Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Estatística

Dissertação de Mestrado

Clusterização de textos aplicada ao tratamento de dados jurídicos desbalanceados

por

Lucas José Gonçalves Freitas

Brasília, 09 de Fevereiro de 2023

Clusterização de textos aplicada ao tratamento de dados jurídicos desbalanceados

por

Lucas José Gonçalves Freitas

Dissertação apresentada ao Departamento de Estatística da Universidade de Brasília, como requisito parcial para obtenção do título de Mestre em Estatística.

Orientadora: Prof. Dra. Thaís Carvalho Valadares Rodriques

Brasília, 09 de Fevereiro de 2023

Dissertação submetida ao Programa de Pós-Graduação em Estatística do Departamento de Estatística da Universidade de Brasília como parte dos requisitos para a obtenção do título de Mestre em Estatística.

Texto aprovado por:

Prof. Thaís Carvalho Valadares Rodrigues

Orientador, EST/UnB

Prof. Nádia Félix Felipe da Silva

INF/UFG

Prof. Rafael Bassi Stern

DES/UFSCAR

You have to prove the computer's assessment with very strong moves and
if you don't find these moves you may lose very quickly.
For a computer this isn't a problem, but for humans it is not so easy.
(Vassily Ivanchuk)

Para minha mãe

Agradecimentos

À minha avó, Neyde Pereira Gonçalves, por cuidar de mim todos os dias da vida. À minha mãe, Ângela Regina Pereira Gonçalves, por me mostrar que o amor pode se reinventar em momentos de dificuldade. O sentimento que me une a vocês cresce a cada dia, apesar da ausência física. Ao meu pai, Acir Mário Freitas, por me ensinar o valor da natureza e do trabalho. Aos três, agradeço pela maior honra da minha vida, que é torcer pelo Clube de Regatas do Flamengo. À SRD Gaia, motivo de seguir em frente mesmo nos piores momentos.

Agradeço à minha namorada, Juliana Venafro, pelo amor e confiança incondicionais. Você me torna melhor a cada dia. Aos meus irmãos Euler Alencar e Pamella Edokawa, que acreditaram em mim desde quando cheguei em Brasília. Sou feliz por trabalhar sob a liderança de vocês e por tê-los como amigos de todas as horas. Aos irmãos Wesley Freitas, Otávio Vasconcelos e Thiago Carolino. Vocês me mantém conectado às origens e são meu apoio para os momentos difíceis longe de casa. Meus sinceros agradecimentos aos professores Guilherme e Thaís Rodrigues do PPGEST/UnB, pela orientação, pelos ensinamentos e pela paciência em reuniões intermináveis. Devo este trabalho à vocês. À Ariane Hayana Thomé de Farias, por me mostrar que o cuidado com apresentações é fundamental e por me ensinar *Shiny*.

Agradeço aos profissionais de saúde que cuidaram de todos os brasileiros durante a pandemia e aos servidores do Supremo Tribunal Federal, com os quais partilho a alegria e a responsabilidade de trabalhar na mais alta instância do judiciário brasileiro.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

O Supremo Tribunal Federal (STF), instância máxima do sistema judiciário brasileiro, produz, assim como tribunais de outras instâncias, imensa quantidade de dados organizados em forma de texto, por meio de decisões, petições, liminares, recursos e outros documentos legais. Tais documentos são classificados e agrupados por servidores públicos especializados em autuação e catalogação de processos judiciais, que em casos específicos usam ferramentas tecnológicas de apoio. Alguns processos que chegam ao STF, por exemplo, são classificados em um ou mais objetivos de desenvolvimento sustentável (ODS) da Agenda 2030 da Organização das Nações Unidas (ONU). Como se trata de uma tarefa repetitiva e relacionada à detecção de padrões, é possível desenvolver ferramentas baseadas em aprendizagem de máquina para tal finalidade. Neste trabalho, são propostos modelos de Processamento de Linguagem Natural (NLP) para agrupamento de processos, com objetivo de aumentar a base de dados em determinados objetivos de desenvolvimento sustentável (ODS) com poucas entradas naturalmente. A atividade de clusterização ou agrupamento, que tem enorme importância por si só, também é capaz de reunir entradas sem etiqueta em torno de processos já classificados por funcionários do tribunal, permitindo, assim, que novas etiquetas sejam alocadas em processos similares. Os resultados obtidos mostram que os conjuntos aumentados por clusterização podem ser utilizados em fluxos de aprendizagem supervisionada para auxílio na classificação processual, especialmente em contextos com dados desbalanceados.

Palavras-chave: Agenda 2030 da ONU; Aprendizagem de Máquina; Agrupamento e Classificação de Textos.

Abstract

The Federal Supreme Court (STF), the highest instance of the Brazilian judicial system, produces, as well as courts of other instances, an immense amount of data organized in text form, through decisions, petitions, injunctions, appeals and other legal documents. Such documents are classified and grouped by public employees specialized in cataloging of judicial processes, which in specific cases use technological support tools. Some processes in the STF, for example, are classified under one or more sustainable development goals (SDGs) of the United Nations (UN) 2030 Agenda. As it is a repetitive task related to pattern recognition, it is possible to develop tools based on machine learning for this purpose. In this work, Natural Language Processing (NLP) models are proposed for clustering processes, in order to increase the database on certain sustainable development goals (SDGs) with few inputs naturally. The activity of clustering, which is of enormous importance in its own right, is also able to gather unlabeled entries around cases already classified by court officials, thus allowing new labels to be allocated to similar cases. The results of the work show that cluster-augmented sets can be used in supervised learning flows to aid in the classification of legal texts, especially in contexts with unbalanced data.

Keywords: 2030 UN Agenda; Machine Learning; Clustering and Text Classification.

Sumário

Abreviações e Siglas

STF	Supremo Tribunal Federal
ODS	Objetivos de Desenvolvimento Sustentável
ONU	Organização das Nações Unidas
NLP	Processamento de Linguagem Natural
CNJ	Conselho Nacional de Justiça
SVM	Support Vector Machine
IA	Inteligência Artificial
MIT	Massachussets Institute of Technology
GPU	Graphics Processing Unit
PCA	Principal Component Analysis
LLE	Locally Linear Embedding
t-SNE	Distributed Stochastic Neighbor Embedding
RNA	Rede Neural Artificial
CNN	Convolutional Neural Net
RNN	Recurrent Neural Net
LSTM	Long Short Term Memory
GRU	Unidade Recorrente Fechada

EDA	Easy Data Augmentation
SMOTE	Synthetic Minority Over-sampling Technique
KNN	K Nearest Neighbors
ROC	Receiver Operating Characteristic Curve
AUC	Area under the ROC Curve
TF-IDF	Term Frequency-Inverse Document Frequency
CBOW	Continuous Bag of Words
PV-DM	Distributed Memory Version of Paragraph Vector
PV-DBOW	Distributed Bag of Words Version of Paragraph Vector
STJ	Superior Tribunal de Justiça
TJGO	Tribunal de Justiça de Goiás
CSVM	Clustered Support Vector Machine
LGPD	Lei Geral de Proteção de Dados
BERT	Bidirectional Encoder Representations from Transformers
FAIR	Facebook Artificial Intelligence Research
BOW	Bag of Words
BCELoss	Binary Cross Entropy Loss

Capítulo 1

Introdução

A Agenda 2030 da ONU é um plano de ação para as pessoas, o planeta, a prosperidade, a paz universal e a liberdade. Criada em 2015, envolve objetivos como erradicação da pobreza, redução da desigualdade, preservação do meio ambiente e economia sustentável. A Agenda 2030 está organizada em 17 ODS (Objetivos de Desenvolvimento Sustentável), como pode ser visto em ONU, 2022. Os 17 ODS são integrados – a ação em uma área afeta os resultados de outras – mas os textos de cada ODS são completamente diferentes em sua essência, de forma que identificá-los como um rótulo em textos legais é um importante passo para automatizar tarefas repetitivas, melhorar fluxos de trabalho e melhor avaliar a prestação jurisdicional de cada tribunal e do poder judiciário como um todo. Incluir a Agenda 2030 da ONU no dia a dia dos tribunais é uma estratégia para tornar a justiça mais eficiente, uma vez que as decisões judiciais devem afetar positivamente a vida dos jurisdicionados.

O Supremo Tribunal Federal possui um sítio eletrônico (STF, 2020) onde informações sobre a Agenda 2030 estão disponíveis, bem como um painel contendo os metadados de processos atualmente etiquetados com ODS na corte. Tais dados podem ser facilmente baixados em formato .xlsx (Excel) através do painel. Este também apresenta *links* de outras iniciativas sobre a Agenda 2030 no poder judiciário, especificamente no âmbito do Conselho Nacional de Justiça (CNJ), cuja missão constitucional é controlar a atuação administrativa e financeira de mais de

90 orgãos do poder judiciário, como tribunais estaduais, tribunais regionais federais, tribunais regionais do trabalho e tribunais regionais eleitorais, por exemplo. O Poder Judiciário Brasileiro é pioneiro, no mundo, na institucionalização da Agenda 2030 e a indexação de sua base de dados (processos judiciais) nos 17 ODS busca responder perguntas como: Quais são os direitos fundamentais presentes nos ODS da Agenda 2030 com maior número de violações observadas pelo Poder Judiciário? É possível estabelecer precedência nos julgamentos que envolvem ODS da Agenda 2030? A Figura 1.1 mostra exemplos de ODS da Agenda 2030.

Figura 1.1: *Exemplos de ODS da Agenda ONU 2030*

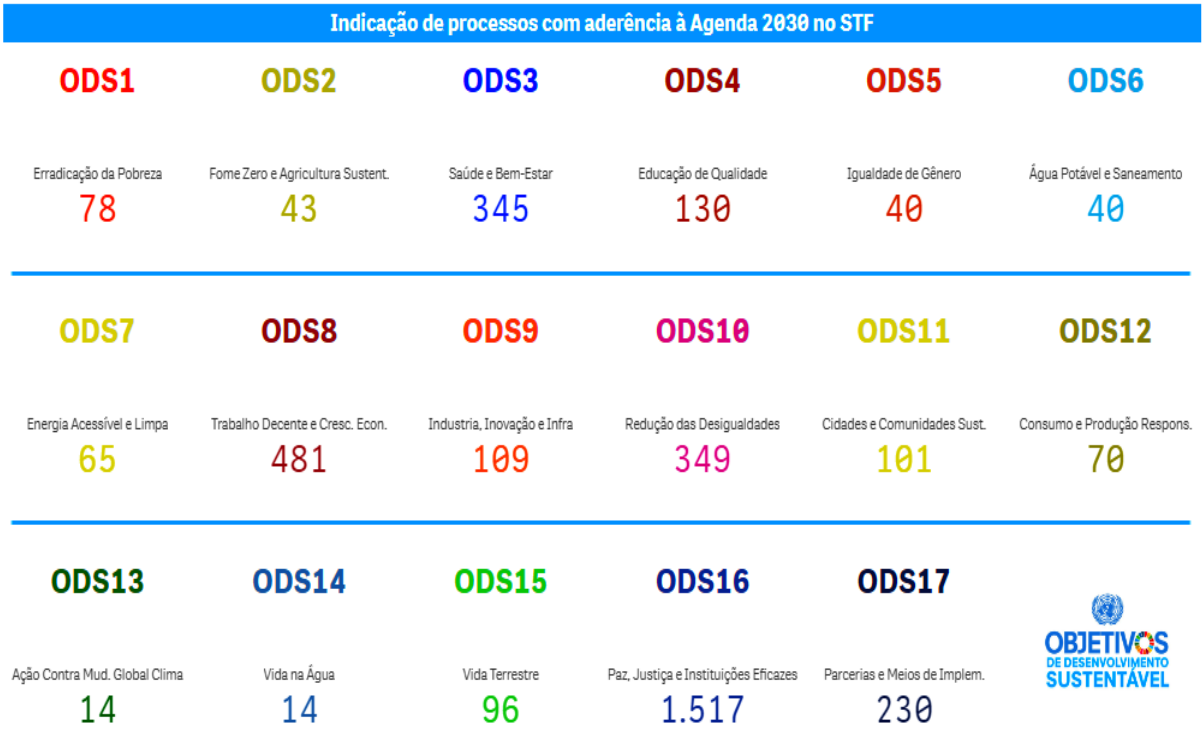
Fonte: Compilação do autor [1]

Dado que funcionários do STF em seu fluxo diário de trabalho classificam manualmente processos novos e antigos que compõem o acervo da corte em ODS da Agenda 2030, o objetivo principal desta dissertação é estender as funcionalidades dos métodos de agrupamento textual

[1] Montagem a partir de imagens extraídas de https://brasil.un.org/pt-br/sdgs.

(aprendizagem não supervisionada) para contextos de dados desbalanceados. O caso específico da Agenda 2030, com a classificação de textos jurídicos em objetivos de desenvolvimento sustentável (ODS), é apenas uma entre tantas aplicações possíveis da abordagem de agrupamento. Se trata, porém, de um caso com dados extremamente desbalanceados, onde alguns poucos ODS concentram mais de 90% das etiquetas realizadas, como mostra o painel gerencial disponível no sítio eletrônico do STF e ilustrado na Figura 1.2.

Figura 1.2: *Quantidade de etiquetas por ODS*



Fonte: Compilação do autor [2]

A abordagem proposta para aumento de dados baseada em clusterização seguirá resumidamente os seguintes passos: primeiramente, os processos classificados e processos sem etiqueta serão reunidos e agrupados de acordo com a similaridade textual, via aprendizagem de máquina.

[2]Montagem a partir de imagens extraídas de https://portal.stf.jus.br/hotsites/agenda-2030/.

Na sequência, os *clusters* formados serão etiquetados com base nas etiquetas dos processos rotulados ali presentes. As etiquetas serão replicadas nos processos inicialmente não rotulados e próximos do centro de cada grupo, para fugir das bordas, onde pode haver confusão entre os elementos de cada *cluster*. Este conjunto aumentado pode ser utilizado em fluxos de aprendizagem supervisionada, especificamente como *inputs* de algoritmos como *Support Vector Machine* (SVM), *Naive Bayes*, *XGBoost*, *CatBoost* e Redes Neurais, sendo os dois últimos modelos utilizados no presente trabalho. Como objetivos específicos têm-se: i) avaliar diferentes técnicas de limpeza de textos, envolvendo atividades de *pos tag* (*part of speech*), ii) comparar a performance de redes neurais, modelos *CatBoost* e de similaridade ajustados com as bases original e aumentada, iii) disponibilização das bases aumentadas para posterior avaliação por servidores do Supremo Tribunal Federal.

