                               'ds_identificador_unico_simplificado',
                               'ds_orgao_julgador',
                               'ds_orgao_julgador_colegiado', 'dt_juntada',
                               'texto_processado',
                               'texto_stemizado', 'sigla_trt'])
    print(df.shape)
    total_time = time.time() - start_time
    print("Tempo para recuperar amostra de todos os regionais ",
          str(timedelta(seconds=total_time)))
    return df


results = []

# ----------------------------------------------
# Funcao que mostra a distribuicao de elementos por assunto
# ----------------------------------------------
import matplotlib.pyplot as plt


def mostra_balanceamento_assunto(data, title, ylabel, xlabel, path, qnt_elem):
    """
```

```
    Funcao que cria um grafico de barras a partir de um conjunto de dados para
    mostrar a quantidade de elementos por
        assunto
    :param data: dados a serem processados
    :param title: titulo do grafico
    :param ylabel: label do eixo y
    :param xlabel: label do eixo x
    :param path: local onde sera gravado o grafico gerado
    :param qnt_elem: quantidade de ementos em data.
    """
    plt.clf()
    plt.cla()
    plt.close()
    data.plot.bar(ylim=0)
    plt.title(title)
    plt.ylabel(ylabel)
    plt.xlabel(xlabel)
    plt.legend()
    # plt.bar(data)
    plt.savefig("{0}{1}.png".format(path, "Balanceamento_Assuntos_" + str(
        qnt_elem) + "_Elementos"))
    # plt.show()

    # df = pd.DataFrame(y_train.value_counts())
    # df = df.reset_index()
    # df.columns = ['assunto_nivel_3', 'qnt_documentos']
    # plt.bar(df['assunto_nivel_3'], df['qnt_documentos'], align='center', alpha=0.5)


# ----------------------------------------------------
# Funcao que divide conjunto de treinamento e teste de stratificado por assunto
# ----------------------------------------------------
from sklearn.model_selection import train_test_split


def splitTrainTest(df_amostra_final):
    X_train, X_test, y_train, y_test = train_test_split(
        df_amostra_final[['sigla_trt', 'nr_processo', 'id_processo_documento',
                          'texto_stemizado']],
        df_amostra_final['cd_assunto_nivel_3'], test_size=0.2,
        random_state=42,
        stratify=df_amostra_final['cd_assunto_nivel_3'])
    return X_train, X_test, y_train, y_test
```

```
##############################################
##############################################
# FUNCOES AUXILIARES DE MODELOS
##############################################
##############################################

# ------------------------------------------------
# Salva os valores preditos
# ------------------------------------------------
def salvaPredicao(modelo, X_test, y_true, y_pred, y_pred_proba_df,
                  df_resultados, path_resultados):
    nome_arquivo_predicao = path_resultados + 'predicao_' + modelo.getNome() + '.csv'
    df_pred = X_test[['sigla_trt', 'nr_processo', 'id_processo_documento']]
    df_pred['y_true'] = y_true
    df_pred['y_pred'] = y_pred
    df_pred = df_pred.reset_index(drop=True)
    # y_pred_proba_df = y_pred_proba_df.reset_index(drop=True)
    df_pred = df_pred.join(y_pred_proba_df)
    df_pred['modelo'] = modelo.getNome()
    df_pred.to_csv(nome_arquivo_predicao)


# ------------------------------------------------
# Grava metricas de execucao e modelo
# ------------------------------------------------
def salvaModelo(modelo, path_resultados, df_resultados, nome_arquivo_destino):
    modelo.salvaClassificationReport(
        path_resultados + 'ClassificationReport_' + modelo.getNome() + '.csv')
    modelo.salvaModelo(path_resultados)
    df_resultados = df_resultados.append(modelo.__dict__, ignore_index=True)
    with open(nome_arquivo_destino, 'a') as f:
        df_resultados.tail(1).to_csv(f, header=False)


# ------------------------------------------------
# Chama o treinamento do modelo
# ------------------------------------------------
def chama_treinamento_modelo(x_trainamento, y_train, x_teste, y_test, modelo,
                             nomeModelo, feature_type, param_grid,
                             n_iteracoes, n_jobs, id_execucao, data,
                             path_resultados, df_resultados,
                             nome_arquivo_destino, X_test_original):
    modelo = treina_modelo_grid_search(x_trainamento, y_train, modelo,
                                       nomeModelo, feature_type, param_grid,
                                       n_iteracoes, n_jobs)
```

```
    modelo, y_pred, y_pred_proba_df = testa_modelo(x_teste, y_test, modelo)
    modelo.setIdExecucao(id_execucao)
    modelo.setData(data)
    modelo.imprime()
    salvaModelo(modelo, path_resultados, df_resultados, nome_arquivo_destino)
    salvaPredicao(modelo, X_test_original, y_test, y_pred, y_pred_proba_df,
                  df_resultados, path_resultados)
    return modelo


def teste():
    print('it works')


# -----------------------------------------------
# Grava modelo de transformacao de features
# -----------------------------------------------
import pickle


def salvaTransformer(transformer, nome, path):
    nomePicke = path + nome + '.p'
    arquivoPickle = open(nomePicke, 'wb')
    pickle.dump(transformer, arquivoPickle)
    arquivoPickle.close()


def carregaModelo(arquivo):
    with open(arquivo, "rb") as input_file:
        return pickle.load(input_file)


