DISTRIBUIÇÃO DE PEDIDOS DE PATENTE EM PORTUGUÊS

Rafael de Sant'Anna Corrêa Nunes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Agosto de 2022

DISTRIBUIÇÃO DE PEDIDOS DE PATENTE EM PORTUGUÊS

Rafael de Sant'Anna Corrêa Nunes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Geraldo Bonorino Xexéo

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2022

Nunes, Rafael de Sant'Anna Corrêa

DISTRIBUIÇÃO DE PEDIDOS DE PATENTE EM PORTUGUÊS/Rafael de Sant'Anna Corrêa Nunes. – Rio de Janeiro: UFRJ/COPPE, 2022.

XVII, 108 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2022.

Referências Bibliográficas: p. 101 – 108.

1. classificação de texto. 2. aprendizado de máquina. 3. roteamento de patentes. 4. distribuição de patentes. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicada a
Daniela,
amada esposa.

Agradecimentos

Agradeço a **Vida**, pela saúde, **bem-estar** e liberdade para escolher livremente.

Agradeço aos meus **pais**, Jupira e Sergio, por deixar clara a necessidade de **aprender a pesquisar**.

Agradeço à **Daniela**, pela dedicação, paciência e cuidados.

Agradeço ao **PESC** pela oportunidade de pesquisar no Brasil e em especial ao meu orientador Geraldo Bonorino Xexéo.

Agradeço ao **INPI** por proporcionar o contexto, os dados e o tempo para pesquisar. Em especial ao colega Jaime Neiva Miranda de Souza, pelo incentivo e pela ajuda com os dados do primeiro ciclo e com a produtização do Application Router v1, que permitiu colocá-lo em produção.

Agradeço ao Governo Federal, a FAPERJ, CAPES e CNPq pelo apoio financeiro à Universidade e ao Programa.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DISTRIBUIÇÃO DE PEDIDOS DE PATENTE EM PORTUGUÊS

Rafael de Sant'Anna Corrêa Nunes

Agosto/2022

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Este trabalho propôs uma solução para a distribuição automática de pedidos de patente em português baseada em técnicas de Aprendizado de Máquina e Classificação de Textos, que obteve como melhor resultado uma acurácia média de 62,32%. Divulga um conjunto de dados de pedidos de patente construído a partir de dados públicos do Instituto Nacional da Propriedade Industrial (INPI), e do *European Patent Office (EPO)*, estuda formas de limpar e preparar os dados, extrair *features* e reduzir dimensionalidade; além de avaliar o desempenho de diversos algoritmos de classificação baseados em aprendizado de máquina utilizando a técnica de validação cruzada.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BRAZILIAN PORTUGUESE PATENT APPLICATIONS ROUTING

Rafael de Sant'Anna Corrêa Nunes

August/2022

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

This work proposes a Machine Learning and Text Classification-based solution to automate the distribution of Brazilian Portuguese patent applications that achieves a mean accuracy of 62.32%. It discloses a new dataset based on public data from the Instituto Nacional da Propriedade Industrial (INPI) and the European Patent Office (EPO); studies different ways to clean, sanitize, and prepare data, extract features, and reduce dimensionality; and uses cross-validation to evaluate various estimator's performance.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AED	Análise Exploratória de Dados, p. 48
BERT	*Bidirectional Encoder Representations from Transformers*, p. 8, 100
BOHB	*Bayesian Optimization and Hyperband*, p. 100
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, p. v
CIP	Classificação Internacional de Patentes, p. 2
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico, p. v
EPO	*European Patent Office*, p. 88
FAPERJ	Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro, p. v
HPO	*Hyper-Parameter Optimization*, p. 100
INPI	Instituto Nacional da Propriedade Industrial, p. v, 2
IPC	International Patent Classification, p. 2
MLP	*Multi-Layer Perceptron*, p. 80, 100
MLP	*Multi-layer Perceptron*, p. 8

NBG Naïve-Bayes Gaussiano, p. 8

NBG *Naïve-Bayes Gaussiano*, p. 80, 100

OMPI Organização Mundial da Propriedade Intelectual, p. 2

OPS *Open Patent Service*, p. 88

PCT *Patent Cooperation Treaty*, p. 39

RPI Revista da Propriedade Industrial, p. 8

SVD Singular Value Decomposition, p. 21

SVM *Support Vector Machine*, p. 8, 80, 100

WIPO World Intellectual Property Organization, p. 2

Capítulo 1

Introdução

Este trabalho explora técnicas de aprendizado de máquina para propor uma solução de distribuição de pedidos de patentes em português. Ele utiliza técnicas de processamento de linguagem natural para criar e avaliar classificadores automáticos de texto, que atribuem uma categoria a cada pedido de patente recebido pelo INPI.

No Brasil, o INPI (2022b) é o responsável pela concessão de patentes, e conforme seu site[1]: "patente é o direito, temporário, concedido pelo Estado, de impedir terceiros, sem o seu consentimento, de produzir, usar, colocar à venda, vender ou importar produto objeto de sua patente e/ou processo ou produto obtido diretamente por processo por ela patenteado". Em contrapartida, o Estado exige que o inventor revele detalhadamente todo o conteúdo técnico protegido pela patente.

Ainda segundo o INPI (2022b), as patentes podem ser:

- **Patente de Invenção (PI)**, para as quais os requisitos para a concessão são a atividade inventiva, a novidade e a aplicação industrial, tendo validade de 20 anos a partir da data de depósito;
- **Patente de Modelo de Utilidade (MU)**, para objetos de uso prático suscetível de aplicação industrial, com validade de 15 anos a partir da data de depósito, e, ainda,

[1] `https://www.gov.br/inpi/pt-br/servicos/perguntas-frequentes/patentes#patente`

- **Certificado de Adição de Invenção (C)**, que introduz um aperfeiçoamento ou desenvolvimento no objeto da invenção, sendo que o certificado não altera a data de vigência da patente.

De acordo com o Manual Básico para Proteção por Patentes de Invenções, Modelos de Utilidade e Certificados de Adição (INPI, 2021, p.9), a intenção do sistema de registro de patentes é estimular o investimento no desenvolvimento de novas tecnologias e produtos, uma vez que durante a vigência da patente, o titular poderá ser recompensado pelos esforços e gastos despendidos. Garantir a divulgação do conhecimento, uma vez que a patente é pública. E permitir que toda a sociedade, após o período de exclusividade, explore comercialmente o conhecimento, não apenas o titular.

A Organização Mundial da Propriedade Intelectual (OMPI), a fim de estabelecer a novidade e avaliar a atividade inventiva ou não obviedade, publica anualmente a *International Patent Classification (IPC*[2]*)*, ou em português, Classificação Internacional de Patentes (CIP), que consiste em uma classificação uniforme de documentos de patente, com o objetivo de estabelecer uma ferramenta de busca para a recuperação de documentos de patentes pelos escritórios de propriedade intelectual e demais usuários, conforme descrito no Guia da IPC (OMPI, 2020).

O exame técnico é a atividade onde é avaliada a presença dos requisitos necessários à concessão da patente. Sendo assim, antes do técnico examinar se um pedido de patente de invenção atende aos requisitos de patenteabilidade – novidade, atividade inventiva e aplicação industrial – os pedidos precisam ter sido direcionados para a área técnica de exame mais apropriada. Portanto, a distribuição dos pedidos de patentes é uma necessidade para qualquer escritório de patente no mundo. Um exemplo são as patentes relativas as máquinas para produção agrícola que serão examinadas por uma área diferente daquelas que examinarão patentes relativas a equipamentos de telecomunicações ou daquela que examinará medicamentos.

[2] `https://www.wipo.int/classifications/ipc/en/`

1.1 Motivação

O INPI, até 2018, contava com um sistema chamado e-Distribuidor, responsável por fazer o roteamento ou distribuição dos pedidos de patente entre as áreas técnicas de exame. Esse sistema apresentava uma acurácia média inferior a 30%, causando muito retrabalho.

Conforme o Relatório de Atividades do INPI (2018), o volume de depósitos foi da ordem de 30.000 pedidos por ano, nos últimos anos, conforme a Figura 1.1. Desse total, pouco mais de 7.000 pedidos são feitos por residentes no Brasil e precisam ser direcionados para as áreas de exame sem que tenham recebido, ainda, a classificação internacional de patentes. Os outros são pedidos de patentes que já contam com a classificação internacional de patentes (IPC) e podem ser distribuídos para as áreas de exame, de maneira determinística, com base em uma simples tabela de mapeamento que lista os símbolos IPC que estão sob responsabilidade de cada área de exame técnico.

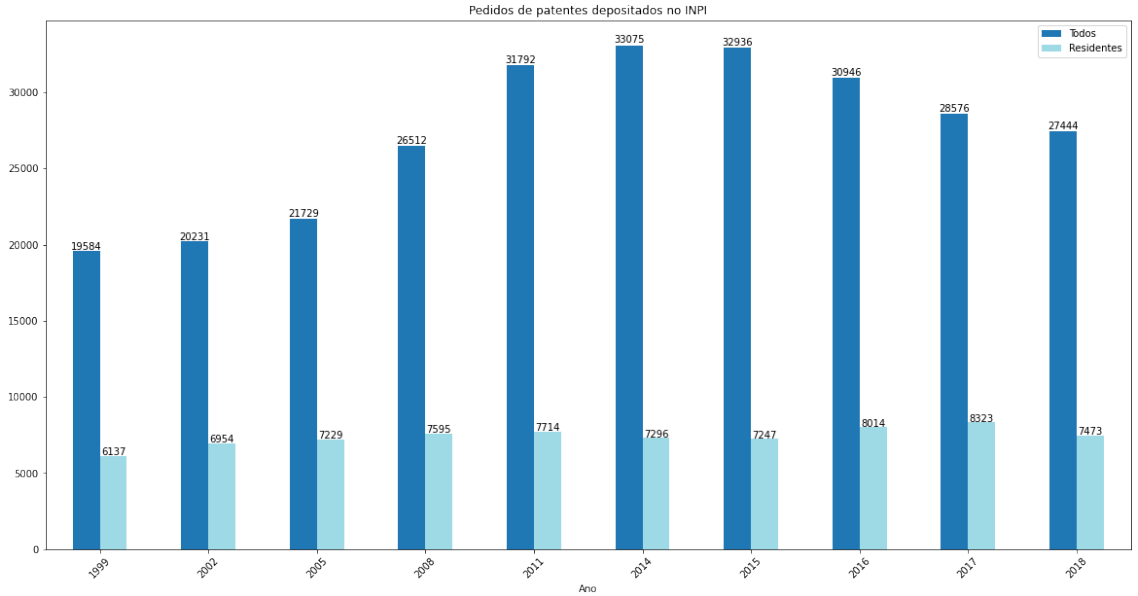

Figura 1.1: Pedidos de patentes depositados no INPI, baseada no Relatório de Atividade do INPI (2018)

Quando um pedido é roteado incorretamente, um examinador investe seu tempo lendo um pedido que não poderá ser examinado por ele, perde tempo com a leitura até perceber que houve o erro, e precisa escolher outra área para enviar o pedido.

Além do impacto direto na redução de produtividade e no trabalho de redirecionar manualmente os pedidos, existe a frustração e a quebra de raciocínio do examinador. Isso termina por impactar negativamente o esforço de redução do backlog[3] de patentes do INPI.

Dessa forma, esse trabalho teve como motivação contribuir com a diminuição do backlog de patentes do INPI através da redução do desperdício de tempo gerado pelo erro de distribuição. Com o volume de pedidos em torno de 7.000 por ano, e a acurácia média de 30%, estima-se que 4.900 pedidos geravam retrabalho dos examinadores todo ano.

Considerando um tempo médio de 15 minutos[4] para identificar o erro e redistribuir o pedido, aproximadamente 1.225 horas desperdiçadas por ano. Se for calculado o esforço disponível com base em 6 horas por dia, 5 dias úteis por semana e 48 semanas no ano, chega-se a um total de 1.440 horas por ano por examinador. Portanto, o aumento da acurácia tem o potencial de praticamente incluir um examinador a mais na equipe, sem os custos e o tempo de treinamento associados a contratação de um servidor público para cumprir a função. Principalmente, permitindo contribuir para a redução do déficit de examinadores sem aumentar os custos.

De acordo com o Plano de Ação 2022 do INPI (INPI, 2022a), estima-se uma tendência de alta no número de pedidos de patentes de residentes no Brasil. O plano considera um aumento de 15% na comparação de 2022 com 2021, elevando para 8.400 a quantidade de pedidos a serem distribuídos automaticamente entre as áreas de exame. Essa tendência de alta na quantidade de pedidos de patentes se justifica pela relação direta entre o avanço tecnológico e a proteção do investimento que financia esse avanço através das patentes.

Além da motivação com base na redução do desperdício existem também motivações de ordem operacional. Devido ao fato de a IPC ser atualizada anualmente, e pelo mapeamento da IPC para as áreas técnicas de exame ser dinâmico, é possível

[3] `https://www.gov.br/inpi/pt-br/servicos/patentes/plano-de-combate-ao-backlog`

[4] Identificado em conversa informal com examinador de uma das áreas que mais recebe pedidos distribuídos erradamente.

que uma IPC antes examinada pela área A, passe a ser examinada pela área B. Sendo assim, a solução de distribuição precisa ter a capacidade de aprender novamente. O e-Distribuidor havia sido desenvolvido externamente e utilizava tecnologia que não estava disponível no INPI, impossibilitando novo desenvolvimento e treinamento.

1.2 Problema

Quando não se tem uma forma analítica de solução, é possível aprender a partir dos dados. No caso específico desta pesquisa, o objetivo é enviar cada pedido de patente para a área correta de exame técnico, sendo essa tarefa chamada distribuição. Existe uma solução analítica para isso, que consiste na tabela "de-para" que fornece o mapeamento $\langle$ IPC[5] $\longrightarrow$ Área de Exame $\rangle$.

Porém, essa solução depende de ter a IPC do pedido de patente, e apenas os pedidos que já foram depositados em outro escritório de patente ao redor do mundo têm a possibilidade de já contarem com a classificação IPC no momento do seu depósito no INPI. Aproximadamente $\frac{1}{3}$ dos pedidos recebidos anualmente pelo INPI não têm essa informação, assim para esses pedidos, a solução analítica não se aplica.

Existem problemas de ciência de dados cujo objetivo é obter um modelo que permita prever comportamentos, FERNÁNDEZ *et al.* (2018, cap.1, p.12) os chamam de problemas preditivos. Em ambientes de Inteligência Artificial, esses problemas são chamados de aprendizado supervisionado, porque os dados contêm a resposta esperada, que define o comportamento a ser aprendido.

Ainda é possível analisar esses problemas conforme o tipo de variável que se pretende prever. Existem problemas onde é estudada uma variável contínua, por exemplo, a probabilidade de um paciente ter uma determinada doença, apesar de limitado entre 0 e 1, existem infinitos valores possíveis para a probabilidade. Ou variáveis categóricas que podem assumir um número limitado de valores discretos. As variáveis categóricas são estudadas nos problemas de classificação, enquanto as variáveis contínuas são estudadas nos problemas de regressão.

[5]International Patent Classification

Esta dissertação comunica o trabalho de pesquisa para resolver um problema de aprendizado supervisionado, para **prever uma variável categórica a partir de textos em linguagem natural**. A pesquisa buscou prever uma variável categórica a partir de textos dos pedidos de patentes em idioma português, portanto, **problema de classificação de textos, no domínio dos documentos de patentes em idioma português**. E propôs resolver o problema através do aprendizado de características dos textos presentes no título e no resumo dos pedidos de patentes previamente classificados com 1 categoria dentre 19 possíveis, dessa forma, o **problema de aprendizado, supervisionado, de máquina utilizando processamento de linguagem natural**. A distribuição dos exemplos do conjunto de dados entre as categorias, pode ser vista na figura 1.2.

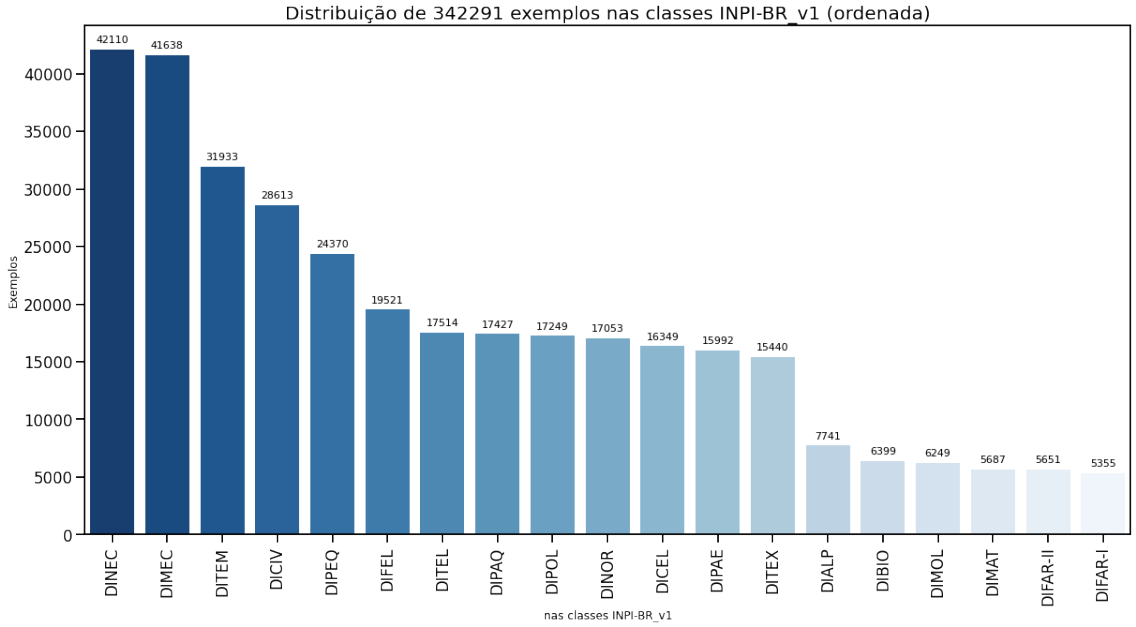

Figura 1.2: Distribuição das observações entre as classes do conjunto de dados

Para permitir sua reprodução parcial, este trabalho incluiu a construção do conjunto de dados. Esse conjunto é composto de textos relativos ao título e ao resumo dos pedidos de patentes publicados pelo INPI através da versão eletrônica da Revista da Propriedade Industrial (RPI)[6], em formato texto, desde 11/08/1992. A distribuição dos exemplos nos anos pode ser vista nas figuras 1.3 e 1.4, tanto com

[6] `http://revistas.inpi.gov.br/rpi/`

base no ano de classificação, como também com base no ano de depósito do pedido de patente.

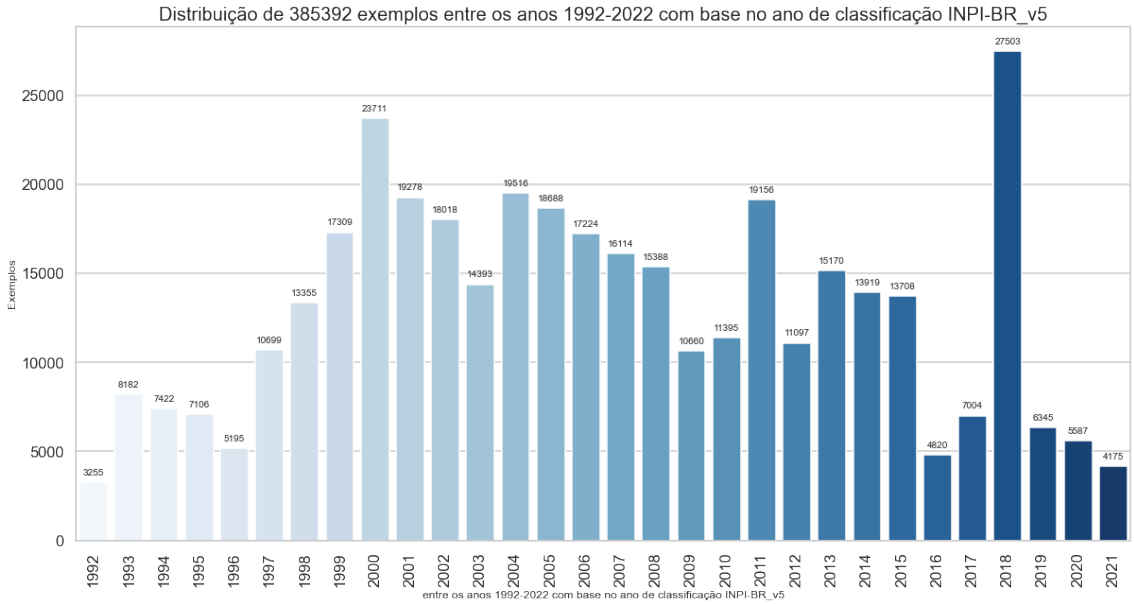

Figura 1.3: Distribuição das observações nos anos com base na data de classificação

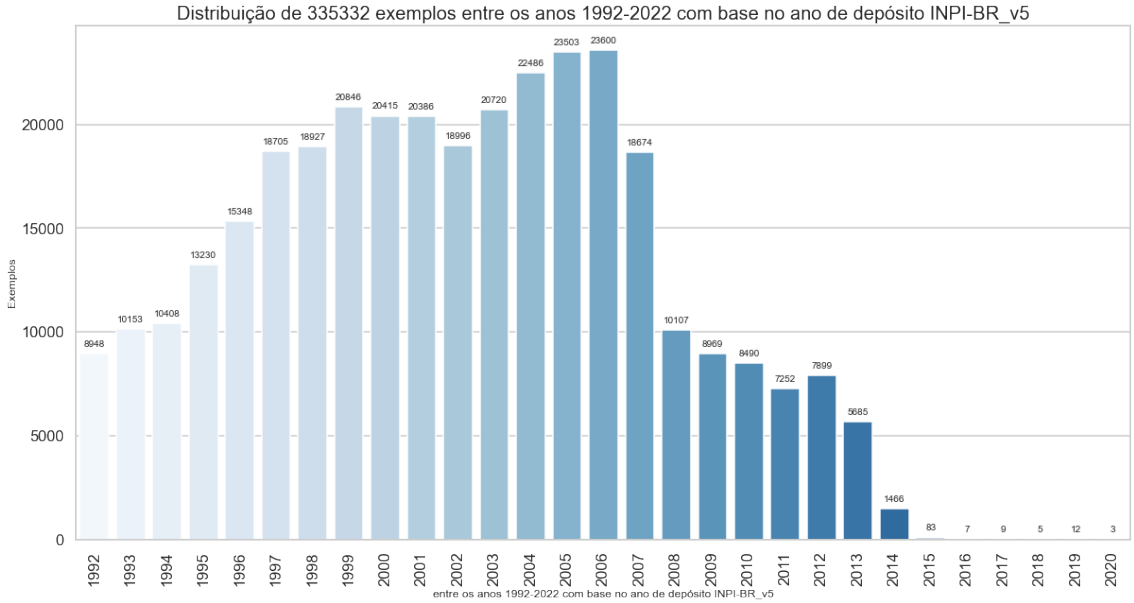

Figura 1.4: Distribuição das observações nos anos com base na data de depósito

O trabalho contribui para a área de classificação de textos ao comparar a acurácia média de vários algoritmos de aprendizado de máquina NBG, FERREIRA (2017, p.30); SVM, WU *et al.* (2007, p.10); MLP, ABU-MOSTAFA *et al.* (2012, p.7);

e BERT, DEVLIN *et al.* (2018); HOREV (2018), em diferentes configurações de hiper-parâmetros, aplicados à classificação de pedidos de patentes em português. E também por disponibilizar um conjunto de dados de patentes depositadas no INPI, que pode ser utilizado em outras pesquisas.

O objetivo do trabalho foi produzir um modelo de classificação com acurácia média estimada superior a acurácia média do e-Distribuidor (solução em uso no INPI quando foi iniciada esta pesquisa), para melhorar o desempenho de distribuição de pedidos de patentes entre as áreas de exame técnico do INPI. Os resultados da pesquisa foram utilizados para criar versões do artefato chamado "Application Router" no INPI.

1.3 Estrutura desse trabalho

Além desta Introdução, este trabalho está dividido em outros 4 capítulos. O capítulo 2 apresenta o básico dos conceitos utilizados para construir a solução e as referências para aprofundar o entendimento. O capítulo 3 apresenta o processo de pesquisa proposto. O capítulo 4 apresenta os resultados, uma breve discussão sobre os achados e os experimentos executados para selecionar, treinar e estimar a acurácia média esperada do modelo escolhido como substituto do e-Distribuidor. O capítulo 5 encerra essa dissertação apresentando a conclusão, as limitações, as dificuldades e os trabalhos futuros.

Capítulo 2

Revisão

Com o intuito de conhecer o estado da arte da área de classificação de textos, principalmente através de trabalhos focados na tarefa específica de classificação de pedidos de patentes, foi feita uma revisão, não sistemática, da literatura, ainda em 2017, através de busca livre pelos termos "patent" e "classification" nos sites Google Scholar[1] e Periódicos Capes[2].

Os artigos mais relevantes, segundo a ordenação feita pelos mecanismos de busca, e mais citados dentre eles foram revisados superficialmente. Primeiro identificados aqueles que tinham acesso aberto, ou permitido pela UFRJ, ou pelo INPI, depois através da leitura do resumo, em busca de trabalhos contendo termos como "IPC", "WIPO", "routing" ou expressões como "International Patent Classification" e "Patent Applications". O mesmo foi feito com a versão em português dos termos.

Buscas e filtragens adicionais foram feitas a partir das referências e orientações obtidas nas disciplinas cursadas na UFRJ/COPPE/PESC. Assim foi possível identificar vários artigos sobre o tema "Classificação de Patentes", a partir dos quais foram identificadas tecnologias, como: algoritmos, técnicas de extração de *features*, métricas, dentre outras.

Outra fonte inicial de pesquisa foi a documentação das bibliotecas utilizadas para desenvolver os programas de computador que implementam o classificador e os ex-

[1] `https://scholar.google.com`
[2] `https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br`

perimentos utilizados durante a pesquisa, sobretudo a documentação da biblioteca Scikit-Learn[3] para Python, principalmente as referências citadas nessa documentação.

Foi possível observar através das pesquisas que SEBASTIANI (2001, p.1) define *Text Categorization - TC, também chamado: text classification, ou topic spotting* (em tradução livre, algo como: Categorização de Texto, Classificação de Texto, ou Identificação de Tópicos), como a atividade de rotular textos de linguagem natural com categorias temáticas de um conjunto predefinido.

A partir daí estava identificada na literatura a área de pesquisa que abriga o problema abordado neste trabalho: **classificação de textos em linguagem natural**, e também sua sub-área focada em **documentos relativos à patentes**. Buscou-se entender os conceitos básicos envolvidos na tarefa, seguindo as referências citadas nos diversos artigos relevantes até alcançar um conceito onde fosse possível ancorar a compreensão e basear as escolhas feitas ao longo do trabalho. Ao longo do texto, houve um esforço para fazer todas as citações necessárias a correta indicação da autoria das ideias e também àquelas que foram julgadas relevantes para que um leitor interessado possa seguir com a pesquisa. Foram destacados os trabalhos relacionados onde ocorreu a obtenção de dados específicos relativos as métricas de desempenho que serão utilizadas na etapa de análise e discussão dos experimentos.

2.1 *Design Science Research Methodology*

Para DRESCH *et al.* (2014, p.67), um processo adequado à pesquisa com objetivo de construir artefatos que permitam a solução de problemas tanto no ambiente acadêmico quanto no ambiente corporativo é a *Design Science Research (DSR)*.

[3] `https://scikit-learn.org`

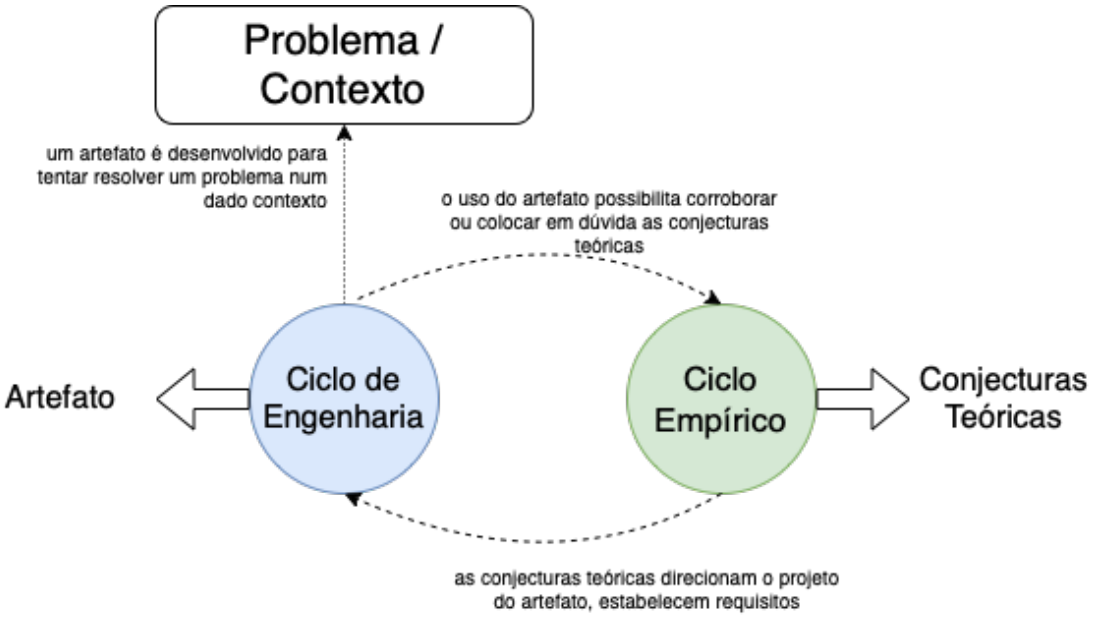

Figura 2.1: DSR em 2 ciclos, adaptado de DRESCH *et al.* (2014)

Para PIMENTEL *et al.* (2019, p.6), DSR é o modo de resolver um problema prático em um contexto específico, por meio de artefatos gerando novo conhecimento científico. A figura 2.1 apresenta a DSR com dois ciclos: o Ciclo de Engenharia, que tem o objetivo de projetar um artefato para solucionar um problema real; e o Ciclo Empírico, onde as conjecturas teóricas são formuladas e testadas.

HEVNER (2007) identifica, além dos ciclos de engenharia e empírico, o ciclo de relevância que liga a pesquisa ao contexto para o qual o artefato é projetado. A figura 2.2 apresenta a relação entre os 3 ciclos que compõem o ciclo completo de pesquisa DSR.

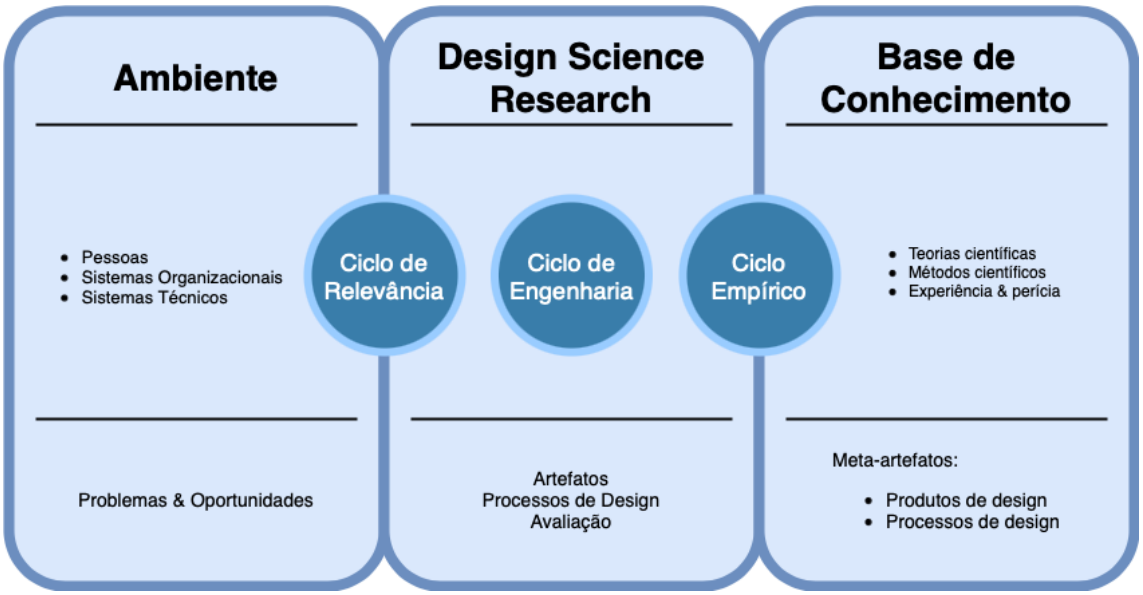

Figura 2.2: Ciclo completo de pesquisa DSR, adaptado de PIMENTEL *et al.* (2019)

PEFFERS *et al.* (2014, p.49) propõem uma metodologia composta de 6 atividades, baseada em DSR, que implementa um processo de pesquisa para produzir um artefato capaz de endereçar o problema original como uma solução relevante do ponto de vista do negócio, o *Design Science Research Methodology (DSRM)*, conforme figura 2.3.

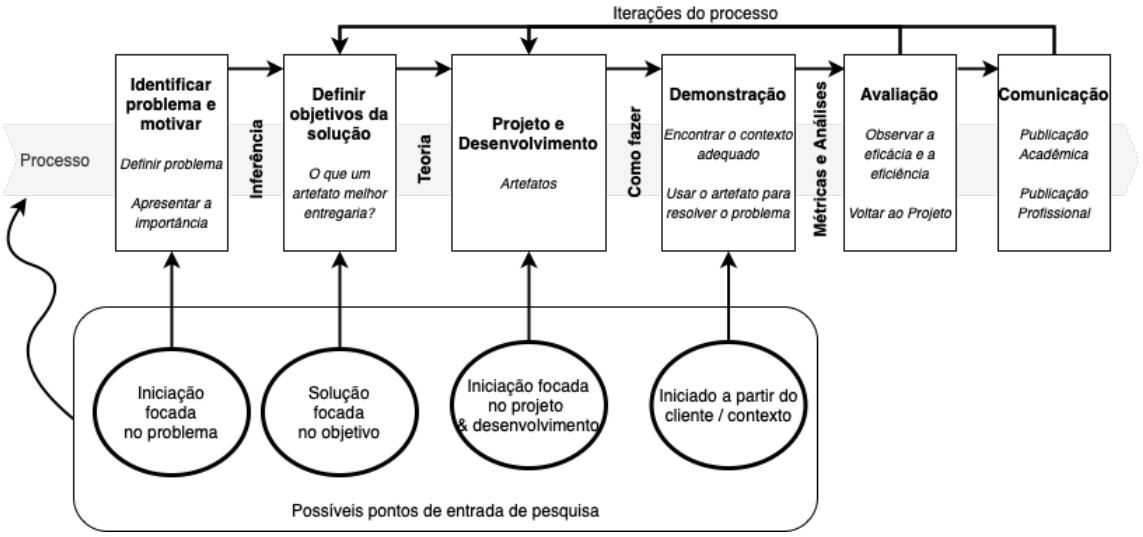

Figura 2.3: Modelo de Processo DSRM Adaptado de PEFFERS *et al.* (2014)

2.2 Ciência de Dados

Tanto os problemas de aprendizado não supervisionado, onde se pretende construir uma descrição da fonte de dados, como os problemas preditivos, ou de aprendizado supervisionado, segundo FERNÁNDEZ *et al.* (2018) podem sem abordados pela disciplina de Ciência de Dados.

FERNÁNDEZ *et al.* (2018) divide os problemas preditivos em problemas de classificação e de regressão. Nos problemas de regressão o objetivo é prever o comportamento de uma variável contínua, ou seja, que assume valores no domínio dos números reais. Já nos problemas de classificação o objetivo é prever o comportamento de uma variável discreta, ou seja, que assume valores dentro de um conjunto finito enumerável.

Ainda segundo FERNÁNDEZ *et al.* (2018), Ciência de Dados pode ser considerada uma disciplina para descobrir novos relacionamentos, padrões e tendências a partir de um grande volume de dados. Portanto, as suas técnicas têm o objetivo de descobrir, de maneira automatizada, o conhecimento contido na informação armazenada em grandes bancos de dados.

A tarefa de classificação de texto é uma das técnicas de Ciências de Dados apropriadas para descobrir conhecimento contido em dados, e o artigo de SEBASTIANI (2001) oferece uma excelente introdução à Classificação de Texto, apresentando conceitos e definições básicas para o entendimento da área, além de apresentar algumas referências importantes.

A tese fundamental pode ser encontrada em (MARON, 1961, p.405) e sugere que estatísticas baseadas no tipo, frequência, localização, ordem, dentre outros, de palavras selecionadas são adequadas para fazer previsões razoavelmente boas sobre o assunto de documentos contendo essas palavras.