Recentemente, alguns trabalhos internacionais tiveram grande impacto na área de classificação em textos jurídicos. Katz, Bommarito e Blackman, 2017 utilizaram algoritmos da família *random forest* para prever decisões da Suprema Corte norte americana. Medvedeva, Vols e Wieling, 2020, apresentaram um método baseado em modelos SVM para prever decisões de casos futuros no Tribunal Europeu de Direitos Humanos. Hausladen, Schubert e Elliot, 2020, apresentaram aplicações de aprendizagem de máquina para prever decisões liberais e conservadoras em vários tribunais norte americanos. Radygin et al., 2021, desenvolveram uma ferramenta de inteligência artificial para buscar violações na Legislação Federal Russa. No Brasil, destacam-se os trabalhos de Junior, Calixto e Castro, 2020, Nascimento e Oliveira, 2022, Menezes e Clementino, 2022 e Zanuz e Rigo, 2022, nos quais os autores trabalharam, respectivamente, com ontologias, clusterização, arquitetura de *transformers* e reconhecimento de entidades nomeadas, sempre no contexto do poder judiciário.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 é feita uma introdução à aprendizagem de máquina, envolvendo treinamento supervisionado, não supervisionado e aprendizagem profunda (*Deep Learning*). O Capítulo 3 apresenta técnicas de Processamento de Linguagem Natural (NLP), para tarefas específicas como vetorização, clusterização e clas-

sificação de textos. No Capítulo 4 são apresentadas as bases de dados e as estratégias para agrupamento e posterior classificação dos textos. O Capítulo 5 apresenta os principais resultados obtidos. As principais conclusões e caminhos para trabalhos futuros estão no Capítulo 6. Os Apêndices tratam dos *frameworks* mais populares de NLP em `R` e Python e também da atividade de *pos tag – part of speech*.

Capítulo 2

Aprendizagem de Máquina

2.1 Introdução

Em linhas gerais, a aprendizagem de máquina trata de transformar dados em informação útil, através de métodos com pouca ou nenhuma interferência humana. É uma área de pesquisa situada na interseção entre a computação científica, inteligência artificial (IA) e a estatística. As aplicações de aprendizagem de máquina podem resolver problemas práticos de diversas áreas, tais como medicina, comércio virtual, serviços de *streaming*, automação industrial, direito e inúmeras outras.

O termo aprendizagem de máquina passou a ser utilizado com notável frequência na década de 50, quando Samuel, 1959, então pesquisador do MIT, desenvolveu um programa chamado de *Game of Checkers*, capaz de jogar damas e aprender com seus próprios erros, partida por partida. Para desenvolver este programa, Arthur Samuel aproveitou o fato de que muitos jogadores de dama anotam suas partidas, conforme indicam McCarthy e Feigenbaum, 1990. Também percebeu que se trata de um jogo com boas e más estratégias bem definidas e conhecidas pelos jogadores, ideal para um primeiro programa de aprendizagem de máquina. Ainda nesta década, Alan Turing criou o *Turing Test* (Turing, 1950), procedimento que testa a capacidade de uma

máquina exibir comportamento inteligente, Frank Rosenblatt criou o *Perceptron* (Rosenblatt, 1958), primeira máquina com desenho de rede neural, e Minsky e McCarthy organizaram a conferência de Darthmouth (1956) inaugurando, oficialmente, o campo de pesquisa em inteligência artificial. Apesar da criação do algoritmo *K Nearest Neighbor*, em 1967, as áreas de inteligência artificial e aprendizagem de máquina passaram por um longo inverno, até a década de 80. Neste período, poucos avanços foram dignos de registro e a diferença entre aprendizagem de máquina e inteligência artificial era praticamente imperceptível ou de difícil compreensão.

O período compreendido entre a metade da década de 80 e o final dos anos 90 foi importante na distinção destas duas áreas, especialmente pelo surgimento de sistemas comerciais inteligentes e pela popularização da *internet* e de computadores pessoais. Nesta época, foi possível entender que os computadores são capazes de desempenhar tarefas difíceis tão bem ou melhor do que seres humanos, se programados com técnicas de inteligência artificial. Quando tais tarefas mudam em complexidade ou natureza, no entanto, as regras estabelecidas pelos procedimentos de inteligência artificial podem deixar de fazer sentido, sendo necessário um período de treinamento, até que os computadores possam voltar ao trabalho com boa performance.

Desde o início do século XXI, quando as duas áreas passaram a caminhar de forma isolada, os mais recentes avanços em aprendizagem de máquina possuem relação com a criação de redes neurais, maiores em relação ao *Perceptron*, enquanto as novas abordagens em inteligência artificial usam processamento em paralelo, especialmente com GPUs (Unidades Gráficas de Processamento), para grandes conjuntos de dados – *Big Data*. Um sistema ou algoritmo de aprendizagem de máquina é um programa de computador capaz de tomar decisões baseadas em experiências acumuladas com a solução de problemas anteriores. Na prática, segundo Lantz, 2013, um algoritmo de aprendizagem de máquina pode ser implementado seguindo os passos:

- **Coleta de Dados:** Obtenção, organização e preparação dos dados que serão utilizados pelo algoritmo para a geração de informação útil. Na maioria das vezes, os bancos de dados estão disponíveis em arquivos de texto ou tabelas.

- **Análise Exploratória:** Etapa inicial que mede a qualidade dos dados disponíveis. Envolve o tratamento de dados faltantes, medidas de correlação entre variáveis e a visualização, ainda que preliminar, de comportamentos como agrupamentos e observações extremas (*outliers*).

- **Treinamento do Modelo:** Dada uma escolha de algoritmo, é fundamental entender o quanto é possível aprender com os dados disponíveis. Após o período de treinamento, onde apenas uma parte dos dados será utilizada, os algoritmos poderão representar o próprio conjunto de dados na forma do modelo escolhido.

- **Avaliação do Modelo:** Utilização de métricas para a avaliação de performance do modelo selecionado. Envolve a aplicação do modelo fora do conjunto de treinamento e o cálculo da métrica de interesse. Ex: Acurácia, Sensibilidade e etc.

Quando tais passos estão completos, os modelos podem ser utilizados para descrição ou previsão e as informações disponíveis podem, enfim, ser apresentadas de maneira clara aos usuários finais. Os algoritmos de aprendizagem de máquina se diferenciam, fundamentalmente, pelo tipo de aprendizagem, que pode ser supervisionada, não supervisionada ou por reforço e pelas tarefas básicas executadas, que podem ser de classificação, regressão, agrupamento ou ordenação. A aprendizagem também pode ser profunda, quando os algoritmos utilizados são baseados em redes neurais e possuem muitas camadas. No presente trabalho, são utilizados métodos não supervisionados para agrupamento de peças jurídicas e algoritmos em ambientes supervisionados para posterior classificação dos textos em ODS da Agenda 2030 da ONU.

2.2 Aprendizagem Supervisionada

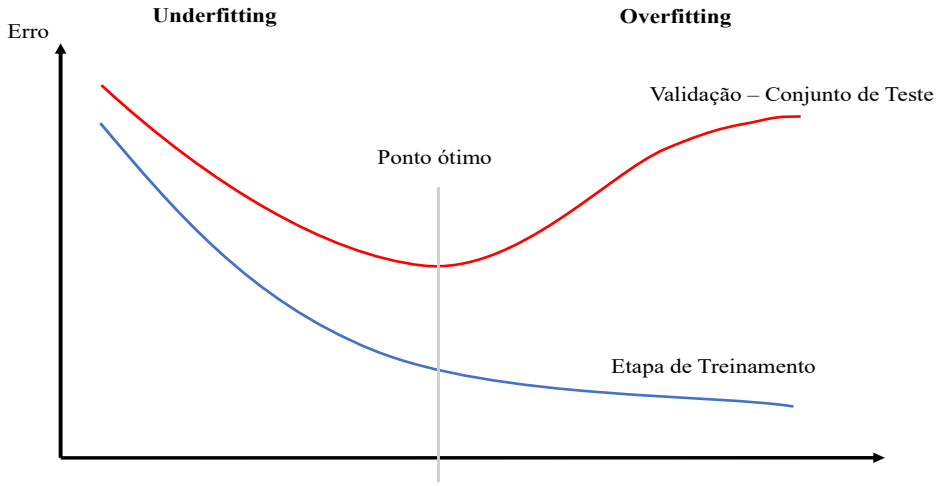
Os tipos mais utilizados de algoritmos de aprendizagem de máquina são aqueles que generalizam conclusões a partir de exemplos conhecidos. Neste tipo de problema, conhecido como aprendizagem supervisionada, o usuário fornece ao algoritmo pares de entradas e saídas para treinamento e espera saídas coerentes dada uma nova entrada. Em linhas gerais, o algoritmo deve ser capaz de criar saídas para entradas até então desconhecidas, sem qualquer ajuda humana, apenas usando a experiência e as regras produzidas a partir de conjunto inicial de treinamento. Como exemplo, pode-se utilizar a classificação de *e-mails* em *spam*. Usando aprendizagem de máquina, o usuário fornece ao algoritmo um razoável número de *e-mails* (que servem de entrada), juntamente com classificações de não *spam* ou *spam* (que é a saída desejada). Dado um novo *e-mail*, ao final do algoritmo, o usuário terá uma sugestão sobre a natureza da mensagem (*spam vs* não *spam*).

Algoritmos desta natureza são chamados de supervisionados porque são fornecidos exemplos de saídas desejadas para cada entrada, com as quais os próprios algoritmos aprendem. Existem dois principais tipos de problemas supervisionados de aprendizagem de máquina, chamados de classificação e regressão. Em problemas de classificação, o objetivo é predizer a classe de uma observação, em duas ou mais possibilidades. No primeiro caso, que inclui o problema de decidir se um dado texto jurídico tem uma etiqueta ou não, temos um problema de classificação binária. Em resumo, o algoritmo deve rotular uma observação quando apenas duas classes são possíveis. Quando, no entanto, existem 3 ou mais classes, o problema é multiclasse. Um exemplo, apresentado por Nunes, Carvalho e Lucena, 2009, envolve a escolha do melhor hospital de referência, entre 3 possibilidades, para gestantes de alto risco do estado do Rio de Janeiro. Também existem problemas onde uma entrada pode receber mais de uma etiqueta simultaneamente. Um exemplo ocorre quando se deseja avaliar, conjuntamente, todas as etiquetas de uma peça jurídica. Em geral, pode-se associar cada classe a um número natural,

de forma que as saídas esperadas são discretas e representam categorias. Quando a predição envolve valores contínuos, temos problemas de regressão.

Na aprendizagem supervisionada, se um modelo é capaz de fazer previsões precisas sobre dados não vistos, diz-se que ele é capaz de generalizar do conjunto de treinamento para conjuntos de teste. Pode haver, contudo, *overfitting*. Isto ocorre quando o algoritmo se ajusta bem ao conjunto de treinamento, mas se mostra inadequado para novas observações. Na prática, em contextos onde existe *overfitting*, há claro prejuízo na generalização e as predições se tornam ruins. Quando o algoritmo apresenta pouco ajuste no conjunto de teste ou até mesmo nos conjuntos de treino e teste ao mesmo tempo, há *underfitting*. Um bom modelo deve apresentar equilíbrio em relação ao *overfitting* e *underfitting*, conforme mostra a Figura 2.1.

Figura 2.1: *Underfitting e Overfitting*



Fonte: Próprio do autor

O ponto ótimo representa a etapa limite do processo de aprendizagem, em relação ao decaimento do erro de ajuste. Basicamente, espera-se maior erro no início do treinamento. Os desafios para utilização de modelos de aprendizagem supervisionada não se limitam aos casos onde há *over* ou *underfitting*. A base de treinamento pode ser insuficiente ou não representativa, bem como as características (*features*) associadas ao problema podem não ser boas. Quando a base de treinamento é insuficiente, uma alternativa é utilizar técnicas de aumento de dados (*data augmentation*), tema a ser tratado em maiores detalhes na Seção 2.5. Por outro lado, os dados podem ser adequados e o modelo pode ter problemas de ajuste. Para evitar problemas de ajuste dos algoritmos e facilitar a seleção dos modelos, a divisão tradicional dos dados de entrada e saída em treinamento e teste é substituída pela divisão treinamento *vs* validação *vs* teste, sendo a base de validação utilizada para ajuste dos hiperparâmetros de cada modelo. Os dados de validação, assim como os de teste, não são utilizados no treinamento dos algoritmos. Tal abordagem será utilizada no presente trabalho. Uma extensa análise dos possíveis problemas em fluxos de aprendizagem supervisionada está disponível em Geron, 2021.

O principal método de aprendizagem supervisionada associado à algoritmos de *machine learning* utilizado neste trabalho são os algoritmos *CatBoost*. Os modelos da família *CatBoost* usam *gradiente boosting* e *decision trees* em esquemas de *ensemble*, para combinar *features* categóricas, numéricas e *features* em texto. De forma resumida, são modelos de *ensemble* que combinam modelos menores, no caso, árvores de decisão. Servem para associar textos de peças jurídicas com os metadados dos processos em tela, em aplicações de tribunais e escritório de advocacia, por exemplo. A principal diferença do algoritmo *CatBoost* para outros métodos *boosting* (*XGBoost* e *LightGBM*) é, sem dúvida, seu suporte nativo para dados categóricos. Outras diferenças notáveis são o uso de árvores simétricas e a estratégia de otimização centrada em *ordered boosting*.

No presente trabalho, os dados categóricos e númericos utilizados foram classe processual, contagem de palavras-chave associadas à cada ODS e ramo do direito. As palavras-chave foram

criadas por especialistas do tribunal para facilitar a classificação em ODS da Agenda 2030. "Feminicídio", por exemplo, quando aparece muitas vezes numa peça processual, indica presença do ODS 5 - Igualdade de Gênero. As variáveis classe processual e ramo do direito são bastante comuns e indicam a natureza da provocação ao poder judiciário e o assunto do processo em tela. O algoritmo *CatBoost*, junto das redes neurais e abordagens de similaridade, serão utilizados no presente trabalho na parte de classificação textual, após a etapa de clusterização. Maiores informações sobre os algoritmos da família *CatBoost* podem ser encontradas em Prokhorenkova et al., 2018.