##############################################
##############################################
# FUNCOES DE GERACAO DE FEATURES
##############################################
##############################################

# -----------------------------------------------
# Funcao de geracao de matriz TF-IDF
# -----------------------------------------------
from sklearn.feature_extraction.text import TfidfVectorizer


def extraiFeaturesTFIDF_train_test(df, X_train, X_test, path):
```

```
    tfidf_vectorizer = TfidfVectorizer(token_pattern=r'(?u)\b[A-Za-z]+\b',
                                       max_df=0.8, min_df=5)
    tfidf_transformer = tfidf_vectorizer.fit(df.texto_stemizado.astype(str))
    salvaTransformer(tfidf_transformer, 'TFIDF', path)
    x_tfidf_train = tfidf_transformer.transform(X_train)
    x_tfidf_test = tfidf_transformer.transform(X_test)
    return tfidf_transformer, x_tfidf_train, x_tfidf_test


# ------------------------------------------------
# Funcao de geracao de matriz BM25
# ------------------------------------------------
from BM25_Transformer import *


def extraiFeaturesBM25(df_amostra_final, tfidf_transformer, x_tfidf_train,
                       x_tfidf_test, path):
    df_amostra_final_tfidf = tfidf_transformer.transform(df_amostra_final)
    bm25_transformer = BM25Transformer()
    bm25_transformer.fit(df_amostra_final_tfidf)
    salvaTransformer(bm25_transformer, 'BM25', path)
    x_bm25_train = bm25_transformer.transform(x_tfidf_train)
    x_bm25_test = bm25_transformer.transform(x_tfidf_test)
    return bm25_transformer, x_bm25_train, x_bm25_test


# ------------------------------------------------
# Funcao de geracao de matriz LSA
# ------------------------------------------------
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import Pipeline


def recupera_lsi_transformer(df, topics):
    tfidf_vectorizer = TfidfVectorizer(token_pattern=r'(?u)\b[A-Za-z]+\b',
                                       max_df=0.8, min_df=5)
    svd_model = TruncatedSVD(n_components=topics, algorithm='randomized',
                             n_iter=10, random_state=42)
    svd_transformer = Pipeline([('tfidf', tfidf_vectorizer),
                                ('svd', svd_model)])
    svd_transformer = svd_transformer.fit(df.texto_stemizado.astype(str))
    return svd_transformer


def extraiFeaturesLSI(df_amostra_final, X_train, X_test, topics, path):
```

```
    svd_transformer = recupera_lsi_transformer(df_amostra_final, topics)
    salvaTransformer(svd_transformer, 'LSI' + str(topics), path)
    x_lsi_train = svd_transformer.transform(X_train)
    x_lsi_test = svd_transformer.transform(X_test)
    return svd_transformer, x_lsi_train, x_lsi_test


##############################################
##############################################
# FUNCOES DE MODELAGEM
##############################################
##############################################

# ------------------------------------------------
# Funcao que faz a busca de hiper-parametros para um modelo
# ------------------------------------------------
from imblearn.ensemble import BalancedBaggingClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import StratifiedKFold


def treina_modelo_grid_search(x_tfidf_train, y_train, classificador, nomeModelo,
                              feature_type, param_grid,
                              n_iterations_grid_search, n_jobs):
    print(">> Fazendo Grid Search para classificador " + nomeModelo)
    # max_samples=round(x_tfidf_train.shape[0] * 0.6)
    stratify_5_folds = StratifiedKFold(n_splits=5, random_state=42)
    start_time = time.time()
    classificadorBag = BalancedBaggingClassifier(classificador, n_jobs=1,
                                                 bootstrap=False,
                                                 random_state=42)
    classificadorOVR = OneVsRestClassifier(classificadorBag, n_jobs=1)
    grid_search = RandomizedSearchCV(estimator=classificadorOVR,
                                     param_distributions=param_grid,
                                     cv=stratify_5_folds,
                                     n_jobs=n_jobs, verbose=2, refit=True,
                                     n_iter=n_iterations_grid_search,
                                     scoring='precision_weighted')
    grid_search.fit(x_tfidf_train, y_train)
    grid_results = ""
    means = grid_search.cv_results_['mean_test_score']
    stds = grid_search.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds,
                                 grid_search.cv_results_['params']):
```

```
        grid_results += "%0.3f (+/-%0.03f) for %r \n" % (mean, std * 2, params)

    total_time = time.time() - start_time
    print(
        "Tempo para execucao do GridSearch para OVR Balanced Bagging "
        + nomeModelo + " para " + str(
            x_tfidf_train.shape[0]) + " elementos: ",
        str(timedelta(seconds=total_time)))
    modelo = Modelo(feature_type + '_' + nomeModelo)
    # modelo.setMaxSamples(max_samples)
    modelo.setTamanhoConjuntoTreinamento(x_tfidf_train.shape[0])
    modelo.setTempoProcessamento(
        str(timedelta(seconds=grid_search.refit_time_)))
    modelo.setFeatureType(feature_type)
    modelo.setBestEstimator(grid_search.best_estimator_)
    modelo.setBestParams(grid_search.best_params_)
    modelo.setGridCVResults(grid_results)
    return modelo