Essas estatísticas podem ser utilizadas, conforme descrito por ALLAHYARI *et al.* (2017); DUMAIS *et al.* (1998); KOWSARI *et al.* (2019), para alimentar algoritmos de aprendizado de máquina que produzirão classificadores. Sendo assim, todos os algoritmos de aprendizado de máquina aplicados aos problemas preditivos de clas-

sificação buscam aprender, a partir de um conjunto de **vetores numéricos**, as *features*, uma função que consiga indicar corretamente a classe associada a vetores numéricos semelhantes.

2.2.1 Conjunto de Dados

Para HE e GARCIA (2009) a maioria dos algoritmos de aprendizado presume ou espera uma distribuição balanceada das observações entre as diversas classes a serem aprendidas. Quando são apresentados a conjuntos de dados desbalanceados, esses algoritmos tendem a apresentar acurácia muito diferente entre as classes.

Ainda para HE e GARCIA (2009, p.1264), apesar de tecnicamente qualquer conjunto de dados com diferente quantidade de observações/exemplos entre as classes poder ser chamado "desbalanceado", o entendimento geral é de que "dados desbalanceados" são aqueles com um desbalanceamento significativo ou extremo. De maneira geral, o desbalanceamento entre duas classes é considerado significativo a partir de 100:1, sendo comum desbalanceamentos da ordem de 1.000:1 e até 10.000:1.

2.3 Espaço Vetorial de Documentos

SEBASTIANI (2001, p.3) define *Text Categorization*, como a tarefa de associar um valor verdadeiro ou falso a cada par $\langle d_j, c_i \rangle \in \mathbb{D} \times \mathbb{C}$, onde $\mathbb{D}$ é um domínio de documentos e $\mathbb{C} = \{c1, ..., c_{|\mathbb{C}|}\}$ é um conjunto predefinido de categorias.

Para MANNING *et al.* (2009), um documento pode ser descrito na forma: $d \in \mathbb{X}$, onde $\mathbb{X}$ é o espaço de documentos; e um conjunto fixo de classes $\mathbb{C} = \{c_1, c_2, ..., c_j\}$. Tipicamente, o espaço de documentos $\mathbb{X}$ é algum tipo de espaço de alta dimensionalidade e as classes são derivadas da aplicação específica.

2.4 Extração de *Features*

Segundo FERNÁNDEZ *et al.* (2018); KOWSARI *et al.* (2019), *features* são os valores numéricos utilizados como entrada pelos algoritmos de aprendizado de máquina.

Divididas em **variáveis independentes** (X) as que representam o documento e **variável dependente** (y) a classe do documento, aquela que o classificador aprende a prever.

As *features* podem ser criadas a partir de informação que não faz parte, originalmente, do conjunto de dados como combinações/transformações de informações presentes nos dados originais, nesse caso o processo de criação é chamado Engenharia de *Features*.

Na classificação de documentos, as *features* não são os termos do vocabulário do idioma. O processo de conversão dos dados disponíveis no conjunto de dados de treinamento em *features* propriamente ditas é chamado de extração de *features*. O TF-IDF é um exemplo de método de extração de *features*, outros exemplos são a construção de uma matriz de presença, ou um matriz de frequência de termos; ou ainda a transformação de cada palavra em um vetor de números reais, como no caso dos *word embbedings*, como: Word2Vec (MIKOLOV *et al.*, 2013a,b) e GloVe (PENNINGTON *et al.*, 2014).

A *tokenização*, uma etapa do processo de extração de *features*, pode retirar as palavras consideradas frequentes no idioma, as chamadas *stop-words*, e incluir processos de lematização e extração de radical. Costuma ser antecedida por uma etapa de pré-processamento responsável por atividades como retirar os caracteres estranhos ao idioma e converter todo o texto para letras minúsculas. Esses processos são úteis para retirar o ruído e diminuir o número de dimensões da matriz de *features* X. As várias conjugações de um mesmo verbo e os plurais são exemplos de palavras que são agrupadas em menos dimensões.

Para KOWSARI *et al.* (2019), a etapa de extração de *features* pode ser feita de inúmeras formas. Para GOMEZ (2019), as técnicas TF-IDF e entropia produzem resultados similares, sendo que o TF-IDF é calculado mais rápido. GOMEZ (2019) também relata que tanto o TF-IDF quanto a entropia produzem resultados melhores que a representação *word embedding: Word2Vec*.

2.4.1 Bag-of-Words

Segundo KOWSARI *et al.* (2019), a forma mais básica de transformar um texto ou documento em um vetor numérico é utilizar o *Bag-of-Words - BoW*, que representa cada palavra que aparece no texto como uma dimensão em um vetor.

De acordo com DANISMAN e ALPKOCAK (2008) trata-se de um espaço onde cada dimensão de um documento $\vec{d}$ corresponde a um termo do vocabulário. Se um termo ocorre em um documento, então a posição do vetor correspondente àquele termo terá um valor diferente de zero. Considerando-se um vocabulário com n termos, então o vocabulário V é um conjunto ordenado de termos, definido como:

$$V = \{t_1, t_2, ..., t_n\} \tag{2.1}$$

então, um vetor de documento arbitrário, $\vec{d_i}$, é definido como:

$$\vec{d_i} = \langle w_{1i}, w_{2i}, ..., w_{ni} \rangle \tag{2.2}$$

onde w_{ki} representa o peso do k^{esimo} termo no documento i.

O BoW é chamado assim porque não leva em consideração nem a posição nem a vizinhança da palavra, apenas a sua presença, como se todas as palavras que compõem o texto fossem colocadas em um saco, em inglês *bag*. Segue exemplo abaixo.

Textos

0: 'a presente invenção se refere a um aparelho para balancear',

1: 'patente de invenção aparelho trata se de um sistema de',

2: 'patente de invenção aparelho um aparelho fornece energia a uma'

Vocabulário

{ 'aparelho', 'balancear', 'de', 'energia', 'fornece', 'invenção', 'para', 'patente', 'presente', 'refere', 'se', 'sistema', 'trata', 'um', 'uma' }

Vetores de presença de cada termo em cada texto

	aparelho	balancear	de	energia	fornece	invenção	para	patente	presente	refere	se	sistema	trata	um	uma
0	1	1	0	0	0	1	1	0	1	1	1	0	0	1	0
1	1	0	1	0	0	1	0	1	0	0	1	1	1	1	0
2	1	0	1	1	1	1	0	1	0	0	0	0	0	1	1

Figura 2.4: Matriz de presença

A informação de presença, mostrada na figura 2.4, pode ser enriquecida com a quantidade de vezes que cada palavra aparece no texto, criando assim a versão do BoW baseado na frequência dos termos, conforme a figura 2.5.

Vetores de frequência de cada termo em cada texto

	aparelho	balancear	de	energia	fornece	invenção	para	patente	presente	refere	se	sistema	trata	um	uma
0	1	1	0	0	0	1	1	0	1	1	1	0	0	1	0
1	1	0	3	0	0	1	0	1	0	0	1	1	1	1	0
2	2	0	1	1	1	1	0	1	0	0	0	0	0	1	1

Figura 2.5: Matriz de frequência

Ponderação

Agregando ainda mais informação ao vetor de características de cada texto, é possível substituir a simples informação da frequência do termo em um documento, pelo peso dele em relação a todo o conjunto de documentos. Sendo assim, um exemplo de técnica de ponderação é o TF-IDF, que divide uma parcela baseada na frequência de um termo em um dado documento, por uma parcela baseada na quantidade de documentos onde o termo aparece.

2.4.2 Term Frequency - Inverse Document Frequency

Multiplicar a frequência de um dado termo em um documento específico pelo inverso da frequência desse mesmo termo dentre todos os documentos de um conjunto, ainda é um dos melhores procedimentos de cálculo automático de pesos. Essa abordagem visa reduzir o peso dos termos mais frequentes dentre os documentos e que, portanto, são menos interessantes para diferenciar os documentos entre si. Entretanto, como mostram JONES (1972); SALTON *et al.* (1975); SALTON e BUCKLEY (1988), existem várias formas de calcular o $TF \times IDF$.

A fórmula clássica do $TF \times IDF$, conforme definida em DANISMAN e ALPKOCAK (2008); MANNING *et al.* (2009); SALTON e BUCKLEY (1988) é normalizada pelo cosseno:

$$tfidf(t_k, d_i) = \frac{tf_{ik}.\log\left(\frac{N}{n_k}\right)}{\sqrt{\left(\sum_{k=1}^{m}\left(tf_{ik}.\log\frac{N}{n_k}\right)^2\right)}} \tag{2.3}$$

onde:

$t_k = k^{esimo}$ termo em um documento d_i

tf_{ik} = frequência do termo t_k no documento d_i

n_k = frequência de t_k dentre os documentos do conjunto de dados

$idf = log(\frac{N}{n_k})$ é a parcela proporcional n_k

N = número total de documentos do conjunto de dados.

A biblioteca scikit-learn de PEDREGOSA *et al.* (2011) implementa a técnica de extração de *features* TF-IDF através da classe "sklearn.feature_extraction.text.TfidfVectorizer", e utiliza versões diferentes da fórmula de cálculo do $TF \times IDF$. A função conta com 2 parâmetros que ajustam a fórmula a ser utilizada: Para o IDF:

- smooth_idf = True $\rightarrow idf(t_k) = \log\frac{N+1}{n_k+1} + 1$, para prevenir divisão por zero, simula a existência de um documento contendo exatamente 1 vez cada um dos termos do vocabulário.
- smooth_idf = False $\rightarrow idf(t_k) = \log\frac{N}{n_k} + 1$

Para o TF:

- sublinear_tf = False $\rightarrow tf_{ik}$ = frequência do termo t_k no documento d_i
- sublinear_tf = True $\rightarrow tf_{ik} = 1 + \log(tf)$, ou seja, reduz a escala de tf e garante valor mínimo igual a 1, uma vez que o $\log(tf)$ não retorna valores negativos.

Os N documentos do conjunto de dados, eventualmente serão convertidos em um conjunto de vetores numéricos composto pelos elementos $\langle \vec{d}, c \rangle$, onde $\langle \vec{d}, c \rangle \in \mathbb{X} \times \mathbb{C}$. O sequenciamento da combinação $\langle$ representação do documento $\vec{d}$, classe c_i do documento $\rangle$ de cada documento do conjunto de dados cria a **matriz de treinamento** $\mathbb{D}$, que será utilizada como entrada pelos estimadores, também chamados de algoritmos de aprendizado.

2.4.3 Word Embeddings

Para (ABDELGAWAD *et al.*, 2020, p.5), *embbeding* é um espaço de poucas dimensões para onde você pode traduzir vetores de maior dimensionalidade. No contexto do Processamento de Linguagem Natural, *word embbedings* significa representar cada palavra através de um vetor de números reais que codificam o significado das palavras, sendo esperado que palavras próximas no espaço vetorial tenham significado similar.

2.4.4 Seleção de *Features*

Segundo MANNING *et al.* (2009), o vetor $\vec{d}$ resultante do método de extração de *features* escolhido tende a ter um número muito elevado de dimensões (todos os termos do vocabulário). Reduzir o número de dimensões do espaço de documentos pode ser útil para acelerar o processo de aprendizado e também para reduzir o ruído. Uma *feature* com ruído é uma *feature* que quando adicionada à representação do documento (vetor de variáveis independentes) aumenta o erro de classificação de novos documentos.

MANNING *et al.* (2009) define a seleção de *features* como o processo de escolha do subconjunto das *features* que será oferecido aos algoritmos de aprendizado de máquina. O algoritmo básico para seleção de *feature* calcula uma medida de utilidade $A(t, c)$ para cada termo do vocabulário e seleciona os k termos com o maior valor de A.

2.5 Redução de Dimensionalidade

A quantidade de palavras em um conjunto de documentos pode ser muito maior que a quantidade de documentos, gerando vetores com muitas dimensões. Sendo assim, a matriz resultante da concatenação de todos os vetores calculados para representar cada documento, pode ter dimensões muito elevadas, da ordem de milhares de colunas.

Reduzir a dimensão da matriz que alimentará o algoritmo de aprendizado de máquina pode ser interessante. Quando é utilizada a técnica de decomposição em valores unitários, em inglês *Singular Value Decomposition*, a matriz resultante fica ordenada com os valores mais relevantes para diferenciar os documentos nas primeiras colunas, permitindo que escolha um conjunto muito menor de colunas, mas que tem capacidade de separar muito bem os documentos, possibilitando assim, que os algoritmos de aprendizado de máquina aprendam com menos dados, e potencialmente em menos tempo. Em contraposição, a matriz fica muito mais densa, geralmente com densidade 1, e o projetista tem então que balancear tempo de cálculo do SVD, ganho e perda de informação, tamanho da matriz resultante e complexidade do algoritmo usado.

2.5.1 *LSA - Latent Semantic Analysis*

A ideia por trás da Análise de Semântica Latente, em inglês *Latent Semantic Analysis - LSA*, é mapear cada documento em um espaço com menos dimensões.

De acordo com MANNING *et al.* (2009), o modelo Bag-of-Words é incapaz de tratar dois problemas clássicos em linguagem natural: sinonímia e polissemia. A representação falha em capturar o relacionamento entre termos sinônimos, cada um é representado como uma dimensão separada no vetor $\vec{d}$. E na polissemia, quando um mesmo termo tem significados diferentes, o modelo captura todos os significados na mesma dimensão. O LSA ajuda a amenizar esses problemas porque geralmente é escolhido um valor de k da ordem de poucas centenas. E quando forçado a comprimir a representação dos documentos em um espaço k-dimensional, o SVD deve reunir

termos com co-ocorrências semelhantes. Esta intuição sugere que a qualidade da representação não deve sofrer muito com a redução de dimensionalidade, inclusive pode até melhorar.

CARDOSO-CACHOPO e OLIVEIRA (2003) informa que o LSA começa com uma matriz de termos por documento X que é decomposta utilizando a técnica de Decomposição em Valores Unitários, em inglês *Singular Value Decomposition - SVD*, no produto de 3 outras matrizes: $X = T_0 S_0 D_0$, de forma que T_0 e D_0 têm colunas ortogonais e S_0 é diagonal. T_0 e D_0 são as matrizes de vetores unitários da esquerda e da direita e S_0 é a matriz diagonal de valores unitários. Se os valores singulares em S_0 estiverem ordenados (e as linhas e colunas em T_0 e D_0 forem permutadas corretamente), os k maiores valores poderão ser mantidos e os demais zerados. O produto das matrizes resultantes é a matriz $\hat{X}$ que é aproximadamente igual a X e terá ordem igual a k. Idealmente, escolhe-se um valor de k grande o bastante para capturar toda a informação relevante dos dados, e pequena o suficiente para não distorcer dos dados.

Segundo DEERWESTER *et al.* (1990), idealmente o que se quer é um k que seja grande o suficiente para capturar toda estrutura relevante da matriz de termos por documento, mas não grande demais que comece a modelar o ruido ou detalhes irrelevantes dos dados. Como escolher o melhor número de dimensões no LSA é uma questão aberta à pesquisa.

2.6 Algoritmos de Aprendizado (Estimadores)

Os algoritmos de aprendizado de máquina, também chamados de estimadores, serão utilizados para encontrar um classificador (estimador treinado), uma função de

classificação γ que mapeie documentos para classes:

$$\gamma : \mathbb{X} \rightarrow \mathbb{C} \tag{2.4}$$

Para MANNING *et al.* (2009), aprendizado supervisionado é chamado assim porque um supervisor (o especialista humano que define as classes dos documentos) funciona como um professor dirigindo o processo de aprendizado. O algoritmo, ou estimador, ou método de aprendizado, Γ recebe o conjunto de treinamento $\mathbb{D}$ como entrada e retorna a função de classificação, ou classificador, γ.

$$\Gamma(\mathbb{D}) = \gamma \tag{2.5}$$

Ainda segundo MANNING *et al.* (2009), o classificador γ costuma receber o mesmo nome do algoritmo/estimador Γ, por exemplo, Naive Bayes (NB) é um algoritmo de aprendizado Γ, mas ao afirmar que "O Naïve Bayes teve uma taxa de erro de 20%", está sendo descrito um experimento no qual um classificador NB em particular, γ (que foi produzido pelo algoritmo de aprendizado NB), teve uma taxa de erro de 20% naquela aplicação.

Aprende-se a partir de dados quando não existe uma forma analítica de solução, mas tem-se dados a partir dos quais é possível produzir uma solução empírica.

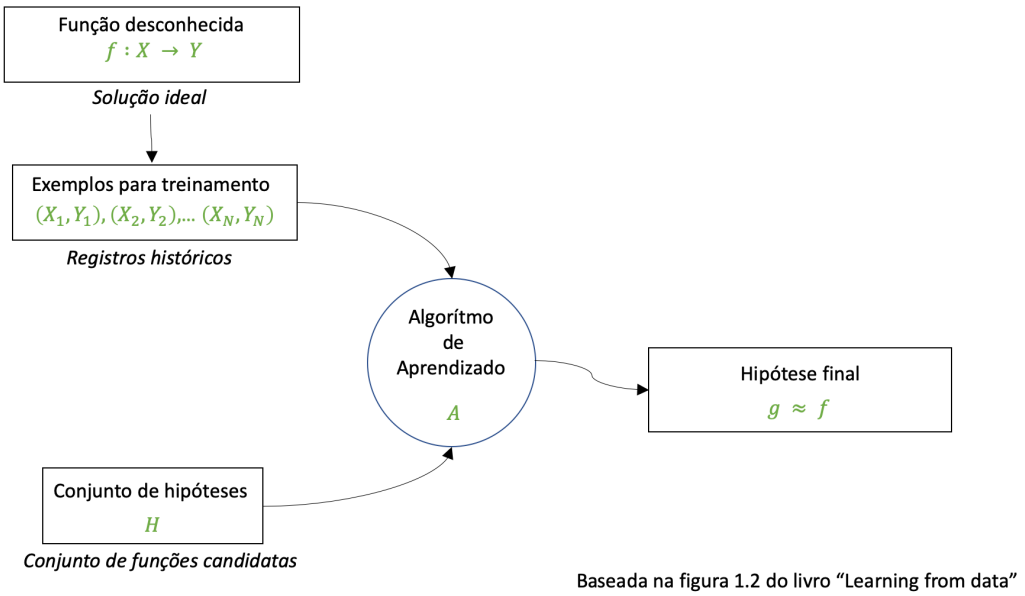

Figura 2.6: O Problema do Aprendizado

ABU-MOSTAFA *et al.* (2012) define os componentes principais do "Problema do Aprendizado", conforme a figura 2.6, como:

- a entrada X (representação, *features*, dos documentos);
- a função alvo desconhecida $f : \mathbb{X} \rightarrow \mathbb{Y}$ (classificação ideal), onde $\mathbb{X}$ é o espaço da entrada (*features*), e $\mathbb{Y}$ é o espaço da saída (conjunto das classes);
- o conjunto de dados $\mathbb{D}$ com exemplos de entrada e saída $(X_1, y_1), ..., (X_N, y_N)$, onde $y_n = f(X_n)$ para todo $n = 1, ..., N$ (são os dados históricos disponíveis, as observações, ou, simplesmente, os exemplos);
- o algoritmo de aprendizado A, que usa o conjunto de dados $\mathbb{D}$ para escolher uma função $g : \mathbb{X} \rightarrow \mathbb{Y}$ que aproxima f. O algoritmo seleciona g a partir de um conjunto de candidatas, chamado conjunto das hipóteses $\mathbb{H}$

Em um problema específico de aprendizado, tanto a função alvo quanto o conjunto de exemplos são dados do problema, enquanto é escolhido o "Modelo de Aprendizado", composto pelo algoritmo de aprendizado e o conjunto de hipóteses específica do algoritmo escolhido.

Esse conjunto de hipóteses de cada algoritmo é fruto do ajuste dos seus hiper-parâmetros, ou seja, como cada algoritmo tem um conjunto diferente de hiper-parâmetros e existem inúmeras combinações para os valores de cada hiper-parâmetro de cada algoritmo, existem inúmeros "classificadores" possíveis, ou hipóteses de classificadores.

O trabalho de treinamento envolve uma etapa de aprendizado específica de cada algoritmo, que ocorre após a escolha dos valores de seus hiper-parâmetros. Esse aprendizado é sempre binário, ou seja, o classificador decide se um determinado exemplo pertence ou não a uma classe.

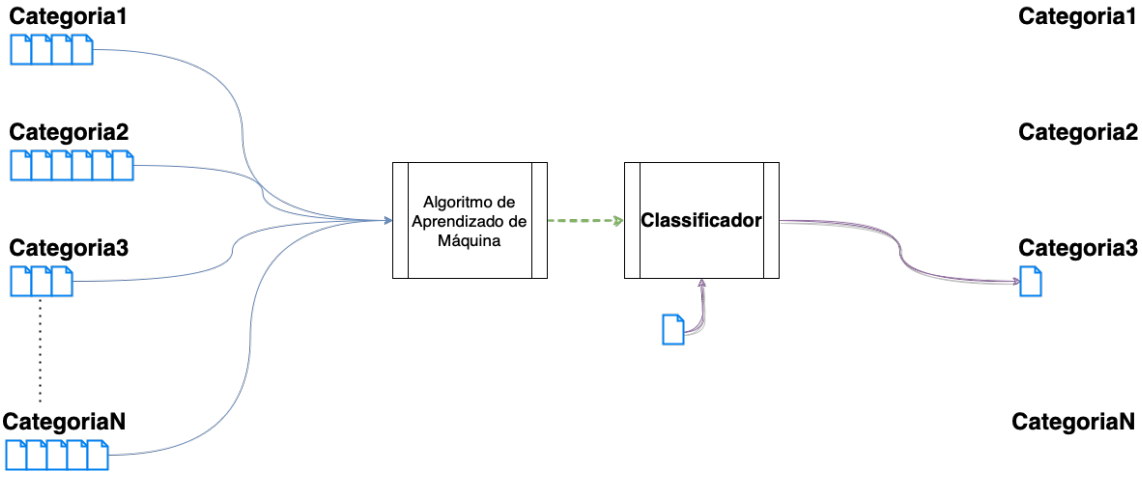

Figura 2.7: Processo automatizado de categorização de patentes baseado em aprendizado de máquina, inspirado na Fig.1 de (MATHIASSEN e ORTIZ-ARROYO, 2006, p.1040)

2.6.1 Problemas multi-classe

Os problemas de classificação com múltiplas classes são resolvidos a partir de classificadores binários. Uma abordagem para utilizar classificadores binários para resolver problemas multi-classe é considerar todos os exemplos em classes diferentes da classe para a qual o classificador está sendo treinado como exemplos negativos, e os exemplos da classe como positivos, essa abordagem recebe o nome de *One-vs-All (OvA)* ou *One-vs-Rest (OvR)*, e produz um modelo composto por um classificador para cada classe aprendida.

Outra abordagem é chamada *One-vs-One (OvO)*, onde é construído um classificador para cada par de classes, produzindo assim um modelo com muito mais classificadores que a abordagem OvA/OvR.

O número de classificadores é igual ao número de classes n, na abordagem OvA/OvR. E $(n * (n-1))/2$ na abordagem OvO.

Sendo assim, um problema com 10 classes, quando resolvido pela abordagem:

- OvA/OvR: produz 10 classificadores diferentes;
- OvO: produz 45 classificadores diferentes;

Nas duas abordagens, uma previsão passa pela avaliação do exemplo em questão por todos os classificadores do modelo e depois por uma forma de decisão sobre que classe escolher para o exemplo. Na abordagem OvA/OvR cada classificador atribui uma probabilidade de o exemplo em questão pertencer àquela classe (um número real entre 0 e 1), e a decisão é baseada na maior probabilidade. Na abordagem OvO, cada classificador do modelo será considerado um voto e o exemplo pertencerá a classe mais votada. Em caso de empate é a implementação que determina como proceder.

As abordagens também diferem quanto aos dados utilizados no treinamento de cada classificador do modelo, na abordagem OvA/OvR os n classificadores serão treinados com todo o conjunto de dados. Mas na abordagem OvO, o conjunto de dados de treinamento de cada classificador vai variar conforme o par de classes.

2.6.2 Naïve-Bayes

A classificação Naïve Bayes, conforme FALL e BENZINEB (2002), é um técnica probabilística simples, onde a probabilidade de um documento em particular pertencer a uma dada classe é baseada na premissa de que as palavras que compõem o documento formam uma distribuição independente entre si. Daí a utilização da expressão *"Naïve" (ingênuo)*, pois as palavras em um documento não costumam

ser totalmente independentes, o que, na prática, não impede o funcionamento da classificação.

De acordo com MANNING *et al.* (2009), a probabilidade de um documento d pertencer a classe c é:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{2.6}$$

onde: $P(t_k|c)$ é a probabilidade condicional do termo t_k ocorrer em um documento da classe c. Interpreta-se $P(t_k|c)$ como uma medida de quanto t_k contribui para c ser a classe correta. $P(c)$ é a probabilidade anterior de um documento pertencer a classe c. Se um termo do documento não fornecer clara indicação para o documento pertencer a uma classe, é escolhida a classe que tem a maior prioridade anterior. $\langle\ t_1, t_2, ..., t_{n_d}\ \rangle$ são os termos em d que são parte do vocabulário usado para a classificação e n_d é o número desses termos em d. O objetivo é encontrar a "melhor" classe para o documento. No classificador NB, a melhor classe para o documento é a classe mais provável ou *maximum a posteriori (MAP)* c_{map}:

$$c_{map} = \arg\max_{c \in C} \hat{P}(c|d) = \arg\max_{c \in C} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c) \tag{2.7}$$

Não é conhecido $P(c)$ nem $P(t_k|c)$, estima-se a partir do conjunto de dados de treinamento, por isso é chamado $\hat{P}$. Na equação (2.7), várias probabilidades condicionais são multiplicadas, uma para cada posição $1 \leq k \leq n_d$. Isso pode gerar valores muito pequenos, por isso o mais comum é somar logaritmos de probabilidades ao invés de multiplicar as probabilidades. A classe com o maior log continua sendo a mais provável; além disso o logaritmo é uma função monotônica $log(xy) = log(x) + log(y)$, portanto, a maioria das implementações do NB calcula:

$$c_{map} = \arg\max_{c \in C} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)] \tag{2.8}$$

2.6.2.1 Naïve-Bayes Gaussiano

A *sci-kit learn* de PEDREGOSA *et al.* (2011) implementa o algoritmo de aprendizado **Naïve-Bayes Gaussiano (NBG)**, através da função *sklearn.naive_bayes.GaussianNB*, que assume que a distribuição de probabilidade das *features* é gaussiana, onde:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.9}$$

com os parâmetros σ_y e μ_y estimados utilizando a máxima probabilidade.

2.6.2.2 Naïve-Bayes Bernoulli

A *sci-kit learn* de PEDREGOSA *et al.* (2011) implementa o algoritmo de aprendizado **Naïve-Bayes Bernoulli (NBB)**, através da função *sklearn.naive_bayes.BernoulliNB*, que assume valores binários para as *features* e tem regra de decisão baseada em:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \tag{2.10}$$

que difere da regra utilizada pela implementação multinomial do Naive-Bayes, uma vez que penaliza a não ocorrência de uma *feature* i que é indicativa para a classe y, enquanto a implementação multinomial apenas ignora a não existência da *feature*.

2.6.3 Decision Tree

De acordo com a documentação da *sci-kit learn* de PEDREGOSA *et al.* (2011) é um método de aprendizado supervisionado não paramétrico utilizado tanto para classificação como para regressão. O objetivo é criar um modelo que estime o valor de uma variável alvo aprendendo regras simples de decisão a partir dos dados. Uma aproximação visual do método é uma árvore de decisão. Em Python, o algoritmo de aprendizado é implementado através da função *sklearn.tree.DecisionTreeClassifier*.

2.6.4 Random Forest

Trata-se de um conjunto de árvores de decisão calculados sobre amostras dos dados, obtidas de maneira aleatória e com reposição. A *sci-kit learn* de PEDREGOSA *et al.* (2011) implementa o algoritmo de aprendizado **Random Forest (RDF)**, através da função *sklearn.ensemble.RandomForestClassifier*.

2.6.5 Stochastic Gradient Descendent

Trata-se de uma método de otimização. A *sci-kit learn* de PEDREGOSA *et al.* (2011) implementa o algoritmo de aprendizado **Stochastic Gradient Descendent (SGD)**, através da função *sklearn.linear_ model.SGDClassifier*. O algoritmo itera sobre as observações de treinamento e para cada uma atualiza os parâmetros do modelo de acordo com a regra de atualização:

$$w \leftarrow w - \eta\Big[\alpha\frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w}\Big] \tag{2.11}$$

onde η é a taxa de aprendizado que controla o tamanho do passo no espaço de parâmetros.

2.6.6 Support Vector Machine

Support Vector Machine (SVM) (LI e BONTCHEVA, 2008) é um algoritmo que não exige ajuste de parâmetros nem a seleção de termos, segundo FALL e BENZINEB (2002). O objetivo do algoritmo é encontrar uma superfície de decisão no espaço de todos os possíveis documentos que melhor separe os documentos relevantes para uma dada categoria. Para isso, o SVM foca nos *outliers*, aqueles documentos que estão nos limites do espaço que delimita a categoria. O SVM pode suportar vocabulários grandes com facilidade, por isso, talvez não precise de seleção de termos.

De acordo com CHEN e CHANG (2012), o SVM apresenta desempenho muito bom na classificação de documentos de patentes em inglês e alemão. Conforme relatou FALL *et al.* (2004), em geral os classificadores baseados no SVM têm capacidade

de generalização em dados desconhecidos (palavras que não existiam no conjunto de dados de treinamento) melhor que algoritmos de aprendizado baseados em distância e similaridade, como kNN e árvores de decisão.

Segundo GOMEZ e MOENS (2014), o SVM também pode suportar um elevado número de *features*, sendo possível não utilizar redução de dimensionalidade de maneira satisfatória. Originalmente, o SVM foi definido para classificação binária, mas ELISSEEFF e WESTON (2002) informam que existem modificações que permitem a sua utilização em problemas multi-classe e multi-label.

Na *sci-kit learn* de PEDREGOSA *et al.* (2011), o SVM é implementado de várias maneiras diferentes. Um exemplo é a classe *sklearn.svm.LinearSVC*[4], que implementa a fórmula:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} max(0, y_i(w^T \phi(x_i) + b)), \tag{2.12}$$

onde ϕ é a função identidade.

Outro exemplo de implementação é o da classe *sklearn.svm.SVC*[5], conforme formulação: dados vetores de treinamento $x_i \in \mathbb{R}^p$, $i = 1, ..., n$ e um vetor $y \in \{1, -1\}^n$, o objetivo é encontrar $w \in \mathbb{R}^p$ e $b \in \mathbb{R}$ de modo que a previsão dada por $sign(w^T \phi(x) + b)$ esteja correta para a maioria dos exemplos. O SVC resolve o

[4]https://scikit-learn.org/stable/modules/svm.html
[5]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

problema primário:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$\text{subject to} y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$

$$\zeta_i \geq 0, i = 1, ..., n$$

E o problema secundário:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

$$\text{subject to} y^T \alpha = 0$$

$$0 \leq \alpha_i \leq C, i = 1, ..., n$$

onde e é o vetor de 1 e Q é uma matriz $n \times n$, onde $Q_i j \equiv y_i y_j K(x_i, x_j)$, e $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ é o núcleo. Os termos α_i são chamados coeficientes duais e têm como limite superior C. Uma vez que o problema de otimização está resolvido, a saída da 'função de decisão' para um exemplo x se torna:

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b, \tag{2.13}$$

a classe prevista corresponde ao sinal.

2.6.7 Redes Neurais

Uma rede neural artificial (RNA) é uma rede de unidades simples, posicionadas em camadas sucessivas, como na figura 2.8. FALL e BENZINEB (2002) explicam que a ideia é simular redes biológicas, as unidades são conectadas e cada conexão tem um peso associado, como os neurônios e suas sinapses.

Nas RNAs um nível da rede recebe as entradas na forma de uma coleção de pesos, um para cada termo, representando um documento. Por exemplo, esses pesos

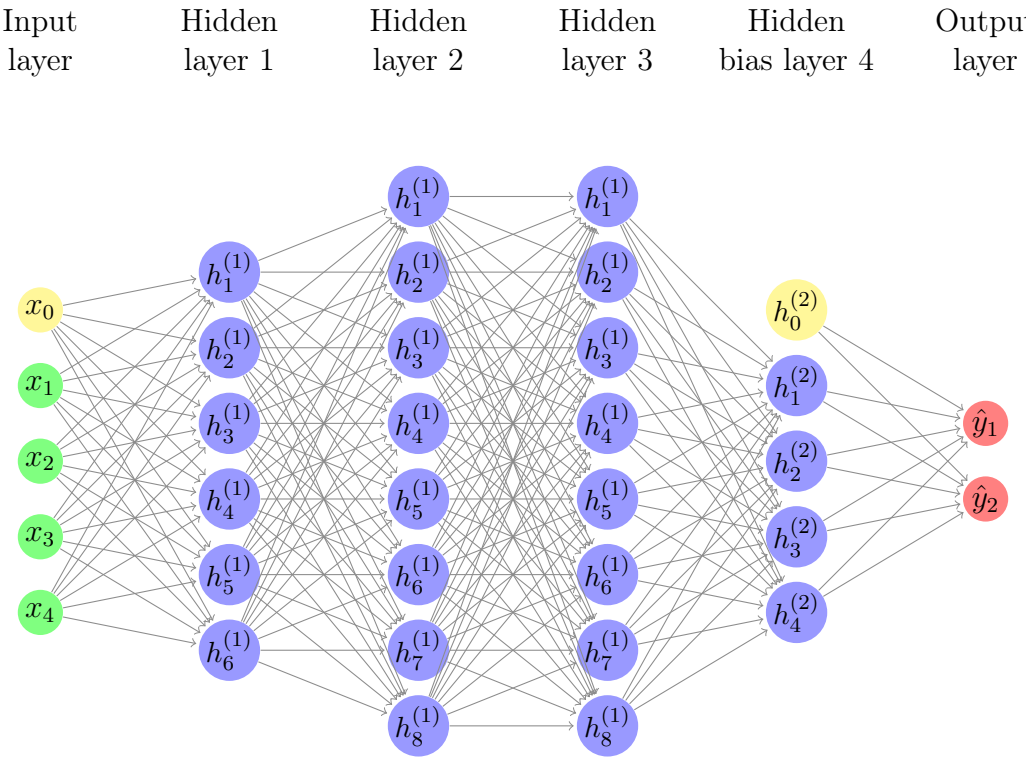

Figura 2.8: Modelo conceitual de RNA

são processados pelos demais níveis da rede e o último nível irá apresentar a saída da rede, que nos problemas de classificação, será a sugestão de categoria.

2.6.7.1 Multi-layer Perceptron (MLP)

Como qualquer Rede Neural Artificial (RNA), o Multi-layer Perceptron é uma rede de unidades simples, posicionadas em camadas sucessivas, por isso o nome Multi-layer.

Seja $\mathbb{X} = \mathbb{R}^d$ o espaço de entrada, onde $\mathbb{R}^d$ é um espaço Euclidiano d-dimensional, e seja $\mathbb{Y} = \{+1, -1\}$ o espaço de saída, de uma decisão binária. O vetor $\mathbf{x} \in \mathbb{R}^d$ é composto pelas *features*, e y é uma saída binária indicando o pertencimento ou não a uma classe específica. A função $h(x)$, escolhida dentre todas as hipóteses $h \in \mathbb{H}$, fornece pesos diferentes para cada coordenada de x, refletindo a importância relativa de cada uma para a decisão. Estes pesos são combinados e o resultado é comparado

com um limite para que seja escolhido o valor de y.

$$+1 \longleftarrow \sum_{i=1}^{d} w_i x_i > threshold,$$
$$-1 \longleftarrow \sum_{i=1}^{d} w_i x_i < threshold$$

De forma mais compacta:

$$h(x) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) + b\right) \tag{2.14}$$

onde $x_1, ..., x_d$ são os componentes do vetor x; $h(x) = +1$ significa que pertence a classe e $h(x) = -1$ significa que não pertence a classe; $sign(s) = +1$ if $s > 0$ e $sign(s) = -1$ if $s < 0$.

A *sci-kit learn* de PEDREGOSA *et al.* (2011) implementa o algoritmo de aprendizado Multi-layer Perceptron, através da classe *sklearn.neural_network.MLPClassifier*, que aprende utilizando *Backpropagation.*

2.6.8 Redes Neurais Profundas

Segundo ABDELGAWAD *et al.* (2020, p.2), a comunidade de Processamento de Linguagem Natural (PLN) experimentou grandes avanços na classificação de textos, impulsionado pelo desenvolvimento de modelos de redes neurais recorrentes e convolucionais. Esses modelos são mais comumente aplicados à análise de sentimentos e classificação de perguntas em textos curtos (twitter e comentários sobre filmes), como em KALCHBRENNER *et al.* (2014), eles não costumam ser utilizados em documentos longos como patentes.