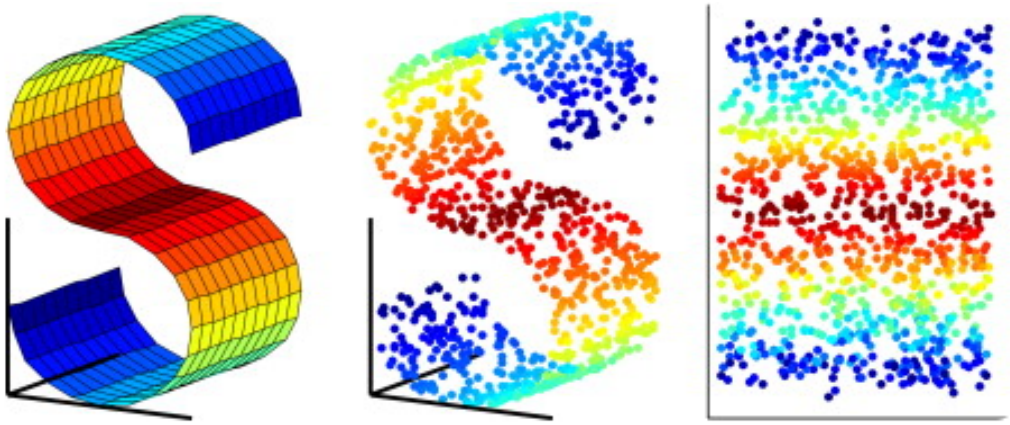
2.3 Aprendizagem Não Supervisionada

Em abordagens de aprendizagem não supervisionada não existem rótulos ou saídas esperadas. O algoritmo, de modo geral, tenta aprender sem exemplos, ou seja, sem supervisão para correlacionar os dados de entrada com resultados (*outputs*) desejados. As tarefas não supervisionadas mais comuns em aprendizagem de máquina são de transformação dos dados e clusterização (agrupamento).

Segundo Müller e Guido, 2016, transformações não supervisionadas são tarefas que criam novas representações para os dados, tornando-os de mais fácil entendimento para humanos ou outros algoritmos de aprendizagem de máquina quando comparados com a representação original. A aplicação mais comum de transformações não supervisionadas é a redução de dimensionalidade, que busca representar, em menores espaços, dados complexos de altas dimensões, preservando relações de distância e posições entre seus elementos. É bastante comum empregar redução de dimensionalidade para tornar os dados bidimensionais e assim melhor visualizá-los. Um exemplo clássico ocorre em processamento de textos e imagens. Ambos são objetos em forma matricial, com representação em altas dimensões. Ao reduzir a dimensão de tais objetos, é possível plotar gráficos onde importantes relações podem ficar evidentes. A Figura 2.2 mostra

um exemplo de redução de dimensionalidade.

Figura 2.2: *Redução de Dimensionalidade*



Fonte: Compilação do autor [1]

A partir da Figura 2.2, nota-se que os dados originais estão em três dimensões e a sua representação fica, após aplicação de técnicas de redução de dimensionalidade, em apenas duas. Importante notar que os pontos estão dispostos, na nova representação, de forma a preservar internamente (apenas na parte colorida da figura) as relações originais de distância.

Quando existem *features* correlacionadas, abordagens de redução de dimensionalidade podem se transformar em ferramentas para extração de características importantes. Um exemplo pode ser visto em Geron, 2021 e envolve veículos de transporte. A quilometragem de um carro pode estar bastante correlacionada com seu ano de fabricação e o algoritmo de redução de dimensionalidade, quando bem ajustado, provavelmente combinará as duas características, transformando-as em uma única variável. Os métodos mais comuns para reduzir dimensionalidade são PCA (*principal component analysis*), LLE (*locally linear embedding*) e t-SNE (*distributed stochastic neighbor embedding*).

[1] Montagem a partir de imagens extraídas de https://iq.opengenus.org/kernal-principal-component-analysis/.

A atividade de clusterização, que é foco deste trabalho, consiste em particionar dados em grupos de elementos parecidos, chamados *clusters*. A ideia principal é que pontos pertencentes a um mesmo *cluster* sejam parecidos e pontos de *clusters* diferentes sejam diferentes. Em outras palavras, os *clusters* devem ser internamente homogêneos e heterogêneos entre si, quando bem ajustados. O problema de agrupar dados é objeto de estudo em contextos de *data mining* e aprendizagem de máquina, mas suas aplicações abrangem outras áreas, como passo intermediário em outros problemas fundamentais de *data science*, segmentação de clientes, detecção de tendências em geral e análise de redes sociais, entre outros.

Os mais populares algoritmos de clusterização são baseados em distância. Tais métodos são simples de interpretar e simples de implementar, sendo aplicáveis numa imensa variedade de cenários. Métodos baseados em distância podem ser divididos em duas categorias: não hierárquicos e hierárquicos. Os métodos hierárquicos não são apresentados em detalhes neste trabalho, pois as aplicações em processamento de linguagem natural costumam utilizar abordagens não hierárquicas. Os métodos não hierárquicos iniciam com partições arbitrárias e possuem flexibilidade, uma vez que os elementos podem mudar de *cluster* durante a execução do algoritmo. O procedimento geral utilizado em métodos não hierárquicos é: i) definir o número de *clusters*; ii) definir os centróides de forma aleatória; iii) calcular o centróide mais próximo de cada elemento usando uma métrica de distância; iv) calcular a nova posição dos centróides com base nos elementos obtidos no passo anterior; iv) repetir os dois últimos passos até obter a posição ideal para os centróides. Os principais métodos não hierárquicos são:

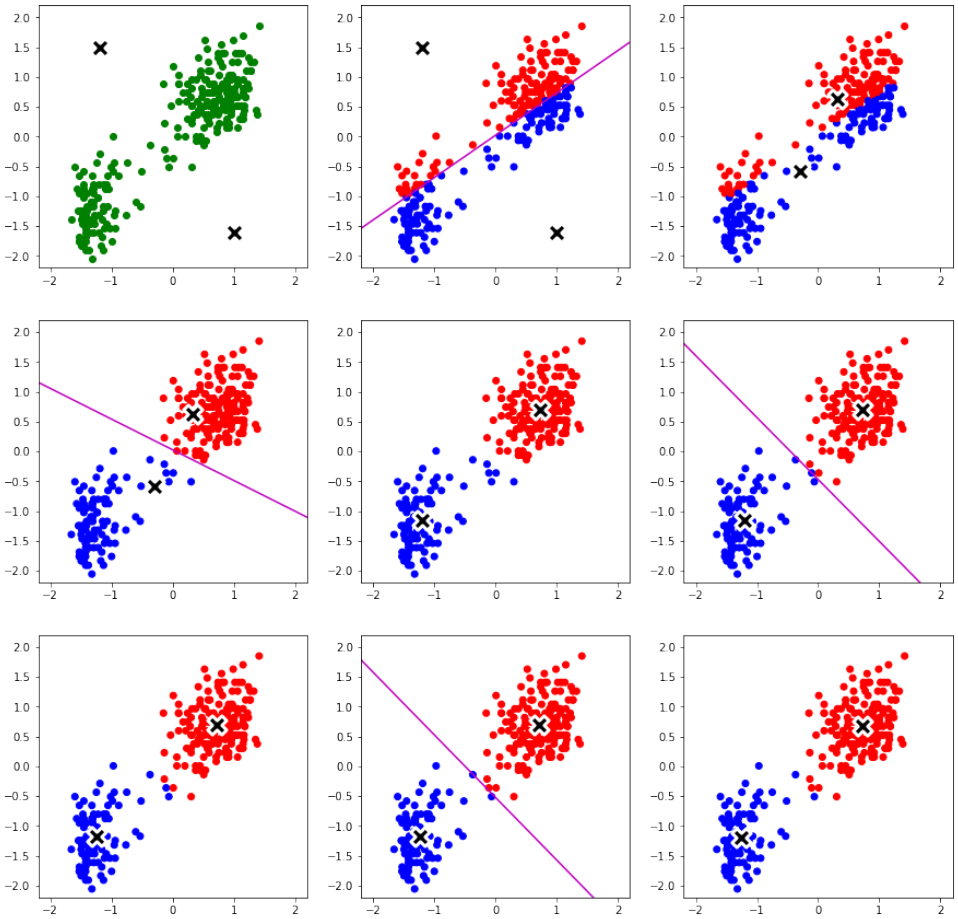
- ***K-Means*:** Neste método, os centróides correspondem à posição média de cada *cluster*, sendo, então, uma função dos dados subjacentes. A distância euclidiana é usada para calcular as distâncias entre os pontos e os centróides, sendo cada ponto atribuído ao *cluster* de centróide mais próximo em cada iteração. O método *k-means* é considerado um dos mais simples para agrupamento de dados, sendo também o mais utilizado, segundo Aggarwal e Reddy, 2014.

- ***K-Medians*:** Neste método, a mediana, em vez da média, é utilizada para criar os centróides. Como no caso da abordagem *k-means*, os centróides não são, necessariamente, observações do conjunto de dados original. A abordagem *k-medians* é mais estável em relação à ruídos e *outliers*, porque a mediana de um conjunto de valores é geralmente menos sensível a valores extremos nos dados.

- ***K-Medoids*:** Neste método, os centróides são amostrados a partir dos dados originais. Tais técnicas são particularmente úteis nos casos em que os pontos de dados a serem agrupados são objetos para os quais não faz sentido falar sobre funções de média ou mediana. Em cada iteração, um dos centróides é substituído por um representante dos dados atuais, a fim de verificar se a qualidade do agrupamento melhora. Esses métodos geralmente requerem muito mais iterações do que *k-means* e *k-medians*.

A Figura 2.3 mostra um exemplo da evolução, por iteração, de um algoritmo da família *k-means*, em um conjunto arbitrário de dados, onde as marcações em x correspondem aos centróides de cada iteração.

Figura 2.3: *Algoritmo K-Means*



Fonte: Compilação do autor [2]

Os algoritmos da família *k-means* serão utilizados na atividade de agrupamento de textos, explicada em detalhes na Seção 3.3. Maiores informações sobre os recentes avanços em *data clustering* podem ser encontradas em Scitovski et al., 2022, Aljarah, Faris e Mirjalili, 2020 e Selvaraj e Choi, 2021.

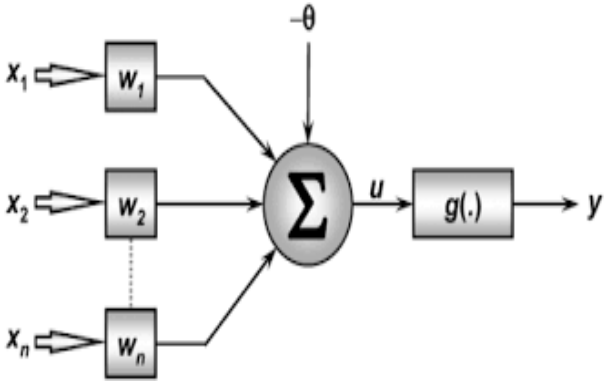
[2]Montagem a partir de imagens extraídas de http://dendroid.sk/2011/05/09/k-means-clustering/.

2.4 Aprendizagem Profunda

Redes Neurais Artificiais (RNA) são modelos computacionais inspirados no funcionamento do sistema nervoso dos animais. Possuem, como os neurônios interligados do cérebro humano, capacidade de adquirir, manter e manipular conhecimentos prévios e informações novas. As conexões entre neurônios artificiais são chamadas de sinapses, sendo representadas por vetores ou matrizes de pesos associados ao problema de interesse. A presente seção é inteiramente dedicada às redes neurais, arquitetura dos modelos atualmente utilizados para a tarefa de classificar ODS da Agenda 2030 em processos do Supremo Tribunal Federal. A inclusão de dois outros modelos na metodologia desta dissertação - *CatBoost* e similaridade - é uma tentativa de melhorar o modelo atualmente em produção.

O *Perceptron* é a rede neural artificial mais simples, cuja implementação se baseia no funcionamento da retina humana. Em sua concepção inicial, a rede *Perceptron* recebia sinais elétricos oriundos de fotocélulas com o objetivo de mapear padrões geométricos em sua forma. Durante a fase de treinamento, pesos iniciais representados por resistores sintonizáveis eram ajustados, enquanto um operador de soma efetuava a composição de todos os sinais. A Figura 2.4 traz uma representação possível para a fase de operação da rede *Perceptron*.

Figura 2.4: Rede *Perceptron*



Fonte: Silva, Spatti e Flauzino, 2016

De acordo com a Figura 2.4, cada uma das entradas $x_1, x_2, \ldots, x_n$ é ponderada por um peso w_i. Em seguida, o resultado da composição de todas as entradas ($\Sigma \rightarrow U$) é adicionado ao limite de ativação θ, sendo este total o argumento da função de ativação $g(U)$, responsável pelas saídas y do algoritmo. Matematicamente,

$$\begin{aligned} U &= \sum_{i=1}^{n} w_i x_i - \theta, \\ y &= \begin{cases} 1, & \text{se } U \geq 0. \\ -1, & \text{caso contrário.} \end{cases} \end{aligned}$$

O objetivo da função de ativação $g(.)$ é limitar as saídas do algoritmo em intervalos razoáveis, tornando a imagem o mais funcional possível, dada a natureza do problema de interesse. No caso do *Perceptron*, apenas duas saídas são possíveis, pois o objetivo inicial desta rede é a classificação binária. As funções de ativação normalmente usadas no *Perceptron* são a função degrau ou degrau bipolar. Outras funções de ativação são possíveis, como as de primeira derivada conhecida (função logística, tangente hiperbólica e função gaussiana) e de derivadas parcialmente conhecidas (função degrau), conforme Silva, Spatti e Flauzino, 2016.

Se as saídas da fase de operação da rede *Perceptron* não são precisas, isto é, se o algoritmo não possui boa acurácia ao tentar prever resultados reais, podem ser feitos ajustes nos pesos e no limiar de ativação através da regra de aprendizado de Hebb, definida pelas equações

$$\begin{aligned} w_i^{atual} &:= w_i^{anterior} + \eta(d^k - y)x_i^k, \\ \theta_i^{atual} &:= \theta_i^{anterior} + \eta(d^k - y)(-1), \end{aligned}$$

onde d^k é o valor desejado para a k-ésima amostra de treinamento, y é a saída produzida pelo *Perceptron* e η é uma constante que define a taxa de aprendizagem da rede. Normalmente,

$0 < \eta < 1$. Este conjunto de operações é a fase de treinamento da rede *Perceptron*, necessária quando em uma época (execução da rede) não foi atingida convergência.