# ------------------------------------------------
# Funcao que faz o teste de um modelo
# ------------------------------------------------

from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import classification_report


def testa_modelo(x_tfidf_test, y_test, modelo):
    print(">> Testando classificador " + modelo.getNome())
    start_time = time.time()
    y_pred = modelo.getBestEstimator().predict(x_tfidf_test)
    y_pred_proba = modelo.getBestEstimator().predict_proba(x_tfidf_test)
    y_pred_proba_df = pd.DataFrame(y_pred_proba,
                                   columns=modelo.getBestEstimator().classes_)
    total_time = time.time() - start_time
    print("Tempo para fazer a predicao de  " + str(
        x_tfidf_test.shape[0]) + " elementos: ",
          str(timedelta(seconds=total_time)))

    start_time = time.time()
```

```
    accuracy = accuracy_score(y_test, y_pred)
    balanced_accuracy = balanced_accuracy_score(y_test, y_pred)
    macro_precision, macro_recall, macro_fscore = score(y_test, y_pred,
                                                        average='macro',
                                                        labels=np.unique(
                                                            y_pred))[:3]
    micro_precision, micro_recall, micro_fscore = score(y_test, y_pred,
                                                        average='weighted',
                                                        labels=np.unique(
                                                            y_pred))[
                                                  :3]
    confusion_matrix = multilabel_confusion_matrix(y_true=y_test, y_pred=y_pred)
    classes = y_test.unique().astype(str).tolist()
    # print(classification_report(y_test, y_pred, target_names=classes))
    classification_report_dict = classification_report(y_test, y_pred,
                                                       target_names=classes,
                                                       output_dict=True)
    total_time = time.time() - start_time
    # print('Confusion matrix:\n', conf_mat)
    print(
        "Tempo para recuperar metricas:  " + str(timedelta(seconds=total_time)))

    modelo.setAccuracy(accuracy)
    modelo.setBalancedAccuracy(balanced_accuracy)
    modelo.setMacroPrecision(macro_precision)
    modelo.setMacroRecall(macro_recall)
    modelo.setMacroFscore(macro_fscore)
    modelo.setMicroPrecision(micro_precision)
    modelo.setMicroRecall(micro_recall)
    modelo.setMicroFscore(micro_fscore)
    modelo.setConfusionMatrix(confusion_matrix)
    modelo.setClassificationReport(classification_report_dict)