MINAEE *et al.* (2021) e BEKAMIRI *et al.* (2021) destacam que as Redes Neurais Recorrentes, em inglês *Recurrent Neural Networks (RNN)*, têm dificuldades como a necessidade do processamento serial de vários parâmetros de uma só vez, tornando ineficaz o uso de hardware de processamento paralelo, como GPUs e TPUs. E as

Redes Neurais Convolucionais, na sigla em inglês *Convolutional Neural Networks (CNN)*, têm dificuldades para manipular dados sequencias e contextuais.

2.6.9 Transformers

VASWANI *et al.* (2017) propõem os "Transformadores" (*Transformers*) como um modelo de arquitetura que evita a recorrência adotando um mecanismo de atenção para definir dependências globais entre entrada e saída, permitindo muito mais paralelização.

2.6.9.1 BERT

Conceitualmente simples e experimentalmente poderoso, o *Bidirectional Encoder Representations from Transformers (BERT)* (DEVLIN *et al.*, 2018), é um modelo de representação de linguagem baseado em ajuste-fino.

DEVLIN *et al.* (2018) dizem que o MLM, do inglês *Masked Language Model*, do BERT aleatoriamente mascara alguns tokens da sequência de entrada e aprende a prever a identificação deles no vocabulário, baseando-se exclusivamente no contexto, ou seja, nos tokens ao redor do token mascarado. O BERT pode aprender não só olhando o contexto à esquerda, mas também o contexto à direita, por isso o bidirecional do nome. Adicionalmente, o MLM do BERT implementa a "previsão da sentença seguinte", portanto, o BERT foi preparado para representar inequivocamente uma única sentença e um par de sentenças em uma sequência de tokens. Para ABDELGAWAD *et al.* (2020, p.4) e BEKAMIRI *et al.* (2021, p.2), a característica de avaliar o contexto de forma bidirecional do BERT permite a ele "entender" o contexto muito bem.

O BERT é dividido em dois passos: pré-treinamento e ajuste-fino. Durante o pré-treinamento o MLM é executado em textos comuns do idioma que se quer modelar. E no ajuste-fino o BERT é primeiro inicializado com os parâmetros calculados no pré-treinamento, e então esses parâmetros são atualizados utilizando novos textos específicos da tarefa para a qual está sendo feito o ajuste fino.

A arquitetura do BERT conta com codificadores bidirecionais multi-camadas baseados na implementação original de transformadores de VASWANI *et al.* (2017). O número de camadas é representado pelo hiper-parâmetro L, a dimensão das camadas escondidas é representada pelo hiper-parâmetro H e o números de cabeças de autoatenção é representado pelo hiper-parâmetro A.

Originalmente foram produzidos dois modelos, tanto o $BERT_{BASE}$ (L=12, H=768, A=12 $\rightarrow$ 110M de parâmetros) quanto o $BERT_{LARGE}$ (L=24, H=1024, A=16 $\rightarrow$ 340M de parâmetros) foram pré-treinados utilizando textos do BooksCorpus ZHU *et al.* (2015) e da Wikipedia, ambos em inglês.

DEVLIN *et al.* (2018, p.4) dizem que para o BERT uma **sentença** é uma extensão contígua de texto de tamanho arbitrário, não uma frase linguística real. E uma **sequência** refere-se a sequência de tokens utilizada como entrada do BERT, que pode ser relativa a uma única sentença ou a duas sentenças agrupadas. O BERT usa embeddings WordPiece (WU *et al.*, 2016), com um vocabulário de 30.000 tokens. O primeiro token de toda sequência é sempre um token especial de classificação ([CLS]), e o estado final correspondente a esse token é usado como representação para as tarefas de classificação. Pares de sentenças são agrupadas em uma única sequência, e as sequências são diferenciadas de duas formas: primeiro são separadas com um token especial ([SEP]); segundo, é adicionado um embedding aprendido a todos os tokens indicando se ele pertence a sentença A ou a sentença B. Para um dado token sua representação é construída somando-se o seu próprio embedding (WordPiece) com os correspondentes embeddings de segmento (A ou B) e de posição (índice na sequência de tokens).

2.7 Métricas

FALL *et al.* (2003) compararam vários classificadores treinados para classificar documentos de patentes em inglês e alemão, utilizando o WIPO-alpha e o WIPO-de de FALL e BENZINEB (2002), introduzindo um conjunto de métricas para essa comparação. Outros trabalhos utilizaram os mesmos conjuntos de dados para expe-

rimentar diversos algoritmos de aprendizado de máquina CHEN e CHANG (2012); GOMEZ e MOENS (2014); MATHIASSEN e ORTIZ-ARROYO (2006); QIU *et al.* (2011); SENEVIRATNE *et al.* (2015); SILLA e FREITAS (2011) especialmente o WIPO-alpha, sempre com foco na classificação de patentes segundo a Classificação Internacional de Patentes (CIP), IPC na, sigla em inglês.

A tarefa de classificação pode produzir quatro tipos de resultados:

- Verdadeiro Positivo (VP) - dada uma categoria qualquer dentre as possíveis, quando ela é escolhida pelo classificador, e o item pertence à categoria escolhida;
- Verdadeiro Negativo (VN) - dada uma categoria qualquer dentre as possíveis, quando ela não é escolhida pelo classificador, e o item não pertence à categoria escolhida;
- Falso Positivo (FP) - dada uma categoria qualquer dentre as possíveis, quando ela é escolhida pelo classificador, e o item não pertence à categoria;
- Falso Negativo (FN) - dada uma categoria qualquer dentre as possíveis, quando ela não é escolhida pelo classificador, e o item pertence à categoria;

Seguindo a definição dada por YANG e LIU (1999, p.43), estas são as medidas mais utilizadas para avaliar a efetividade de classificadores:

- **Recuperação**: para uma dada categoria, razão entre a quantidade de itens corretamente classificados e o total de itens daquela categoria;

$$\frac{VP}{VP + FN} \tag{2.15}$$

- **Precisão**: razão entre a quantidade de itens corretamente classificados e todos os itens classificados;

$$\frac{VP}{VP + FP} \tag{2.16}$$

- **Acurácia**: razão entre a quantidade total de itens classificados corretamente e o total de itens;

$$\frac{VP + VN}{VP + FP + VN + FN} \tag{2.17}$$

- **F1-score**: combina Recuperação (R) e Precisão (P) com pesos iguais;

$$F_1(R, P) = \frac{2RP}{R + P} \tag{2.18}$$

- **Matriz de confusão**: compara graficamente a previsão de um classificador, apresentando linha a linha, coluna a coluna, a previsão do classificador para cada classe comparada com a classe verdadeira. O classificador perfeito teria uma matriz de confusão com valores apenas na diagonal que indica a previsão correta na classe verdadeira.

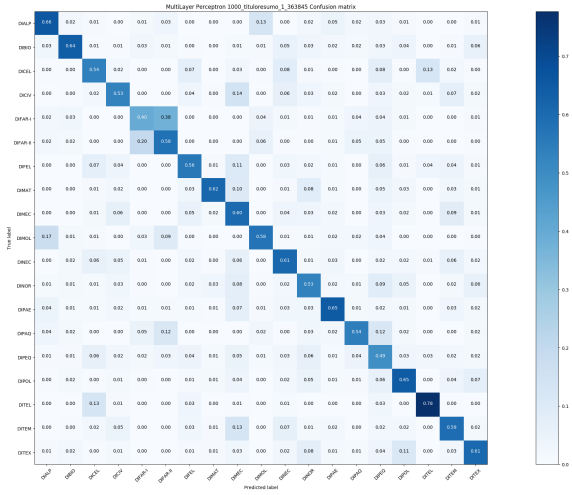

Figura 2.9: Exemplo de matriz de confusão

As medidas podem ser calculadas primeiro para as decisões binárias em cada categoria individual e depois ter a média geral calculada. Ou podem ser calculadas globalmente sobre todas as $n \times m$ decisões binárias, onde n é número total de itens classificados e m é o número total de categorias possíveis. O primeiro modo de cálculo é chamado média-micro e o último é chamado de média-macro. Ainda é possível calcular a média ponderada pela quantidade de observações de cada classe, essa é a média balanceada.

2.8 Trabalhos Relacionados

A revisão da literatura mostrou que pelo menos desde LARKEY (1999, p.183) o *TF-IDF* é utilizado para a representação de documentos de patentes, e também que a acurácia esperada para classificadores k-NN, em um problema com 76 classes de uma coleção de patentes em inglês, estava entre 25% e 32%, de acordo com LARKEY (1999, p.186).

Segundo KOSTER *et al.* (2001), o *European Patent Office (EPO)* também precisa distribuir entre os seus examinadores de patentes os pedidos que o escritório recebe. KOSTER *et al.* (2001) comparou algoritmos de recuperação de informação Rocchio e Winnow (KOSTER *et al.*, 2003), treinados a partir de um conjunto de 68.000+ pedidos de patentes contendo texto completo e resumo, e distribuídos em 44 classes. Sendo que o pré-processamento aplicado aos textos foi apenas substituir as letras maiúsculas por minúsculas e substituir os caracteres especiais por espaço, não sendo utilizada nenhuma técnica de lematização nem de eliminação de *stop words*, mas considerando apenas as palavras que aparecem pelo menos três vezes no pedido. Para estimar o desempenho do classificador foi utilizada a métrica F1-score, que ficou perto de 60% quando o treinamento utilizou o texto completo, e perto de 50% quando o treinamento utilizou o resumo.

A Organização Mundial da Propriedade Intelectual (OMPI), através de FALL e BENZINEB (2002), publicou uma revisão da literatura focada nos problemas específicos da classificação automática de patentes. Essa publicação teve como ponto de partida uma iniciativa da OMPI chamada *The CLAIMS project* e seu objetivo era identificar o estado da arte no momento de sua publicação, para subsidiar a criação de um "categorizador" com objetivo de sugerir 3 a 4 subclasses da classificação internacional de patentes que resultassem em uma acurácia entre 80 - 90%, após a decisão final de um humano dentre as sugestões do classificador. Esse trabalho tinha o propósito de funcionar nos idiomas Inglês e Francês.

O processo geral apresentado na figura 2.10, baseado na imagem utilizada em (FALL e BENZINEB, 2002, p.9), mostra os passos e opções da classificação.

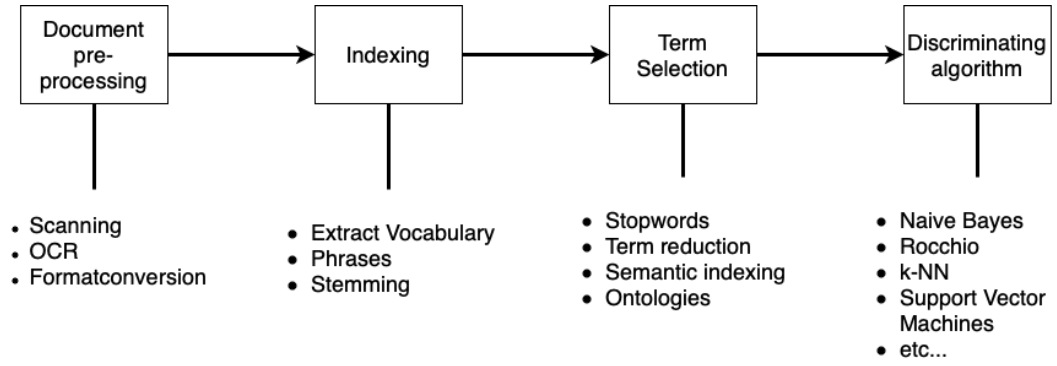

Figura 2.10: adaptado da figura 1 de FALL e BENZINEB (2002) - Passos e opções do processo de classificação

A implementação de um classificador automático conta com uma etapa de treinamento, durante a qual um conjunto de exemplos classificados manualmente é apresentado aos algoritmos de classificação seguindo os passos apresentados na figura 2.10. E uma etapa de teste, ou avaliação do desempenho, onde o classificador é utilizado para indicar a classe dos exemplos que não lhe foram apresentados durante a etapa de treinamento, e, em seguida, avaliando se a classificação indicada pelo classificador coincide com a classificação manual.

O artigo de FALL e BENZINEB (2002) reconhece que é difícil extrapolar a efetividade de um algoritmo de classificação de um corpus e conjunto de categorias para outro. Apenas experimentos controlados utilizando o mesmo corpus e o mesmo conjunto de categorias podem ser usados para identificar a acurácia relativa dos algorítimos. Mas, em geral, os estudos demonstram que SVM e k-NN desempenham melhor que as redes neurais e que todos são mais acurados que as abordagens Roccio e Naïve-Bayes.

De acordo com FALL e BENZINEB (2002), o Escritório Europeu de Patentes *European Patent Office (EPO)* fez o estudo mais completo sobre software de categorização de patentes. O objetivo era construir um roteador de pedidos de patentes para o time técnico de especialistas que inicia a análise dos pedidos.

Segundo FALL e BENZINEB (2002), no EPO, os especialistas estavam divididos em 44 diretorias e 549 times, cada um responsável por processar um grupo de símbolos *International Patente Classification (IPC)*. A acurácia da distribuição ma-

nual feita pelo pessoal administrativo foi anteriormente medida como sendo 81.2%, o objetivo do classificador automático era superar essa acurácia.

Apesar do trabalho do EPO, em 2002, a pesquisa acadêmica sobre técnicas de categorização concentrava-se em dados genéricos da Reuters, de grupos de troca de mensagens (newsgroups) ou coleções de resumos médicos. A dificuldade de obter conjuntos de dados de patentes fez a comunidade acadêmica não utilizar esse tipo de dado em pesquisas de categorização de texto, conforme conclui FALL e BENZINEB (2002).

Para superar essa dificuldade de obter conjunto de dados específico de patentes FALL *et al.* (2003) apresentam o conjunto de dados WIPO-alpha, que consiste em pedidos de patente submetidos a OMPI sob o Tratado de Cooperação em Patentes, PCT na sigla em inglês *Patent Cooperation Treaty*, entre 1998 e 2002. No mesmo trabalho apresenta o seguinte esquema para avaliar o sucesso na tarefa de classificação das patentes segundo a IPC:

- *top-prediction*: compara a previsão principal do classificador com o símbolo IPC principal da patente
- *three-guesses*: compara as três primeiras previsões do classificador com o símbolo IPC principal da patente
- *all-categories*: compara a previsão principal do classificador com os três primeiros símbolos IPC da patente

MATHIASSEN e ORTIZ-ARROYO (2006) combinaram diversos algoritmos de aprendizado para categorizar pedidos de patentes, utilizando diferentes formas de representação para os pedidos presentes no conjunto de dados WIIPO-alpha modificado, e concluíram que a melhor combinação obteve f1-score de 0.8667, representando 6,51% de melhora sobre o melhor classificador individual, que atingiu f1-score de 0.8137.

Dentre os trabalhos direcionados para a classificação utilizando a IPC, uma outra abordagem é estudada por QIU *et al.* (2011), aproveitando-se do caráter hierárquico

da classificação internacional de patentes. Eles demonstraram que a abordagem deles era competitiva utilizando métricas como f1-score e acurácia. Outro trabalho que trata da abordagem hierárquica é o de SILLA e FREITAS (2011), que faz um revisão do tema.

CHEN e CHANG (2012) utilizaram a acurácia para avaliar o método *Three Phase Categorization - TPC*, baseado no SVM e no k-NN, que obteve 36,07% de acurácia no nível do subgrupo da IPC, utilizando o conjunto de dados WIPO-alpha.

GOMEZ e MOENS (2014) fazem uma extensa revisão dos trabalhos de classificação de patentes indicando parâmetros como o algoritmo, o conjunto de dados e a métrica de avaliação utilizada, concluindo que não há padronização nem na metodologia nem nos resultados, ficando destacada a ausência de um padrão de avaliação tanto para os dados quanto para as métricas. Sendo comum que os trabalhos não apresentem os detalhes usados nas suas implementações.

SENEVIRATNE *et al.* (2015) apresentam uma abordagem baseada em assinatura para representar os documentos que se mostrou competitiva e bastante escalável e eficiente, uma vez que o treinamento, utilizando o texto do resumo de um conjunto de dados com mais de 1.300.000 exemplos (USPTO 2006-2013), levou pouco mais de 1 minuto, sendo que o tempo de predição ficou abaixo de 1 segundo.

Está disponível no site da OMPI[6] um conjunto de documentos que conta a história da categorização automática de texto no contexto da classificação internacional de patentes. Desde 2002, a OMPI desenvolve o produto IPCCAT, fruto do projeto CLAIMS, com foco no suporte à aplicação de símbolos IPC a textos com o resumo dos pedidos de patentes.

Conforme o relatório de FIÉVET (2018), o desempenho desse classificador não é medido com base na acurácia, o desempenho vem sendo medido utilizando-se a precisão, calculada com base na previsão de três classes, das quais basta que uma esteja correta. Seguindo essa estratégia de medição o IPCCAT atingiu níveis de precisão da ordem de 80% em 2018, utilizando aproximadamente 8.000 redes

[6]https://www.wipo.int/classifications/ipc/en/ITsupport/Documentation/categorization.html

neurais para isso. O relatório também informa que a utilização de n-gram teve efeito colateral.

Em outra apresentação, FIÉVET e GUYOT (2018) revelam que os campos de texto utilizados para o desenvolvimento são o título e o resumo, e a expectativa que redes neurais profundas não tragam benefícios sobre as redes neurais tradicionais.

FIÉVET e GUYOT (2018) ainda revelam que o texto completo do pedido de patente só foi avaliado no experimento inicial em 2003. Durante todos esses anos a OMPI construiu não só o conjunto de dados WIPO-alpha, mas também outras versões como a gama e a delta. Porém, em 01 de junho de 2022, uma consulta ao site que oferecia esses conjuntos de dados[7] mostrou que a iniciativa foi descontinuada em 2021.

BEKAMIRI *et al.* (2021) aplicam o estado da arte em matéria de aprendizado profundo e processamento de linguagem natural, os modelos de linguagem pré-treinados, à tarefa de classificação de patentes tanto utilizando a IPC, quanto utilizando a CPC. Fizeram um ajuste fino do SBERT utilizando textos de patentes em inglês e obtiveram 66,48% de f1-score e 58% de acurácia.

FERREIRA (2021) usou um conjunto de dados produzido e disponibilizado pelo INPI de Portugal para explorar algoritmos tradicionais de aprendizado de máquina e também metodologias de transferência de aprendizado, baseadas em aprendizado profundo, na tarefa de classificação de patentes em português, no contexto da Classificação Internacional de Patentes, até o segundo nível, totalizando 124 categorias. Os melhores resultados foram obtidos utilizando o LinearSVC com TF-IDF e o BERTimbau, sendo que o desempenho do BERTimbau foi 4% superior ao desempenho do LinearSVC.

[7]https://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/index.html

Capítulo 3

Proposta

Este trabalho utiliza a adaptação da DSRM de PEFFERS *et al.* (2014) descrita na figura 3.1 para comparar o desempenho de vários classificadores baseados em aprendizado de máquina em busca de uma solução que permita construir um artefato que supere o desempenho do e-Distribuidor[1].

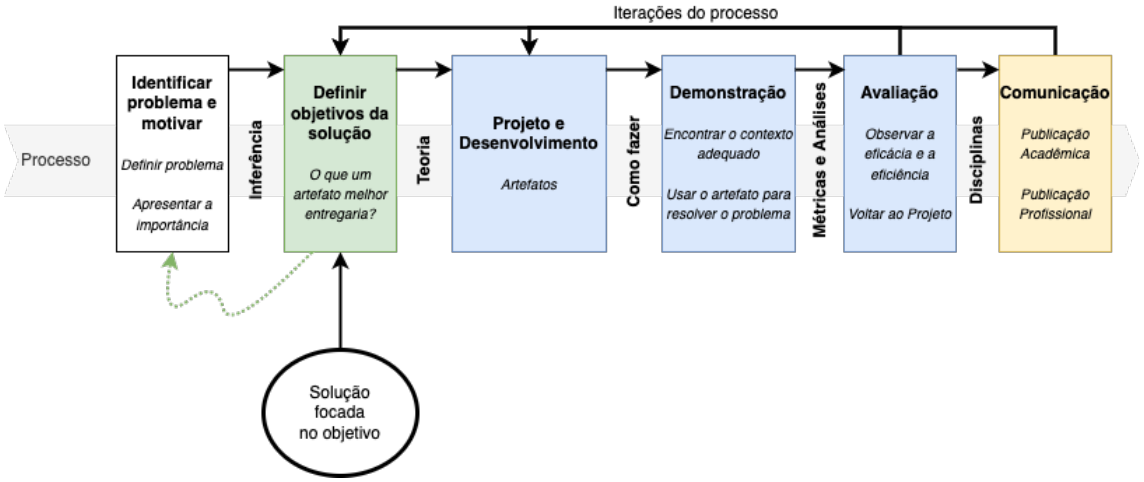

Figura 3.1: Modelo de Processo DSRM Adaptado

Iniciando pela etapa **"Solução focada no objetivo"** do método, executa dois ciclos de pesquisa durante os quais são investigadas conjecturas teóricas apresentadas nos mapas das figuras 3.2 e 3.4, respectivamente ciclo 1 e ciclo 2, para orientar a construção de artefatos que superem o desempenho do e-Distribuidor. Utiliza dados históricos dos pedidos já distribuídos para treinar diversos classificadores,

[1]solução de distribuição automática dos pedidos de patentes, em uso no INPI, quando esta pesquisa foi iniciada.

com diversas configurações de hiper-parâmetros a partir de diferentes recortes dos dados, tratados com diferentes técnicas de pré-processamento e extração de *features*.

Tendo como referência as abordagens definidas por FALL *et al.* (2003) (*top-prediction*; *three-guesses*; *all-categories*), adota a abordagem *top-prediction* para a comparação entre algoritmos. Utilizando a acurácia média balanceada como principal métrica de comparação e tomada de decisão, valendo-se de outras métricas para aprofundar a análise e permitir a comparação com outros trabalhos.

3.1 Ciclo 1 - Prova de Conceito

3.1.1 Objetivo da solução

O objetivo deste ciclo é demonstrar a possibilidade de aumentar a acurácia média da distribuição dos pedidos de patentes além dos 30% do e-Distribuidor, investigando as conjecturas apresentadas no mapa da figura 3.2.

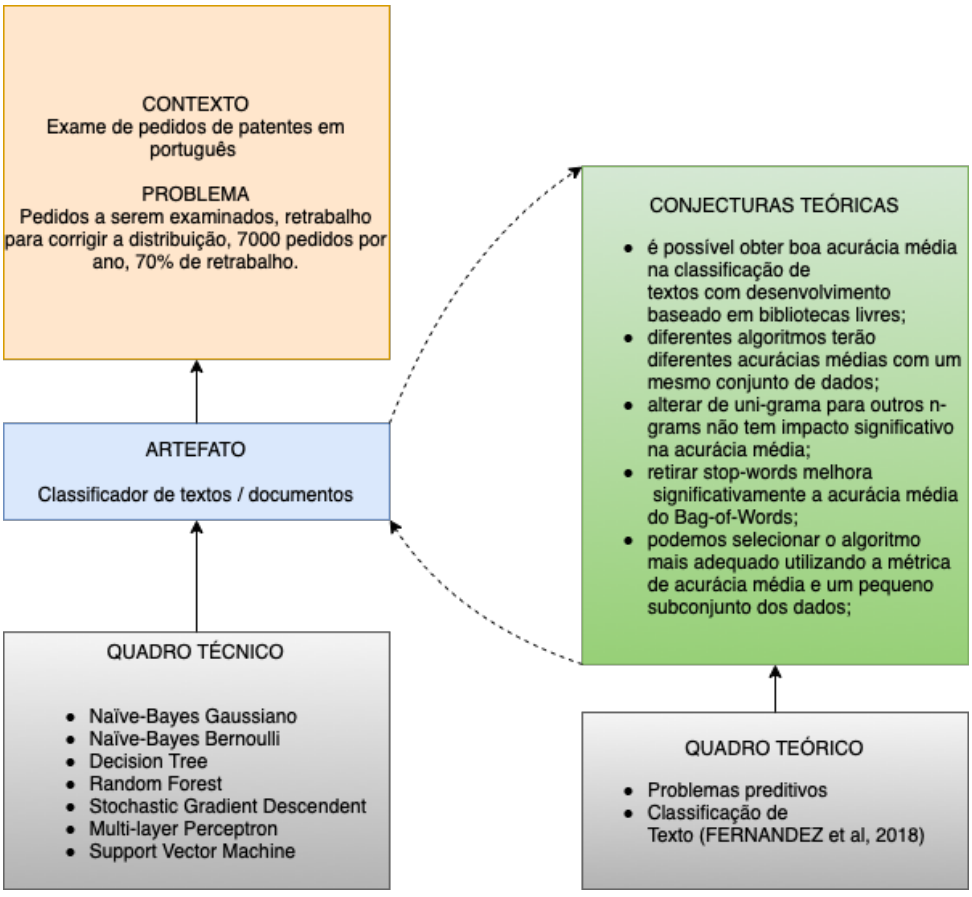

Figura 3.2: Mapa do primeiro ciclo de pesquisa DSRM

3.1.2 Desenvolvimento

A etapa de "Projeto e Desenvolvimento" se utiliza de ferramentas como a revisão da literatura, registrada no capítulo 2, e a experimentação através de programação em Python para executar as atividades relativas aos ciclos de engenharia e empírico, orientados conforme o passo a passo apresentado na figura 3.3.

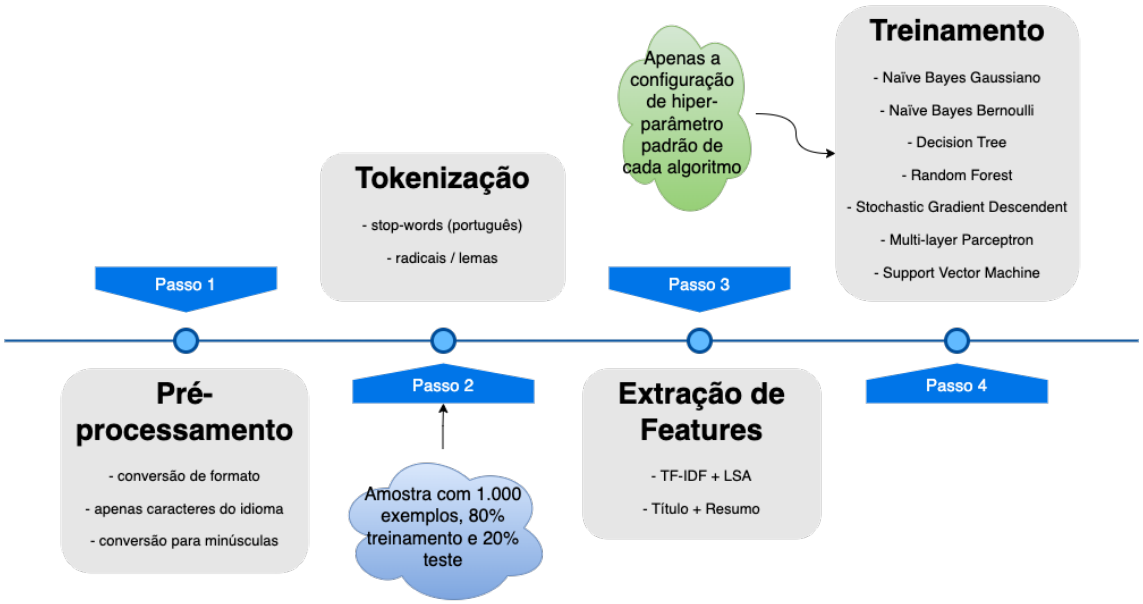

Figura 3.3: Passo-a-passo para a construção dos classificadores do ciclo 1

3.1.2.1 Conjunto de dados

O conjunto de dados utilizado neste ciclo conta com dados internos do INPI, que não estarão disponíveis para a comunidade acadêmica. Cada exemplo/observação conta com `título ; resumo ; data de depósito ; área de exame`.

3.1.2.2 Extração de *features*

Este experimento utiliza todo o texto disponível no conjunto de dados: título e resumo dos pedidos.

Optou-se por extrair as *features* através de TF-IDF, seguida da aplicação de técnica de redução de dimensionalidade por decomposição em valores singulares,

resultando em um LSA calculado separadamente para os campos título e resumo, respectivamente com 20 e 30 dimensões.

Partindo do conjunto de dados original, retira os valores nulos e os valores duplicados; converte todo o texto para letras minúsculas; aceita apenas caracteres do idioma português; considera apenas as palavras com 2 letras ou mais; e substitui as palavras por seus radicais.

3.1.2.3 Treinamento

O processo de extração de *features* descrito acima será aplicado apenas em uma amostra de 1.000 observações do conjunto de dados original. E as *features* serão utilizadas para ranquear os algoritmos: NBG, NBB, DCT, RDF, SGD, MLP e SVM. O treinamento utilizará 80% das observações e os testes para aferir a acurácia média balanceada serão executados com 20% das observações.

3.1.3 Demonstração

Dois algoritmos selecionados serão treinados com o conjunto de dados completo. E com mais dimensões no LSA, nessa etapa as dimensões do título serão reduzidas para 200, e as dimensões do resumo serão reduzidas para 800, totalizando 1.000 dimensões que formarão, por concatenação, a matriz de *features* X.

3.1.4 Avaliação

Os 2 algoritmos treinados com o conjunto de dados completo vão produzir 2 classificadores, e aquele que apresentar a melhor acurácia média estimada fará parte do artefato "Application Router v1". Esse artefato será avaliado por experimento *"in vivo"*, conforme definido por TRAVASSOS e BARROS (2003), tendo sua acurácia média aferida em situação real de uso.

3.2 Ciclo 2 - Melhorar o desempenho

3.2.1 Objetivo da solução

Neste ciclo ocorre um avanço na pesquisa e são identificadas formas de otimizar o desempenho do artefato produzido como solução para o problema da distribuição de pedidos de patentes. É seguida a investigação das conjecturas do ciclo anterior e adicionadas outras, apresentadas no mapa da figura 3.4.

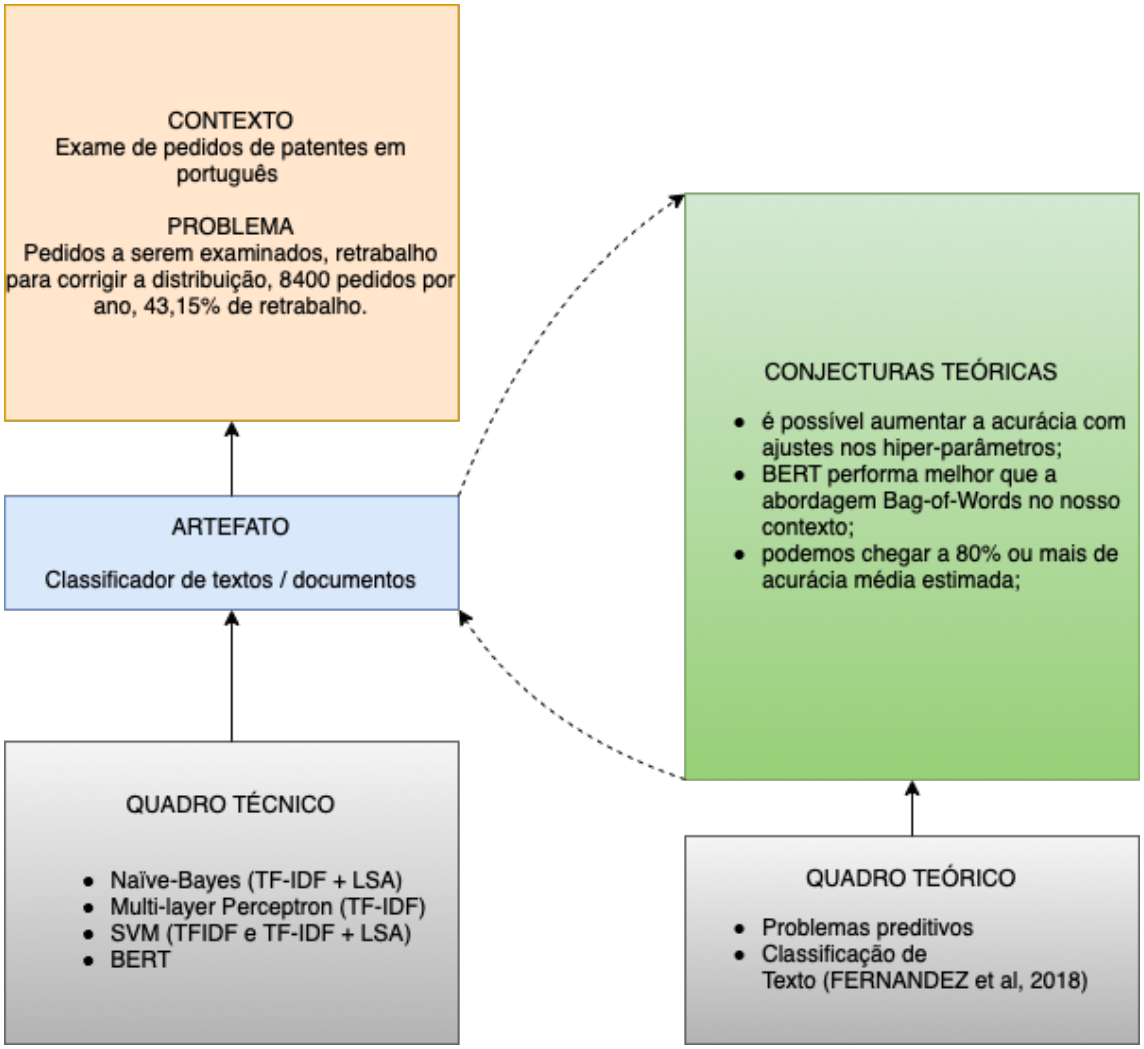

Figura 3.4: Mapa do ciclo #2 de pesquisa DSRM

3.2.2 Desenvolvimento

A etapa de "Projeto e Desenvolvimento" se utiliza de ferramentas como a revisão da literatura, registrada no capítulo 2, e a experimentação através de programação em

Python para executar as atividades relativas aos ciclos de engenharia e empírico, orientados conforme o passo a passo apresentado na figura 3.5.

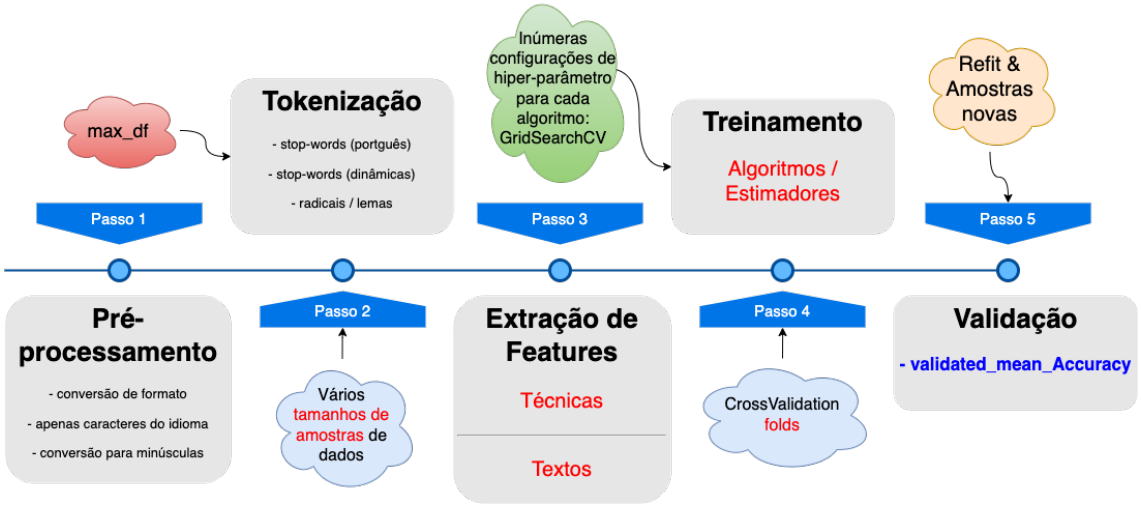

Figura 3.5: Passo-a-passo para a construção dos classificadores do ciclo 2

3.2.2.1 Conjunto de dados

Utiliza o trabalho executado durante a disciplina UFRJ/COPPE/PESC/COS738[2] NUNES *et al.* (2017), adaptando o processo de construção do conjunto de dados para obter dados públicos de pedidos de patentes do INPI e associar a eles uma classe (área de exame), a partir da classificação IPC e de uma tabela "de-para"[3] fornecida pelo INPI. Esse conjunto de dados fica como contribuição para a comunidade acadêmica para futuros experimentos e evolução do próprio conjunto de dados[4].

3.2.2.2 Extração de *features*

Partindo do conjunto de dados original, retira os valores nulos e os valores duplicados; converte todo o texto para letras minúsculas; e aceita apenas caracteres do idioma português. Desconsidera as *stop words* do idioma português e substitui as palavras por seus radicais.

[2] `https://github.com/rafaelscnunes/COS738-AutomaticPatentClassification`
[3] O INPI autorizou a divulgação da tabela
[4] `https://github.com/rafaelscnunes/app-router/tree/main/dataset`

Como parte da investigação para identificar formas de otimizar o desempenho, será feita uma análise exploratória dos dados (AED) para selecionar o texto a ser utilizado na tarefa de extração de *features* (matriz X).