O *Perceptron* é bastante utilizado em tarefas simples de classificação, dado que problemas complexos exigem a utilização de redes com mais neurônios, maior número de camadas (blocos de neurônios) e arquiteturas recorrentes. *Deep Learning* é uma área do aprendizado de máquina que se ocupa de produzir representações para dados de entrada, através do processamento sequencial de estímulos com redes neurais artificiais de muitas camadas ocultas, isto é, blocos de neurônios internos à rede. Basicamente, os algoritmos de aprendizagem profunda diferem de redes neurais simples, tais como o *Perceptron*, pela quantidade de camadas e, consequentemente, pelo número de neurônios, além dos métodos de atualização dos pesos. Com a adição de mais pesos e limiares, as redes profundas possuem melhor capacidade de aproximar funções complexas e performam melhor em tarefas de mapeamento em grandes conjuntos de dados. As principais arquiteturas de redes profundas são:

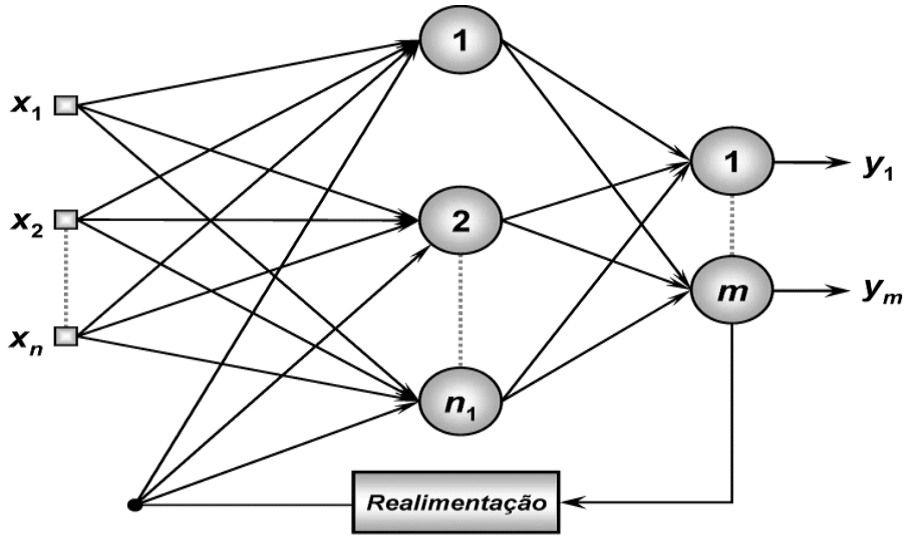
- **Redes Convolucionais (CNN's)** - LeNet, AlexNet e ResNet.
- **Redes Recorrentes (RNN's)** - Redes de Elman, Jordan, Hopfield e Redes de Longa Memória em Prazo Curto (LSTM).

As redes convolucionais, que possuem arquitetura *feedforward* (fluxo de informações em uma única direção), são também conhecidas por redes invariantes a deslocamento ou invariantes no espaço. Neste tipo de rede neural, a camada de entrada é dividida, de forma que cada neurônio da primeira camada oculta seja responsável por processar apenas alguns lotes de informação inicial, com pesos compartilhados de camada para camada. A convolução propriamente dita ocorre quando estes pequenos lotes, depois do processamento em algumas camadas, são utilizados para filtrar e mapear o espaço inicial de informações, através de novos vetores ou tensores. Esta arquitetura, que percebe imagens como volumes, permite reconhecimento espacial com menos parâmetros do que redes neurais artificias totalmente conectadas, por exemplo.

Trata-se de uma rede de rápida execução, pois existem camadas de *pooling* responsáveis por simplificar informações de camadas anteriores e fundamentalmente utilizadas para reconhecimento de imagens. A atualização dos pesos são feitas, nestas arquiteturas, através de otimizadores baseados em *backpropagation*, gradiente descendente estocástico (Adam) ou métodos de momento (RMSprop).

As redes neurais recorrentes são poderosos algoritmos para processamento de dados sequenciais, tais como sons, séries temporais ou linguagem natural. Tais redes diferem de arquiteturas *feedforward* porque incluem *loops* de realimentação, havendo memória entre as camadas da rede. Objetivamente, as redes *feedforward* produzem modelos de atualização mais lenta e as redes recorrentes produzem modelos mais dinâmicos em relação à memória. A Figura 2.5 apresenta uma rede neural recorrente simples, com uma única camada oculta.

Figura 2.5: Rede neural recorrente



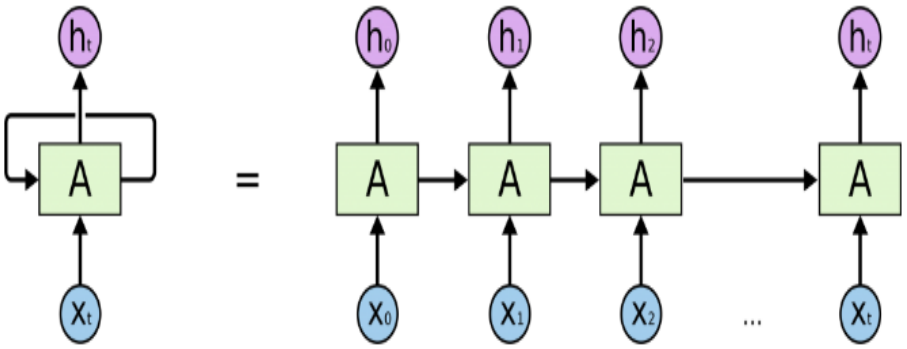
Fonte: Silva, Spatti e Flauzino, 2016

É possível notar, pela Figura 2.5, que os neurônios são realimentados depois que as saídas da rede $y_1, y_2 \ldots, y_m$ são produzidas. Este *looping* ou sistema de realimentação pode ocorrer até que uma determinada regra seja atingida e pode ser visto como parte da etapa de treinamento

da rede.

Em muitos problemas, apenas informações recentes são necessárias para execução de uma tarefa. Nestes casos, onde o espaço de tempo entre a informação necessária e a construção de *outputs* (saídas de rede - h_t) é pequeno, as redes recorrentes são bastante úteis. A Figura 2.6 mostra uma rede neural recorrente estendida.

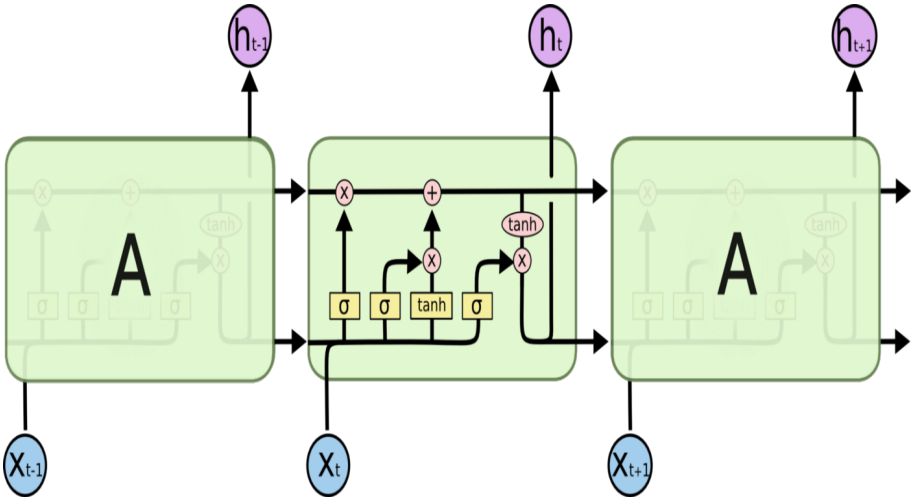
Figura 2.6: Rede recorrente estendida



Fonte: Deep Learning Book, 2022

Quando, no entanto, a rede cresce, isto é, quando as informações necessárias (X_t) para a atualização de neurônios (A) e pesos estão distantes da iteração ou tempo atual, as redes recorrentes perdem parte da capacidade de conectar tais informações. Isto significa que as funções de ativação perdem a capacidade de realçar os neurônios com informações importantes. Para tarefas desta natureza, existem redes com memória a longo prazo ou de arquitetura LSTM, introduzidas por Hochreiter e Schmidhuber, 1997, capazes de aprender com dependências no tempo. Nesta estrutura de rede, que será utilizada neste trabalho para capturar dependências entre partes distantes de um mesmo texto, existem 4 camadas internas, próximas da função de ativação. A Figura 2.7 mostra um nó de rede LSTM.

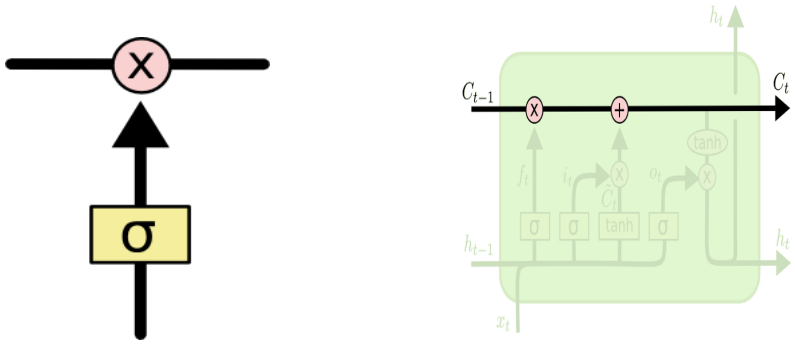
Figura 2.7: Neurônio na arquitetura LSTM



Fonte: Deep Learning Book, 2022

Na Figura 2.7, as caixas amarelas representam camadas escondidas, os círculos cor de rosa representam operações pontuais com vetores, tais como multiplicação por escalar e soma, as setas simples representam vetores, as setas que se fundem representam concatenação de informação e as setas de bifurcação indicam que alguma informação está sendo copiada para lugares diferentes. A ideia central da arquitetura LSTM é atualizar o estado da seta horizontal, que passa na parte de cima da Figura 2.7, no neurônio central. Esta linha de estados transporta informações com algumas modificações lineares. Existem, no entanto, algumas portas (*gates*) neste tipo de arquitetura, com capacidade de adicionar ou remover informações à linha de estados da célula. A porta inicial (*forget gate*) e a linha de estados são descritas pela Figura 2.8.

Figura 2.8: Porta Inicial e Linha de Estados na arquitetura LSTM



Fonte: Deep Learning Book, 2022

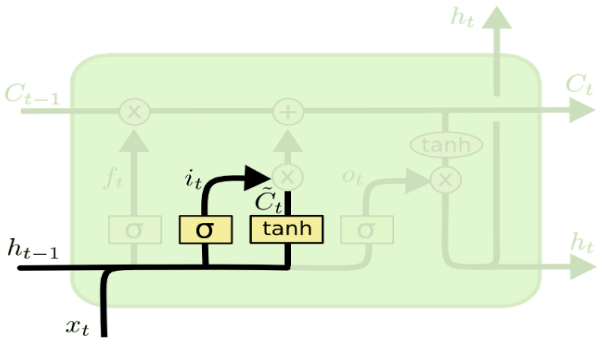
A camada sigmóide produz números entre 0 e 1 para indicar o quanto de informação atual (x_t) é útil. Valores extremos, como 0 e 1, indicam, respectivamente, que nada da informação atual será aproveitada e toda a informação atual pode ser aproveitada. Matematicamente, ocorre

$$f_t = \sigma(W_{t-1}.[h_{t-1}, X_t]) + b_f,$$

onde b_f é um valor de correção. Depois que apenas uma fração da informação atual é considerada útil, uma camada tangente hiperbólica cria um vetor de informação candidata ($\tilde{C}_t$), através de pesos iniciados aleatoriamente (W_i,W_c), que será combinado à informação útil (i_t) definida pela camada sigmóide. Em resumo, a porta central do neurônio, descrita pela Figura 2.9, é dada pelas equações:

$$\begin{aligned} i_t &= \sigma(W_i.[h_{t-1}, X_t]) + b_i, \\ \widetilde{C}_t &= tanh(W_c.[h_{t-1}, X_t]) + b_C, \\ C_t &= f_t + (i_t \times \widetilde{C}_t), \end{aligned}$$

Figura 2.9: Porta Central na arquitetura LSTM



Fonte: Deep Learning Book, 2022

onde a combinação, de fato, ocorre quando f_t é somado ao vetor de candidatos submetido à atualização. i_t é chamado de *input gate* e os valores b_i e b_C servem para correção. A última camada antes das saídas de rede é dada por $o_t = \sigma(W_o.[h_{t-1}, X_t]) + b_o$. Finalmente, a saída de rede é dada por $h_t = o_t(tanh(C_t))$, conforme ilustrado anteriormente pela Figura 2.7.

O principal objetivo da escolha desta arquitetura é avaliar se os dados aumentados melhoram as redes atualmente em uso, que são de natureza LSTM. Comparações entre as redes ajustadas com a base original e as bases aumentadas são apresentadas no capítulo 5. A próxima seção trata de *data augmentation*.

2.5 *Data Augmentation*

A performance de modelos de aprendizagem profunda (*deep learning*) depende fortemente da qualidade, quantidade e, especialmente em processamento de linguagem natural, do contexto dos dados de exemplo. Dados em pequenas quantidades são, talvez, o maior desafio para utilização dos modelos de aprendizagem profunda. *Data augmentation* é, em último grau, uma estratégia para diminuir o peso deste problema em fluxos de aprendizagem de máquina e *deep*

learning.

Data augmentation é o processo de aumentar artificialmente a quantidade original de dados gerando novas entradas a partir de dados existentes. Isso inclui adicionar pequenas alterações ou ruídos aos dados originais, promover ciclos de anotações simplificadas ou usar modelos de aprendizado de máquina para gerar novas entradas no espaço dos dados originais. Uma questão que se apresenta, quando estratégias de *data augmentation* são necessárias, é a diferença entre dados sintéticos e dados adicionais:

- **Dados sintéticos:** dados gerados artificialmente, sem utilização dos objetos originais. Podem ser gerados através de simulações ou redes neurais.
- **Dados adicionais:** gerados através dos objetos originais, com transformações tais como substituição controlada, adição de ruídos, translações e rotações.