    return modelo, y_pred, y_pred_proba_df
```

Listing D.3: Classe Modelo

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 27 16:08:24 2018

@author: anarocha
"""
import pandas as pd
```

```
import os
import pickle

class Modelo:
    def __init__(self, nome, feature_type = None, tempo_processamento = None,
                 best_params_ = None, best_estimator_ = None, grid_scores_ = None,
                 accuracy = None,balanced_accuracy = None,  macro_precision = None,
                 macro_recall = None, macro_fscore = None, micro_precision = None,
                 micro_recall = None, micro_fscore = None, confusion_matrix = None,
                 num_estimators = None, max_samples = None,
                 tamanho_conjunto_treinamento = None, id_execucao = None,
                 data = None,classification_report = None, grid_cv_results = None):
        self.nome = nome
        self.feature_type = feature_type
        self.tempo_processamento = tempo_processamento
        self.best_params_ = best_params_
        self.best_estimator_ = best_estimator_
        self.grid_scores_ = grid_scores_
        self.accuracy = accuracy
        self.balanced_accuracy = balanced_accuracy
        self.macro_precision = macro_precision
        self.macro_recall = macro_recall
        self.macro_fscore = macro_fscore
        self.micro_precision = micro_precision
        self.micro_recall = micro_recall
        self.micro_fscore = micro_fscore
        self.confusion_matrix = confusion_matrix
        self.num_estimators = num_estimators
        self.max_samples = max_samples
        self.tamanho_conjunto_treinamento = tamanho_conjunto_treinamento
        self.id_execucao = id_execucao
        self.data = data
        self.classification_report = classification_report
        self.grid_cv_results = grid_cv_results

    def setNome(self, nome):
        self.nome = nome

    def setFeatureType(self, feature_type):
        self.feature_type = feature_type

    def setTempoProcessamento(self, tempo_processamento):
        self.tempo_processamento = tempo_processamento

    def setBestParams(self, best_params_):
```

```
        self.best_params_ = best_params_

    def setBestEstimator(self, best_estimator_):
        self.best_estimator_ = best_estimator_

    def setGridScores(self, grid_scores_):
        self.grid_scores_ = grid_scores_

    def setAccuracy (self, accuracy):
        self.accuracy = accuracy

    def setBalancedAccuracy(self, balanced_accuracy):
        self.balanced_accuracy = balanced_accuracy

    def setMacroPrecision(self, macro_precision):
        self.macro_precision = macro_precision

    def setMacroRecall(self, macro_recall):
        self.macro_recall = macro_recall

    def setMacroFscore(self, macro_fscore):
        self.macro_fscore = macro_fscore

    def setMicroPrecision(self, micro_precision):
        self.micro_precision = micro_precision

    def setMicroRecall(self, micro_recall):
        self.micro_recall = micro_recall

    def setMicroFscore(self, micro_fscore):
        self.micro_fscore = micro_fscore

    def setConfusionMatrix(self, confusion_matrix):
        self.confusion_matrix = confusion_matrix

    def setNumEstimators(self, num_estimators):
        self.num_estimators = num_estimators

    def setMaxSamples(self, max_samples):
        self.max_samples = max_samples

    def setTamanhoConjuntoTreinamento(self, tamanho_conjunto_treinamento):
        self.tamanho_conjunto_treinamento = tamanho_conjunto_treinamento

    def setIdExecucao(self, id_execucao):
```

```
        self.id_execucao = id_execucao

    def setData(self, data):
        self.data = data

    def setClassificationReport(self, classification_report):
        self.classification_report = classification_report

    def setGridCVResults(self,grid_cv_results):
        self.grid_cv_results = grid_cv_results

    def getNome(self):
        return self.nome

    def getFeatureType(self):
        return self.feature_type

    def getTempoProcessamento(self):
        return self.tempo_processamento

    def getBestParams(self):
        return self.best_params_

    def getBestEstimator(self):
        return self.best_estimator_

    def getGridScores(self, grid_scores_):
        return self.grid_scores_

    def getAccuracy (self):
        return self.accuracy

    def getBalancedAccuracy (self):
        return self.balanced_accuracy

    def getMacroPrecision(self):
        return self.macro_precision

    def getMacroRecall(self):
        return self.macro_recall

    def getMacroFscore(self):
        return self.macro_fscore

    def getMicroPrecision(self):
```

```
        return self.micro_precision

    def getMicroRecall(self):
        return self.micro_recall

    def getMicroFscore(self):
        return self.micro_fscore

    def getConfusionMatrix(self):
        return self.confusion_matrix

    def getNumEstimators(self):
        return self.num_estimators

    def getMaxSamples(self):
        return self.max_samples

    def getTamanhoConjuntoTreinamento(self):
        return self.tamanho_conjunto_treinamento

    def getIdExecucao (self):
        return self.id_execucao

    def getData (self):
        return self.data

    def getClassificationReport(self):
        return self.classification_report

    def getGridCVResults(self):
        return self.grid_cv_results

    def imprime(self):
        print(" ")
        print("Nome modelo: " + self.nome)
        print("Quantidade de elementos de treinamento: "
              + str(self.tamanho_conjunto_treinamento))
        print("Tempo de treinamento: " + str(self.tempo_processamento))
        print("Feature Type: " + self.feature_type)
        print("Accuracy: " + str(self.accuracy))
        print("Balanced Accuracy: " + str(self.balanced_accuracy))
        print('macro_precision %s \nmacro_recall     %s \nmacro_fscore     %s'
              % (self.macro_precision, self.macro_recall, self.macro_fscore))
        print('micro_precision %s \nmicro_recall     %s \nmicro_fscore     %s'
              % (self.micro_precision, self.micro_recall, self.micro_fscore))
```

```
    def salvaClassificationReport(self, arquivo):
        df = pd.DataFrame.from_dict(self.classification_report)
        df['nome_algoritmo']=self.nome
        df['id_execucao']=self.getIdExecucao()
        if not (os.path.isfile(arquivo)):
            with open(arquivo, 'a') as f:
                df.to_csv(f, header=True)
                f.close()
        else:
            with open(arquivo, 'a') as f:
                df.to_csv(f, header=False)
                f.close()

    def salvaModelo(self, path):
        nomePicke = path + 'Modelo_' + self.nome.replace(' ', '') + '.p'
        arquivoPickle = open(nomePicke, 'wb')
        pickle.dump(self.getBestEstimator(), arquivoPickle)
        arquivoPickle.close()
```