3.2.2.3 Treinamento

O treinamento será feito através do esquema de validação cruzada, no qual todo o conjunto de dados é utilizado como dado de treinamento e também de testes, utilizando-se várias rodadas com amostras de teste e treinamento que são alteradas a cada rodada, até todas as observações terem sido usadas tanto para treinar como para testar o classificador.

Para avaliar a conjectura sobre a possibilidade de aumentar a acurácia com ajustes nos hiper-parâmetros, além do esquema de validação cruzada, será usada uma função implementada na biblioteca Sci-kit Learn (PEDREGOSA *et al.*, 2011) pela classe sklearn.model_selection.GridSearchCV que executa o *Cross Validation* a partir de um conjunto pré-definido de hiper-parâmetros, identificando o ajuste que obtém o melhor desempenho. Ao final do processo o classificador (ajuste de hiper-parâmetros) que tiver atingido o melhor desempenho (acurácia) é treinado com todo o conjunto de dados.

Para aferir o desempenho esperado, utiliza-se uma etapa de validação, com dados não utilizados nas fases anteriores. Isso é possível porque mesmo a maior amostra de dados (20.000 observações) representa menos de 10% do total de observações disponíveis no conjunto de dados completo. Será utilizada a acurácia média balanceada como métrica de desempenho para comparar os classificadores e identificar o melhor para nosso trabalho.

A etapa de desenvolvimento do ciclo 2 será divida em 3 experimentos:

Experimento *Features*, cujo objetivo é identificar o texto e o ajuste de seleção dinâmica de stop-words (max_df) que produz o melhor desempenho. Segue os passos da figura 3.6 para avaliar, em amostras reduzidas do conjunto de dados, se existe diferença significativa de desempenho quando utilizados apenas o título em

comparação com o uso apenas do resumo, ou da combinação dos dois. Também avalia se existe diferença significativa do desempenho quando retiradas as palavras que se repetem mais vezes nos documentos.

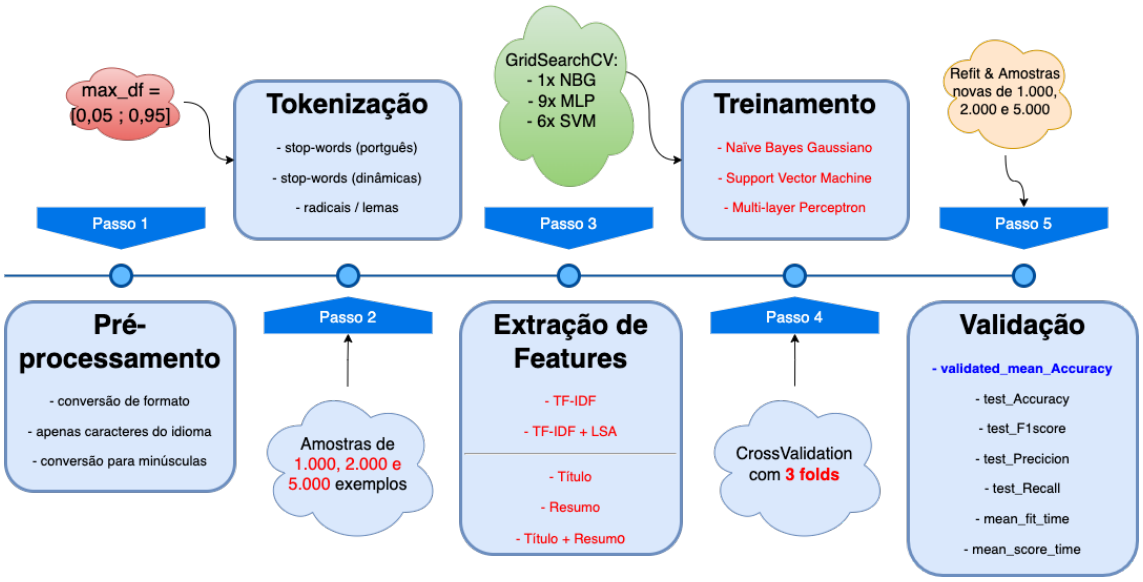

Figura 3.6: Ciclo 2 - Desenvolvimento - Experimento *Features*

Experimento Estimadores, cujo objetivo é identificar o algoritmo que apresenta o melhor desempenho e técnica de extração de *features* mais adequada para cada algoritmo. Segue os passos da figura 3.7 para investigar a diferença de acurácia média balanceada quando utilizados TF-IDF + LSA, e quando usados apenas o TF-IDF. Avaliando também o tempo para extração das *features* e o tempo de treinamento de cada algoritmo. Avalia se a posição relativa dos algoritmos no ranking se mantém consistente conforme é aumentada a quantidade de observações utilizadas no treinamento, e utiliza amostras do conjunto de dados com 1.000, 2.000, 5.000, 10.000, 15.000 e 20.000 observações.

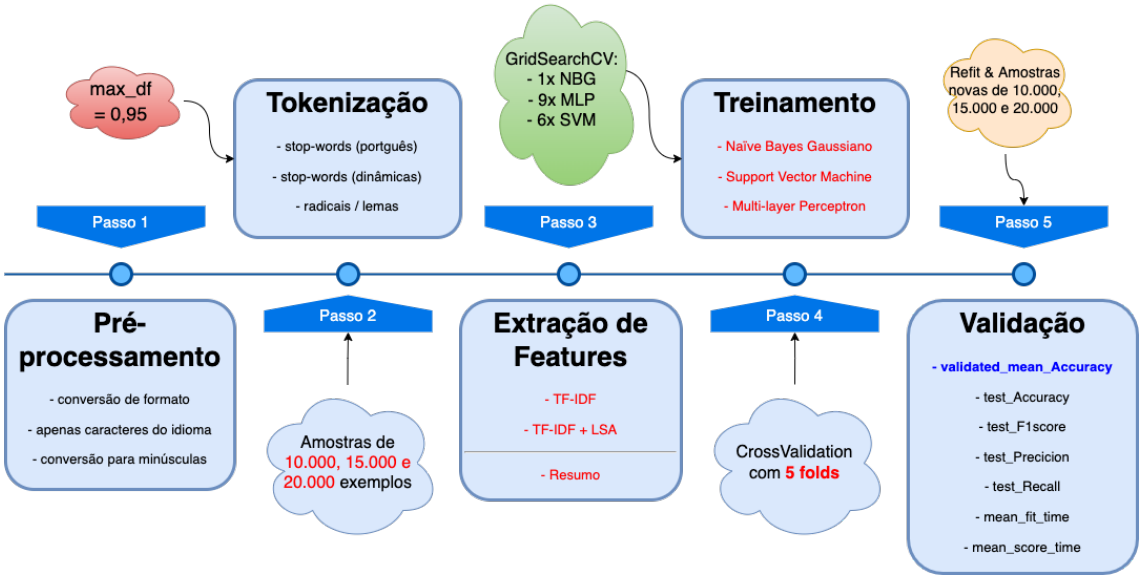

Figura 3.7: Ciclo 2 - Desenvolvimento - Experimento *Estimadores*

Experimento BERT, cujo objetivo é identificar o melhor ajuste para o parâmetro *Learning Rate* (taxa de aprendizado), seguindo os passos da figura 3.8.

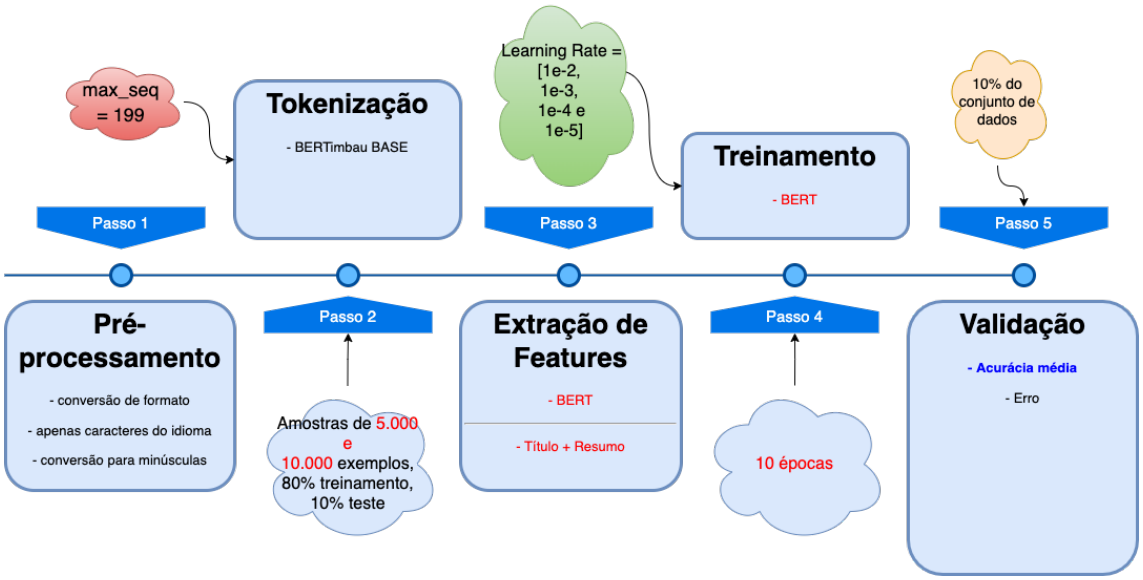

Figura 3.8: Ciclo 2 - Desenvolvimento - Experimento *Estimadores*

3.2.3 Demonstração

OS 2 algoritmos de melhor desempenho na etapa de treinamento e o BERTimbau[5] de SOUZA *et al.* (2020) serão treinados com 90% dos exemplos do conjunto de dados completo, e validados com 10% dos exemplos, depois terão seus desempenhos comparados.

3.2.4 Avaliação

Os 2 algoritmos de melhor desempenho na etapa de demonstração serão avaliados por experimento *"in vitro"*, conforme definido por TRAVASSOS e BARROS (2003), tendo sua acurácia média apenas estimada. Uma versão atualizada do conjunto de dados INPI-BR será produzida com novas observações que serão utilizadas para avaliar o desempenho. Ou seja, 100% do INPI-BR_v1 será utilizado para treinar os 2 classificadores de melhor desempenho nas fases anteriores, e os dados novos serão utilizados para estimar a acurácia.

O classificador que apresentar a melhor acurácia média estimada equipará o artefato "Application Router v2" depois de novamente treinado com todos os exemplos disponíveis na versão mais completa do conjunto de dados INPI-BR.

[5] `https://huggingface.co/neuralmind/bert-base-portuguese-cased`

Capítulo 4

Experimentos

Os experimentos compõem as tarefas de desenvolvimento, demonstração e avaliação do processo DSRM proposto na figura 3.1, e, como parte da avaliação, este capítulo inclui também a apresentação e a análise dos resultados.

Conforme proposto no capítulo 3, os experimentos estão divididos em 2 ciclos de pesquisa: o primeiro ciclo de pesquisa produziu o artefato "Application Router v1", o segundo ciclo de pesquisa produzirá o artefato "Application Router v2". Ambos os ciclos avaliaram conjecturas teóricas e produziram os mapas atualizados 4.13 e 4.39.

4.1 Ciclo de Pesquisa #1 - Prova de Conceito

4.1.1 Desenvolvimento

O primeiro ciclo da pesquisa funciona como Prova de Conceito com o objetivo de demonstrar que a solução baseada em aprendizado de máquina tem acurácia média superior ao e-Distribuidor, além de ter fácil implementação através de algoritmos disponíveis na biblioteca scikit-learn[1] em Python 3.

Estudos anteriores[2] não identificaram melhora significativa de acurácia quando utilizado bi-grama nos textos referentes ao título e ao resumo dos pedidos de paten-

[1] `https://scikit-learn.org/stable/`

[2] `https://github.com/rafaelscnunes/COS738-AutomaticPatentClassification`

tes em português, mas indicaram um aumento significativo do vocabulário, o que impacta negativamente o tempo de extração das *features* e também o tempo gasto para aplicar o LSA, sendo que o LSA já termina por escolher os melhores termos e combinações de termos do vocabulário (FERNÁNDEZ *et al.*, 2018). Por isso, decidiu-se não utilizar n-gramas nos experimentos.

4.1.1.1 Conjunto de Dados

O conjunto de dados utilizado para comparação entre os diversos algoritmos de aprendizado tinha a distribuição de observações nas classes conforme a figura 4.1, similar a distribuição do conjunto completo de dados e contendo exemplos de todas as classes.

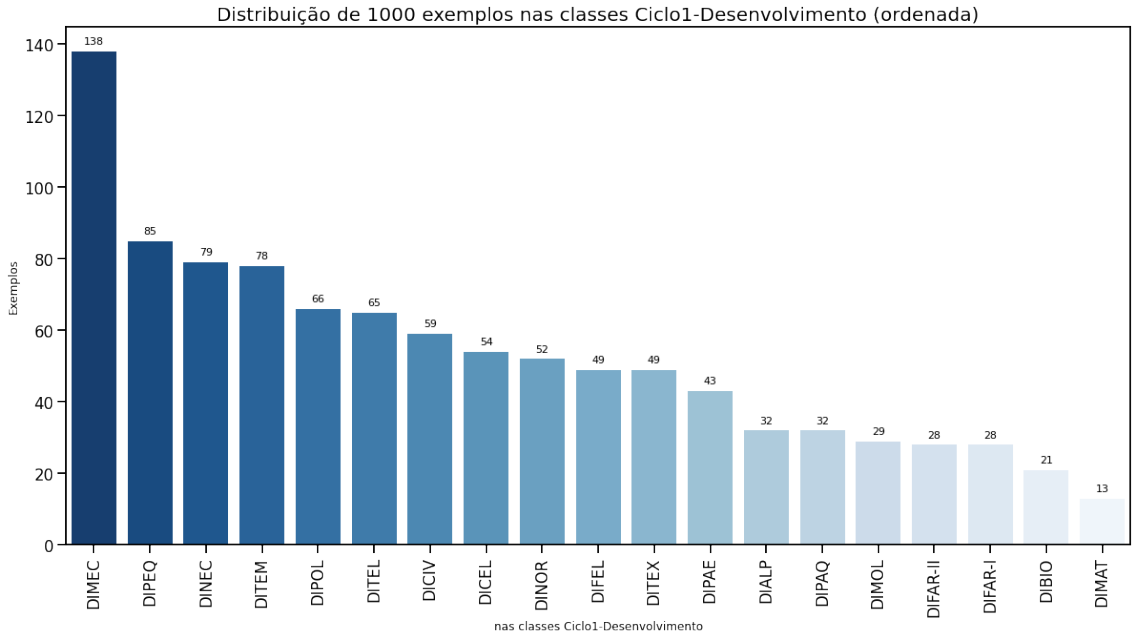

Figura 4.1: Distribuição de classes no conjunto de dados com 1.000 observações

4.1.1.2 Extração de Features

O TF-IDF e o LSA foram obtidos através do código fonte abaixo aplicado sobre as colunas título e resumo do conjunto de dados.

Pré-processamento:

```
text = text.lower()
```

```
text = re.sub(r'[^a-z\- àáâãçéêëíóôõöúü]', '', text)
words = [RSLPStemmer().stem(word) for word in text.split() if len(word) > 2]
```

Cálculo do TF-IDF:

```
tfidf = TfidfVectorizer(
    analyzer='word', tokenizer=word_tokenizer,
    strip_accents='unicode', stop_words=STOP_WORDS,
    ngram_range=(1, n_grams), max_df=.95, min_df=1,
    sublinear_tf=True)
```

Cálculo do LSA:

```
svd = TruncatedSVD(n_components=20, algorithm='randomized')
svd = TruncatedSVD(n_components=30, algorithm='randomized')
```

4.1.1.3 Treinamento

Como referência inicial, estimou-se a acurácia de um classificador aleatório e um classificador aleatório com a mesma distribuição do conjunto de dados original. A figura 4.2 apresenta o resultado dessa simulação, incluindo a estimativa de acurácia ao se escolher sempre a classe mais frequente do conjunto de dados.

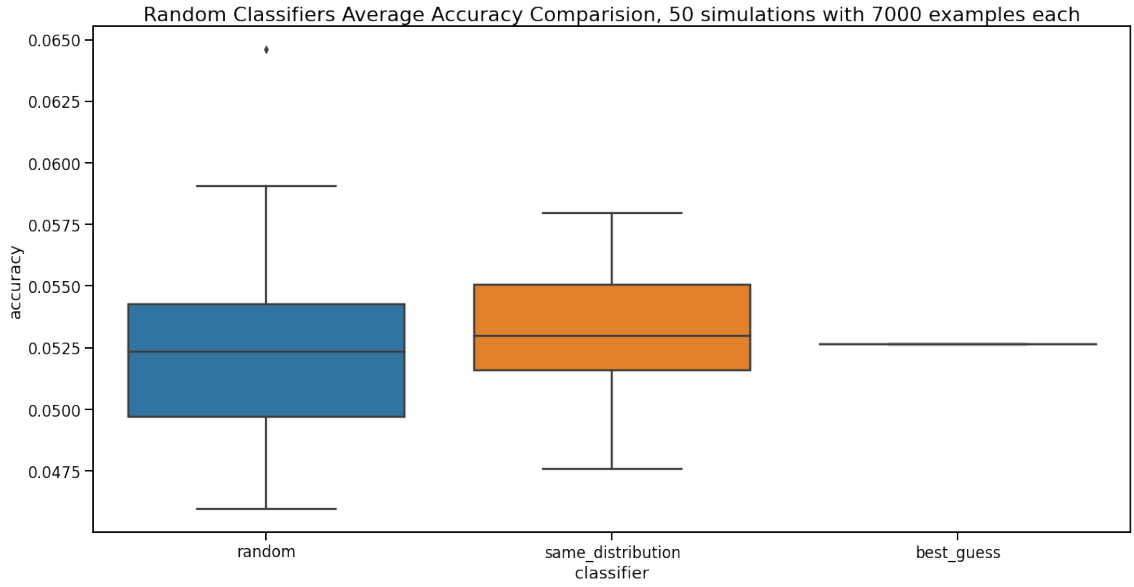

Figura 4.2: Classificadores aleatórios - boxplot

A tabela 4.1 apresenta a acurácia média de cada classificador aleatório, e demonstra que mesmo os 30% de acurácia do e-Distribuidor já correspondiam a um avanço significativo em relação às opções aleatórias e à escolha da classe mais frequente.

Tabela 4.1: Ciclo 1 - Classificadores aleatórios - acurácia

Classificador	Acurácia média
Totalmente aleatório	5.91%
Aleatório, mas seguindo a distribuição do conjunto de dados	4.92%
Escolhendo sempre a classe mais frequente	5.26%

Algoritmos avaliados

Naïve-Bayes Gaussiano - NBG - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.3 e com acurácia de 39,00%.

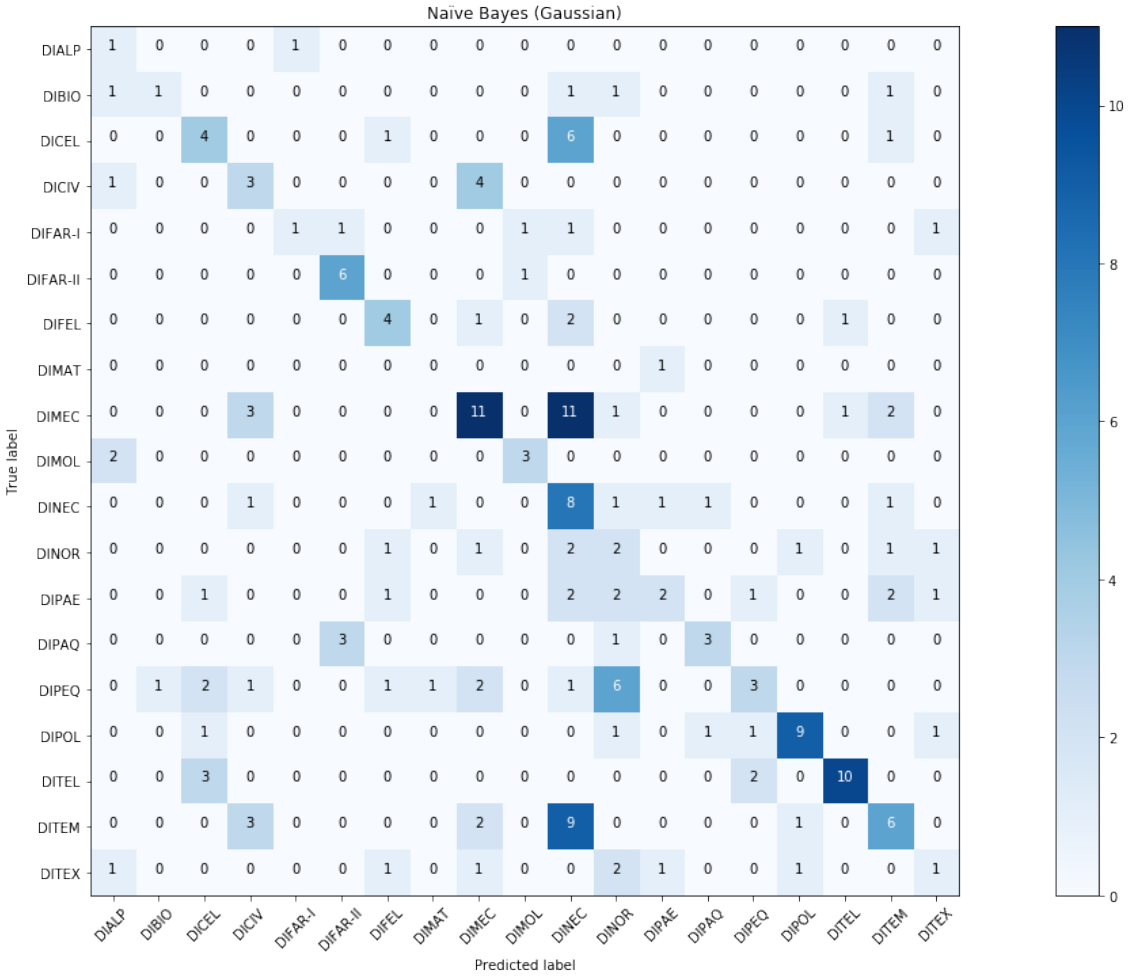

Figura 4.3: Matriz de Confusão do Naive-Bayes Gaussiano

Naïve-Bayes Bernoulli - NBB - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.4 e com acurácia de 33,50%.

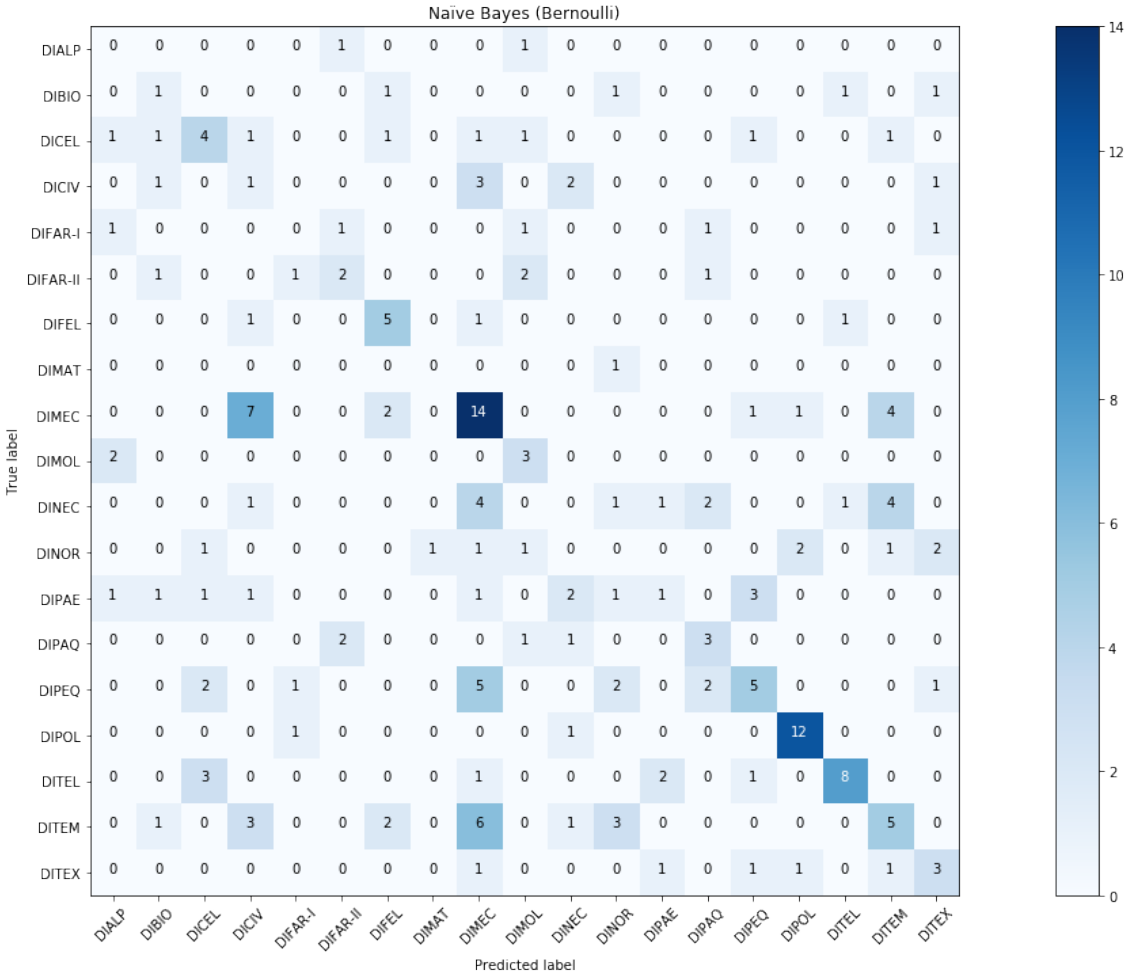

Figura 4.4: Matriz de Confusão do Naive-Bayes Bernoulli

Decision Tree - DCT[3] - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.5 e com acurácia de 20,5%.

[3]Há um erro no texto da imagem. Diz Stochastic Gradient Descent, mas o correto, de acordo com o código fonte do Jupyter Notebook, em anexo, é Decision Tree

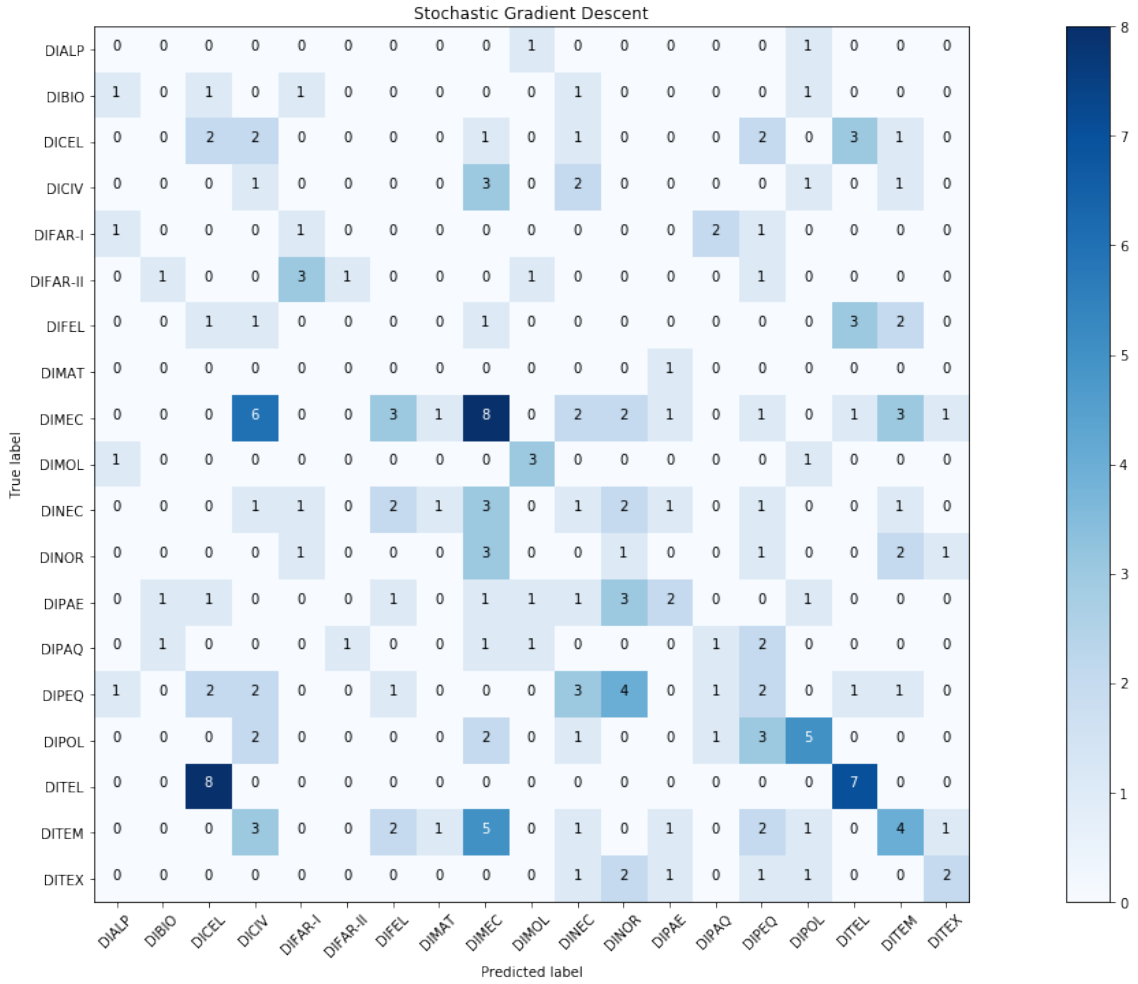

Figura 4.5: Matriz de Confusão do Decision Tree

Random Forest - RDF - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.6 e com acurácia de 25,00%.

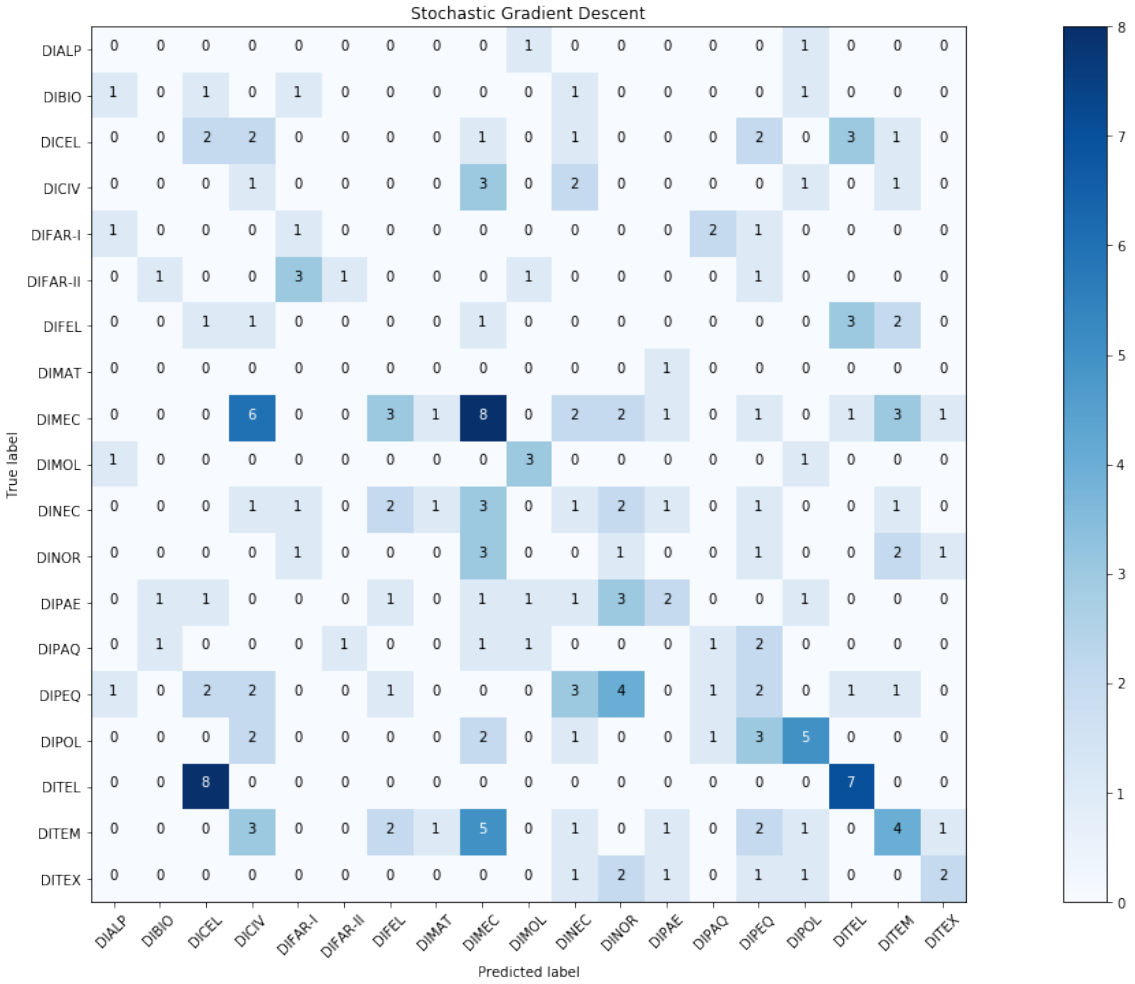

Figura 4.6: Matriz de Confusão do Random Forest

Stochastic Gradient Descendent - SGD - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.7 e acurácia de 31,50%.

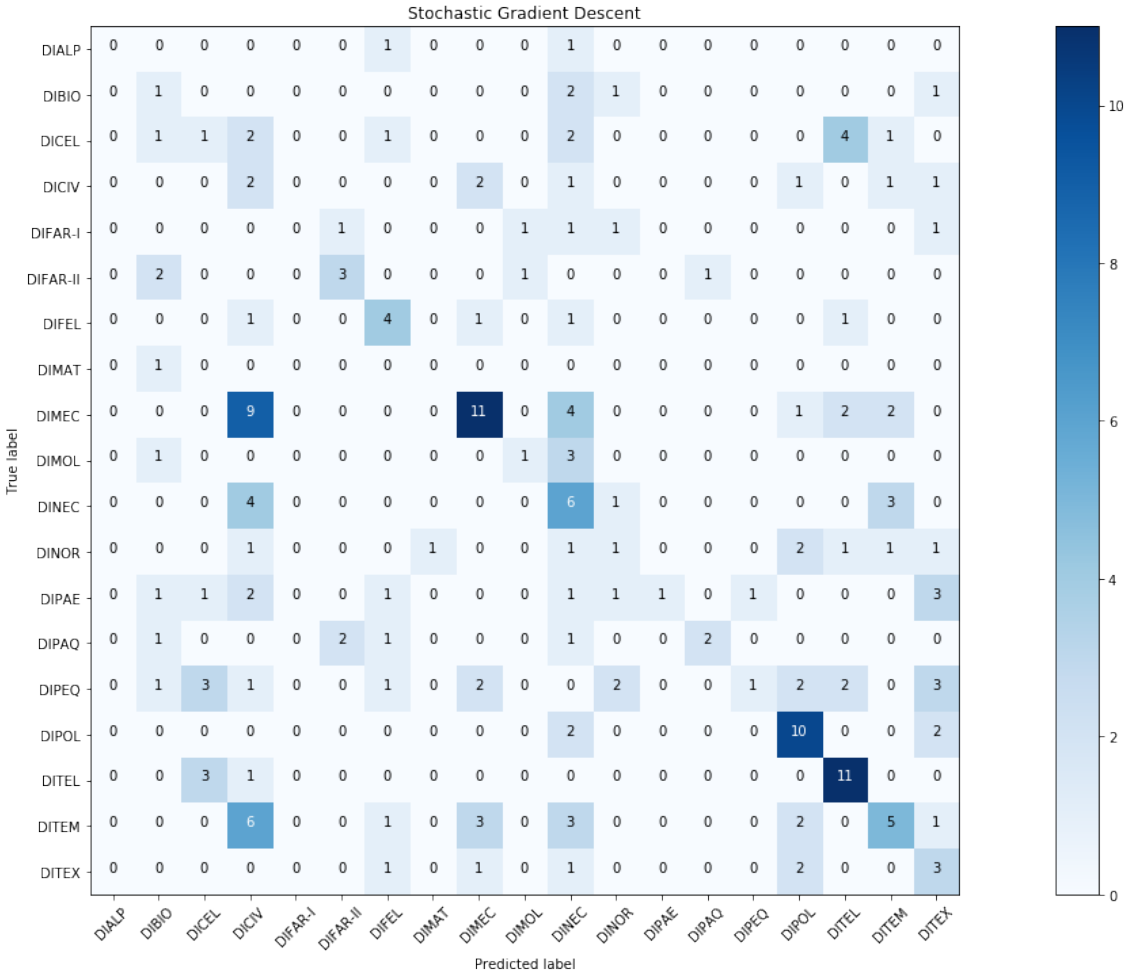

Figura 4.7: Matriz de Confusão do Stochastic Gradient Descendent

Multi-layer Parceptron - MLP - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.8 e com acurácia de 37,00%.