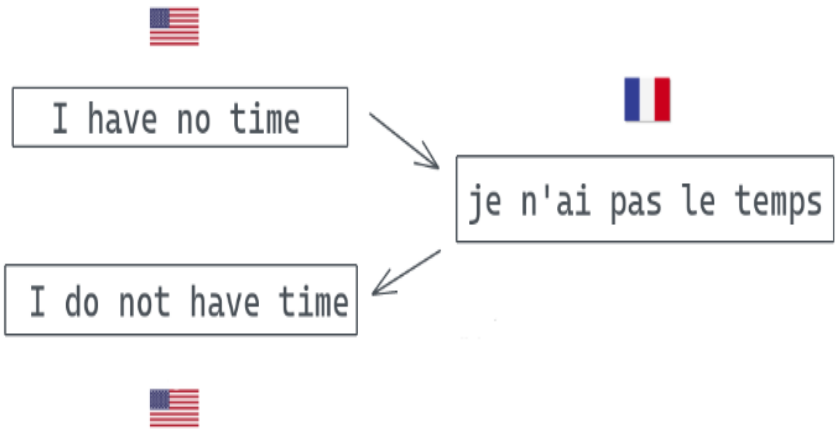
Em visão computacional, por exemplo, estratégias para geração de dados adicionais são óbvias e imediatas. Rotacionando uma figura, temos uma nova figura. Ao diminuir o brilho de uma figura, há uma nova figura. Em processamento de linguagem natural (NLP), no entanto, as técnicas de geração de dados adicionais precisam ser mais robustas, dado que pequenas transformações implicam em mudanças semânticas, gramaticais e de contexto nos textos originais. As técnicas mais populares de *data augmentation* em NLP são:

- ***Back Translation:*** consiste na tradução dos textos para um novo idioma, seguida de nova tradução, retornando ao idioma original. Estratégia produz ruídos razoavelmente controlados nos textos originais.
- ***Easy Data augmentation* (EDA):** consiste na aplicação, simultânea ou não, de 4 operações: substituição por sinônimos, inserção aleatória de sinônimos, troca aleatória de posições entre palavras e remoção probabilística de palavras.

- **Transformações em sentenças:** sobe um nível em relação às técnicas anteriores e faz substituições e inserções de sentenças (frases) ao invés de palavras.

Tais técnicas podem ser combinadas com estratégias de balanceamento de dados em contextos de classificação, tais como *upsample*, *downsample* e SMOTE. *Upsample* busca injetar dados da classe minoritária, enquanto que a estratégia de *downsample* objetiva diminuir a quantidade de entradas da classe majoritária. A técnica SMOTE, que é da família *upsample*, faz inserção de dados baseando-se no algoritmo KNN (K vizinhos mais próximos). A abordagem de tradução reversa, como fica claro na Figura 2.10, consiste numa forma sofisticada de obter sinônimos ou até mesmo frases e períodos de mesmo sentido.

Figura 2.10: Estratégia de tradução reversa



Fonte: Compilação do autor [3]

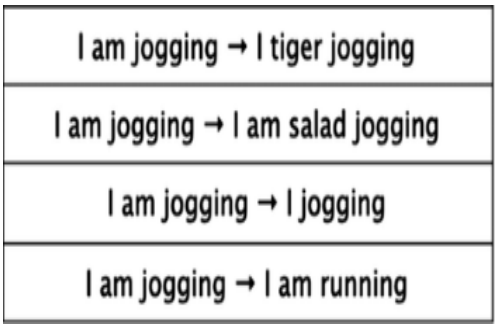
Como se pode observar, a frase em inglês *I have no time* foi traduzida para francês como *je n'ai pas le temps* e retraduzida como *I do not have time*. Um ponto positivo desta abordagem é a utilização imediata de redes neurais *transformers*, arquitetura focada em atenção primeiramente voltada para tradução de textos, conforme Vaswani et al., 2017. *Transformers* são a

[3]Montagem a partir de imagens extraídas de https://amitness.com/2020/02/back-translation-in-google-sheets/

última grande inovação em NLP e fazem parte da sequência esperada deste trabalho. Um ponto negativo desta abordagem é que nem sempre tradutores de bom nível estão disponíveis, além da inclusão de um passo de tradução tornar os fluxos mais lentos.

As técnicas EDA e de transformações em sentenças oferecem diversas opções de *data augmentation*, mas a substituição por sinônimos parece ser a abordagem mais utilizada em NLP atualmente. Um ponto positivo desta estratégia é que não se faz necessário ter um dicionário de sinônimos à disposição, muito embora tal opção seja a melhor. Palavras parecidas podem ser obtidas diretamente do corpus textual, isto é, do conjunto de todos os objetos, na etapa de vetorização dos textos - a ser melhor explicada em seções subsequentes deste trabalho. A Figura 2.11 mostra exemplos das técnicas de EDA.

Figura 2.11: *Estratégia Easy Data Augmentation*



Fonte: Shorten et al, 2021

O presente trabalho pretende gerar dados novos com base em classificações de textos originais, através de *upsample* e clusterização. Combina, assim, aprendizagem de máquina e transformações. Transformações (EDA) serão utilizadas para balancear classes em ODS com poucos registros, enquanto técnicas de clusterização serão utilizadas para agrupar processos avaliados e peças sem avaliação, com o objetivo de projetar etiquetas que aparecem em maior proporção aos textos próximos dos centróides dos *clusters*. O objetivo central da estratégia de *data augmentation* é avaliar se o aumento da base de dados gera melhoria no desempenho dos algoritmos

empregados na classificação dos textos em ODS da Agenda 2030.

2.6 Métricas de Avaliação

As métricas escolhidas para avaliar a performance de modelos de *machine learning*, especialmente em tarefas de classificação, impactam diretamente nos processos de tomada de decisão. De forma resumida, o valor de cada métrica reflete a qualidade do modelo. Se a escolha for errada, é impossível avaliar corretamente se um determinado modelo é útil e atende aos requisitos que a solução do problema deve atender. Existem casos onde erros diferentes possuem custos diferentes e casos onde um tipo específico de erro deve ser terminantemente evitado, sob pena de inviabilizar completamente a solução. Outro aspecto importante no processo de escolha da métrica a ser utilizada é a sua explicação a todas as pessoas envolvidas no projeto. A interpretação de cada métrica é tão importante quanto seu valor absoluto.

As métricas mais populares em tarefas de classificação envolvem quantidades presentes na matriz de confusão, ilustrada pela Figura 2.12.

Figura 2.12: Matriz de Confusão

Fonte: Compilação do autor [4]

onde os valores TP, FN, FP e TN são tais que:

- **Verdadeiro Positivo (TP, do inglês *True Positive*):** ocorre quando o modelo entende que a classe é positiva (1) e, ao verificar o *label* real, vê-se que a classe era realmente positiva.
- **Verdadeiro Negativo (TN, do inglês *True Negative*):** ocorre quando o modelo entende que a classe é negativa e, ao verificar o rótulo, descobre-se que a classe era realmente negativa;
- **Falso Positivo (FP, do inglês *False Positive*):** acontece quando o classificador diz que a classe é positiva, mas ao verificar a resposta, vê-se que a classe era negativa;
- **Falso Negativo (FN, do inglês *False Negative*):** acontece quando o output do modelo é negativo, mas ao verificar a resposta, vê-se que a classe era positiva.

Pode-se notar que as quantidades TP e TN representam acertos, enquanto FP e FN indicam erros. A primeira métrica de interesse em tarefas de classificação decorre deste fato e se chama acurácia. A acurácia avalia simplesmente o percentual de acertos, ou seja, pode ser obtida pela razão $\frac{TP+TN}{TP+TN+FP+FN}$. Avaliar apenas a acurácia é um risco, dado que bases desbalanceadas são extremamente comuns em problemas reais. Supondo um problema de classificação onde, num total de 10 mil entradas apenas 1 mil são da classe A, um classificador que rotula todas as entradas com rótulo B acertará 90% das vezes, errando todas as entradas com rótulo A. Isso é claramente ruim e deve ser evitado em problemas de classificação.

Em problemas desbalanceados, a classe com menos entradas de exemplo normalmente é aquela onde se registra a presença de uma característica interessante ao problema em si. Exemplos são dados pela análise de impressões digitais em homicídios, em que o número de homens é maior que o de mulheres e nos casos onde um processo judicial deve ser classificado em uma etiqueta rara, com poucas entradas. Na prática, problemas desbalanceados são encontrados em

[4]Montagem a partir de imagens extraídas de https://www.kunumi.com/2022/05/18.

maior número. Duas métricas são indicadas neste caso: sensibilidade (também conhecida como *recall*) e especificidade. A primeira métrica indica a capacidade do modelo de detectar com sucesso resultados classificados como positivos. A especificidade avalia o contrário, ou seja, a capacidade de detectar resultados rotulados como negativos. Em termos das quantidades da matriz de confusão, temos, respectivamente, as seguintes fórmulas: $\frac{TP}{TP+FN}$ e $\frac{TN}{TN+FP}$. Este trabalho dará maior ênfase à métrica sensibilidade, uma vez que a abordagem de *data augmentation* será aplicada à ODS da Agenda 2030 com poucos registros.

Existem mais duas métricas populares, chamadas de precisão e *F-score*. A precisão é dada por $\frac{TP}{TP+FP}$ e o *F-score* é dado pela média harmônica entre a precisão e a sensibilidade. A precisão pode ser utilizada em filtros de *spam*, por exemplo. Uma mensagem importante ser considerada *spam* é pior do que um *spam* ser considerado importante. O próximo capítulo do trabalho se dedica à atividade de processamento de linguagem natural, centro do problema de classificação de processos jurídicos em ODS da Agenda 2030 da ONU.

Capítulo 3

Processamento de Linguagem Natural

3.1 Introdução

Processamento de linguagem natural (em inglês, NLP) é uma coleção de métodos que tornam a linguagem humana acessível a computadores e máquinas. Recentemente, tarefas de processamento de linguagem natural invadiram o cotidiano de muitas pessoas, dado que a tradução automática de textos é rápida e viável, *e-mails* são classificados automaticamente como *spam*, motores de busca como o *Google* se tornaram comuns e sistemas de diálogo para auto atendimento são utilizados por muitas empresas.

A identificação do processamento de linguagem natural como desafio computacional, no entanto, data de décadas passadas. Quando Alan Turing criou o *Turing Test*, cujo objetivo era avaliar a capacidade de uma máquina alcançar a inteligência humana, uma tarefa linguística foi escolhida. O termo *chatbot* foi criado mais tarde, mas o Jogo da Imitação (base do teste de Turing) é uma máquina de diálogo. Alguns anos depois, Joseph Weizenbaum criou, no MIT, o primeiro grande exemplo de validação do teste de Turing. A ELIZA (Weizenbaum, 1966), cuja implementação é surpreendentemente simples, foi projetada para substituir terapeutas respondendo questões básicas através de reconhecimento de padrões.

Atualmente, o processamento de linguagem natural é próximo de áreas como Linguística Computacional, Aprendizagem de Máquina, Inteligência Artificial e Ciência da Computação e compreende tarefas como classificação e clusterização de textos, análise de sentimentos, tradução automática, reconhecimento de entidades nomeadas e *part-of-speech* (*pos tag*). As duas primeiras tarefas mencionadas são o foco desta dissertação, dado que o objetivo é aumentar a base de rótulos de ODS da Agenda 2030 com poucas entradas naturalmente, via agrupamento de textos. A atividade de *pos tag* foi utilizada no fluxo de pré-processamento dos textos e está melhor explicada no Apêndice B. Todas as tarefas citadas, no entanto, envolvem vetorizar textos em maior ou menor grau. A etapa de vetorização (*embedding*) consiste na transformação de textos em vetores ou matrizes de números, objetos que os computadores conseguem ler e processar mais facilmente.

3.2 Vetorização de Textos - *Embedding*

Vetores são, em essência, listas de objetos. Já os espaços vetoriais são coleções de vetores de alguma dimensão. Representar objetos em espaços vetoriais significa incorporá-los em ambientes dotados de distância e um dos motivos de se fazer isso é justamente promover comparações entre tais objetos. Representar textos em forma de vetores de números (*embedding*) é, portanto, uma maneira de fazer comparações de textos via algoritmos, modelos ou métodos computacionais. Trata-se de uma etapa fundamental para qualquer atividade envolvendo processamento de textos.

A forma mais simples de representar textos dada uma coleção (*corpus*) é colocar todas as palavras do *corpus* e as identificações dos textos em uma matriz, denominada matriz de coocorrência, como na Tabela 3.1.

Tabela 3.1: Tabela de Coocorrência

	texto 1	texto 2	...	texto n
palavra 1	x_{11}	x_{12}	...	x_{1n}
palavra 2	x_{21}	x_{22}	...	x_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
palavra k	x_{k1}	x_{k2}	...	x_{kn}

Os textos contém, ao todo, k palavras e x_{td} é a frequência da palavra t no documento d. Ainda que palavras inúteis sejam retiradas (*stopwords*), não é razoável pensar que a matriz de co-ocorrência será densa, isto é, com poucas entradas zero, pois é completamente esperado que nem todas as palavras estejam em todos os textos do *corpus*. Além da esparsidade, que ocorre quando a matriz possui muitos zeros em suas entradas, outro ponto de atenção nas matrizes de coocorrência é que, em abordagens exclusivamente baseadas em dicionário, palavras mais frequentes devem ter menor peso, ou seja, menores entradas em matriz de coocorrência. Isso é feito para reforçar, na representação matricial, as palavras com maior potencial de identificar assuntos e que apareçam pouco no *corpus*. Um jeito elegante de aumentar o peso de palavras com maior representatividade no *corpus* é dado pelo algoritmo TF-IDF.