Listing D.4: BM25 Transformer

```
# coding: UTF-8
# Codigo retirado de: https://github.com/arosh/BM25Transformer

from __future__ import absolute_import, division, print_function, \
    unicode_literals
import numpy as np
import scipy.sparse as sp
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.utils.validation import check_is_fitted
from sklearn.feature_extraction.text import _document_frequency


class BM25Transformer(BaseEstimator, TransformerMixin):
    """
    Parameters
    ----------
    use_idf : boolean, optional (default=True)
    k1 : float, optional (default=2.0)
    b : float, optional (default=0.75)
    References
    ----------
    Okapi BM25: a non-binary model - Introduction to Information Retrieval
    http://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.htm
```

```
    """

    def __init__(self, use_idf=True, k1=2.0, b=0.75):
        self.use_idf = use_idf
        self.k1 = k1
        self.b = b

    def fit(self, X):
        """
        Parameters
        ----------
        X : sparse matrix, [n_samples, n_features]
            document-term matrix
        """
        if not sp.issparse(X):
            X = sp.csc_matrix(X)
        if self.use_idf:
            n_samples, n_features = X.shape
            df = _document_frequency(X)
            idf = np.log((n_samples - df + 0.5) / (df + 0.5))
            self._idf_diag = sp.spdiags(idf, diags=0, m=n_features,
                                        n=n_features)
        return self

    def transform(self, X, copy=True):
        """
        Parameters
        ----------
        X : sparse matrix, [n_samples, n_features]
            document-term matrix
        copy : boolean, optional (default=True)
        """
        if hasattr(X, 'dtype') and np.issubdtype(X.dtype, np.float):
            # preserve float family dtype
            X = sp.csr_matrix(X, copy=copy)
        else:
            # convert counts or binary occurrences to floats
            X = sp.csr_matrix(X, dtype=np.float64, copy=copy)

        n_samples, n_features = X.shape

        # Document length (number of terms) in each row
        # Shape is (n_samples, 1)
        dl = X.sum(axis=1)
        # Number of non-zero elements in each row
```

```
    # Shape is (n_samples, )
    sz = X.indptr[1:] - X.indptr[0:-1]
    # In each row, repeat `dl` for `sz` times
    # Shape is (sum(sz), )
    # Example
    # -------
    # dl = [4, 5, 6]
    # sz = [1, 2, 3]
    # rep = [4, 5, 5, 6, 6, 6]
    rep = np.repeat(np.asarray(dl), sz)
    # Average document length
    # Scalar value
    avgdl = np.average(dl)
    # Compute BM25 score only for non-zero elements
    data = X.data * (self.k1 + 1) / (
                X.data + self.k1 * (1 - self.b + self.b * rep / avgdl))
    X = sp.csr_matrix((data, X.indices, X.indptr), shape=X.shape)

    if self.use_idf:
        check_is_fitted(self, '_idf_diag', 'idf vector is not fitted')

        expected_n_features = self._idf_diag.shape[0]
        if n_features != expected_n_features:
            raise ValueError("Input has n_features=%d while the model"
                             " has been trained with n_features=%d" % (
                                 n_features, expected_n_features))
        # *= doesn't work
        X = X * self._idf_diag

    return X
```

Anexo I

Termo de abertura do projeto Classificador de Assuntos

TERMO DE ABERTURA DO PROJETO TAP

1. IDENTIFICAÇÃO DO DOCUMENTO

Nome do Projeto:	**CLASSSIFICADOR DE TEMAS**	**Código:**	
Demandante:	**CSJT**		

2. NOMEAÇÃO DO GERENTE DO PROJETO

Gerente do Projeto:	ANA CAROLINA PEREIRA ROCHA MARTINS
Unidade Administrativa:	SPTRI - CTPJE
Telefone:	61 3043 7929
E-Mail:	acprocha@tst.jus.br

3. HISTÓRICO DE REVISÕES

Data	Versão	Descrição	Responsável
30/05/2017	1.0	Versão inicial	Ana Carolina P. Rocha Martins

4. DOCUMENTOS DE REFERÊNCIA

Plano Estratégico do Conselho Superior da Justiça do Trabalho - 2015 -2020. Disponível em <http://www.csjt.jus.br/c/document_library/get_file?uuid=a194caba-556e-4cf6-96db-78c0f21b1438&groupId=955023 >

Plano Estratégico da Justiça do Trabalho - 2015 -2020. Disponível em <http://www.csjt.jus.br/c/document_library/get_file?uuid=f525e749-2197-438c-91ae-d31acfe4cbdf&groupId=955023>

FELDMAN, R., & SANGER, J. (2007). **The text mining handbook: advanced approaches in analyzing unstructured data**. Cambridge university press.

CARVALHO, Maximiliano (2017). **O Princípio Da Automatização Do Processo Eletrônico Como Catalisador Da Observância Aos Precedentes Do TST.** Revista Fórum Trabalhista RTF, Nº 24. Disponível em <http://www.editoraforum.com.br/ef/index.php/publicacoes/periodicos/listar-periodicos/revista-farum-trabalhista-rft/ >

5. JUSTIFICATIVA DO PROJETO

O crescimento do volume de processos trabalhistas e do número de documentos dentro do PJe tem tornado análise manual desta quantidade de dados digitais um limitador para a agilidade de tramitação dos processos. Assim, se faz necessário que o PJe KZ adquira funcionalidades baseadas em inteligência artificial que possam auxiliar o homem a

trabalhar com a ferramenta de forma mais eficiente, maximizando-se "a produção com menor consumo de energia, de modo a entregar quantitativa e qualitativamente mais rápido a prestação jurisdicional, ao mesmo tempo em que se melhora a qualidade de vida dos usuários internos e externos do processo eletrônico" (CARVALHO, 2017), poupando ao homem o trabalho que uma máquina pode fazer.

À vista disto, este projeto propõe uma automação de máquina que, a partir da leitura dos documentos de decisões do segundo grau de processos com tema identificado, possa aprender o vocabulário utilizado para abordar cada tema, e a partir disto, classifique de forma automática novos processos como tema ainda não identificado em segundo e terceiro grau. Foram escolhidos os temas, ao invés de assuntos, por serem uma informação mais detalhada e útil a justiça trabalhista.