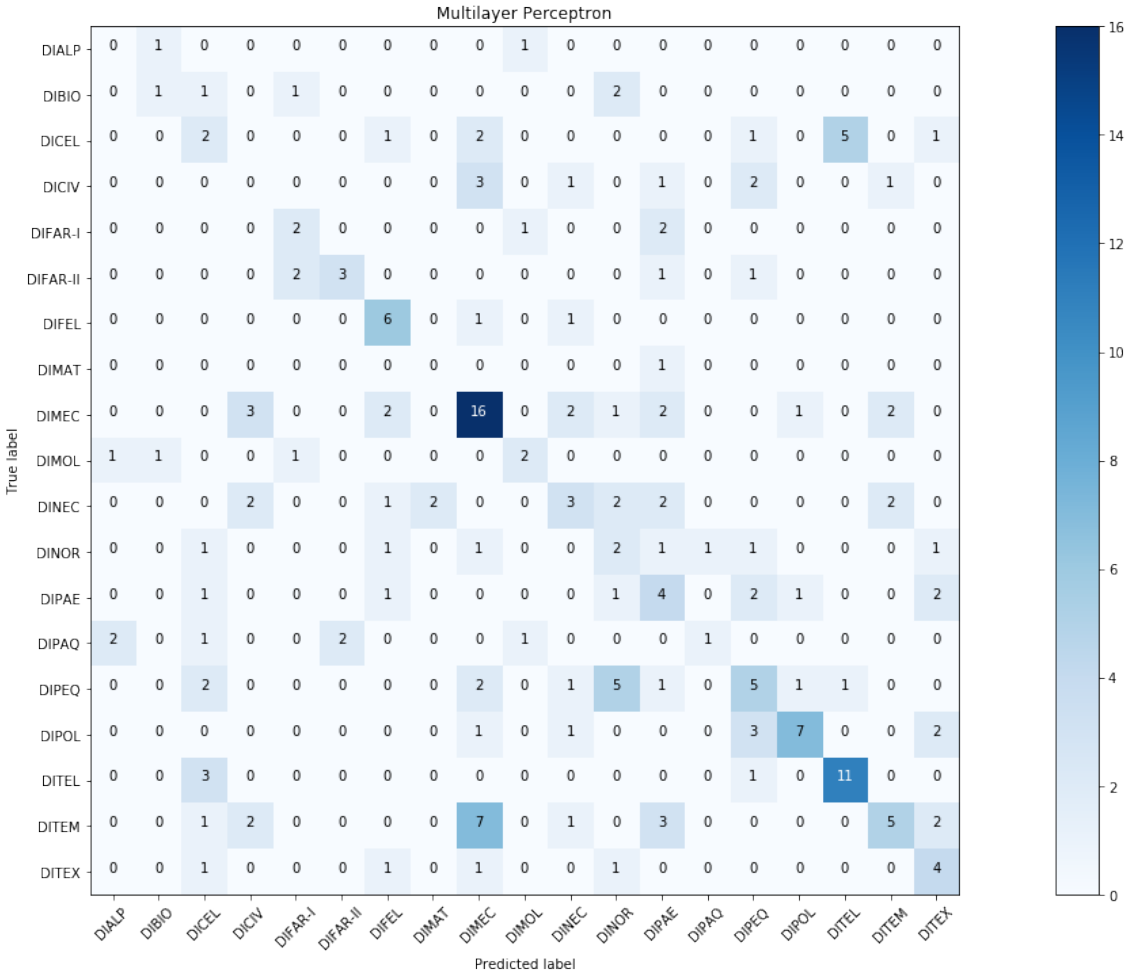

Figura 4.8: Matriz de Confusão do Multi-layer Perceptron

Support Vector Machine - SVM - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.9 e com acurácia de 38,00%.

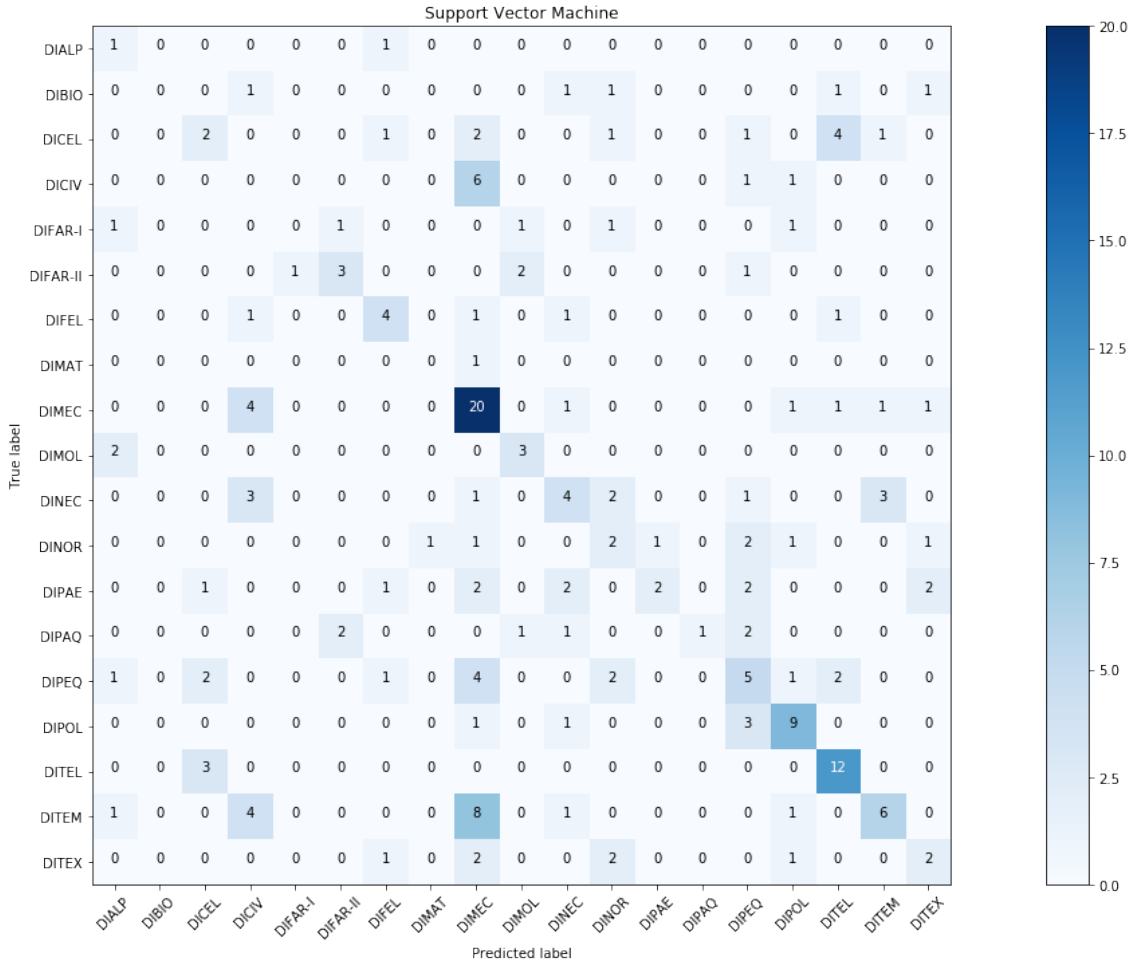

Figura 4.9: Matriz de Confusão do Support Vector Machine

O resultado da etapa de treinamento do ciclo 1 foi um ranking baseado na acurácia, apresentado na tabela 4.2.

Tabela 4.2: Ciclo 1 - Ranking dos Algoritmos - Desenvolvimento

Algoritmo	Acurácia
NBG - Naïve-Bayes Gaussiano	39,00%
SVM - Support Vector Machine	38,00%
MLP - Multi-layer Perceptron	37,00%
NBB - Naïve-Bayes Bernoulli	33,50%
SGD - Stochastic Gradient Descendent	31,50%
RDF - Random Forest	25,00%
DCT - Decision Tree	20,50%

4.1.2 Demonstração

Como o e-Distribuidor era baseado em uma rede neural, e o MLP apresentou desempenho entre os melhores, escolheu-se adotá-lo como algoritmo para o classificador final da solução. Tanto o MLP quanto o NBG foram treinados utilizando o conjunto de dados completo, com mais de 360.000 observações e distribuição de classes conforme a figura 4.10.

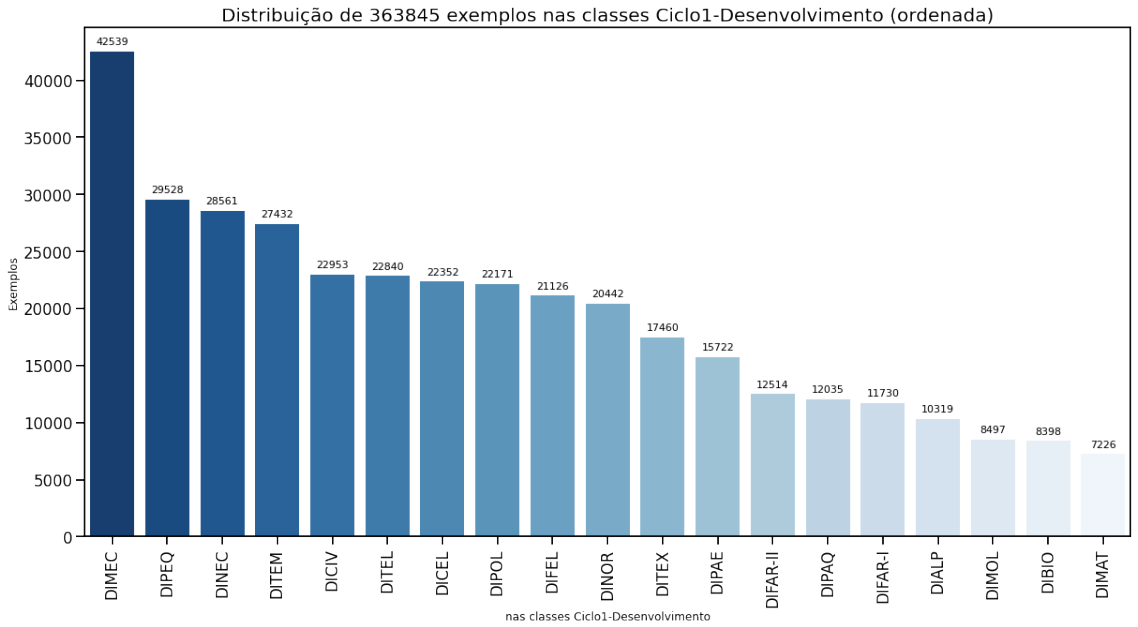

Figura 4.10: Conjunto de dados completo do Application Router v1

Para calcular o TF-IDF foram utilizados tanto o título quanto o resumo, separadamente. O LSA também foi calculado separadamente, retornando 200 dimensões para o título e 800 dimensões para o resumo. O resultado do LSA para cada uma das colunas foi concatenado formando uma matriz com 1.000 colunas que foi a entrada para os algoritmos de aprendizado. A tabela 4.4 mostra o resultado dos testes. A tabela 4.3 mostra o tamanho do vocabulário e os tempos de extração do TF-IDF e do LSA.

Tabela 4.3: Ciclo 1 - Tokens e tempo de extração das *features*

Texto	Tokens	TF-IDF	LSA
Título	65.294	1h18min	65seg
Resumo	277.495	11h42min	9min35seg

Tabela 4.4: Ciclo 1 - Ranking dos Algoritmos - Demonstração

Algoritmo	Acurácia	Tempo de treiamento
MLP - Multi-layer Perceptron	58,71%	3h48min
NBG - Naïve-Bayes Gaussiano	40,44%	6seg

4.1.2.1 Falcon 9

O modelo baseado no algoritmo NBG obteve acurácia de 40,44% quando treinado com 80% dos exemplos e testado com 20% deles. A matriz de confusão pode ser observada na figura 4.11.

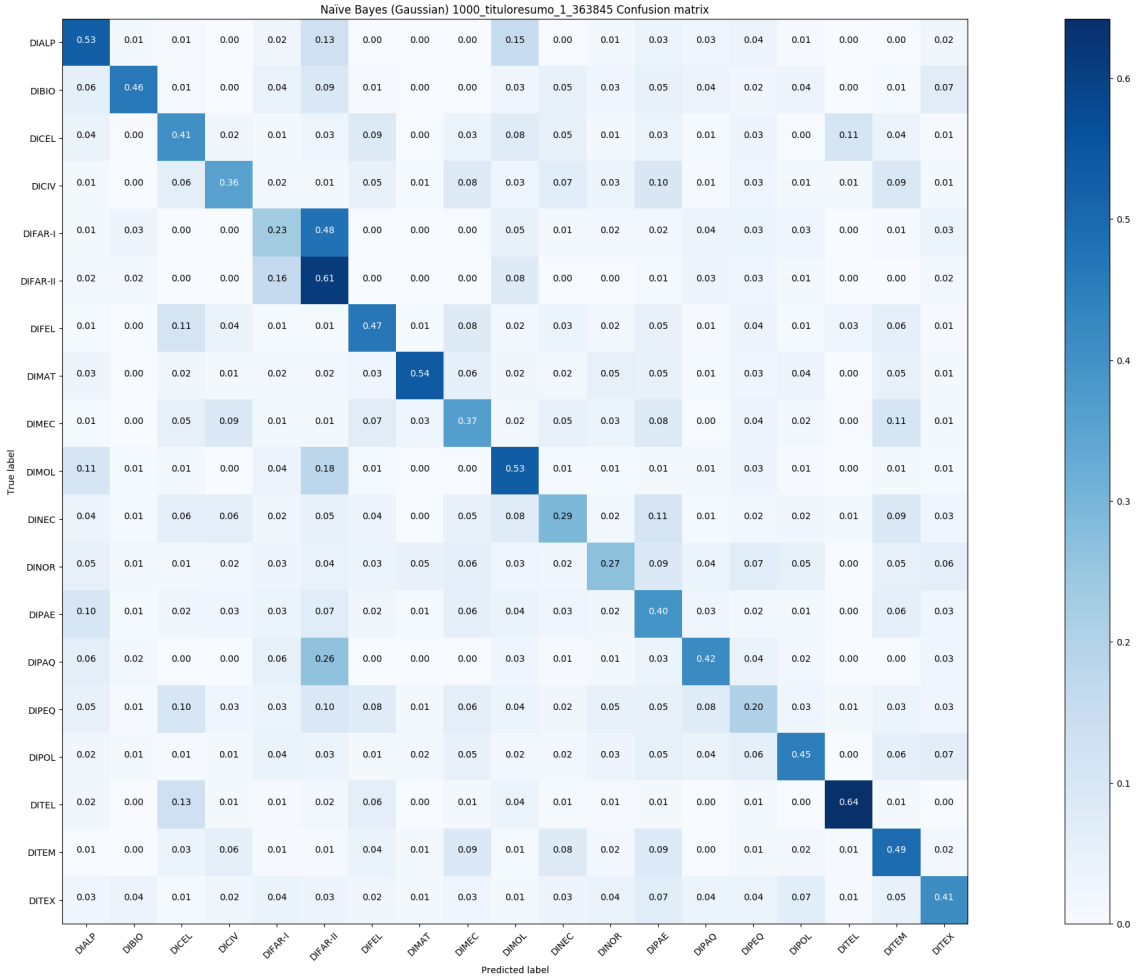

Figura 4.11: Application Router v1 - Falcon 9 Model

4.1.2.2 Falcon Heavy

O modelo baseado no algoritmo MLP obteve acurácia de 58,71% quando treinado com 80% dos exemplos e testado com 20% deles. A matriz de confusão pode ser observada na figura 4.12.

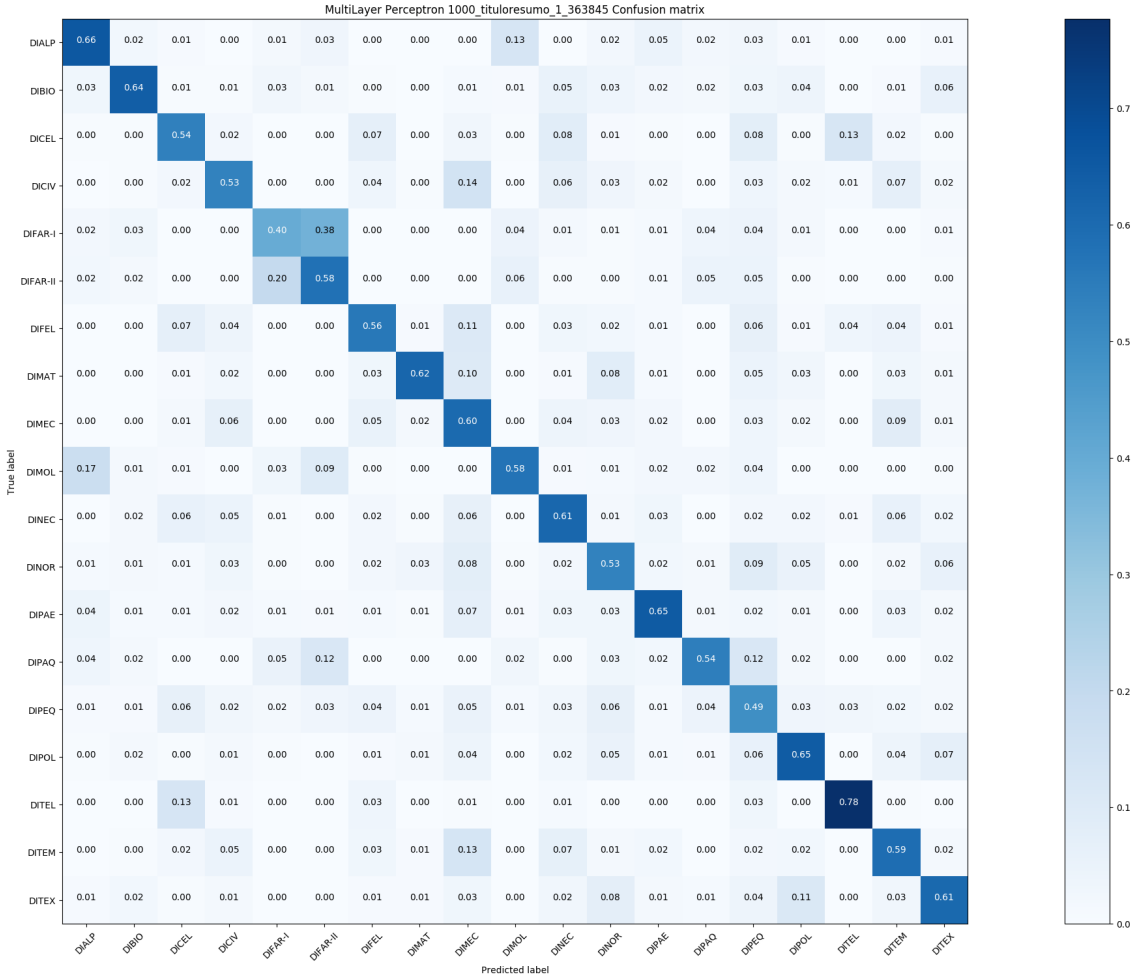

Figura 4.12: Application Router v1 - Falcon Heavy Model

4.1.3 Avaliação

Um experimento *in vivo* foi realizado para avaliar o artefato produzido pelo primeiro ciclo de pesquisa. Desde abril de 2018 o "Application Router v1" está em operação no INPI. Em junho de 2019, foram coletados os dados de 1 ano de operação e efetuada uma análise comparando o resultado da distribuição feita pelo "Application Router

v1" com a distribuição que teria sido feita pelo e-Distribuidor. Essa análise concluiu que a acurácia média do artefato em produção foi de 56,85%.[4]

> Universo: 11224 pedidos
> Percentual do universo corretamente roteado pelo "Application Router": 56.85% [6381 pedidos]
> Percentual do universo corretamente roteado pela Rede Neural Antiga: 29.78% [3342 pedidos]
> Percentual de melhora do roteamento do "Application Router" comparado ao roteamento da Rede Neural Antiga: 90.93% [3039 pedidos]

A análise continuou com a revisão das conjecturas da proposta do ciclo 1, o mapa atualizado pode ser visto na figura 4.13.

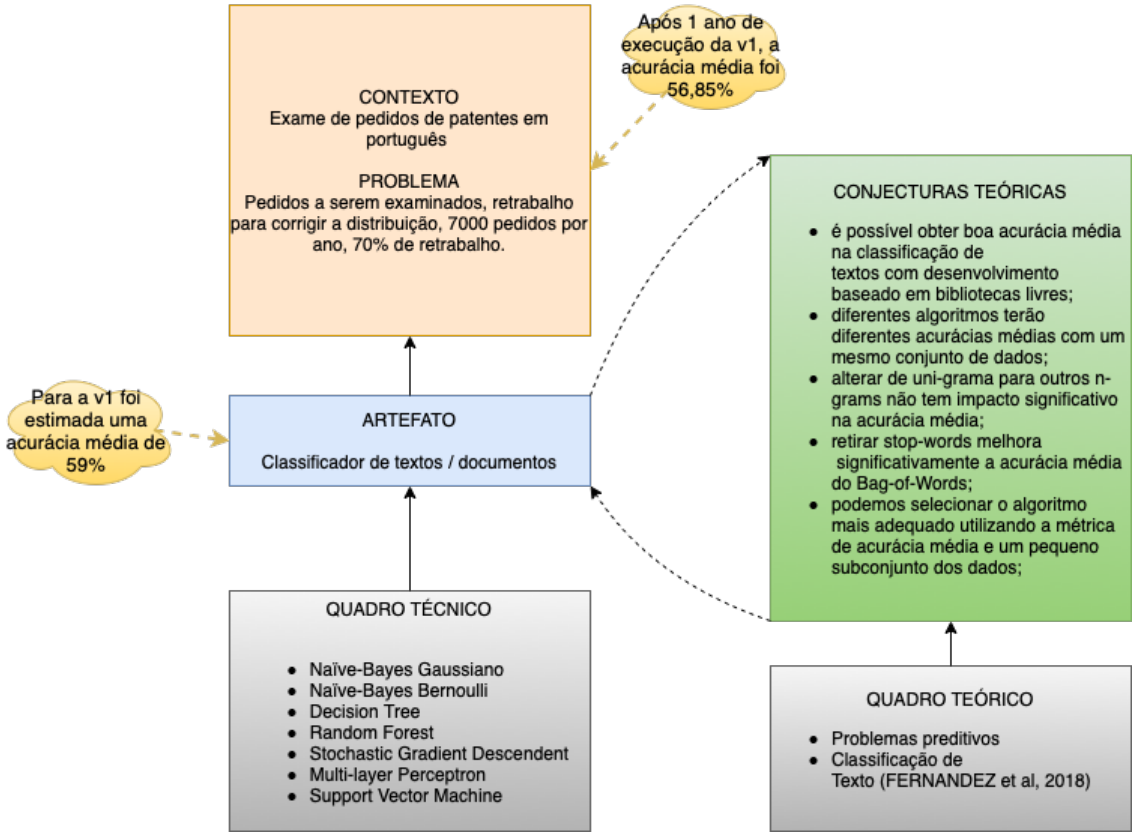

Figura 4.13: Mapa DSR - ciclo de pesquisa #1 - após experimentos

4.1.3.1 Revisão das conjecturas

- "É possível obter boa acurácia média na classificação de textos com desenvolvimento baseado em bibliotecas livres" - aumentou-se a acurácia de aproximadamente 30% para pouco mais de 56%, adotando-se como critério de "boa

[4]Informação retirada do relatório sobre os resultados obtidos com a adoção do "Application Router"

acurácia média" como sendo uma acurácia superior àquela apresentada pela solução em operação anteriormente, a conjectura foi confirmada. O "Application Router v1" atingiu quase o dobro da acurácia do "e-Distribuidor".

- "diferentes algoritmos terão diferentes acurácias médias com um mesmo conjunto de dados" - ficou demonstrado pela variação no valor das acurácias médias ao longo do ciclo de pesquisa. Os experimentos de seleção do estimador/algoritmo permitiram confirmar que os algoritmos NBG, SVM e MLP, atingiram acurácia média superior a acurácia média dos outros estimadores/algoritmos (NBB, SGD, RDF e DCT).

- "alterar de uni-grama para outros n-gramas não tem impacto significativo na acurácia média" - essa conjectura não foi testada. Ensaios anteriores sugeriram que o pequeno aumento da acurácia média não compensava o aumento no tempo de treinamento.

- "retirar stop-words melhora significativamente a acurácia média do Bag-of-Words" - essa conjectura não foi testada. Decidiu-se não retirar as stop-words e deixar o LSA escolher as dimensões. Uma tarefa foi prevista no experimento do ciclo 2 para investigar o assunto.

- "é possível selecionar o algoritmo mais adequado utilizando a métrica de acurácia média em um pequeno subconjunto dos dados" - essa conjectura foi rejeitada, uma vez que o ranking de acurácia média não se manteve da fase de desenvolvimento para a fase de demonstração, como pode ser verificado comparando-se as posições relativas do NBG e do MLP nas tabelas 4.2 e 4.4. Porém, uma tarefa foi adicionada no experimento do ciclo 2 para investigar melhor o assunto, uma vez que o tamanho da amostra do conjunto de dados utilizada para construir o ranking pode ter sido insuficiente.

4.2 Ciclo de pesquisa #2 - Otimizar a performance

Com o conceito provado no primeiro ciclo, o segundo ciclo de pesquisa buscou a evolução do "Application Router v1", ou seja, superar a acurácia média estimada de 59%. Para isso, esse ciclo segue analisando as conjecturas que não foram nem confirmadas nem descartadas totalmente, as conjecturas propostas para o segundo ciclo, e novas conjecturas que surgiram durante os experimentos.

4.2.1 Desenvolvimento

O classificador, baseado no algoritmo Multi-layer Perceptron, criado no ciclo anterior, foi configurado com 2 níveis escondidos de 100 e 50 neurônios. Portanto, uma das conjecturas é a de que novos ajustes de hiper-parâmetros podem render melhoras na acurácia média.

Neste ciclo de pesquisa utilizou-se a classe *sklearn.model_selection.GridSearchCV* para implementar uma busca pela melhor acurácia média ponderada dentre os conjuntos de hiper-parâmetros de cada algoritmo, utilizando o método da validação cruzada, conforme indicado nas figuras 3.6, 3.7 e 3.8. A ponderação da média é feita com base na distribuição das observações nas classes; e os conjuntos de hiper-parâmetros testados pode ser vistos abaixo:

```
param_grid = {
    'nbg': {},

    'mlp': {'hidden_layer_sizes': [(100, 50, 19), (100,), (100, 50)],
            'activation': ['relu'],
            'solver': ['adam'],
            'alpha': [1e-4,],
            'learning_rate': ['adaptive',],
            'learning_rate_init': [1e-3, 1e-4, 1e-5],
            'max_iter': [2000, ]},
```

```
    'svm': {'C': [0.5, 1.0, 1.5],
            'dual': [True, False],
            'random_state': [self.random_seed, ],
            'tol': [1e-4,],
            'multi_class': ['ovr',]},
}
```

Como os experimentos foram executados com amostras reduzidas do conjunto de dados (1.000, 2.000, 5.000, 10.000, 15.000 e 20.000 observações), foi ativada a funcionalidade *refit* para treinar o modelo selecionado (aquele com os hiper-parâmetros que atingiram a melhor acurácia média ponderada) utilizando todo o conjunto de dados. E utilizou-se um segundo conjunto de dados de tamanho semelhante ao da amostra de treinamento para validar o modelo treinado. Nos gráficos e *Jupyter Notebooks*:

- test_Accuracy - é o valor da acurácia média ponderada obtido em cada etapa da validação cruzada;
- mean_Accuracy - é o valor médio da acurácia média ponderada obtido para um conjunto específico de hiper-parâmetros;
- validated_mean_Accuracy - é o valor da acurácia média ponderada obtido na etapa de validação utilizando amostras diferentes daquela utilizada na validação cruzada e *refit*.

4.2.1.1 Conjunto de dados

O dataset gerado e utilizado no segundo ciclo conta com mais de 340.000 exemplos no formato (data de classificação ; título ; resumo ; classificação ipc ; área de exame ; rpi de registro), contando com exemplos de todas as 19 classes, como mostra a figura 4.14.

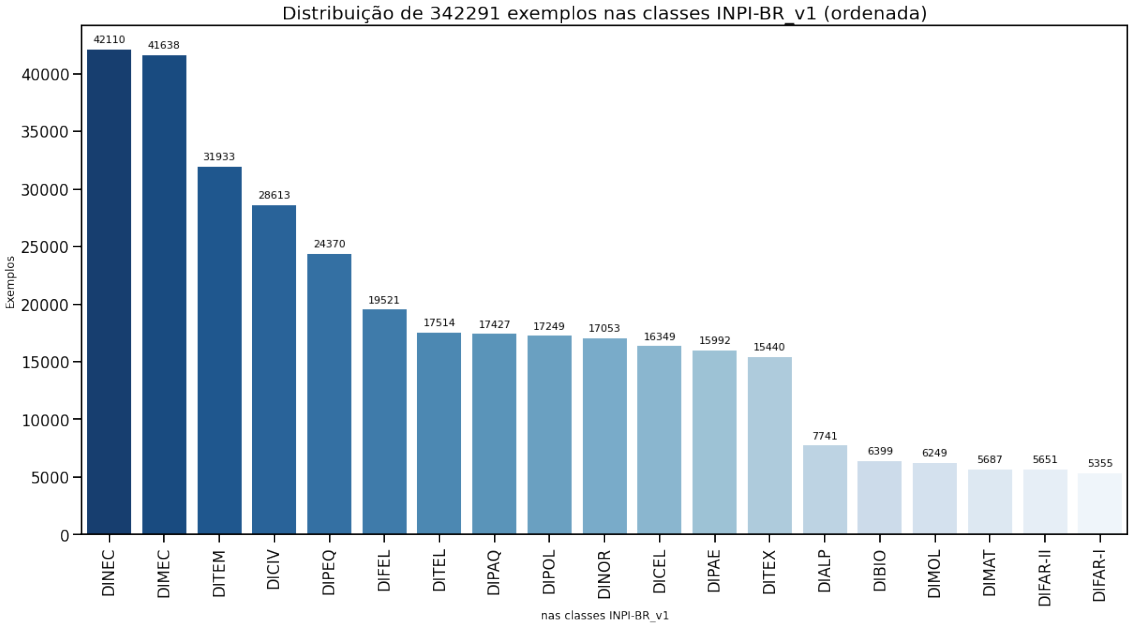

Figura 4.14: Dataset INPI-BR_v1

Apesar do aparente desbalanceamento, a revisão da literatura mostrou que, no contexto do aprendizado de máquina, é considerado desbalanceado um conjunto de dados cuja relação entre a quantidade de observações das classes apresente uma relação da ordem de 100:1 ou mais. Como é possível observar na figura 4.14 a relação entre a classe com maior número de observações e a classe com menor número de observações é de aproximadamente 8:1, sendo assim, decidiu-se não aplicar qualquer técnica de balanceamento nos dados.

A tabela 4.5 mostra um breve resumo da Análise Exploratória de Dados (AED), em inglês, *Exploratory Data Analysis (EDA)*[5], em inglês) que foi feita para decidir entre o título e o resumo, únicos textos disponíveis para serem utilizados na tarefa de extração de *features* (matriz X).

Tabela 4.5: Ciclo 2 - Dataset INPI-BR_v1

	Quantidade	Tamanho médio	Desvio padrão	Vocabulário
Títulos	342.291	12 palavras	10	85.623
Resumo	342.291	130 palavras	69	305.176

[5] `https://github.com/rafaelscnunes/app-router/blob/main/notebooks/dissertation_4_cycle2_1_eda_inpi-br_v1.ipynb`

A AED também mostrou que 8.854 palavras ocorrem exclusivamente no título, sendo que uma verificação aleatória dessas palavras mostrou que elas tendem a ser ruído, erros de digitação. Além disso, 65% dos títulos se repetem no resumo.

4.2.1.2 Extração de Features

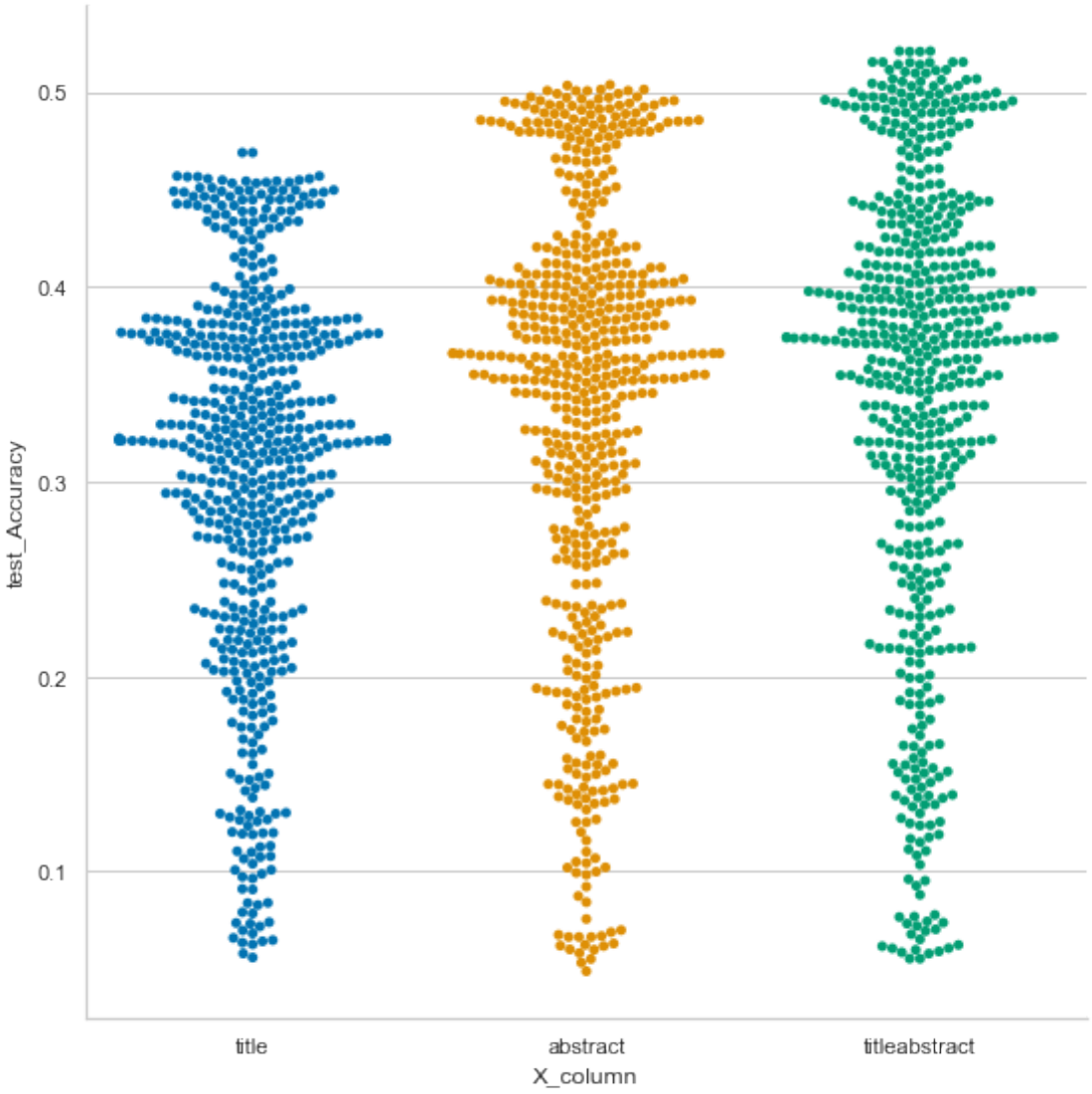

Figura 4.15: Título, Resumo ou Título + Resumo?
(test_Accuracy)

O experimento específico para identificar o melhor texto (título, resumo ou título+resumo) a ser utilizado para treinar os algoritmos concluiu que utilizar apenas o resumo resulta em maior acurácia na classificação quando comparado com o uso exclusivo do título, como observado na figura 4.16. Sendo assim, apesar de o melhor

desempenho ser obtido pelo texto: título+resumo, como na figura 4.15, decidiu-se utilizar exclusivamente o resumo para a extração das *features* que vão alimentar os algoritmos de aprendizado no segundo experimento (Estimadores) da etapa de desenvolvimento do ciclo 2.