3.3 TF-IDF

A métrica TF-IDF de uma palavra num determinado documento é dada por $TF_{t,d} \times IDF_t$, em que $TF_{t,d}$ (do inglês *Term-Frequency*) são as entradas da matriz de coocorrência e $IDF_t = log_{10}(\frac{N}{df_t})$, com df_t = número de documentos onde a palavra t aparece e N = número de documentos do *corpus*. IDF é a sigla, em inglês, para *Inverse Document Frequency*. Uma forma alternativa de calcular o termo $TF_{t,d}$ é dada por $log_{10}(x_{td}+1)$, onde x_{td} é uma entrada da matriz de coocorrência. A Tabela 3.2 apresenta uma comparação entre a matriz de coocorrência e a matriz obtida pelo algoritmo TF-IDF, baseadas em textos de Shakespeare:

Tabela 3.2: Comparação entre a matriz de coocorrência e o método TF-IDF

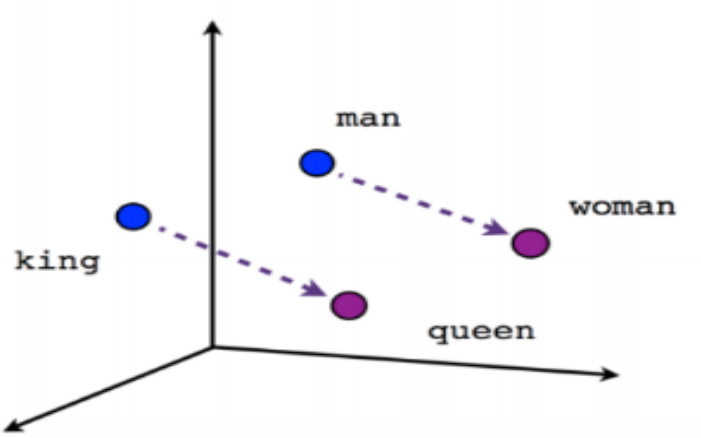
	As You Like It		Twelfth Night		Julius Caesar		Henry V	
batalha	1	0.074	0	0	7	0.22	13	0.28
bom	114	0	80	0	62	0	89	0
idiota	36	0.019	58	0.021	1	0.0036	4	0.0083
sagacidade	20	0.049	15	0.044	2	0.018	3	0.022

Os valores da matriz de coocorrência e TF-IDF são separados por colunas na Tabela 3.2. Importante notar que a palavra "bom" foi eliminada no contexto TF-IDF e a palavra "idiota" teve sua importância drasticamente reduzida. Apesar de não resolver o problema da esparsidade - basta ver que no exemplo existem vários valores 0 - a abordagem TF-IDF é a forma mais popular de *embedding* em processamento de linguagem natural. Um problema da técnica TF-IDF é que palavras com similar distribuição no *corpus* podem ter vetores parecidos, sem, contudo, guardarem proximidade semântica ou de significado. O contrário também vale. Palavras de significado próximo não estão necessariamente próximas no espaço vetorial, dada a esparsidade dos vetores. Os modelos da família *word2vec*, especialmente o método *doc2vec*, resolvem este problema e serão apresentados na subseção a seguir.

3.4 *Doc2Vec*

As representações de texto baseadas em dicionário, comumente chamadas de abordagens *BoW* (*bag-of-words* ou saco de palavras, em português) não se preocupam com contexto e sequer mantém próximas representações de palavras de sentido similar. Considerando a representação da linguagem humana tal qual os próprios humanos a utilizam, vetores de palavras devem formar, idealmente espaços como o da Figura 3.1,

Figura 3.1: Representação de palavras em um espaço vetorial



Fonte: Compilação do autor [1]

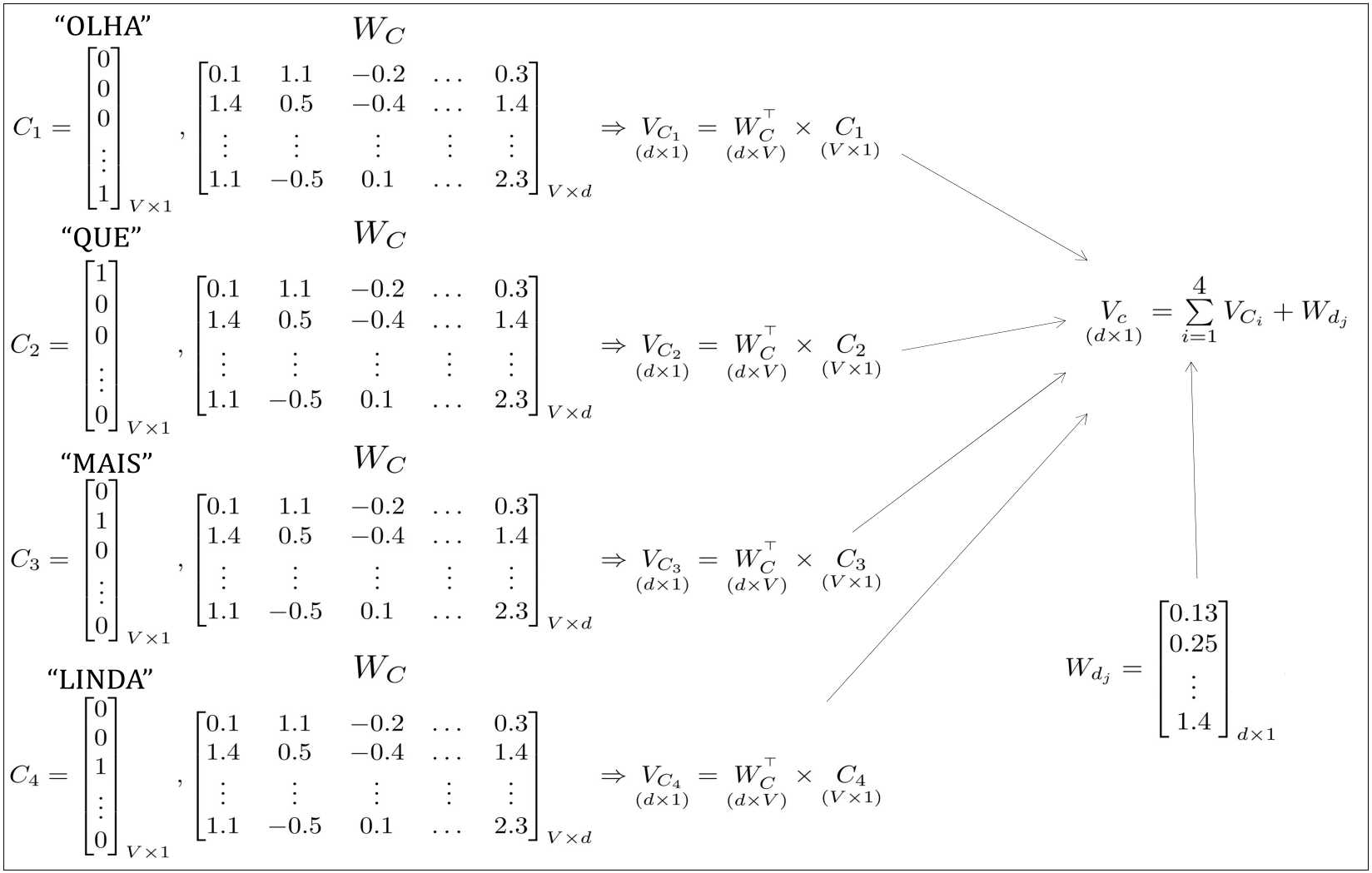
Onde as palavras *"king"* (rei) e *"man"* (homem) estão relativamente próximas e mantém, respectivamente, distâncias parecidas com as palavras *"queen"* (rainha) e *"woman"* (mulher). O método *word2vec*, apresentado por Mikolov et al., 2013, pretende fazer exatamente isso: criar representações numéricas para cada palavra, mantendo as relações de vizinhança e distância tal qual a Figura 3.1 indica. Na prática, em abordagens baseadas em *word2vec*, tem-se que vetor(*"king"*) - vetor(*"man"*) + vetor(*"woman"*) $\approx$ vetor(*"queen"*).

As abordagens *word2vec* utilizam redes neurais para representação de palavras e suas arquiteturas precursoras são a *Feedforward Neural Net Language*, também conhecida como NNLM e proposta por Bengio et al., 2003, e a *Recurrent Neural Net Language Model* (RNNLM), proposta por Mikolov et al., 2010. O modelo *word2vec* é uma rede neural com duas camadas que busca representar palavras artificialmente escondidas de acordo com as palavras da vizinhança *(Continuous Bag of Words - CBOW)* ou prediz a vizinhança com base na palavra central (*Skip-gram*), através dos próprios pesos da rede neural. As duas matrizes de pesos destas ar-

[1] Montagem a partir de imagens extraídas de https://danvitoriano.medium.com.

quiteturas servem para representar as palavras do dicionário, de forma que o foco da rede não é fazer predições de variáveis categóricas ou numéricas. Na primeira abordagem (CBOW), que será utilizada neste trabalho, a primeira camada rede é dedicada a representar as palavras do contexto (vizinhança da palavra alvo escondida) através de uma matriz de pesos iniciada com valores aleatórios e a segunda camada é uma função *softmax* cujos parâmetros são a concatenação dos vetores de contexto e a representação de saída de cada palavra do dicionário (outra matriz de pesos, também iniciada com valores aleatórios). Os pesos da rede são atualizados, época por época, de forma a maximizar os valores obtidos pela função *softmax* da última camada. A arquitetura *doc2vec* (*Distributed Memory Version of Paragraph Vector* – PV-DM) é uma variante do modelo *word2vec* (CBOW) e inclui uma representação, em formato de vetor, para cada texto do *corpus*. A Figura 3.2 mostra uma época do modelo *doc2vec*.

Figura 3.2: Época do modelo *doc2vec* (PV-DM)



Fonte: Próprio autor

Na representação da Figura 3.2, a palavra de interesse é "coisa" (escondida) e as 4 palavras de contexto (C) são "olha", "que", "mais" e "linda". Este procedimento se repete para todas as palavras de todos os textos do *corpus*, de forma sequencial. Os vetores C_1, C_2, C_3 e C_4 são as representações em *one-hot encoding* das palavras "olha", "que", "mais" e "linda", de acordo com a posição de cada uma delas no dicionário de palavras do *corpus*. Na prática, os vetores C_1, C_2, C_3 e C_4 de dimensão $V \times 1$ são formados por números 0 e pelo número 1 na posição de cada palavra entre as V palavras do dicionário. Tais vetores são multiplicados pela primeira matriz transposta de pesos (W_c), que representa as palavras de contexto e é iniciada com valores aleatórios. Os vetores resultantes desta multiplicação $(V_{C1}, V_{C2}, V_{C3}$ e $V_{C4})$ são concatenados entre si e com o vetor W_{d_j}, que é iniciado de forma aleatória e representa o j-ésimo texto, onde as palavras "coisa", "olha", "que", "mais" e "linda" aparecem.

A Figura 3.3 mostra o vetor V_C (concatenação de V_{C_i} com W_{d_j}) como *input* de uma função *softmax*, junto da matriz de pesos W_o, cujas linhas são as representações de palavras alvo. A saída da função *softmax* (h_o) é um vetor de tamanho V, que representa a probabilidade de cada uma das palavras do dicionário ser a palavra escondida $(P(O = w_i|C))$. Os pesos (W_c e W_o) e o vetor W_{d_j} são atualizados via *backpropagation* de forma a maximizar os valores $P(O = w_i|C)$ e assim melhorar a representação das palavras e dos textos, que é o objetivo do modelo *doc2vec*. Na probabilidade condicional obtida via função *softmax*, $w_1, w_2, \ldots, w_v$ são as palavras do dicionário.

Figura 3.3: Camada *softmax* no modelo *doc2vec* (PV-DM)

$$V_C \ , \ W_0 = \begin{bmatrix} 0.2 & -0.3 & 1.2 & \ldots & 0.3 \\ 0.8 & 0.7 & -1.1 & \ldots & -1.5 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.5 & -1.1 & 1.3 & \ldots & 0.7 \end{bmatrix}_{V \times d} \Rightarrow \underset{(V \times 1)}{h_o} = softmax(W_o, V_C) = \begin{bmatrix} P(0 = w_1|C) \\ P(0 = w_2|C) \\ P(0 = w_3|C) \\ \vdots \\ P(0 = w_V|C) \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.03 \\ 0.05 \\ \vdots \\ 0.74 \\ \vdots \\ 0.03 \end{bmatrix}$$

Fonte: Próprio autor

As matrizes W_c e W_o seguem, depois que o número de épocas para treinamento da representação da palavra mascarada é atingido, para a rede neural focada na próxima palavra escondida até que todas as palavras de todos os textos tenham representação. Consequentemente, todos os textos estarão representados pelos vetores da família W_{d_j}. A novidade teórica do *word2vec* e, por consequência, do *doc2vec*, é a adoção de duas matrizes de pesos ou representação de palavras. Uma das matrizes (W_c) gera representações para palavras quando estas aparecem na janela de contexto, ou seja, quando estão na vizinhança de uma palavra escondida. A outra matriz de pesos (W_o) gera representações para as palavras alvo. O modelo *doc2vec*, por sua vez, também inovou ao incluir a representação do texto no *embedding* da cada palavra, tornando-a um objeto atualizável através de *backpropagatio*n. Desta forma, todos os textos influenciam na representação de todas as palavras e isso é o ponto de partida para os modelos mais modernos de atenção, como os *transformers*.

Importante mencionar que os vetores obtidos via *doc2vec* possuem o mesmo tamanho e são todos densos, resolvendo o grande problema da esparsidade nas abordagens TF-IDF. As dimensões sugeridas para vetores obtidos via *doc2vec* oscilam entre 300 e 500, sendo admitidas representações menores em textos curtos. Tais representações podem ser utilizadas isoladamente para classificar ou agrupar textos sem uso de algoritmos de aprendizagem de máquina. Maneiras interessantes para classificar processos parecidos apenas com uso de vetores envolvem calcular distâncias por cosseno (*cosine similarity*) entre as representações (*embeddings*) de dois textos. A similaridade por cosseno é dada por:

$$cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}, \tag{3.1}$$

onde A e B são vetores e $cos(\theta)$ indica a similaridade. Tal métrica varia entre [-1,1], sendo 1 a similaridade total (vetores na mesma direção - textos iguais). A similaridade por cosseno será utilizada no presente trabalho como um dos métodos auxiliares de classificação de textos, junto

das redes neurais e dos modelos *CatBoost*. Para cada texto não rotulado, observa-se o texto etiquetado com maior similaridade por cosseno, replicando suas etiquetas de ODS. Em resumo, a similaridade por cosseno é utilizada para repetir as etiquetas de processos rotulados em processos não etiquetados. A próxima seção se ocupa de apresentar informações mais detalhadas sobre a atividade de agrupar textos, especialmente no contexto do poder judiciário.