Uma vez implementado este projeto, uma série de funcionalidades poderão ser construídas no PJe KZ, tais como triagem por temas, busca textual em peças do processo, busca textual semântica em peças do processo, sugestão de análise de precedentes no momento da redação de decisões, criação de avisos para a identificação de temas de processos repetitivos, sugestão de análise de processos similares no momento da conclusão de uma decisão, entre outros.

6. OBJETIVO DO PROJETO

Construir um aprendizado de máquina que seja capaz de reconhecer informações relevantes nas peças processuais e, a partir destas informações, identificar os temas ali abordados de forma automática.

7. ALINHAMENTO ESTRATÉGICO

Origem (PEI/PDTIC/PETIC)	Objetivo	Indicador	Impacto no indicador
Plano Estratégico 2015-2020 do CSJT	Contribuir para a melhoria da prestação jurisdicional na Justiça do Trabalho de 1o e 2o graus	Índice de Satisfação Interna com o Sistema do Processo Judicial Eletrônico (ISIPJe)	Influenciar positivamente no número de questionários que aprovam o sistema PJe.
Plano Estratégico	Contribuir para a melhoria da prestação jurisdicional	Índice de Satisfação Externa com o	Influenciar positivamente no

2015-2020 do CSJT	na Justiça do Trabalho de 1o e 2o graus	Sistema do Processo Judicial Eletrônico (ISEPJe)	número de questionários que aprovam o sistema PJe, uma vez que haverá maior celeridade com a implementação do projeto.
Plano Estratégico da Justiça do Trabalho 2015 -2020	Assegurar a celeridade e a produtividade na prestação jurisdicional	Tempo Médio de Duração do Processo - 2a Instância (TMDP2)	Diminuir o tempo de tramitação do processo, o que impacta diretamente no tempo médio de duração do processo.
Plano Estratégico da Justiça do Trabalho 2015 -2020	Assegurar a celeridade e a produtividade na prestação jurisdicional	Índice de Processos Julgados (IPJ)*	Ao trazer agilidade ao processo de julgamento, poderá aumentar o número de processos julgados

8. DESCRIÇÃO DO ESCOPO

- Serviço de extração dos documentos do PJe em formato de texto, para que possa ser feita a indexação destes;
- Pré-processamento dos textos extraídos (remoção de tags html por exemplo);
- Indexação dos textos do PJe;
- Criação de uma estrutura de dados que contenha os temas uniformizados pelo Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho[1]. Caso não haja uma priorização do projeto Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho, os temas serão buscados do Banco Nacional de Jurisprudência Trabalhista (BANJUR) e dos temas de Recursos Repetitivos;
- Criação de um modelo de mineração de textos capaz de classificar o tema dos processos que já se encontram classificados (aprendizado);
- Aplicação do modelo de mineração de textos para a classificação dos processos das bases dos tribunais que aderirem à solução (aplicação do aprendizado);

1 O Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho, ganhador do concurso "Projetos e Oportunidades" do Tribunal Superior do Trabalho (TST) de 2017, se propõe a estruturar e uniformizar os temas que são utilizados na classificação de processos nos órgãos judicantes deste órgão.

- Criação de uma tela para alteração do tema classificado, para o caso de haver necessidade de retificação ou complementação.
- Integração com o PJe KZ;
- Guia de instalação;
- Criação do manual referente à funcionalidade.

8.1. Não escopo do projeto

- Criação de novas funcionalidades baseadas na indexação dos textos e no aprendizado construído;
- Criação de um *thesaurus* jurídico (dicionário de conceitos negociais);
- Unificação dos temas processuais da justiça do trabalho;
- Criação de novos temas;

8.2. Premissas Iniciais

- Assinatura da Carta de Compromisso de Liberação, até o dia 1º de junho de 2017 (não prorrogável), para que seja oficializada a permissão para a servidora Ana Carolina Pereira Rocha Martins se ausentar às sextas feiras de sua unidade de trabalho, durante o período de 24 meses, para que possa comparecer às aulas ministradas na UNB e empreender demais estudos do mestrado necessários à conclusão do projeto;
- Permissão para que, a partir do primeiro semestre de 2018, a servidora Ana Carolina Pereira Rocha Martins atue em demandas relacionadas a este projeto em conjunto com as demais demandas de sua atribuição ordinária;
- Aprovação da servidora Ana Carolina Pereira Rocha Martins em todas as matérias do programa de mestrado, com tolerância a apenas uma reprovação;
- Apoio de uma pessoa ou grupo de negócio detentor do saber jurídico inerente ao projeto para apoio do saber negocial e validação da solução. Não há necessidade inicial de dedicação integral ao projeto;
- Apoio da SISUP para auxílio da disponibilização da solução. Não há necessidade inicial de dedicação integral ao projeto;
- Apoio da Seção de Gestão do Produto - SGPROD -para auxílio da validação da solução.Não há necessidade inicial de dedicação integral ao projeto;

- Apoio de uma pessoa da Seção de Métodos e Padrões (SMPAD) para consolidação da arquitetura e apoio no desenvolvimento. Não há necessidade inicial de dedicação integral ao projeto;
- Apoio de uma pessoa da Seção de Análise e Projetos para Tribunais (SPTRI) no desenvolvimento o projeto. Não há necessidade inicial de dedicação integral ao projeto;
- Priorização do projeto por parte da gestão do CSJT dos biênios de 2016-2017 e 2018-2019.
- Acesso às bases bugfix de todos os TRTs, incluindo as bases binárias;
- Acesso aos dados do Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho;
- Caso o Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho não seja implementado, será preciso ter acesso à base do BANJUR e aos temas de recursos repetitivos do TST e dos regionais.