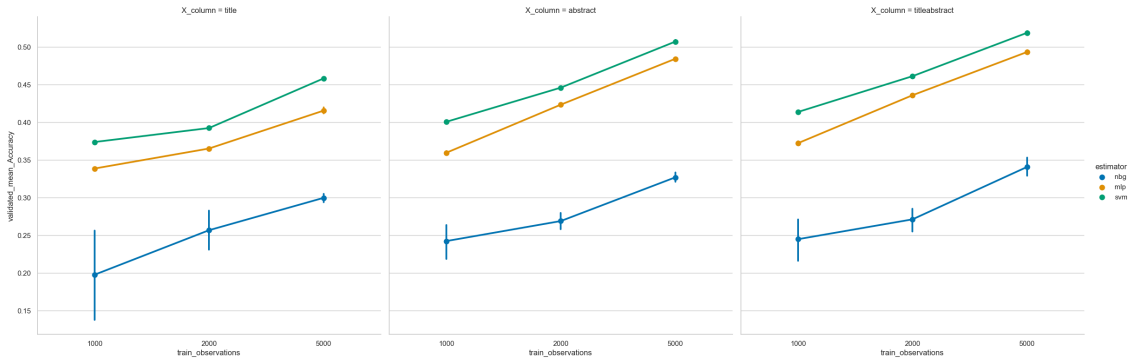

Figura 4.16: Título, Resumo ou Título + Resumo?
(validated_mean_Accuracy)

A figura 4.16 mostra que com 1.000, 2.000 ou 5.000 observações, utilizar a coluna "resumo", em comparação com a coluna "título", sempre resulta em uma acurácia média maior na fase de validação. Lembrando que neste ciclo de pesquisa sempre utiliza-se uma amostra do conjunto de dados para treinar os algoritmos utilizando a técnica de validação cruzada, nesse caso com 3 *folds*, e depois utiliza-se outra amostra de dados para validar a acurácia média do classificador selecionado pelo *GridSearchCV*[6].

Além do pré-processamento tradicional com a remoção dos valores nulos e dos valores duplicados, a conversão de todo o texto para minúsculas e o filtro para descartar qualquer caractere estranho ao idioma português, a tarefa de tokenização deste ciclo de pesquisa contou com a retirada das stop-words padrão do português conforme a biblioteca NLTK de BIRD *et al.* (2001) e também a identificação dos radicais das palavras utilizando a classe *nltk.stem.RSLPStemmer*[7].

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

[7] `https://www.nltk.org/_modules/nltk/stem/rslp.html`

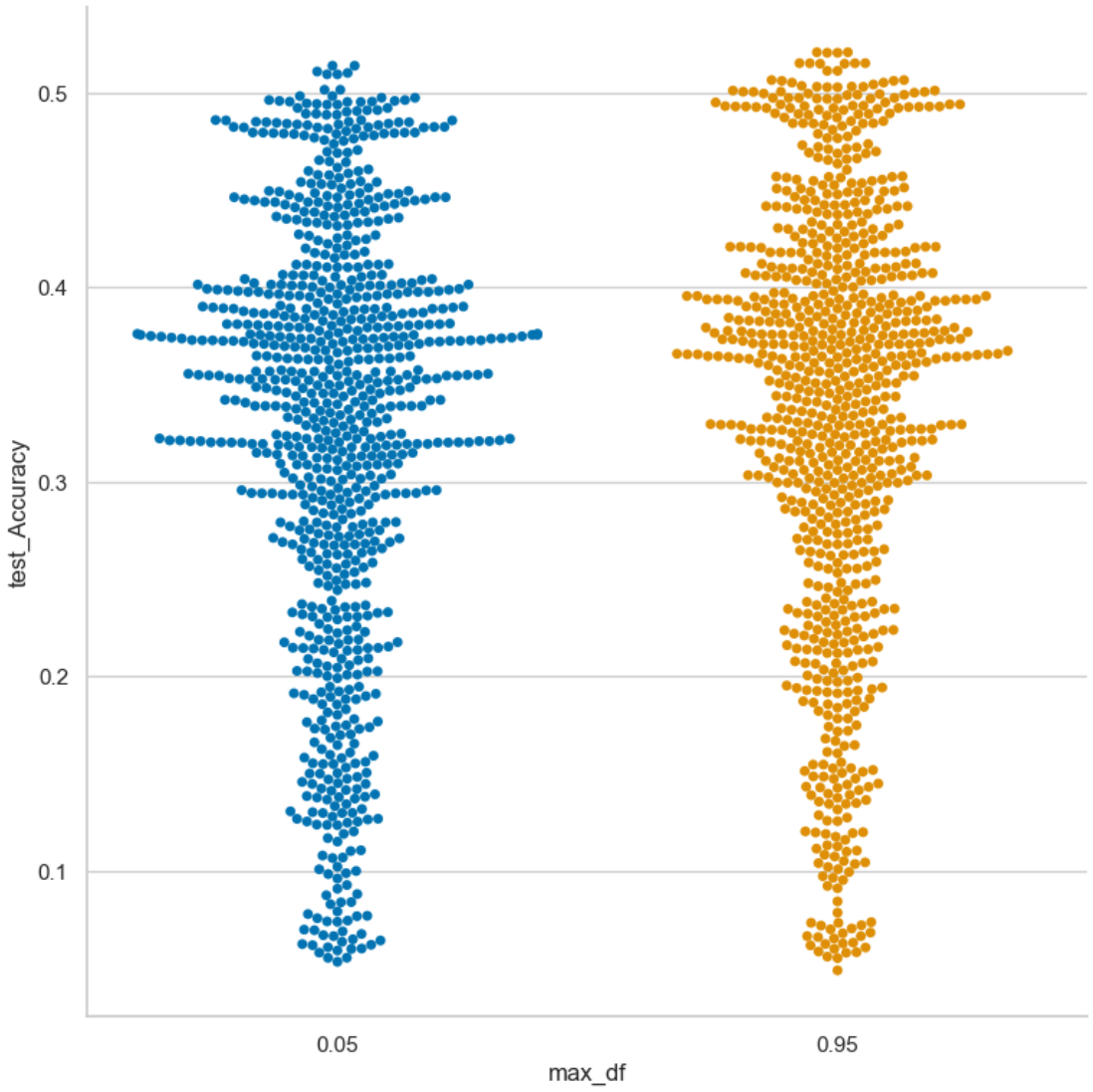

Figura 4.17: Retirar stop-words dinamicamente? (test_Accuracy)

Foi avaliada também a utilização de uma opção da biblioteca scikit-learn de PEDREGOSA *et al.* (2011) para identificar e retirar dinamicamente palavras que aparecem repetidamente nos textos. A figura 4.17 mostra a acurácia média treinando os classificadores com diferentes quantidades de observações e comparando esse desempenho com 2 valores diferentes para o parâmetro max_{df}, $0,95$ é o valor padrão e $0,05$ é o valor que começa a provocar uma mudança significativa ($+$ 100) na quantidade de stop-words consideradas.

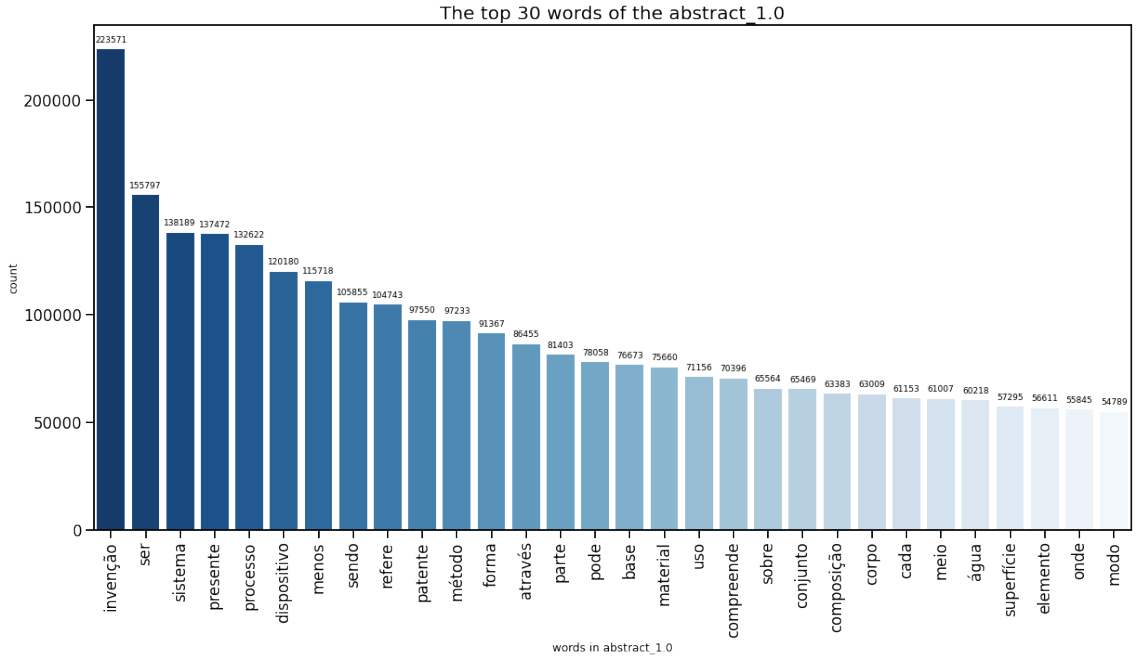

Figura 4.18: Todas as palavras

É possível verificar nos histogramas das figuras 4.18 e 4.19, em conjunto com a variação da performance apresentada na figura 4.17, que apesar da mudança no perfil dos histogramas o impacto na performance dos classificadores foi muito pequeno, e negativo em todos, com exceção do NBG para o impacto negativo.

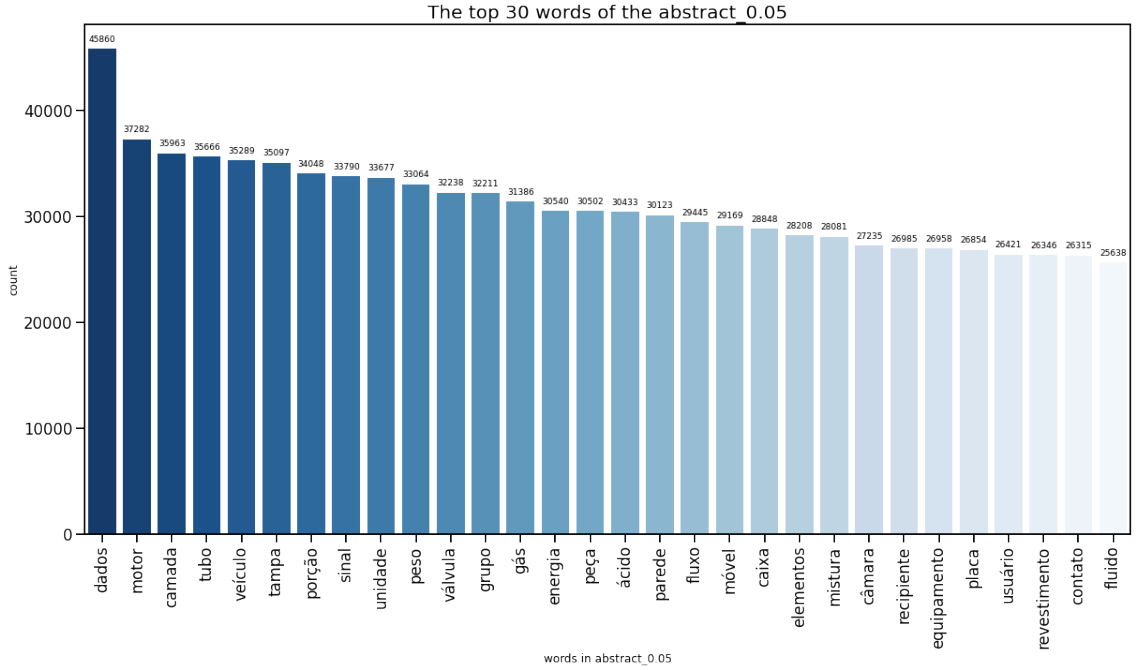

Figura 4.19: Excluídas as palavras que se repetem no resumo de pelo menos 5% dos exemplos

Avaliou-se também o impacto da retirada das stop-words considerando as técnicas de extração de features. A figura 4.20 mostra que tanto utilizando TF-IDF, como utilizando TF-IDF + LSA, um valor de max_{df} próximo de $1,00$ sempre resulta em uma acurácia média superior a valores menores de max_{df}.

Após as análises, decidiu-se descartar dinamicamente apenas as palavras que realmente aparecerem em quase todos os resumos, mantendo o valor padrão do parâmetro $max_{df} = 0,95$. Com esse valor nenhuma palavra foi descartada dinamicamente do conjunto de dados INPI-BR.

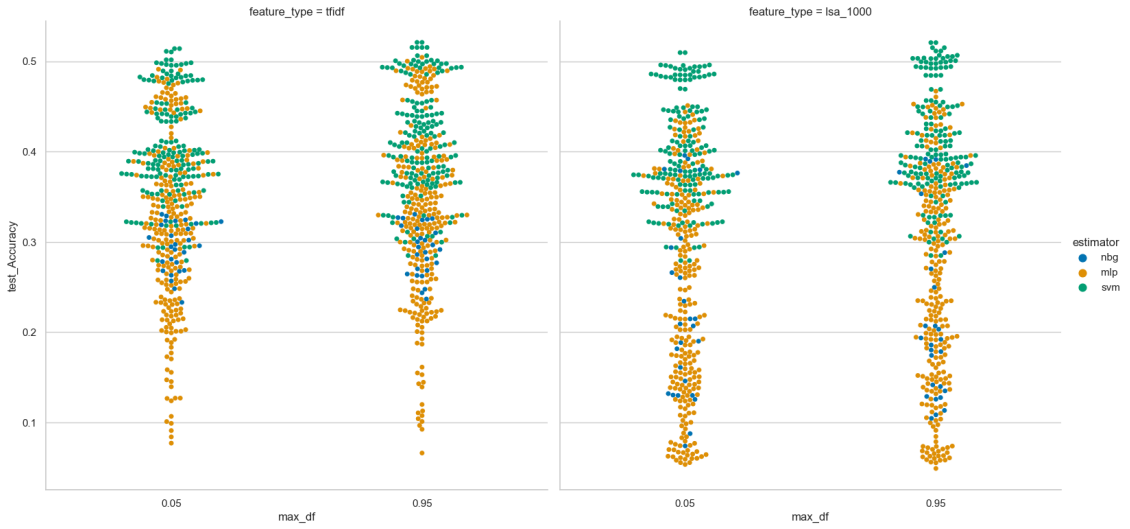

Figura 4.20: Retirar stop-words dinamicamente? (tipo de *feature*)

Comparou-se o desempenho dos classificadores dependendo da técnica de extração de *features*: TF-IDF ou TF-IDF + LSA. A figura 4.21 apresenta o resultado dos testes.

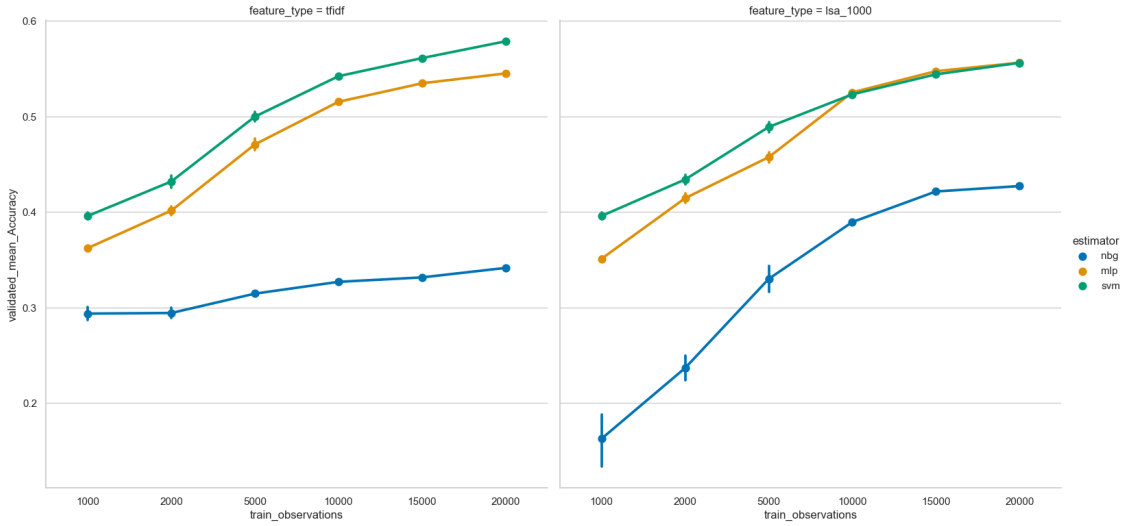

Figura 4.21: TF-IDF ou TF-IDF + LSA?

A figura 4.21 mostra que a melhor acurácia é obtida pelo SVM utilizando TF-IDF, e que o segundo melhor desempenho é do MLP utilizando TF-IDF + LSA.

4.2.1.3 Treinamento

A tabela 4.6 apresenta os tempos de treinamento (mean_fit_time) e predição (mean_score_time), lembrando que todas as implementações de algoritmo utilizadas neste trabalho utilizam a abordagem One-vs-Rest para a classificação multi-classes.

Tabela 4.6: Ciclo 2 - Duração do treinamento (em segundos)

estimator	observations	mean_fit_time	mean_score_time
mlp	1000	547.067900	0.470021
	2000	818.974625	0.713387
	5000	1999.936221	1.058542
	10000	3960.399590	0.769975
	15000	6792.343807	1.140777
	20000	13346.941700	9.857106
nbg	1000	0.114778	0.347543
	2000	0.315059	1.181592
	5000	1.320153	4.694783
	10000	5.380005	8.649023
	15000	10.226935	16.178202
	20000	16.926720	26.803106
svm	1000	1.477000	0.131237
	2000	2.845847	0.195818
	5000	7.484465	0.402968
	10000	19.689743	1.649221
	15000	45.207215	2.193218
	20000	103.311387	15.629600

Algoritmos treinados

Naïve-Bayes Gaussiano - NBG - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.22 e com acurácia de 42,72%, a partir de *features* LSA com 1.000 dimensões baseadas exclusivamente no resumo de 20.000 observações.

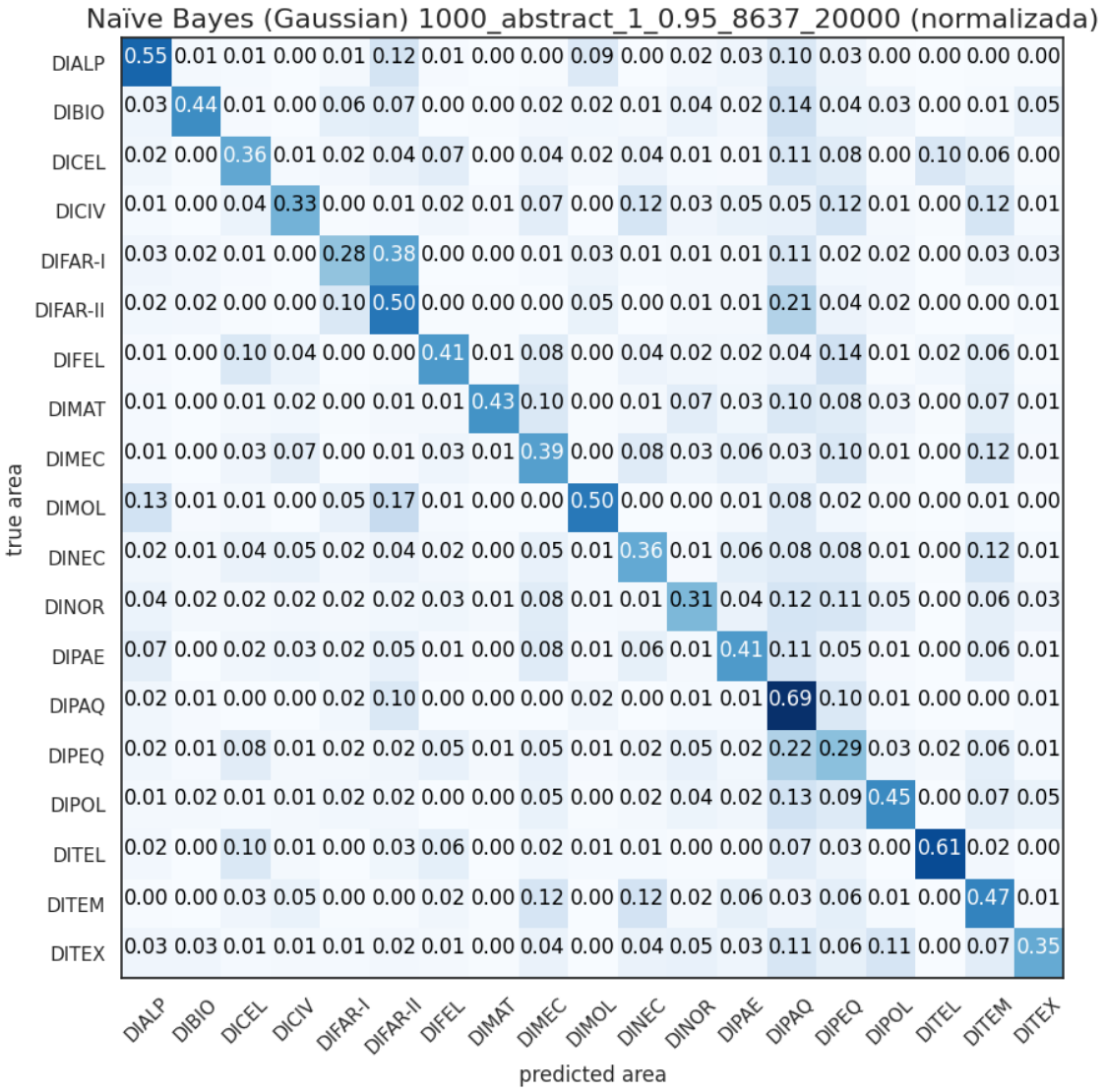

Figura 4.22: Ciclo 2 - Naive-Bayes Gaussiano - Matriz de Confusão

Multi-layer Parceptron - MLP - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.23 e com acurácia de 55,64%, a partir de *features* LSA com 1.000 dimensões baseadas exclusivamente no resumo de 20.000 observações. Sendo que os hyper-parâmetros estão com a seguinte configuração: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 50, 19), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2000, 'solver': 'adam'}

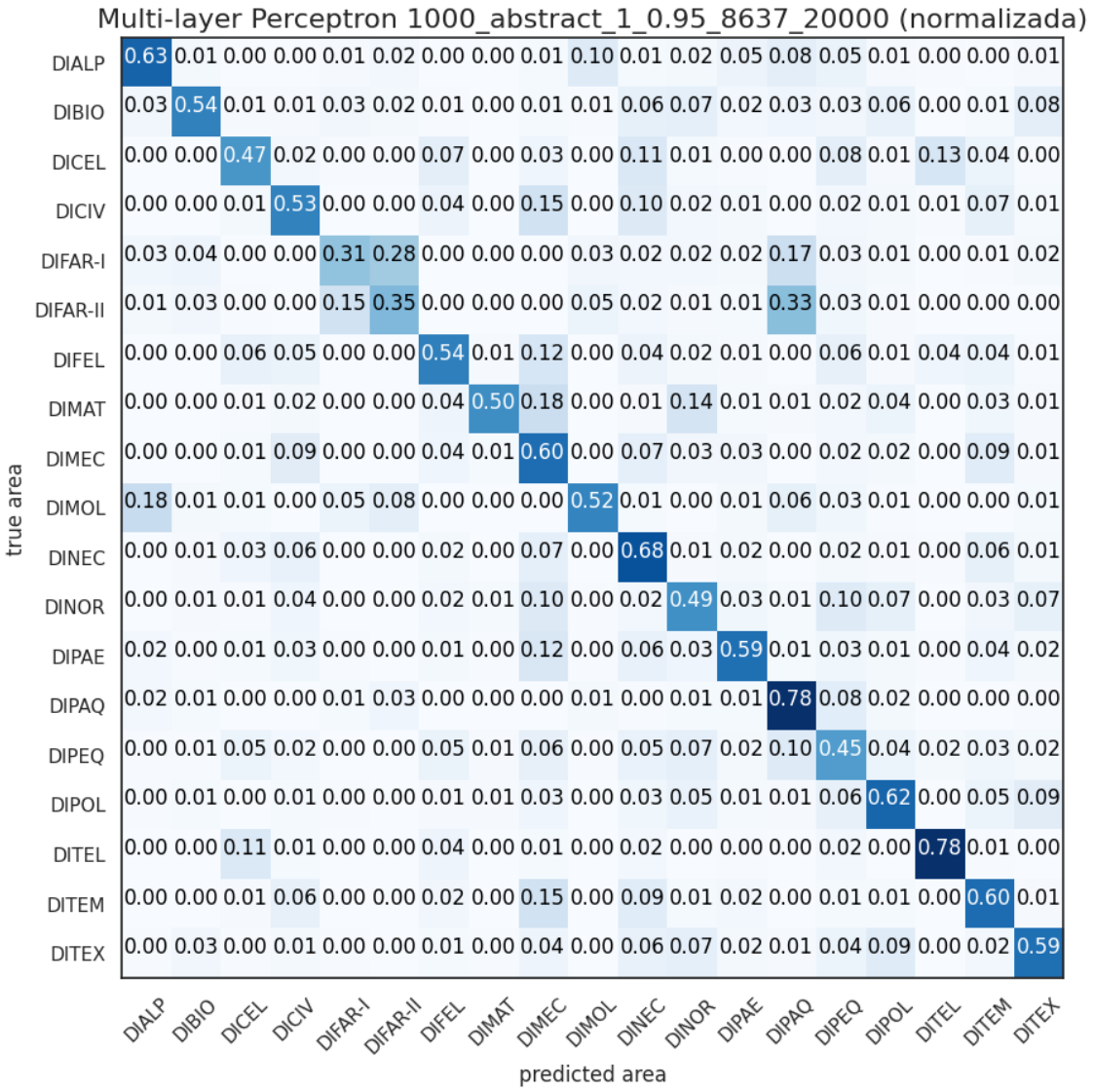

Figura 4.23: Ciclo 2 - Multi-layer Perceptron - Matriz de Confusão

Support Vector Machine - SVM - o experimento chegou em um classificador com a matriz de confusão apresentada na figura 4.24 e com acurácia de 57,88%, a partir de *features* TF-IDF baseadas exclusivamente no resumo de 20.000 observações. Sendo que os hiper-parâmetros estão com a seguinte configuração: {'C': 1.0, 'dual': True, 'multi_class': 'ovr', 'random_state': 8637, 'tol': 0.0001}

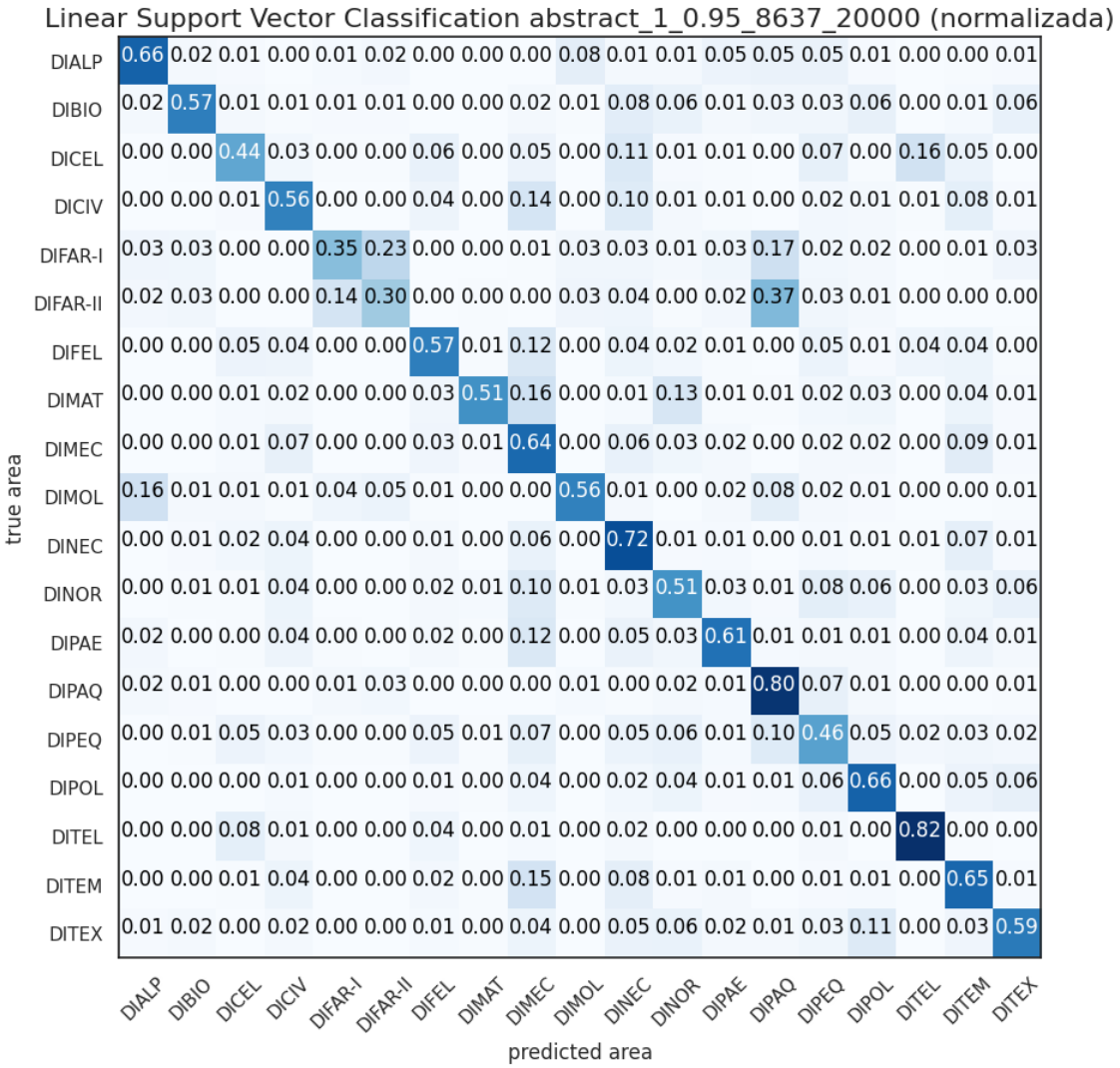

Figura 4.24: Ciclo 2 - Support Vector Machine - Matriz de Confusão

Bidirectional Encoder Representations from Transformers **- BERT -** foi feito um experimento com 5.000 e 10.000 observações, variando a taxa de aprendizado de 1e-5 até 1e-2 e conclui-se que o melhor resultado se obtém com o valor 1e-3, como mostra a figura 4.25.

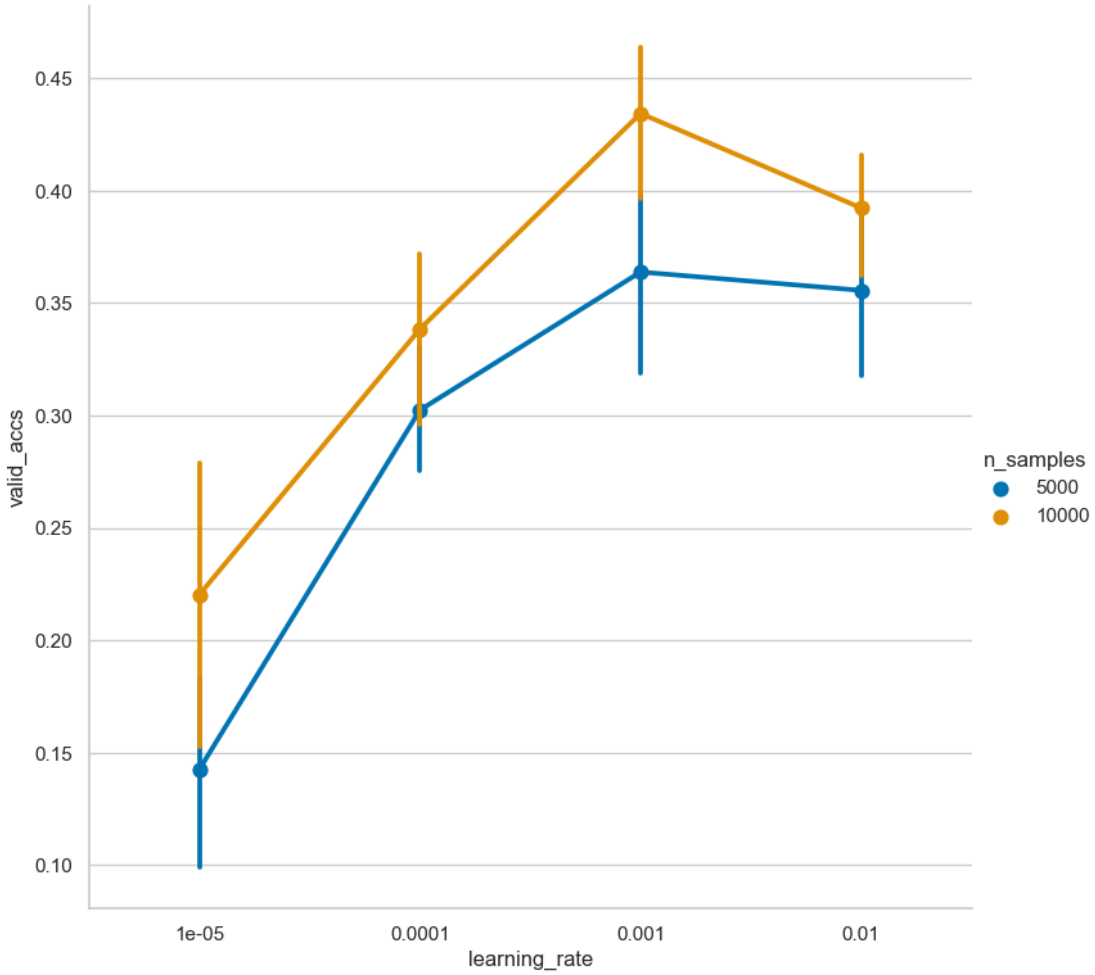

Figura 4.25: Identificação da melhor taxa de aprendizado para o BERT

4.2.2 Demonstração

4.2.2.1 Algoritmos baseados em *features*

Durante a etapa de desenvolvimento, o SVM foi o algoritmo de melhor desempenho e o NBG foi descartado, mantendo-se o MLP por ser o algoritmo que equipa a versão 1 do "Application Router", permitindo assim que a comparação possa ser mais direta. Sendo assim, a etapa de demonstração conta com SVM e MLP configurados com o melhor conjunto de hiper-parâmetros identificado na etapa de desenvolvimento, e treinados com 90% do conjunto de dados e validados com os 10% restantes, conforme apresentado nas figuras 4.26 e 4.28. Vale registrar ainda que durante o desenvolvimento, apesar do texto do título estar repetido no resumo em 65% das observações,

sugerindo que poderia utilizar apenas o resumo, o resultado final de acurácia indicava um desempenho melhor quando utilizava-se a combinação do título e do resumo. Na etapa de demonstração, foi comparado o desempenho dos classificadores treinados apenas com o texto do resumo, e daqueles treinados com o texto da combinação do título+resumo. Sendo que o melhor desempenho sempre foi obtido pelo classificador treinado com a combinação.

MLP - Multi-Layer Perceptron

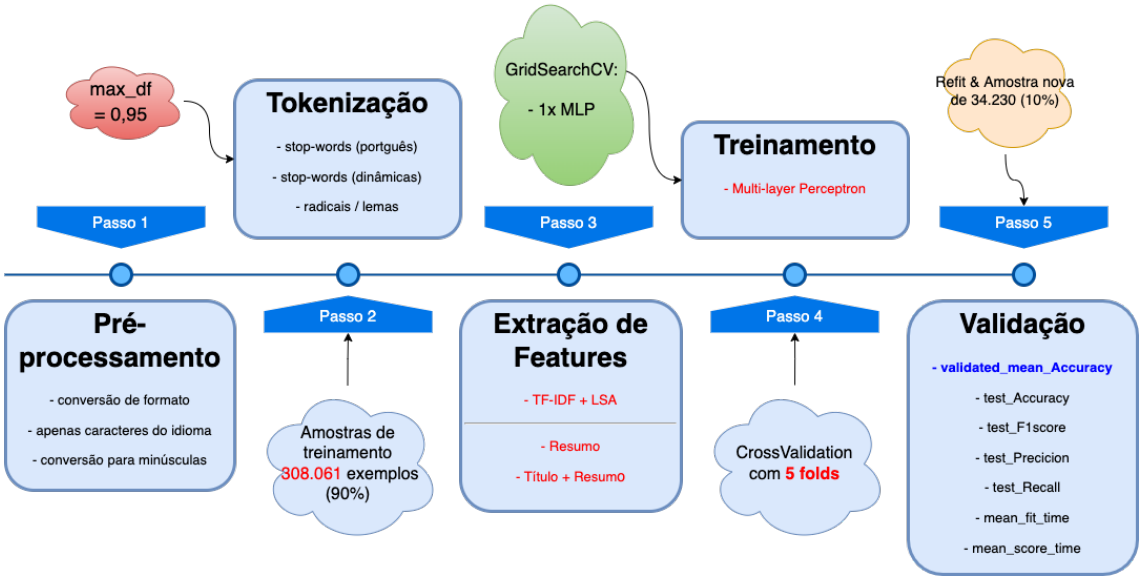

Figura 4.26: Ciclo 2 - Demonstração - MLP

O experimento chegou em um classificador com a seguinte configuração de hiperparâmetros: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 50, 19), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2000, 'solver': 'adam'}, com a matriz de confusão apresentada na figura 4.27 e com acurácia média balanceada de 56,66%, treinado a partir de *features* LSA com 1.000 dimensões, baseadas na combinação do título com o resumo de 90% observações (308.061 observações), e validados com 10% das observações.