3.5 Clusterização de Textos

As principais formas de agrupar textos envolvem comparação dos vetores obtidos na etapa de *embedding*. Tecnicamente, os agrupamentos reúnem textos cujos respectivos vetores são parecidos. No presente trabalho, o algoritmo de clusterização utilizado é da família *k-means*, apresentada na Seção 2.3. Aos vetores, podem se aplicar técnicas de redução de dimensionalidade como PCA, já citadas anteriormente, com foco na avaliação dos *clusters* obtidos por outros métodos ou até mesmo antes do agrupamento, como forma de tornar os vetores obtidos menores e mais densos, conforme Amilar, 2018.

O agrupamento de textos, especialmente no poder judiciário, serve ao propósito de reduzir atividades repetitivas e manuais de servidores e especializar grupos de trabalho com foco em textos parecidos. Os benefícios desta atividade tem grande impacto nos fluxos de trabalhos e, consequentemente, na prestação jurisdicional dos tribunais. Em linhas gerais, processos parecidos podem ser tratados mais rápido com grupos de servidores e juízes especializados na questão ou classe processual em tela. No Supremo Tribunal Federal, por exemplo, há o instrumento da repercussão geral, cujo objetivo é uniformizar a interpretação constitucional sem exigir que o STF decida múltiplos casos idênticos sobre a mesma questão constitucional. Tal atividade é, em última instância, uma tarefa de similaridade e agrupamento processual.

No Superior Tribunal de Justiça (STJ) há uma iniciativa de inteligência artificial focada em agrupamento semântico de textos chamada Athos. A plataforma Athos, hoje totalmente inte-

grada aos fluxos de trabalhos do STJ, percorre algumas atividades básicas de processamento de linguagem natural, tais como conversão de palavras para letras minúsculas, remoção de *stopwords*, remoção de termos entre parênteses, retirada de caracteres especiais (como pontuação), singularização de verbetes e conversão em NGramas (conforme Amilar, 2018). Ngramas são palavras que aparecem mais comumente juntas do que separadas, em determinado contexto. A vetorização de textos é feita via *doc2vec*. No primeiro treinamento realizado pela ferramenta, 300 mil acórdãos foram apresentados e o algoritmo *doc2vec* foi ajustado com 1.000 épocas, organizando os documentos em um espaço vetorial de 300 dimensões. Atualmente, a base de dados do Athos possui mais de 9 milhões de peças vetorizadas e indexadas para comparação e pesquisa textual.

Há também o exemplo da ferramenta Berna, do Tribunal de Justiça de Goiás (TJGO). A ideia é agrupar demandas repetitivas e impedir máculas no princípio do juiz natural, que proíbe a criação de tribunais extraordinários e transferência de causa para outros tribunais. Um exemplo de demanda repetitiva ocorre quando vários processos sobre problemas com a mesma companhia aérea, mesmo número de voo, horários e datas iguais são peticionados em juizados diferentes com diferença de minutos. A iniciativa Berna é um bem sucedido exemplo da combinação entre metadados e textos processuais, dado que a localização da petição (juizados) é parte fundamental do problema.

O treinamento de *embeddings*, bem como a aplicação dos principais algoritmos de clusterização, é bastante custoso computacionalmente e os metadados cumprem importante papel na redução do universo de textos a agrupar. Outra estratégia para contornar o problema do custo computacional envolvido na atividade de clusterização é associar clusterização e classificação de textos. Neste caso, há um agrupamento inicial seguido da aplicação de algoritmos de aprendizagem supervisionada para acertar os *clusters*, com as entradas dadas pelo modelo de *embedding* utilizado no agrupamento. Maiores informações sobre tal abordagem, que é inspirada em técnicas CSVM (*Clustered Support Vector Machines*), podem ser encontradas em Gu

e Han, 2013

A associação entre agrupamento e classificação de textos jurídicos, neste trabalho, tem objetivo de aumentar a base de dados com determinadas etiquetas de ODS da Agenda 2030 da ONU, em uma ordem inversa em relação às abordagens que usam a classificação para acertar *clusters*.

3.6 Classificação de Textos

A classificação textual serve para rotular textos de acordo com etiquetas previamente estabelecidas. Rotular e agrupar podem ser atividades parecidas, mas em aprendizagem de máquina há diferença em relação à construção dos resultados. A atividade de classificação textual normalmente é feita com supervisão e o agrupamento de textos envolve reunir textos de *embeddings* parecidos, ou seja, é algo não supervisionado.

No contexto jurídico, classificar peças é uma atividade básica em qualquer tribunal. Todos os processos são rotulados, uma vez que os metadados extraídos de tais rótulos servem para guiar a autuação, iniciar ou finalizar o rito processual e gerar jurisprudência. Em última análise, metadados podem ser utilizados para avaliar a atuação dos próprios tribunais, como no caso dos ODS da Agenda 2030. A classificação de textos serve, desta forma, para sugerir etiquetas automaticamente e com base na análise rápida de conjuntos de treinamento com rótulos parecidos.

Existem muitos métodos de classificação textual baseados em aprendizagem de máquina e aprendizagem profunda. Algoritmos como *Naive Bayes*, SVM, *XGBoost* e *CatBoost* são muito utilizados para etiquetar textos, sendo este último um imenso avanço recente. Algoritmos da família *CatBoost* permitem utilizar, dentro do próprio método e sem esquemas de votação (*ensemble*), textos e metadados textuais. Redes neurais profundas também são muito utilizadas para classificação textual, sendo as arquiteturas mais escolhidas as LSTM e CNN, adaptadas

da classificação de imagens. O presente trabalho busca comparar e, por consequência, fazer ensemble de modelos baseados em similaridade por cosseno, redes neurais e algoritmos *CatBoost*. Maiores informações sobre classificação de textos podem ser encontradas em Aggarwal e Reddy, 2014 e Hvitfeldt e Silge, 2021.

O próximo capítulo é dedicado à metodologia aplicada no presente trabalho. Apresenta a base de dados, fluxo de limpeza de textos e parâmetros utilizados no *doc2vec*. Apresenta as estratégias de clusterização e classificação, com os parâmetros utilizados no algoritmo *k-means* e a regra para projeção de etiquetas no aumento da base de dados. Por fim, apresenta o ajuste de redes neurais LSTM, modelos *CatBoost* e abordagens de similaridade.

Capítulo 4

Metodologia

4.1 Introdução

A proposta deste trabalho é usar métodos de agrupamento para reunir textos etiquetados e não etiquetados em ODS da Agenda 2030 da ONU, com objetivo de usar a proximidade de processos rotulados em estratégias de *data augmentation*. Ao final do fluxo proposto, é esperado que as etiquetas sintéticas produzidas via clusterização tornem mais fácil o problema de classificação em rótulos desbalanceados, que ocorre em ODS da Agenda 2030 com poucas entradas naturalmente.

É importante entender quais textos jurídicos são utilizados para rotular processos do STF em ODS, bem como a limpeza realizada nos próprios textos. Outro aspecto importante, que afeta diretamente qualquer abordagem de *data augmentation*, é o tamanho do aumento a ser promovido. Na prática, não adianta utilizar dados aumentados ao ponto de desbalancear os dados já existentes. Conjuntos balanceados envolvem quantidades não muito diferentes entre as classes, que no caso em tela são 1, quando um processo tem etiqueta de determinado ODS e 0 caso contrário. Todas as análises foram feitas considerando cada ODS como um problema isolado de aprendizagem de máquina e *deep learning*, pois problemas com rótulos não exclusivos exigem bases maiores. Isso justifica a estratégia de quebrar o problema original em problemas

menores e binários. Maiores explicações sobre a base de dados utilizada serão dadas na seção a seguir.

4.2 Base de Dados

A base etiquetada consiste em 2.100 textos de acórdãos e petições iniciais de processos do Supremo Tribunal Federal (STF). Acórdãos são registros de julgamentos colegiados em órgãos da justiça e petições iniciais, como o próprio nome indica, incluem os primeiros pedidos das partes, quando estas decidem provocar o poder judiciário. A classificação original (rótulo a ser utilizado em procedimentos de aprendizagem supervisionada) é feita por servidores do próprio tribunal, que usam, além da própria experiência, estes dois documentos (acórdãos e petições). A expectativa é replicar as classificações, via NLP, para automatizar sugestões e tornar o processo de trabalho mais rápido.

No sítio eletrônico do STF para a Agenda 2030 há um painel contendo o número de etiquetas por ODS (Figura 1.2). Pode-se verificar que alguns ODS possuem pouquíssimas entradas, enquanto o ODS 16 é utilizado muitas vezes. No recorte da Figura 1.2, em particular, 55% dos processos possuíam a marcação de ODS 16, relacionado à Paz, Justiça e Instituições Eficazes. O número de entradas do ODS 16 e seu natural balanceamento é o que se deseja, em termos de aprendizagem de máquina e automação, para os outros objetivos de desenvolvimento sustentável. A estratégia de *data augmentation* proposta por este trabalho se aplica imediatamente a qualquer ODS. No sítio eletrônico da Agenda 2030 no STF é possível verificar quais processos foram etiquetados até agora e alguns metadados interessantes para análise de dados jurídicos e jurimetria. As peças podem ser obtidas, quando não estão em segredo de justiça e sua exposição não infringe a Lei Geral de Proteção de Dados (LGPD), no próprio site do tribunal.

A base sem etiquetas é composta por mais de 40 mil acórdãos do STF, proferidos entre 2015 e 2018. São processos que não foram avaliados em termos de ODS, dado que tais etiquetas co-

meçaram a ser utilizadas em 2020 e não são obrigatórias para autuação dos processos judiciais. Tais textos serão utilizados neste trabalho para incrementar a base inicial, incluir novos contextos/palavras e balancear os dados originais, fundamentalmente em ODS de poucas entradas. As duas bases são limpas da mesma forma, de modo a manter os textos uniformemente livres de números, palavras de parada, pontuações e demais termos inúteis para aprendizagem de máquina. A Tabela 4.1 mostra um exemplo de texto bruto e seu respectivo texto limpo:

Tabela 4.1: Exemplo de limpeza dos textos

Texto Original: supremo tribunal federal ementa e acórdão inteiro teor do acórdão - página 1 de 6 09/12/2014. primeira turma ag.reg. em tutela antecipada na ação cível originária 1.824 amapá relator : min. marco aurélio agte.(s) : estado do amapá proc.(a/s)(es) : procurador-geral do estado do amapá agdo.(a/s) : união proc. (a/s)(es) : advogado-geral da união pessoa jurídica de direito público
Texto Limpo: primeira turma tutela antecipada cível originária amapá relator marco aurélio estado amapá procurador geral estado amapá união advogado geral união pessoa jurídica direito público

Pode-se notar que o texto limpo é de menor dimensão, mas continua denso. A diminuição dos textos é fundamental para que os *embeddings* sejam representativos. Palavras inúteis prejudicam a arquitetura *doc2vec*, pois tomam espaço de termos significativos na janela de contexto. Uma estratégia adicional é dada pela manutenção de palavras classificadas como substantivos, verbos ou adjetivos. É uma forma agressiva de limpeza, com passo adicional de *pos tag*, com objetivo de manter apenas os termos com significado semântico. A tarefa de *pos tag*, cuja importância transcende sua utilização como ferramenta auxiliar de limpeza de textos, será explicada em maiores detalhes no Apêndice B.

Os textos limpos são organizados em tabelas, onde uma coluna de identificação (ID) guarda a memória sobre os processos etiquetados e os textos não avaliados. Isso é importante nas estratégias de treinamento, validação e teste, principalmente na etapa de agrupamento dos textos,

a ser explicada na seção subsequente.

A estratégia de *upsample*, que conta também com transformações EDA (Seção 2.5) para trocar 10% das palavras por sinônimos contextuais, foi feita através do pacote *nlpaug*, em linguagem Python. Em síntese, a classe com menos representantes é aumentada até se igualar à quantidade de representantes da classe majoritária. No presente trabalho, porém, os textos não são apenas replicados, como no *upsample* tradicional; 10% de suas palavras são trocadas por outras de contexto parecido. Tal troca é feita através de *transformers* BERT (*bert-base-portuguese-cased*), disponíveis no repositório Souza, Nogueira e Lotufo, 2020.

Os modelos BERT, cuja sigla significa *Bidirectional Encoder Representations from Transformers*, possuem arquitetura centrada em atenção *multi-head*, que relaciona todas as palavras dos textos na tarefa de construção dos *embeddings*. A implementação BERT em língua portuguesa (do Brasil) faz uso do *corpus brWaC*, com 3.5 milhões de páginas retiradas de textos gerais da *internet* e 2.7 bilhões de *tokens* (palavras ou grupos de palavras com significado próprio). Os modelos de arquitetura BERT foram treinados para acertar palavras escondidas dado o contexto (estratégia *word mask*) com 1 milhão de épocas para treinamento.

Os textos limpos, organizados em tabelas e com este *upsample* preliminar são, então, submetidos ao modelo *doc2vec*, que foi treinado exclusivamente em textos jurídicos (diferente do BERT) para gerar vetores de 300 posições. Tais vetores são *input* para a etapa de clusterização, melhor explicada a seguir.

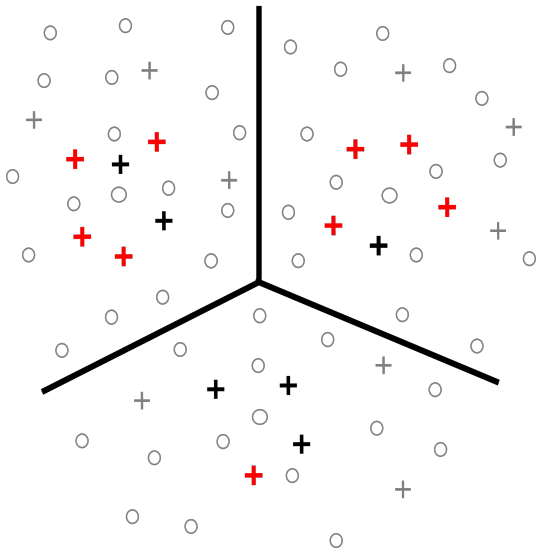
4.3 Estratégia de Clusterização

A estratégia de clusterização apresentada por este trabalho é derivada de um problema de aprendizagem semi-supervisionada e detecção de dígitos em imagens sugerido por Geron, 2021. Considerando as duas bases disponíveis, parte do conjunto com etiquetas (60%) é dedicado à etapa de treinamento ou ajuste dos *clusters* via *k-means*, junto da base não rotulada. Ao todo,

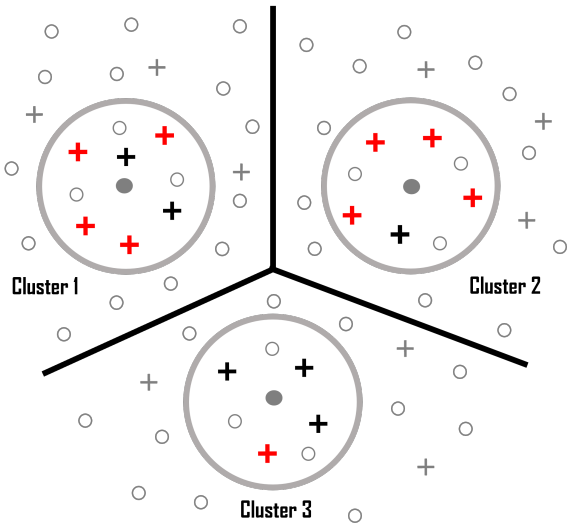
a quantidade de processos utilizados para agrupamento nesta etapa (treinamento) oscila em torno de 44 mil entradas, a depender do *upsample* realizado para cada ODS. Em síntese, esta quantidade de processos (cerca de 44 mil) refere-se à soma de 60% da base etiquetada (cerca de 1.200 processos para cada ODS) com a base não rotulada. As Figuras 4.1 e 4.2 mostram a estratégia de clusterização, que é centrada em determinar os grupos via *k-means*, escolher processos afastados das bordas segundo um raio definido a priori e então replicar as etiquetas para os processos inicialmente não rotulados, conforme limiar de classificação também definido a priori.