8.3. Restrições Iniciais

- Considerando a elevada influência dos dados com os algoritmos de *text mining*[2] no desempenho dos algoritmos, segundo FELDMAN & SANGER (2007) , a performance desses métodos só pode ser medida em caráter experimental, se comportando de forma diferente para cada cenário de aplicação. Neste caso, embora não se possa determinar o percentual do índice de acerto, é sabido que uma margem de erro é esperada neste tipo de técnica. Assim, fica aqui o registro que esta solução não terá 100% de eficácia.

2 Área da inteligência artificial que se propõe a resolver a crise da sobrecarga de informação extraindo as informações relevantes de textos ao combinar técnicas de mineração de dados, aprendizado de máquina, processamento de linguagem natural, recuperação da informação e gerenciamento do conhecimento (FELDMAN & SANGER, 2007)

9. CRONOGRAMA DE MARCOS

Como se trata de um projeto de dissertação de mestrado profissional, o prazo estipulado pelo Programa de Pós Graduação em Ciência Aplicada da UNB é de 2 anos a partir do ingresso como aluno regular, podendo se estender por mais 6 meses, se necessário. Este então é o balizador do prazo necessário para a implementação solução. As datas aqui apresentadas são apenas uma estimativa do tempo de desenvolvimento da solução, podendo ser alteradas à medida que forem sendo desenvolvidas. À pedido da Coordenação Nacional Executiva do PJe(CNE), a implementação do projeto dentro da Coordenação Técnica do PJe (CTPJE) terá início em 2018.

Marcos	Previsão
Criação de serviço para disponibilização das peças processuais em formato de texto para a ferramenta de indexação	Abril/2018
Instalação de uso da ferramenta de indexação no ambiente de desenvolvimento (Ex: Solr, Elasticsearch)	Junho/2018
Qualificação da dissertação junto à UNB	Junho/2018
Separação de processos que serão usados para o treinamento	Julho/2018
Implementação de algoritmos de mineração de textos para a descoberta do modelo de classificação	Fevereiro/2019
Aplicação dos conhecimentos construídos em novas peças processuais dentro do PJe KZ	Março/2019
Homologação da solução no ambiente de desenvolvimento/homologação	Maio/2019
Realimentação da base de conhecimento com as novas decisões julgadas	Junho/2019
Implantação em produção em um Tribunal Regional do Trabalho para testes	Junho/2019
Implantação nos demais regionais	Agosto/2019

10. RISCOS PREVIAMENTE IDENTIFICADOS

Probabilidade		Impacto (Efeito que o risco exerce sobre o projeto)		Grau de Risco	
Índice	Probabilidade de Ocorrência	Índice	Impacto	Índice	Descrição
1	Improvável	1	Muito baixo	1 a 2	Muito Baixo - Impacto mínimo no projeto
2	Pouco provável	2	Baixo	3 a 5	Baixo - Impacto no projeto
3	Provável	3	Médio	6 a 10	Médio - Impacto no projeto
4	Muito provável	4	Alto	12 a 16	Alto - Impacto no projeto
5	Quase Certo	5	Muito alto	20 a 25	Muito Alto - Comprometimento no projeto

Tabela de identificação e plano de resposta aos riscos:

Nº	Risco Encontrado	Probabi-lidade	Impacto	Grau do Risco	Ação de Contingência	Responsável pela ação de contingência
1	Impossibilidade de assinar a Carta de Compromisso de Liberação até 1º de Junho de 2017	3	5	15	Adiamento no ingresso no mestrado por um ano, com possibilidade de atraso no cronograma devido à falta de conhecimento necessário para a implementação do projeto.	-
2	Impossibilidade de permitir que a servidora Ana Carolina atue em demandas relacionadas ao projeto em conjunto com as demais demandas	2	4	8	Não há. Algumas datas provavelmente serão postergadas.	CNE e CTPJe

3	Reprovação da servidora Ana Carolina em mais de uma máteria no programa de mestrado da UNB	2	5	10	Continuação do projeto sem parceria com a UNB com possibilidade de atraso no cronograma devido à falta de conhecimento necessário para a implementação do projeto.	Ana Carolina Pereira Rocha Martins
4	Impossibilidade de se ter uma pessoa disponível que possa responder pelo conhecimento negocial	2	5	10	Construção do conhecimento de forma autodidata e buscando conhecimento com pessoas da área jurídica.	Ana Carolina Pereira Rocha Martins
5	Impossibilidade da SISUP atuar no projeto	2	5	10	Não há. A solução não poderá ser implantada em produção.	CNE e CTPJe
6	Impossibilidade da SGPROD atuar no projeto	2	4	8	A validação da solução terá que acontecer pela pessoa de apoio negocial	CNE e CTPJe
7	Impossibilidade de que a SMPAD disponibilize alguém para auxiliar na proposta de arquitetura do projeto	2	3	6	Uma arquitetura será proposta, mas poderá não ser a mais adequada ao projeto PJe.	Ana Carolina Pereira Rocha Martins
8	Impossibilidade de que a SPTRI disponibilize alguém para auxiliar no desenvolvimento do projeto	2	4	8	Disponibilizar alguém da Seção de Análise e Projeto para Varas (SPVAR) ou SMPAD. Não sendo possível, poderá haver atraso no cronograma.	CNE e CTPJe