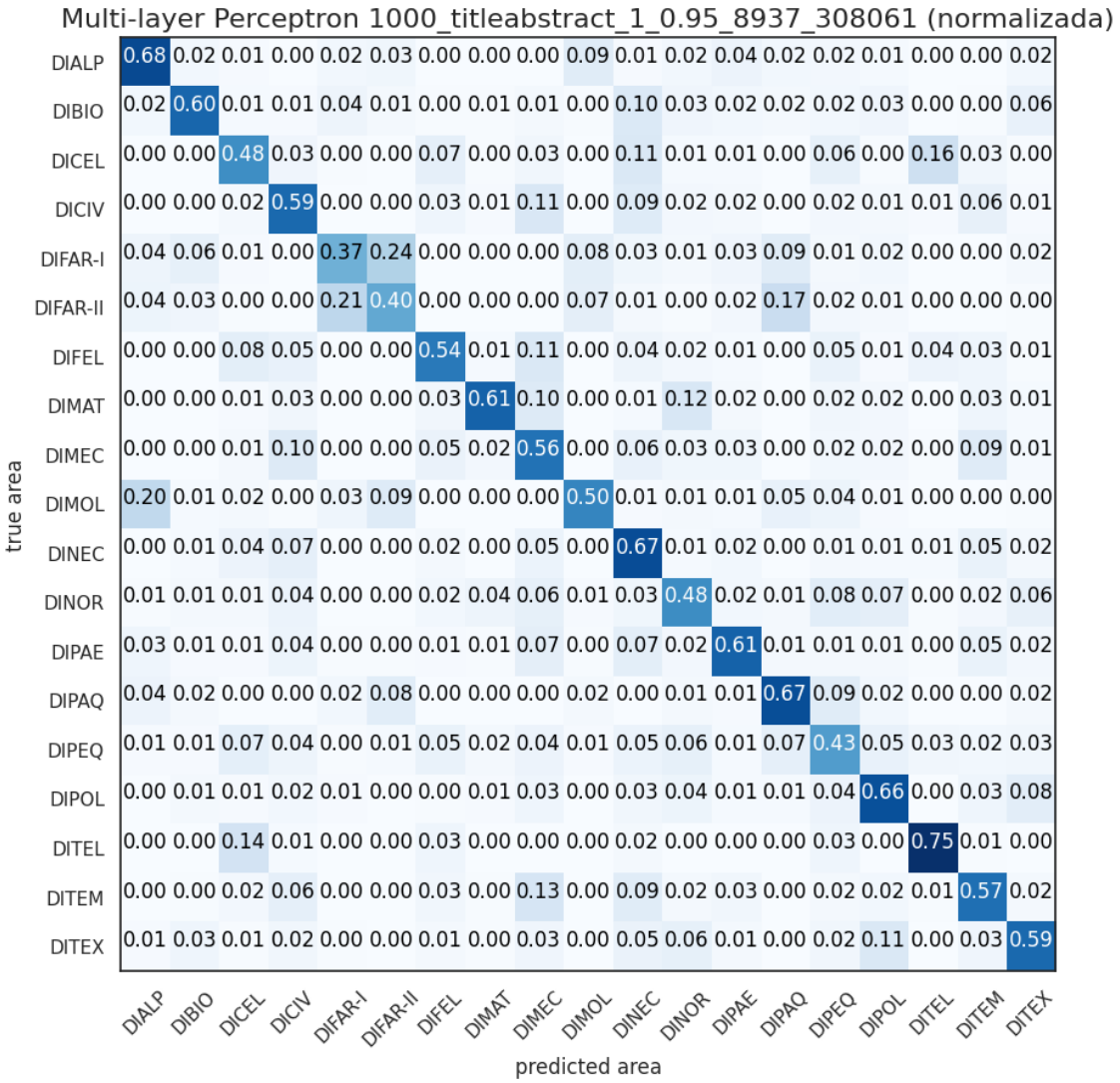

Figura 4.27: MLP - Multi-Layer Perceptron

SVM - Support Vector Machine

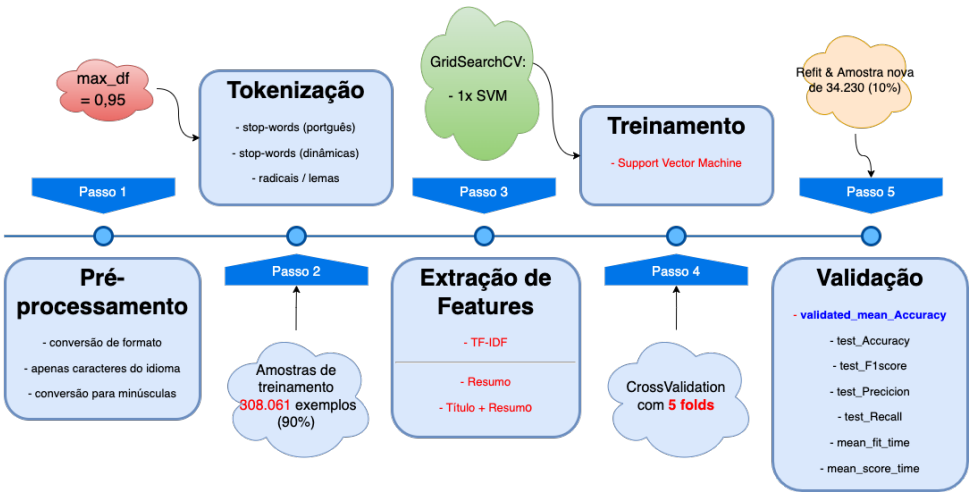

Figura 4.28: Ciclo 2 - Demonstração - SVM

O experimento chegou em um classificador com a seguinte configuração de hiper-parâmetros: {'C': 1.0, 'dual': True, 'multi_class': 'ovr', 'random_state': 8937, 'tol': 0.0001}, com a matriz de confusão apresentada na figura 4.29 e com acurácia média balanceada de 65,41%, treinado a partir de *features* TF-IDF baseadas na combinação do título com o resumo de 90% observações (308.061 observações), e validados com 10% das observações.

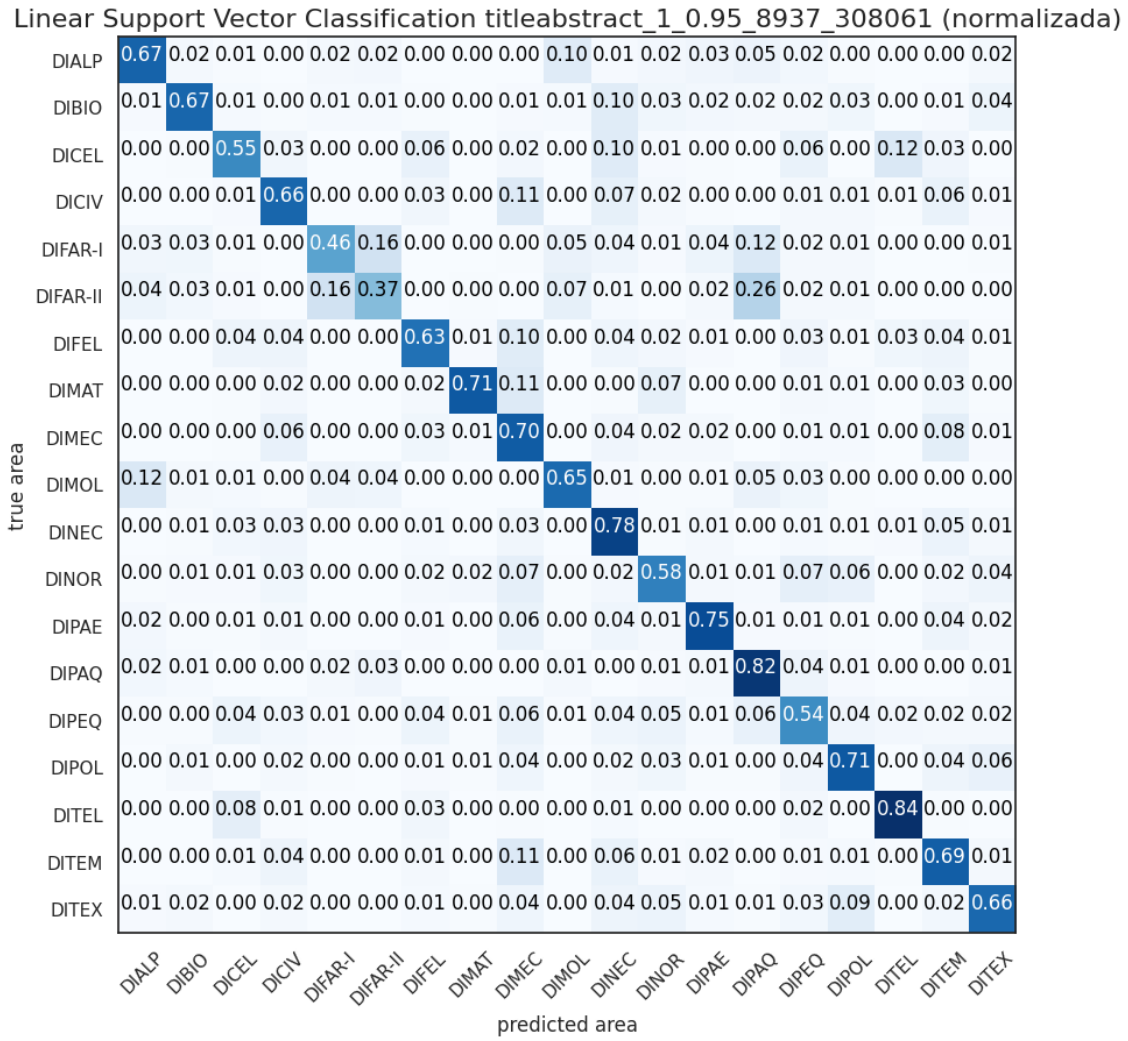

Figura 4.29: SVM - Support Vector Machine

4.2.2.2 Algoritmo baseado em ajuste-fino

Bidirectional Encoder Representations from Transformers (BERT)

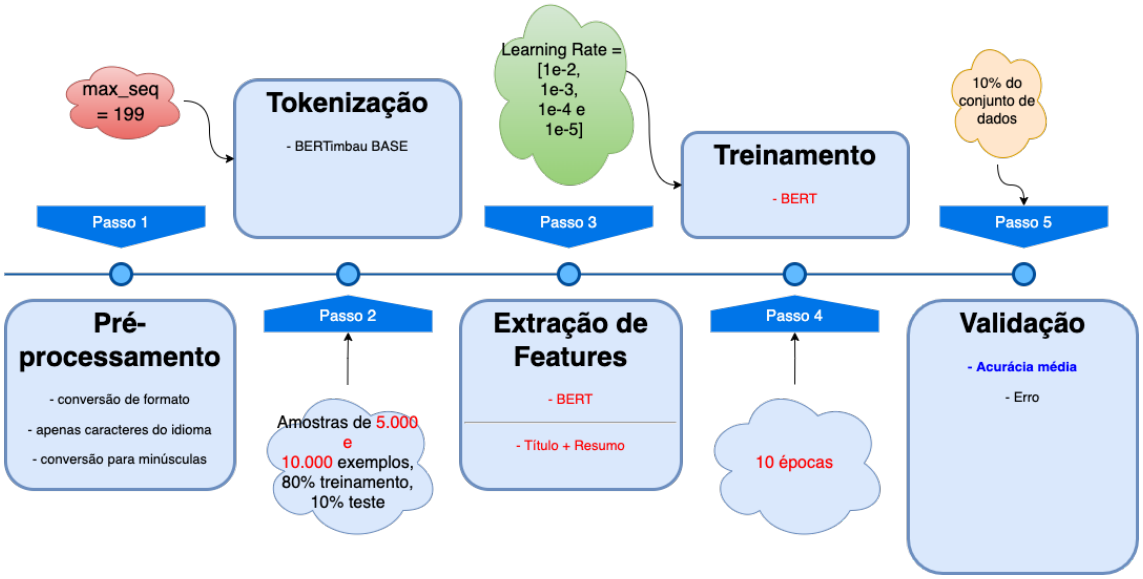

Figura 4.30: Ciclo 2 - Demonstração - BERT

Por se tratar do estado da arte atualmente, decidiu-se avaliar o BERT. Além da análise feita na etapa de desenvolvimento para identificar a taxa de aprendizado do BERTimbau mais interessante para o caso específico, foi feita também uma análise para identificar o tamanho máximo de sequência mais apropriado, conforme figura 4.30. A análise exploratória de dados permitiu montar a tabela 4.7 que apresenta um resumo das estatísticas e as figuras 4.31 e 4.32 apresentam a distribuição de quantidade de palavras (tokens) no texto do resumo.

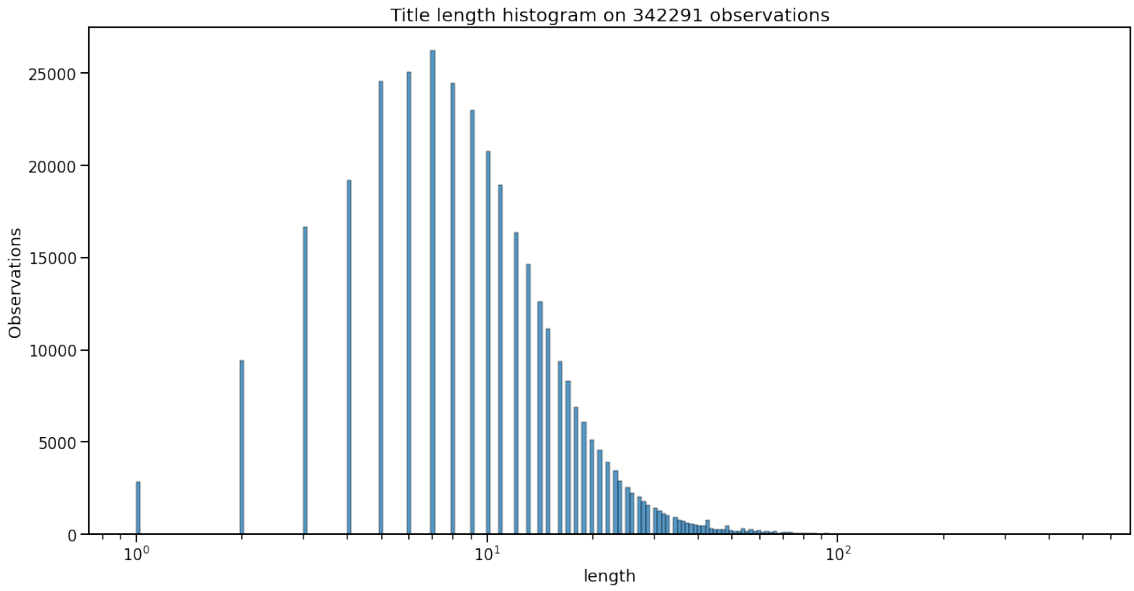

Figura 4.31: Histograma do comprimento dos títulos

Tabela 4.7: Ciclo 2 - Estatísticas dos textos

Texto	Tamanho médio	Desvio padrão	Mediana
Título	12	10	9
Resumo	130	69	121

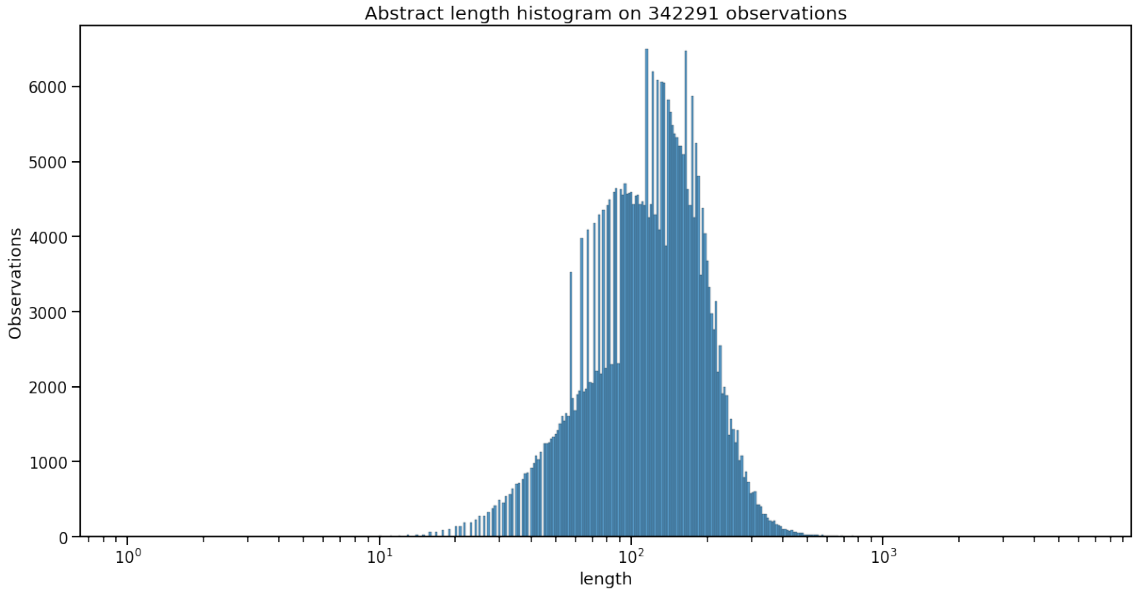

Figura 4.32: Histograma do comprimento dos resumos

Utilizou-se a implementação *"neuralmind/bert-base-portuguese-cased"* para treinar duas versões do BERTimbau BASE como classificador. O resultado pode ser observado na tabela 4.8 e nas figuras 4.33 e 4.34, com as respectivas matrizes de confusão.

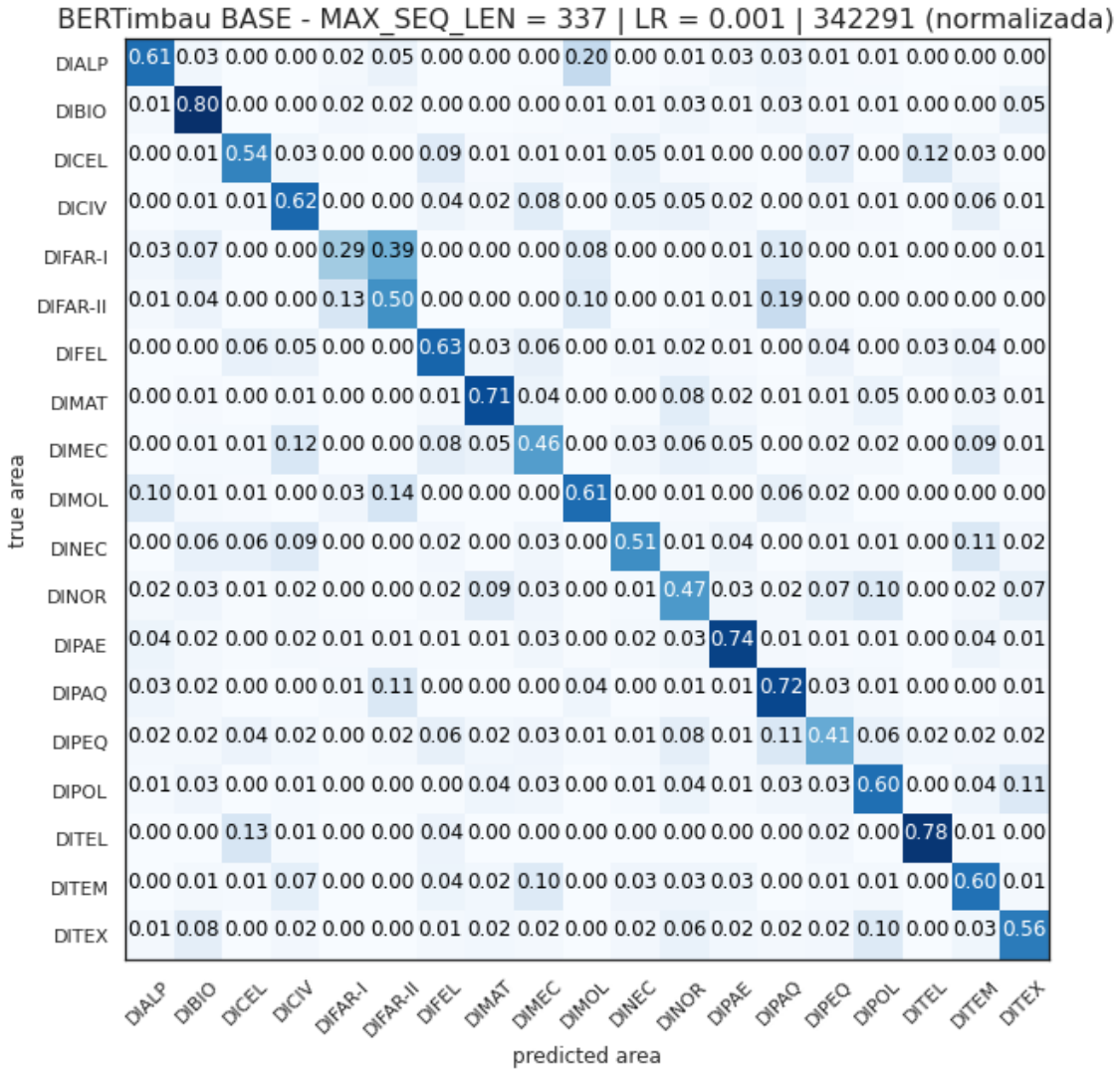

Figura 4.33: bert-base-portuguese-cased com MAX_SEQ_LAN = 337

Tabela 4.8: Ciclo 2 - Performance do BERTimbau, treinado com todo o conjunto de dados

Model	MAX_SEQ_LEN	Acurácia	Tempo de treinamento
bert-base-portuguese-cased	199	58,24%	5h 00min
bert-base-portuguese-cased	337	57,74%	9h 03min

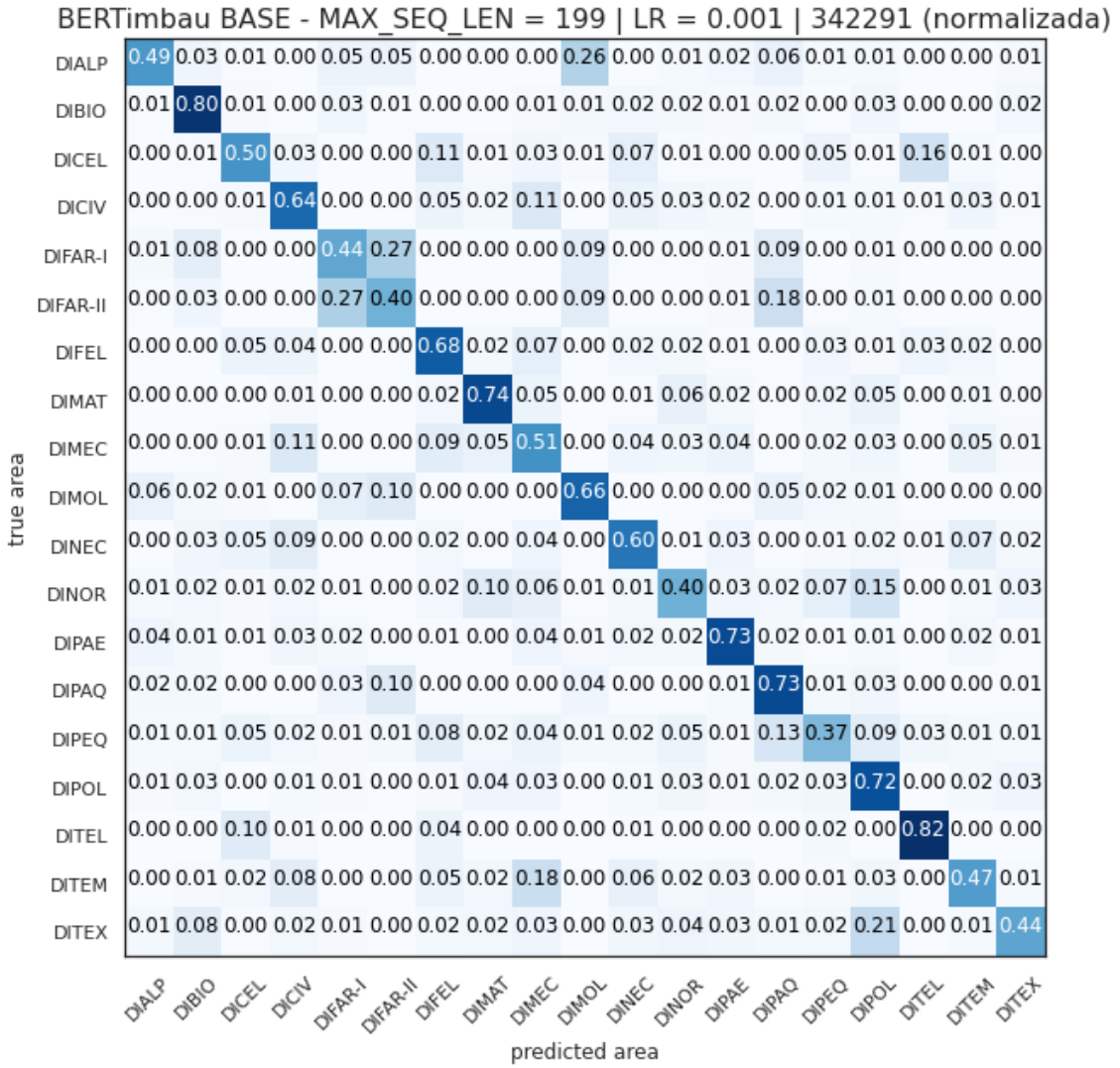

Figura 4.34: bert-base-portuguese-cased com MAX_SEQ_LAN = 199

4.2.3 Avaliação

Conforme pode ser visto na figura 1.4, a data de depósito das observações presentes no conjunto de dados INPI-BR se distribui entre os anos de 1992 e 2015. A análise exploratória de dados revelou que a última RPI que publicou informação sobre o resumo dos pedidos foi a de número 2.335 de 06 de outubro de 2015, a partir da RPI 2336 de 13 de outubro de 2015 (inclusive) não se encontra mais a informação relativa ao resumo.

Para efetuar a etapa de avaliação do ciclo 2, optou-se por produzir uma versão atualizada do INPI-BR, recuperando os resumos disponíveis através do *Open Patent Service* (OPS[8]) do *European Patent Office (EPO)*.

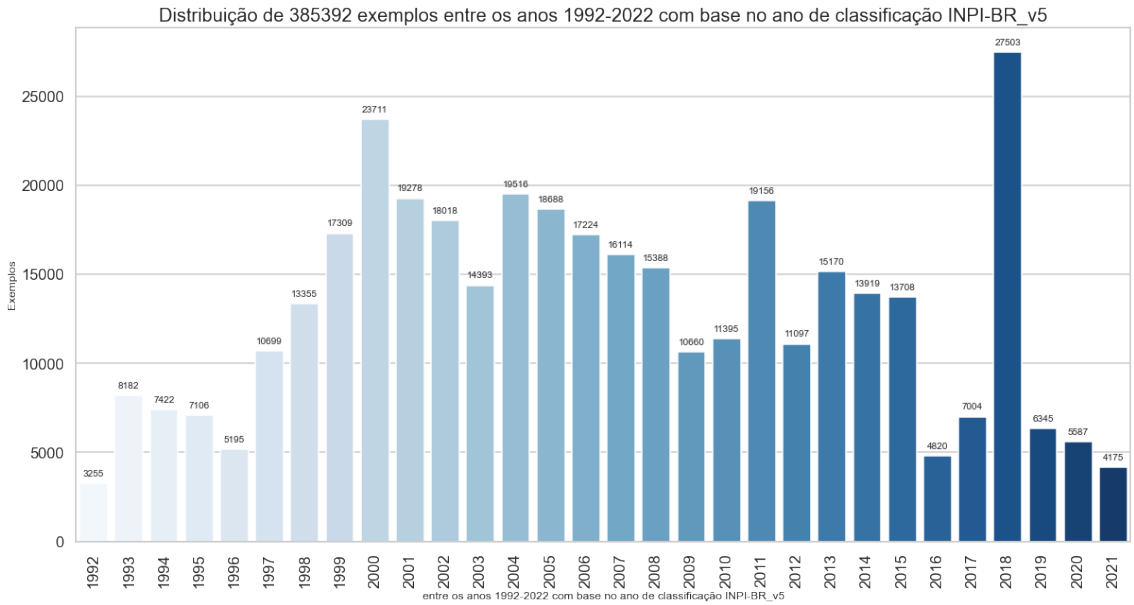

Figura 4.35: inpi-br_v5 - distribuição segunda a data de classificação

O conjunto de dados atualizado conta com +385k exemplos, e a distribuição dos exemplos pode ser visualizada nas figuras 4.35, 4.36 e 4.37, respectivamente: distribuição segundo a data de classificação dos pedidos, distribuição segunda a data de depósito dos pedidos e distribuição segunda a área de exame.

[8] `https://www.epo.org/ops`

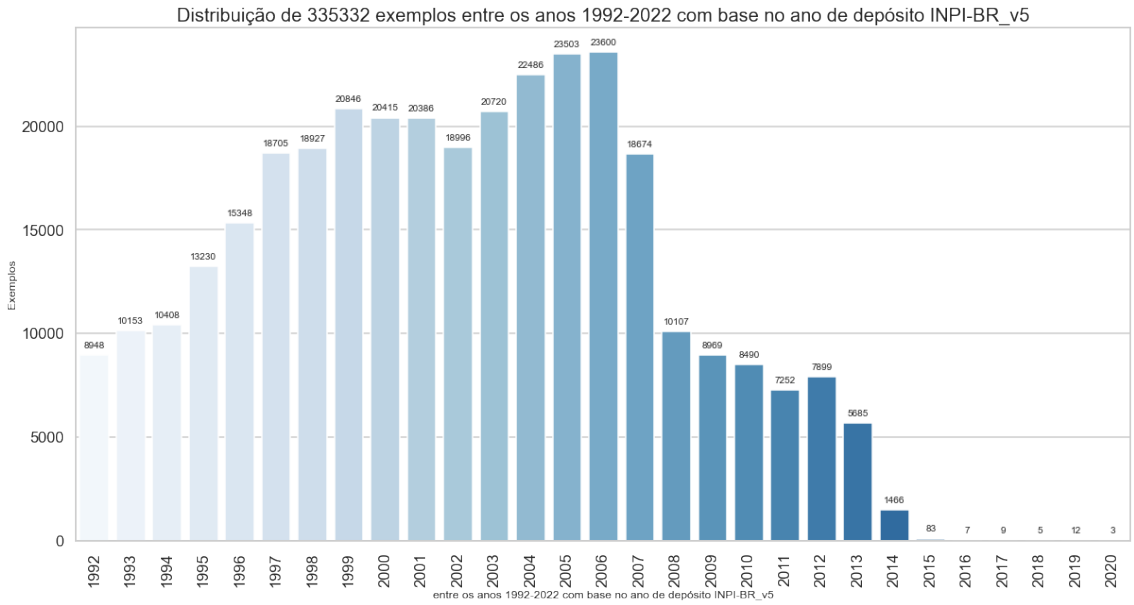

Figura 4.36: inpi-br_v5 - distribuição segunda a data de depósito

A diferença no número de observações registrada nas imagens e as aproximadamente 385k observações do conjunto de dados, presentes na distribuição de classes 4.37, se deve ao fato de as observações conterem, necessariamente, apenas resumo e área de exame. Sendo assim, muitas observações não contam com a data de depósito, e/ou com a data de classificação, além disso, existem erros de digitação que registram datas em anos como 1900. Por exemplo, as observações com valores nulos ou fora de escala nos campos relativos as datas de depósito e classificação foram descartadas nas respectivas distribuições.

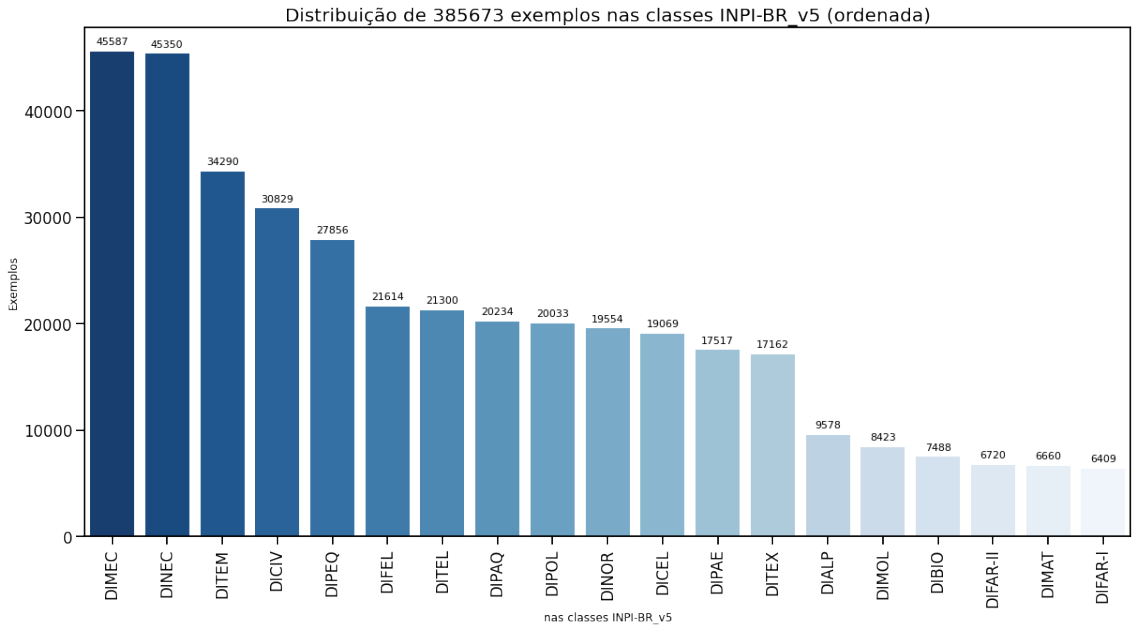

Figura 4.37: inpi-br_v5 - distribuição segunda a área de exame técnico

Como o BERT teve um desempenho semelhante a do Application Router v1, escolheu-se treinar o SVM com TF-IDF e o MLP com LSA como sendo os algoritmos escolhidos como Falcon Heavy e Falcon 9, respectivamente. A partir do INPI-BR_-v2, foram separadas as novas observações, obtidas com a recuperação de resumos disponíveis no EPO e feito um experimento *in vitro*, para estimar a acurácia esperada para as duas versões de classificador. O resultado pode ser verificado na tabela 4.9.

Tabela 4.9: Ciclo 2 - Acurácia média estimada - Avaliação

Classificador	Acurácia	Tempo de treinamento (refit)
Falcon Heavy (SVM | TF-IDF)	62,32%	56,87 segundos
Falcon 9 (MLP | TF-IDF + LSA)	56,28%	2,34 horas

Vale destacar que além do tempo empregado no treinamento do algoritmo de classificação, existe ainda o investimento de tempo para o cálculo do TF-IDF, que custou aproximadamente 6,92 horas, e do LSA, que custou 5,64 minutos, na parcela do conjunto de dados utilizado para o treinamento. Na tabela 4.9 a coluna "Tempo de treinamento (refit)" não inclui esses tempos.

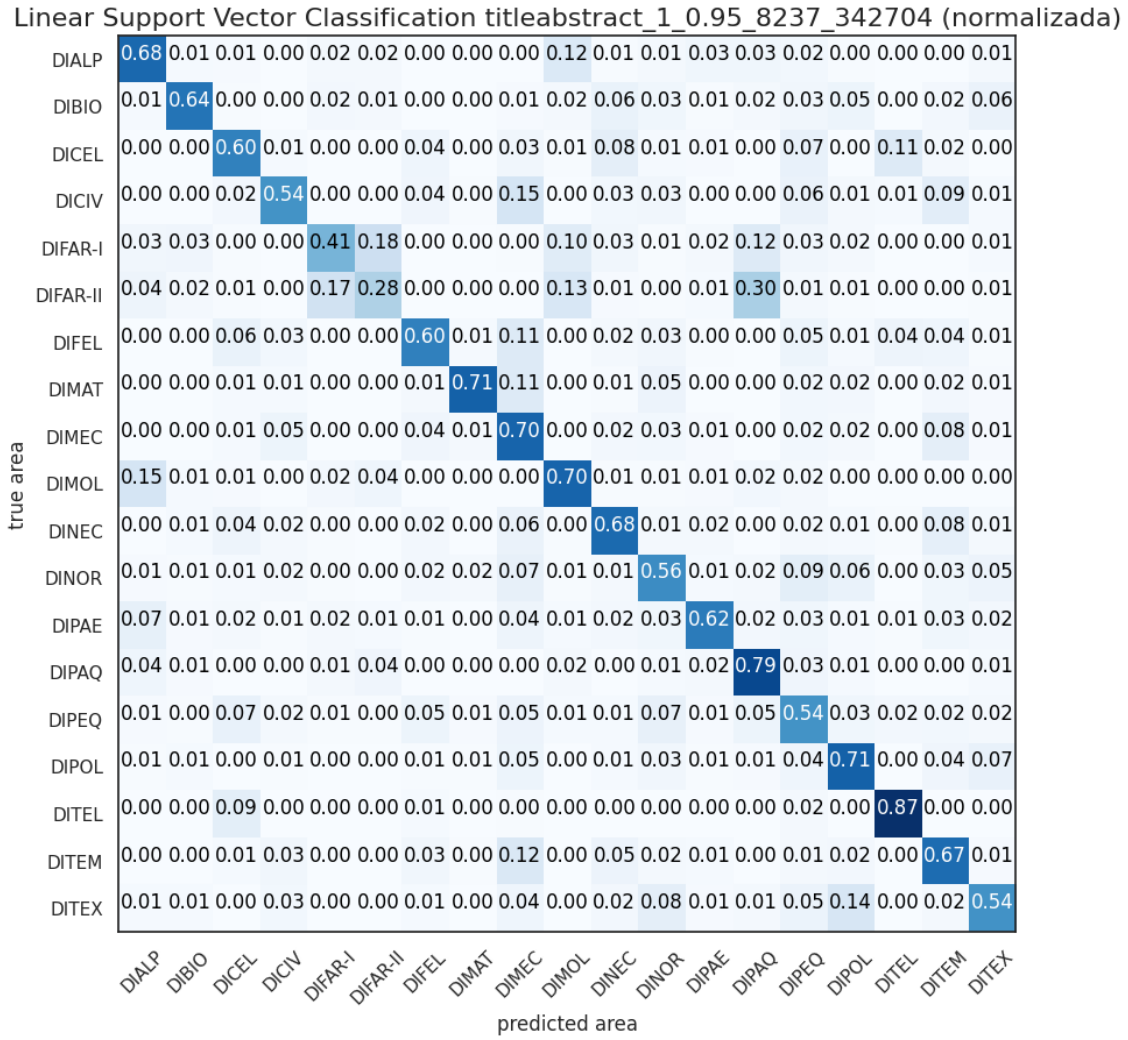

Figura 4.38: Matriz de confusão - Falcon Heavy

Outro destaque é a utilização da matriz de confusão como ferramenta de análise da qualidade da classificação. Uma característica recorrente nas matrizes de confusão apresentadas nesse trabalho é o "quadrado" formado pelas previsões corretas e erradas das classes DIFAR-I e DIFAR-II, um exemplo é a matriz de confusão do Falcon Heavy (SVM com *features* TD-IDF) apresentada na figura 4.38. Ele reflete a dificuldade de separação entre as classes terminando por classificar muito frequentemente exemplos da classe A como sendo B, e vice-versa. No caso deste trabalho, a dificuldade existe na prática por se tratarem de 2 áreas responsáveis pelo exame de patentes relativas a fármacos. Enquanto o restante das classes apresenta um comportamento mais compatível com a diagonal formada na matriz de confusão de um classificador perfeito.

4.2.3.1 Revisão das conjecturas

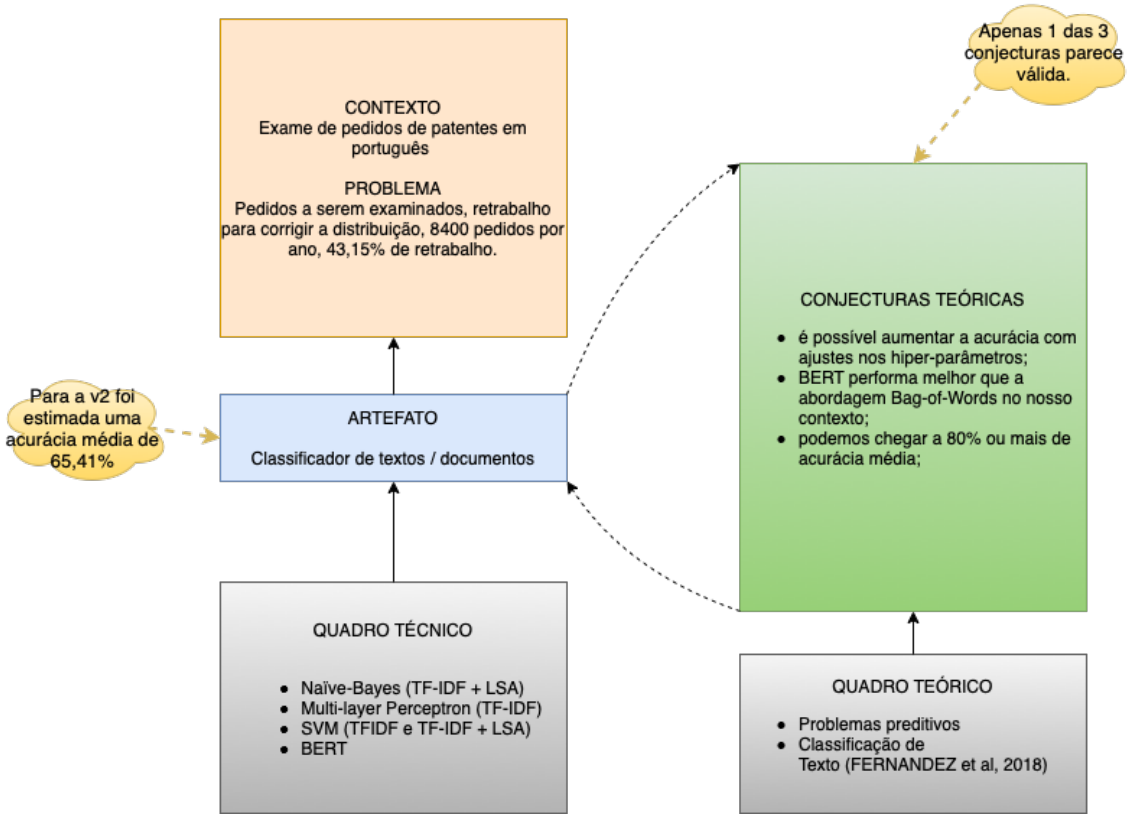

Figura 4.39: Mapa DSR - segundo ciclo de pesquisa - após experimentos

A análise continuou com a revisão das conjecturas da proposta do ciclo 2, o mapa atualizado pode ser visto na figura 4.39.

- "É possível aumentar a acurácia com ajustes nos hiper-parâmetros" - essa conjectura foi confirmada pela acurácia média obtida pelo MLP ajustado com 3 camadas, no ciclo 2 dos experimentos, em comparação com o MLP ajustado com 2 camada, do ciclo 1 dos experimentos.

- "BERT tem desempenho melhor que a abordagem Bag-of-Words no nosso contexto" - essa conjectura foi invalidada. Ocorreu o treinamento do BERTimbau BASE com o conjunto completo de dados e se obteve um máximo de 58,24% de acurácia média, quando utilizado o texto do resumo apenas. Não foi possível determinar o motivo do baixo desempenho do BERT, mas acredita-se que tenha relação com o vocabulário técnico dos textos do título e do resumo. Um

experimento futuro pode atualizar os parâmetros do BERTimbau utilizando os textos do conjunto de dados INPI-BR;

- "É possível chegar a 80% ou mais de acurácia média" - a melhor acurácia média foi estimada em 65,41%. Mas ainda é possível treinar o BERT-LARGE e tentar outras arquiteturas para o MLP.

4.3 Discussão

Os experimentos deste trabalho sugerem a confirmação do que foi dito por FALL e BENZINEB (2002) a respeito do SVM ter desempenho melhor que as redes neurais. No caso desse experimento, o SVM obteve o melhor desempenho, utilizando apenas TF-IDF como *features*, e também foi melhor que o MLP quando utilizou TF-IDF + LSA como *features*.

Segundo FALL e BENZINEB (2002), no EPO, os especialistas estavam divididos em 44 diretorias e 549 times, cada um responsável por processar um grupo de símbolos IPC. A acurácia da distribuição manual feita pelo pessoal administrativo foi anteriormente medida como sendo 81,2%, o objetivo do classificador automático era superar essa acurácia. No INPI, não foi feito nenhum estudo para identificar a acurácia da distribuição manual, uma vez que já estava em operação uma distribuição automatizada.

MATHIASSEN e ORTIZ-ARROYO (2006) combinaram diversos algoritmos de aprendizado para categorizar pedidos de patentes, utilizando diferentes formas de representação para os pedidos presentes no conjunto de dados WIPO-alpha modificado, e concluiu que a melhor combinação obteve F1-score de 0.8667, 6,51% melhor que o melhor classificador individual, que atingiu F1-score de 0.8137. Este trabalho também avaliou a métrica f1-score, mas optou por utilizar apenas a acurácia média balanceada para comparar os classificadores. Além disso, não foi explorada a técnica de combinar diversos algoritmos, algo que pode ser estudado em trabalhos futuros.

De acordo com CHEN e CHANG (2012) o SVM apresenta desempenho muito bom na classificação de documentos de patentes em inglês e alemão, conforme relataram FALL *et al.* (2004), em geral os classificadores baseados no SVM têm capacidade de generalização em dados desconhecidos (palavras que não existiam no conjunto de dados de treinamento) melhor que algoritmos de aprendizado baseados em distância e similaridade, como kNN e árvores de decisão. Os experimentos deste trabalho sugerem que o bom desempenho do SVM se mantém em documentos de patentes em português.

Segundo GOMEZ e MOENS (2014), o SVM também pode suportar um elevado número de *features*, sendo possível não utilizar redução de dimensionalidade de maneira satisfatória. Os experimentos deste trabalho também sugerem a confirmação dessa característica, uma vez que o melhor desempenho foi atingido sem recorrer a redução de dimensionalidade.

Conforme o relatório de FIÉVET (2018), o desempenho do IPCCAT não é medido com base na acurácia, ele vem sendo medido utilizando-se a precisão, calculada com base na previsão de 3 classes, das quais basta que uma esteja correta. Seguindo essa estratégia de medição o IPCCAT atingiu níveis de precisão da ordem de 80% em 2018, utilizando aproximadamente 8.000 redes neurais para isso. O relatório também informa que a utilização de n-gram teve efeito colateral. Os experimentos deste trabalho não podem ser diretamente comparados com as avaliações do IPCCAT, primeiro porque se trata de uma tarefa de classificação com um número muito diferente de classes, e segundo porque a precisão não é a métrica mais relevante para a tarefa de distribuição. É preciso que apenas a primeira sugestão do classificador seja considerada.

FIÉVET e GUYOT (2018) revelam que os campos de texto utilizados para o desenvolvimento são o título e o resumo, e a expectativa que redes neurais profundas não tragam benefícios sobre as redes neurais tradicionais. Os experimentos deste trabalho sugerem a confirmação dessa afirmação. O BERTimbau BASE (rede neural

profunda) não obteve um desempenho melhor que a primeira versão do Application Router.

FIÉVET e GUYOT (2018) ainda revelam que o texto completo do pedido de patente só foi avaliado no experimento inicial em 2003. Durante todos esses anos a OMPI construiu não só o conjunto de dados WIPO-alpha, mas também outras versões como a gama e a delta. Porém, em 01 de junho de 2022, uma consulta ao site que oferecia esses conjuntos de dados[9] mostrou que a iniciativa foi descontinuada em 2021. O conjunto de dados INPI-BR[10] pode ajudar a superar a falta de dados para novos experimentos em português.

Para KOWSARI *et al.* (2019), a etapa de extração de *features* pode ser feita de inúmeras formas. Para GOMEZ (2019) as técnicas TF-IDF e entropia produzem resultados similares, sendo que o TF-IDF é calculado mais rápido. GOMEZ (2019) também relata que tanto o TF-IDF quanto a entropia produzem resultados melhores que a representação *word embedding: Word2Vec*. Os experimentos deste trabalho sugerem a confirmação de que o TF-IDF é uma representação mais interessante para os documentos de patentes, nos experimentos utilizando o BERTimbau BASE, seja com 199 ou 337 tokens de tamanho máximo de sequência, o desempenho sempre foi inferior ao SVM e similar ao MLP.

FERREIRA (2021) menciona um conjunto de dados em português fornecido pelo INPI de Portugal aos participantes de um concurso, que contava com 36.100 patentes entre os anos de 2007 e 2013, contendo textos de reinvidicações e descrição. Este trabalho produziu um conjunto de dados com mais de 380.000 exemplos contendo texto de título e resumo em português do Brasil, relativo a pedidos de patentes depositados entre 1992 e 2020, e disponível publicamente em sua versão csv, bem como o código fonte necessário a extração e construção diretamente a partir da fonte primária no site do INPI do Brasil.

[9] `https://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/index.html`

[10] `https://github.com/rafaelscnunes/app-router/blob/main/dataset/inpi-br_v5.csv`

FERREIRA (2021) também comunica sobre o melhor desempenho ter sido atingido pelo BERTimbau e pelo LinearSVC com TF-IDF. Este trabalho sugere a confirmação de que o SVM (LinearSVC) utilizando *features* baseadas em TF-IDF oferece o melhor desempenho de classificação para patentes em português, os resultados 4% maiores do BERTimbau não se repetiram neste trabalho. FERREIRA (2021) atingiu um f1-score balanceado de 0,608 com LinearSVC, e de 0,636 com o BERTimbau. Este trabalho obteve f1-score de 0,680 para o LinearSVC com TF-IDF, e 0,570 para o BERTimbau BASE. Não ficou claro durante a leitura da dissertação de FERREIRA (2021) qual foi a versão do BERTimbau utilizada. Novos experimentos podem ser feitos em trabalhos futuros treinando o BERTimbau LARGE e também atualizando os parâmetros do BERT para o texto presente nos pedidos de patentes.

4.4 Ambiente Operacional

O código fonte dos experimentos efetuados ao longo desse trabalho está disponível no GitHub[11] através de notebooks e módulos, todos em Python 3[12] & Jupyter[13]. Os dados utilizados para treinamento e teste dos classificadores, no ciclo 2, foram produzidos a partir de dados públicos do INPI[14], e também estão disponíveis no GitHub, assim como os módulos Python 3 responsáveis pela construção do conjunto de dados.

No primeiro ciclo, o código-fonte foi executado exclusivamente em hardware próprio do INPI, já no segundo ciclo, utilizou-se máquinas virtuais do serviço Google Cloud Platform, e também máquinas próprias do INPI.

As máquinas criadas na GCP foram do tipo "compute-optimized", rodavam Debian 11 e Python 3.9, com as especificações abaixo:

1. c2-standard-8 (8 vCPU, 32 GB memory) | onde foram executados os experimentos: cycle2_tfidf_vs_lsa e cycle2_est_selection | até que ocorreu um

[11] `https://github.com/rafaelscnunes/app-router`
[12] `https://www.python.org`
[13] `https://jupyter.org`
[14] `http://revistas.inpi.gov.br/rpi/`

problema para treinar o SVC com dataset com 15.000 observações. Não tinha memória suficiente.

2. c2-standard-30 (30 vCPU, 120 GB memory) | onde foram executados os experimentos: cycle2_features e cycle2_estimators (corrigido para LinearSVC)| a correção foi necessária porque o GridSearchCV estava com uma lista de hyper-parâmetros que gerava muitas execuções, e porque a abordagem One-vs-One do SVC padrão precisa de mais memória para treinar, que a abordagem One-vs-Rest, padrão no LinearSVC.

3. n1-standard-8 (8 vCPU, 30 GB memory, 1x GPU NVIDIA Tesla T4) | que foi usado para conectar e executar os notebooks do Colaboratory[15] com a BASE do BERTimbau[16]

A máquina virtual utilizada no INPI contava com 24 vCPUs e 128 GB de RAM, rodando Red Hat Enterprise Linux 7.9 e Python 3.6.8

Vale destacar que tanto o TruncatedSVD quanto o GridSearchCV se beneficiam grandemente da disponibilidade de vCPU, por isso a troca de uma MV de 8 vCPU para uma de 30 vCPU.

O custo do ciclo 2 de pesquisa foi de aproximadamente R$ 1.000,00 por cerca de 15 dias de uso das máquinas citadas acima.

Os testes foram iniciados para identificação da melhor combinação de coluna de texto, técnica de extração de features, algoritmo e hiper-parâmetros, utilizando a implementação *sklearn.svm.SVC*, por ser a implentação que permitia calcular as probabilidades para cada classe e então aferir a métrica AUC. Entretanto, durante os experimentos ficou evidente que o tempo de treinamento seria um problema, então optou-se por utilizar outra implementação do SVC o LinearSVC.

Durante a fase de demonstração, no treinamento dos estimadores escolhidos (SVM e MLP) com o conjunto completo dos dados, foi identificado um defeito no módulo python responsável pelo treinamento: entre a obtenção das *features* através

[15] `https://colab.research.google.com/`
[16] `https://huggingface.co/neuralmind/bert-base-portuguese-cased`

do TfidfVectorizer() e o cálculo do LSA através do TruncatedSVD estava ocorrendo uma conversão da matrix X do formato denso para o formato esparso. Essa conversão tornava o código dependente de uma disponibilidade de RAM da ordem de 400 GB, o que provocou parada por indisponibilidade dessa RAM. O problema foi resolvido retirando a conversão, desnecessária, uma vez que as funções da sklearn estão preparadas para trabalhar com matriz densa ou esparsa.

Capítulo 5

Conclusão

Avaliaram-se formas de seleção dos textos, formas de extração das features, algoritmos de aprendizado de máquina e hiper-parâmetros diferentes para os algoritmos: NBG, MLP, SVM e BERT. A pesquisa concluiu que utilizar a combinação dos textos do título e do resumo, e utilizar apenas o TF-IDF para treinar o SVM, implementado pela classe *sklearn.svm.LinearSVC* produz o melhor resultado de acurácia média ponderada. Para o caso de estudo, o classificador final atingiu acurácia média estimada em 62,32%.

Este trabalho teve início em 2017, seguindo a metodologia proposta na figura 3.1. O primeiro ciclo de pesquisa contou com uma etapa de "Demonstração" que estimou uma acurácia média de 59% de pedidos distribuídos corretamente pelo "Application Router v1".

Em abril de 2018 o "Application Router v1" foi colocado em produção, e em junho de 2019, foram coletados para análise dados de 1 ano de operação que permitiram a apuração de uma acurácia média de 56,85%, muito próximo da estimativa inicial.

Este resultado encerrou a etapa de "Avaliação" do primeiro ciclo de pesquisa do processo DSRM proposto, cumprindo o objetivo, e confirmando experimentalmente que é possível obter acurácia média superior a 30% utilizando exclusivamente bibliotecas livres como a Sci-kit Learn[1].

[1] `https://scikit-learn.org/`

Comparado com o desempenho do e-Distribuidor, de aproximadamente 30% de acurácia média, o aumento da eficiência do roteamento de pedidos foi mais de 90%. Lembrando que um pedido roteado incorretamente, exige do examinador tempo lendo o pedido até perceber o erro, e depois tempo enviando-o para outra área, implicando na redução de produtividade, o que impacta negativamente o esforço de redução do backlog de patentes do INPI. Assim o "Application Router v1", economizou o trabalho de redistribuição de 3.039 pedidos, reduzindo 39% do retrabalho, e contribuindo para a redução do backlog de patentes no INPI.

Estima-se que o retrabalho de cada pedido distribuído erradamente seja de 15 minutos, sendo assim os 3.039 pedidos corretamente distribuídos pelo Application Router v1 quando comparado com o e-Distribuidor representam aproximadamente 760 horas de retrabalho economizadas a cada ano.

O segundo ciclo de pesquisa permitiu elevar a acurácia média ponderada para aproximadamente 62%, correspondendo a uma expectativa de melhora de aproximadamente 10% na comparação com o primeiro ciclo. Além de disponibilizar para a comunidade acadêmica o conjunto de dados INPI-BR[2], com dados para futuros estudos relativos à classificação de patentes em português.

Como trabalhos futuros, a pesquisa identificou a possibilidade de utilizar a combinação de vários algoritmos (ensemble), a utilização da versão LARGE do BERTimbau, o treinamento do BERT utilizando o corpo de texto do conjunto de dados INPI-BR e a utilização de heurísticas e ferramentas automáticas para obter hiperparâmetros, como o BOHB[3]. Além de continuar o desenvolvimento do "Application Router" para implementar treinamento contínuo a partir dos novos dados gerados a cada dia no INPI.

[2] `https://github.com/rafaelscnunes/app-router/blob/main/dataset/inpi-br_v5.csv`
[3] `https://github.com/automl/HpBandSter`

Referências Bibliográficas

ABDELGAWAD, L., KLUEGL, P., GENC, E., et al., 2020, "Optimizing Neural Networks for Patent Classification", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 11908 LNAI, pp. 688–703. ISSN: 16113349. doi: 10.1007/978-3-030-46133-1_41.

ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T., 2012, *Learning from data*. Pasadena, CA, AMLBook New York, NY, USA:. ISBN: 978-1-60049-006-4.

ALLAHYARI, M., POURIYEH, S., ASSEFI, M., et al., 2017, "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques", *arXiv*. ISSN: 23318422.

BEKAMIRI, H., HAIN, D. S., JUROWETZKI, R., 2021, "PatentSBERTa: A Deep NLP based Hybrid Model for Patent Distance and Classification using Augmented SBERT 1-Preliminary Draft, Work in Progress", Disponível em: <https://huggingface.co/AI-Growth-Lab/PatentSBERTa/>.

BIRD, S., LOPER, E., KLEIN, E., 2001. "NLTK - Natural Language Toolkit". Disponível em: <https://www.nltk.org/index.html>.

CARDOSO-CACHOPO, A., OLIVEIRA, A. L., 2003, "An empirical comparison of text categorization methods", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 2857, pp. 183–196. ISSN: 03029743. doi: 10.1007/978-3-540-39984-1_14.

CHEN, Y. L., CHANG, Y. C., 2012, "A three-phase method for patent classification", *Information Processing and Management*, v. 48, n. 6, pp. 1017–1030. ISSN: 03064573. doi: 10.1016/j.ipm.2011.11.001. Disponível em: <http://dx.doi.org/10.1016/j.ipm.2011.11.001>.

DANISMAN, T., ALPKOCAK, A., 2008. "Feeler: Emotion classification of text using vector space model". .

DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., et al., 1990, "Indexing by Latent Semantic Analysis", *Journal of the American Society for Information Science*, v. 41, n. 6, pp. 391–407. Disponível em: <https://doi.org/10.1002/(SICI)1097-4571(199009)41:6{%}3C391::AID-ASI1{%}3E3.0.CO;2-9https://asistdl.onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-4571(199009)41:6{%}3C391::AID-ASI1{%}3E3.0.CO;2-9>.

DEVLIN, J., CHANG, M.-W., LEE, K., et al., 2018, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Google AI Language*. Disponível em: <https://arxiv.org/pdf/1810.04805.pdf>.

DRESCH, A., LACERDA, D. P., ANTUNES, J. A. V., 2014, *Design Science Research*. Porto Alegre, RS, Springer. ISBN: 978-3-319-07373-6. doi: 10.1007/978-3-319-07374-3.

DUMAIS, S., PLATT, J., HECKERMAN, D., et al., 1998. "Inductive learning algorithms and representations for text categorization". Disponível em: <https://www.microsoft.com/en-us/research/uploads/prod/1998/01/Inductive-Learning-Algorithms-and-Representations-for-Text-Categorization.pdf>.

ELISSEEFF, A., WESTON, J., 2002, "A kernel method for multi-labelled classification", *Advances in Neural Information Processing Systems*. ISSN: 10495258. doi: 10.7551/mitpress/1120.003.0092.

FALL, C. J., BENZINEB, K., 2002. "Literature survey: Issues to be considered in the automatic classification of patents". Disponível em: <https://www.wipo.int/ipc/itos4ipc/ITSupport{_}and{_}download{_}area/Documentation/presentations/wipo-categorizationsurvey.pdf>.

FALL, C. J., TÖRCSVÁRI, A., BENZINEB, K., et al., 2003, "Automated categorization in the international patent classification", *ACM SIGIR Forum*, v. 37, n. 1, pp. 10–25. ISSN: 01635840. doi: 10.1145/945546.945547. Disponível em: <http://portal.acm.org/citation.cfm?doid=945546.945547https://doi.org/10.1145/945546.945547>.

FALL, C. J., TÖRCSVÁRI, A., FIÉVET, P., et al., 2004, "Automated categorization of German-language patent documents", *Expert Systems with Applications*, v. 26, n. 2, pp. 269–277. ISSN: 09574174. doi: 10.1016/S0957-4174(03)00141-6.

FERNÁNDEZ, A., GARCÍA, S., GALAR, M., et al., 2018, *Learning from Imbalanced Data Sets*. Granada, Spain, Springer Nature Switzerland AG. ISBN: 978-3-319-98073-7. doi: 10.1007/978-3-319-98074-4. Disponível em: <https://doi.org/10.1007/978-3-319-98074-4>.

FERREIRA, Á. L. D. O. A., 2021, *PORTUGUESE PATENT CLASSIFICATION A use case of text classification using machine learning and transfer learning approaches*. Tese de Doutorado, Universidade Nova de Lisboa.

FERREIRA, M. C., 2017, *Incident Routing: text classification, feature selection, imbalanced datasets, and concept drift in incident ticket management*. Tese de Doutorado, UFRJ/COPPE/PESC.

FIÉVET, P., 2018, *9.a Report on IPC-related IT systems*. Relatório técnico, WIPO, Geneva. Disponível em: <https://www.wipo.int/ipc/itos4ipc/ITSupport{_}and{_}download{_}area/Documentation/presentations/20180208{_}IPCCE50{_}IPC{_}IT.ppt>.

FIÉVET, P., GUYOT, J., 2018. "IPCCAT-neural : automatic text categorization in the IPC". Disponível em: <https://www.wipo.int/ipc/itos4ipc/ITSupport{_}and{_}download{_}area/Documentation/presentations/20180423{_}IPCCAT{_}ICSDV{_}2018.pdf>.

GOMEZ, J. C., 2019, "Analysis of the effect of data properties in automated patent classification", *Scientometrics*, v. 121, n. 3, pp. 1239–1268. ISSN: 15882861. doi: 10.1007/s11192-019-03246-1. Disponível em: <https://doi.org/10.1007/s11192-019-03246-1>.

GOMEZ, J. C., MOENS, M.-F., 2014, "A Survey of Automated Hierarchical Classification of Patents". In: *Lecture Notes in Computer Science*, Springer, pp. 215–249. doi: 10.1007/978-3-319-12511-4_11. Disponível em: <http://link.springer.com/10.1007/978-3-319-12511-4{_}11>.

HE, H., GARCIA, E. A., 2009, "Learning from Imbalanced Data", *IEEE Transactions on Knowledge and Data Engineering*, v. 21, n. 9, pp. 1263–1284. ISSN: 1860949X. doi: 10.1007/978-3-030-04663-7_4. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5128907>.

HEVNER, A. R., 2007, "A Three Cycle View of Design Science Research", *Scandinavian Journal of Information Systems*, v. 19, n. 2, pp. 87–92. doi: 10.1007/978-1-4842-2866-1. Disponível em: <https://pdfs.semanticscholar.org/fbd9/8df286a9c03843411725bfd3691669bc6407.pdf>.

HOREV, R., 2018. "BERT Explained: State of the art language model for NLP". Disponível em: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>.

INPI, 2018, *Relatório de Atividades INPI 2018*. Relatório Técnico 1, INPI, Rio de Janeiro. Disponível em: <http://www.inpi.gov.br/sobre/estatisticas/RelatoriodeAtividades2018.pdf>.

INPI, 2021, "Manual Básico para Proteção por Patentes de Invenções, Modelos de Utilidade e Certificados de Adição", p. 102. Disponível em: <https://www.gov.br/inpi/pt-br/servicos/patentes/guia-basico/ManualbsicodePatentes20210607b.pdf>.

INPI, 2022a, *Plano de Ação 2022*. Relatório técnico, a. Disponível em: <https://www.gov.br/inpi/pt-br/governanca/planejamento-estrategico/arquivos/documentos/versao-executiva-02-06-2022{_}pa-2022.pdf>.

INPI, 2022b. "Perguntas frequentes sobre Patentes". b. Disponível em: <https://www.gov.br/inpi/pt-br/servicos/perguntas-frequentes/patentes{#}patente>.

JONES, K. S., 1972, "A statistical interpretation of term specificity and its application in retrieval", *Journal of Documentation*, v. 28, n. 1, pp. 11–21. ISSN: 00220418. doi: 10.1108/eb026526.

KALCHBRENNER, N., GREFENSTETTE, E., BLUNSOM, P., 2014, "A convolutional neural network for modelling sentences", *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, v. 1, pp. 655–665. doi: 10.3115/v1/p14-1062.

KOSTER, C., SEUTTER, M., BENEY, J., 2001, *Classifying Patent Applications with Winnow*. Relatório técnico, University of Nijmegen.

KOSTER, C. H. A., SEUTTER, M., BENEY, J., 2003, "Multi-classification of Patent Applications with Winnow". In: *LNCS 2890*, pp. 546–555, Novosibirsk, Russia, jul. Springer. Disponível em: <http://www.cs.kun.nl/pekinghttps://books.google.com.br/books?hl=en{&}lr={&}id=lqL4wL5aSkkC{&}oi=fnd{&}pg=PA1{&}dq=PSI+2003,+LNCS+2890+Springer-Verlag+Berlin{&}ots=HG{_}C9Mfxgo{&}sig=Gsl29e6nD7F08lDWV{_}oGiA0cLsY{#}v=onepage{&}q=PSI2003{%}2CLNCS2890Springer-VerlagBerlin{&}f=f>.

KOWSARI, K., MEIMANDI, K. J., HEIDARYSAFA, M., et al., 2019. "Text classification algorithms: A survey". ISSN: 20782489. Disponível em: <https://www.mdpi.com/2078-2489/10/4/150/pdf>.

LARKEY, L. S., 1999, "A Patent Search and Classification System", *ACM*. Disponível em: <http://delivery-acm-org.ez29.capes.proxy.ufrj.br/10.1145/320000/313304/p179-larkey.pdf?ip=200.130.19.157{&}id=313304{&}acc=ACTIVESERVICE{&}key=344E943C9DC262BB.36BD8EA867D3A5EB.4D4702B0C3E38B35.4D4702B0C3E38B35{&}{_}{_}acm{_}{_}=1557089108{_}48aabd344c18ac16a37f6c0cf8f978>.

LI, Y., BONTCHEVA, K., 2008, "Adapting support vector machines for f-term-based classification of patents", *ACM Transactions on Asian Language Information Processing*, v. 7, n. 2, pp. 1–15. ISSN: 15300226. doi: 10.1145/1362782.1362786.

MANNING, C. D., RAGHAVAN, P., SCHUTZE, H., 2009, *An Introduction to Information Retrieval*. Online edi ed. Cambridge, Cambridge University Press. ISBN: 0521865719. Disponível em: <https://nlp.stanford.edu/IR-book/>.

MARON, M. E., 1961, "Automatic Indexing: An Experimental Inquiry", *Journal of the ACM (JACM)*, v. 8, n. 3, pp. 404–417. ISSN: 1557735X. doi: 10.1145/321075.321084.

MATHIASSEN, H., ORTIZ-ARROYO, D., 2006, "Automatic Categorization of Patent Applications Using Classifier Combinations", *Intelligent Data Engineering and Automated Learning – IDEAL 2006*, pp. 1039–1047. ISSN: 16113349. doi: 10.1007/11875581_124.

MIKOLOV, T., CHEN, K., CORRADO, G., et al., 2013a, "Efficient Estimation of Word Representations in Vector Space", *arXiv preprint arXiv:1301.3781*, pp. 1–12. ISSN: 15324435. doi: 10.1162/153244303322533223. Disponível em: <http://arxiv.org/abs/1301.3781>.

MIKOLOV, T., CHEN, K., CORRADO, G., et al., 2013b. "Distributed Representations of Words and Phrases and their Compositionality". b.

MINAEE, S., KALCHBRENNER, N., CAMBRIA, E., et al., 2021, "Deep Learning Based Text Classification: A Comprehensive Review", *ACM computing surveys (CSUR)*, v. 54, n. 3, pp. 1–40. Disponível em: <http://arxiv.org/abs/2004.03705https://doi.org/10.1145/3439726>.

NUNES, R., AZEVEDO, R., SANTOS, C., 2017. "Automatic Patent Classification based on INPI Brazil online data". .

OMPI, 2020. "Classificação Internacional de Patentes". Disponível em: <http://ipc.inpi.gov.br/classifications/ipc/ipcpub/media/help/pt/guide.pdf>.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al., 2011, "Scikit-learn: Machine Learning in {P}ython", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830.

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A., et al., 2014, "A design science research methodology for information systems research", *Journal of Management Information Systems*, v. 24, n. 3, pp. 45–77. ISSN: 07421222. doi: 10.2753/MIS0742-1222240302.

PENNINGTON, J., SOCHER, R., MANNING, C. D., 2014, "GloVe: Global Vectors for Word Representation". In: *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pp. 1532–1543. EMNLP, oct. Disponível em: <https://aclanthology.org/D14-1162.pdf>.

PIMENTEL, M., FILIPPO, D., SANTORO, F. M., 2019, "Design Science Research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação". In: Bittencourt, I. I., Jaques, P. A., Pimentel, M., et al. (Eds.), *Metodologia de Pesquisa em Informática na Educação: Concepção da Pesquisa*, v. 1 ed., SBC, cap. 5, Porto Alegre. doi: 10.1007/978-3-319-07374-3_4. Disponível em: <https://metodologia.ceie-br.org/livro-1/>.

QIU, X., HUANG, X., LIU, Z., et al., 2011, "Hierarchical Text Classification with Latent Concepts", *49th annual meeting of the associatin of the computational Linguistics*, pp. 598–602. ISSN: 19310145.

SALTON, G., WONG, A., YANG, C. S., 1975, "AVector Space Model for Automatic Indexing", *Communications of the ACM*, v. 18, n. 11. Disponível em: <http://delivery-acm-org.ez29.capes.proxy.ufrj.br/10.1145/370000/361220/p613-salton.pdf?ip=200.130.19.157{&}id=361220{&}acc=ACTIVESERVICE{&}key=344E943C9DC262BB.36BD8EA867D3A5EB.4D4702B0C3E38B35.4D4702B0C3E38B35{&}{_}{_}acm{_}{_}=1556982769{_}8a4dc1a5493bd5c4cbc9d66c0de2c3>.

SALTON, G., BUCKLEY, C., 1988. "Term-weighting approaches in automatic text retrieval". Disponível em: <http://www.ict.nsc.ru/jspui/bitstream/ICT/1231/1/solton-1-29-03.pdf>.

SEBASTIANI, F., 2001, *Machine Learning in Automated Text Categorization*. Relatório técnico, Consiglio Nazionale delle Ricerche. Disponível em: <http://liinwww.ira.uka.de/bibliography/Ai/automated.text.categorization.html>.

SENEVIRATNE, D., GEVA, S., ZUCCON, G., et al., 2015. "A Signature Approach to Patent Classification". Disponível em: <https://eprints.qut.edu.au/95674/1/airs2015{_}signature{_}patent.pdfhttps://doi.org/10.1007/978-3-319-28940-3{_}35>.

SILLA, C. N., FREITAS, A. A., 2011, "A Survey of Hierarchical Classification Across Different Application Domains", *Data Mining and Knowledge Discovery*. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=7C0EC326B47A60B9159A6BF00B4AD105?doi=10.1.1.183.302{&}rep=rep1{&}type=pdf>.

SOUZA, F., NOGUEIRA, R., LOTUFO, R., 2020, "BERTimbau: pretrained BERT models for Brazilian Portuguese". In: *9th Brazilian Conference on Intelligent Systems, BRACIS*, Rio Grande do Sul, oct. BRACIS.

TRAVASSOS, G. H., BARROS, M., 2003, "Contributions of In Virtuo and In Silico Experimentes for the Future of Empirical Studies in Software Engineering", *2nd Workshop in Workshop Series on Empirical Software Engineering The Future of Empirical Studies in Software Engineering*, v. 1, n. January, pp. 1–14.

VASWANI, A., SHAZEER, N., PARMAR, N., et al., 2017, "Attention is all you need", *Advances in Neural Information Processing Systems*, pp. 6000–6010. ISSN: 10495258. Disponível em: <https://arxiv.org/pdf/1706.03762.pdf>.

WU, X., KUMAR, V., ROSS, Q. J., et al., 2007. "Top 10 algorithms in data mining". ISSN: 02191377.

WU, Y., SCHUSTER, M., CHEN, Z., et al., 2016. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". Disponível em: <http://arxiv.org/abs/1609.08144>.

YANG, Y., LIU, X., 1999, "A re-examination of text categorization methods", *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*, pp. 42–49. doi: 10.1145/312624.312647.

ZHU, Y., KIROS, R., ZEMEL, R., et al., 2015, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books", *Proceedings of the IEEE International Conference on Computer Vision*, v. 2015 Inter, pp. 19–27. ISSN: 15505499. doi: 10.1109/ICCV.2015.11.