9	Troca da gestão e despriorização do projeto	4	5	20	Não há. É necessário que este projeto esteja priorizado pela CNE.	Presidente do CSJT;CNE e CTPJe
10	Impossibilidade de acesso às bases bugfix dos TRTs	2	5	10	Não há. O projeto necessita disto.	CNE e CTPJe
11	Impossibilidade de acesso aos dados do Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho	2	4	8	Concessão de acesso aos temas e processos presentes no BANJUR e aos temas e processos de recursos de revista repetitivos e demandas repetitivas.	CNE e CTPJe
12	Projeto Unificação dos Temas Processuais Para Toda a Justiça do Trabalho não concluído a tempo	3	4	12	Concessão de acesso aos temas e processos presentes no BANJUR e aos temas e processos de recursos de revista repetitivos e demandas repetitivas.	CNE e CTPJe
13	Impossibilidade de acesso aos dados do BANJUR e dos temas de processos repetitivos	2	5	10	Trocar a abordagem. Ao invés de se fazer um aprendizado de máquina para a classificação posterior de novas peças, faz-se o reconhecimento de citações de súmulas, dispositivos (entre outras entidades relevantes) para que se faça o agrupamento de processos similares	Ana Carolina Pereira Rocha Martins

14	Não confiabilidade da classificação de temas/assuntos existente atualmente (BANJUR, recursos repetitivos)	3	5	15	Trocar a abordagem. Ao invés de se fazer um aprendizado de máquina para a classificação posterior de novas peças, faz-se o reconhecimento de citações de súmulas, dispositivos (entre outras entidades relevantes) para que se faça o agrupamento de processos similares	Ana Carolina Pereira Rocha Martins
15	Baixa quantidade de processos categorizados corretamente, não sendo suficiente para construir a inteligência de classificação	3	4	12	Trocar a abordagem. Ao invés de se fazer um aprendizado de máquina para a classificação posterior de novas peças, faz-se o reconhecimento de citações de súmulas, dispositivos (entre outras entidades relevantes) para que se faça o agrupamento de processos similares	Ana Carolina Pereira Rocha Martins
16	Impossibilidade de disponibilizar máquina com processamento adequado	3	4	12	Construir a solução de forma distribuída (clusters)	Ana Carolina ,SISUP e equipe de infraestrutura do regional
17	Na impossibilidade de disponibilizar máquina com processamento adequado, impossibilidade de construir a aplicação de forma distribuída	3	5	15	Não há. É necessário que se tenha capacidade computacional disponível.	-

11. PRINCIPAIS ENVOLVIDOS NO PROJETO

Nome	Área	Papel	Telefone(s)	E-mail
Dr. Maximiliano Pereira de Carvalho	Coordenação Nacional Executiva do PJe (CNE)/ CSJT	Patrocinador	061 3043 7384	maximiliano.carvalho@tst.jus.br
Herbert Parente	CTPJE/ CSJT	Coordenador Técnico	061 3043 7711	herbert.parente@tst.jus.br
Ana Carolina Pereira Rocha Martins	SPTRI/ CTPJE/ CSJT	Desenvolvedora, líder técnica do projeto	061 3043 2979 061 99927 5063	acprocha@gmail.com
Glauber Moreira Rocha	SPTRI/ CTPJE/ CSJT	Apoio no desenvolvimento	061 3043 7824	glauber.rocha@tst.jus.br
Christiano Guimaraes de Carvalho	SISUP/ CTPJE/ CSJT	Atuar na disponibilização da solução	061 3043 7927	christiano.carvalho@tst.jus.br
Victor Lopes Dias de Araújo	SGPROD /CTPJE/ CSJT	Validação da demanda	061 3043 7941	victor.araujo@tst.jus.br
Daniel Souto Rocha	SMPAD/ CTPJE/ CSJT	Apoio na definição da arquitetura	061 3043 7582	daniel.rocha@tst.jus.br
A definir	Grupo de Negócios	Apoio negocial	A definir	A definir

12. EQUIPE TÉCNICA INICIAL DO PROJETO

Nome	Papel	Telefone(s)	E-mail
Ana Carolina Pereira Rocha Martins	Analista/Desenvolvedora	61 3043 7929	acprocha@tst.jus.br

13. APROVAÇÃO

Data	Nome	Departamento / Unidade	Assinatura
02/06/17	MAXIMILIANO CARVALHO	PRESIDENCIA CSJT	
02/06/17	FABIANO COELHO DE SOUZA	PRESIDENTE CSJT	

Todas as páginas deverão ser rubricadas