A Figura 4.1 (A) mostra um exemplo de representação em 3 dimensões para vetores obtidos via *doc2vec*, onde existem processos pertencentes à base rotulada (cruzes) e da base não rotulada (círculos), bem como processos com etiqueta 1 para determinado ODS (vermelho) e etiquetados com etiqueta 0 para o mesmo ODS (preto). Ajustando 3 *clusters* e selecionando apenas os processos em um raio de 50% do centróide (procedimento que está graficamente explicado na Figura 4.2), tem-se a Figura 4.1 (B), também ilustrando os processos etiquetados e não etiquetados. No *cluster* 1, dentro do raio especificado, existem 6 processos rotulados, dos quais 4 (67%) possuem etiqueta 1 para o ODS em análise. Dessa forma, considerando neste exemplo o limiar de classificação de 70%, os processos não rotulados deste *cluster* receberão etiqueta 0 para o ODS em tela, como se pode ver na representação do primeiro *cluster* na Figura 4.1 (C), onde os processos inicialmente não rotulados (círculos) estão na cor preta (assim como o círculo de raio 50% tornou-se preto). No *cluster* 2, existem 5 processos etiquetados dentro do raio, dos quais 4 (80%) possuem etiqueta 1 para o ODS avaliado. Os processos inicialmente não rotulados deste *cluster*, então, receberão a etiqueta 1 para o ODS avaliado e estão em cor vermelha na representação da Figura 4.1 (C) (de forma semelhante, o círculo de raio 50% tornou-se vermelho). Neste exemplo, o *cluster* 3 tem apenas 25% dos processos com etiqueta 1 dentro do raio e seus processos não rotulados receberão etiqueta 0, tal qual os do *cluster* 1.

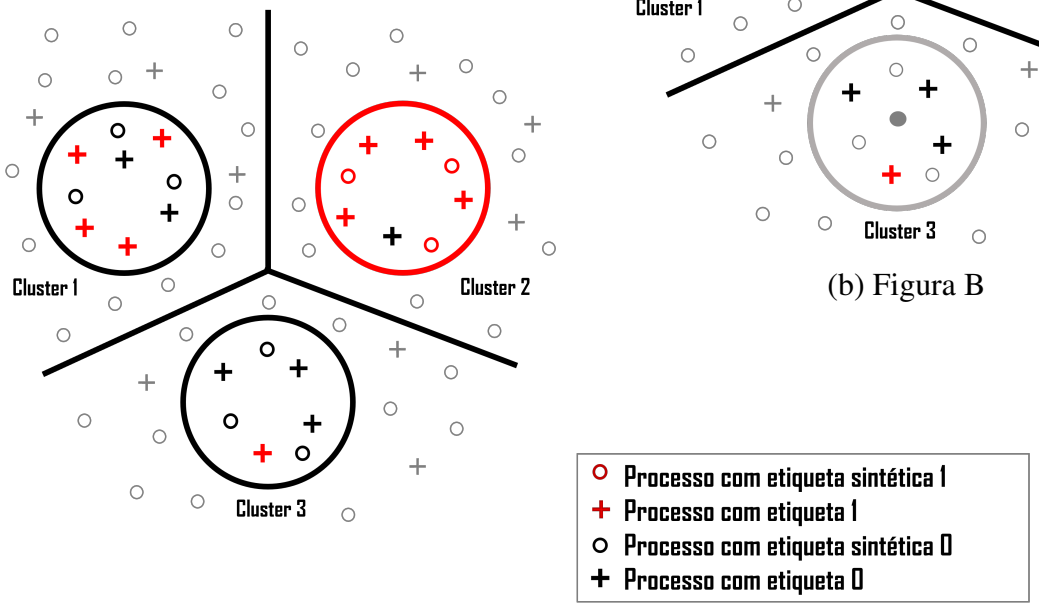
Figura 4.1: Etapa de treinamento da estratégia de clusterização para 1 ODS genérico. (a) Segmentação em 3 *clusters* (b) Ilustração do raio de 50% do centro (c) Ilustração do limiar de 70% para etiqueta sintética 1



(a) Figura A



(b) Figura B

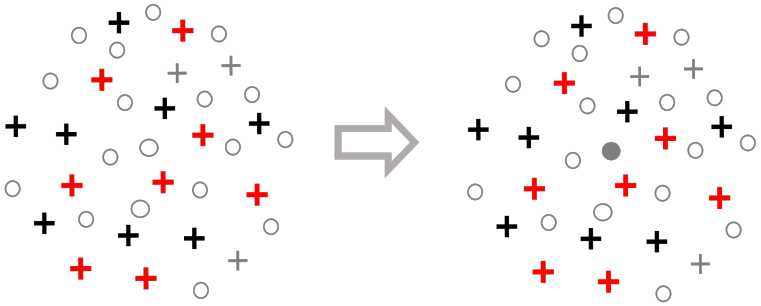


(c) Figura C

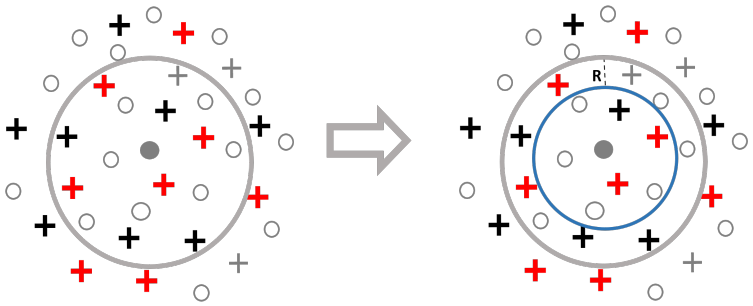
Fonte: Próprio autor

A Figura 4.2 representa graficamente o parâmetro raio do centro. A distância R, obtida depois das etapas de determinação do centróide (Figura 4.2 (A)) e agrupamento (Figura 4.2 (B)), é representada pela diferença entre os círculos cinza e azul e indica que os processos representados pela Figura 4.1 (B) e 4.1 (C) estão longe das bordas, sendo, portanto, mais parecidos entre si. A margem criada pelo parâmetro raio do centro mantém os processos utilizados para propagação das etiquetas teoricamente longe de processos de outros *clusters*.

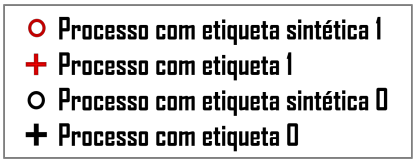
Figura 4.2: Definição do raio na estratégia de data augmentation do processo de clusterização



(a) Figura A



(b) Figura B

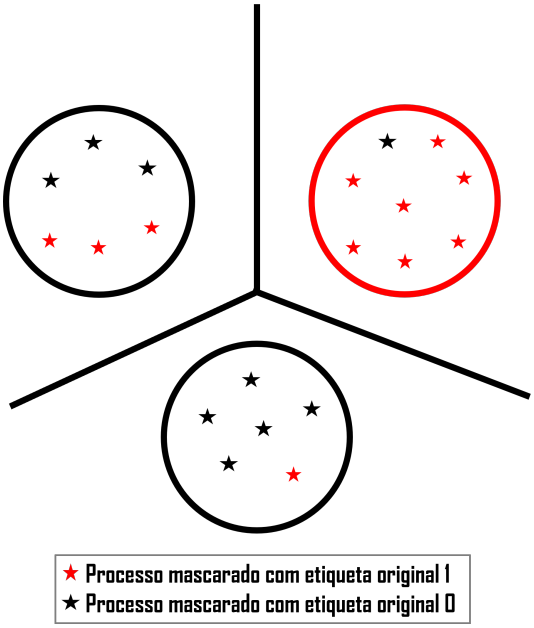


Fonte: Próprio autor

Para verificar qual configuração ótima para os parâmetros - raio do centro, limiar de classifi-

cação e número de clusters - 20% da base etiquetada é utilizada para validação. Estes processos são mascarados na clusterização, de forma a esconder sua etiqueta verdadeira (1 para texto com ODS e 0 para textos sem a marcação de ODS). Para cada uma das combinações de raio, limiar e número de clusters em análise, uma etiqueta sintética é atribuída a estes processos mascarados, com base nas etiquetas dos vizinhos próximos dentro do próprio *cluster*. A comparação entre as etiquetas sintéticas e verdadeiras é a métrica de validação. A Figura 4.3 mostra *clusters* cuja etiqueta sintética atribuída é dada pela cor da borda. Estrelas na cor vermelha indicam processos com etiquetas originais 1 e estrelas na cor preta indicam etiquetas originais 0.

Figura 4.3: Etapa de validação com a visualização das etiquetas originais dos processos mascarados



Fonte: Próprio autor

Pode-se notar que a acurácia é dada por $\frac{3+7+5}{6+8+6} = \frac{15}{20} = 75\%$ e a sensibilidade por $\frac{7}{8} = 87\%$. Tais cálculos permitem a construção de uma matriz de validação para cada tamanho de *cluster*, objeto que será melhor explicado no capítulo seguinte. Determinados os tamanhos de *clusters*, raios e limiares ótimos, é possível avaliar se a clusterização performa bem na base

separada para teste (20% da base original). A estratégia se parece com a adotada na etapa de validação, onde as etiquetas originais são mascaradas e etiquetas sintéticas são geradas através dos *clusters*. Cumpre registrar que tais métricas (acurácia e sensibilidade) são utilizadas para escolher as melhores combinações de parâmetros - número de *clusters*, raio do centro e limiar de classificação - para todos os ODS em análise. O custo computacional de agrupar todos os processos via *kmeans* depende do número de *clusters*, oscilando para o presente trabalho entre 17 minutos e 1 hora. A próxima seção é dedicada à estratégia de classificação, ou seja, aos modelos ajustados para sugerir etiquetas de ODS.

4.4 Estratégia de Classificação

Essa seção tem o objetivo de justificar e apresentar os ajustes de redes neurais LSTM, a abordagem de similaridade e os modelos *CatBoost* utilizados para classificação dos textos (incluindo as bases aumentadas) nos ODS selecionados da Agenda 2030 - 3, 4, 8, 9, 10, 11, 15, 17 e 16. O próximo capítulo, que trata dos resultados, mostra comparações entre a classificação realizada sem as bases aumentadas e os ajustes feitos com as bases de etiquetas sintéticas obtidas pelo agrupamento.

As redes de arquitetura LSTM, já devidamente apresentadas na Seção 2.4 deste trabalho, são melhores que as redes recorrentes de arquitetura padrão (também chamadas de *vanilla* RNN) pois tem maior capacidade de lidar com dependências de longo prazo por meio de sua arquitetura de três portas diferentes: porta de entrada, porta de saída e porta de esquecimento. As três portas operam juntas para decidir quais informações lembrar e o que esquecer na célula LSTM durante um tempo arbitrário.

As redes LSTM foram ajustadas usando a biblioteca Pytorch, desenvolvida para *deep learning*, visão computacional e NLP pelo FAIR (*Facebook AI Research lab*). Atualmente, o projeto Pytorch é gerenciado pela Pytorch Foundation, subsidiária da Fundação Linux. Também

é possível ajustar redes neurais profundas com o *framework* Tensorflow, ambos (Tensorflow e Pytorch) servem para manipular tensores e grafos, objetos básicos para fluxos de *deep learning*. Tutoriais sobre Pytorch estão disponíveis em Paszke et al., 2019. O Apêndice A traz maiores informações sobre *frameworks* para processamento de linguagem natural (NLP).

Como a atividade de classificação é diferente da fase de agrupamento, somente a etapa de limpeza dos textos pode ser reaproveitada. Por se tratar de uma biblioteca de sintaxe própria, o Pytorch usa um instrumento de *embedding* baseado em *one-hot encoding* e pesos. A estratégia *one-hot* preenche vetores esparsos para cada palavra do *corpus* (*bag of words* - BoW). Supondo que o termo "batalha" seja a décima quinta palavra a aparecer no *corpus* de textos de Shakespeare, seu vetor representativo teria dimensão M igual ao tamanho do BoW, com M-1 valores 0 e o valor 1 na posição 15. Textos são representados via *one-hot encoding* pela coleção de vetores de palavras construídos desta forma. O *framework* Pytorch faz a vetorização de palavras (e textos, consequentemente) combinando o índice igual a 1 (15, no exemplo) com pesos inicializados aleatoriamente. Não são indicados, em arquiteturas LSTM, *inputs* oriundos de *embeddings* com contexto, tais como *doc2vec*, uma vez que a própria rede possui memória e guarda relações entre termos distantes nos textos.

As camadas de LSTM da rede recebem, no passo *forward*, as saídas da etapa de *embedding* e fazem seguidas reduções de dimensão, considerando pesos para relação entre termos distantes (portas da arquitetura), camadas de *dropout* e camadas de ativação. *Dropout* é um método de regularização onde as conexões de entrada para unidades LSTM são excluídas probabilisticamente das atualizações de peso durante o treinamento de uma rede. O modelo retorna, ao final da execução, probabilidades de pertencimento para cada categoria (0 sem relação com o ODS em tela e 1 caso contrário). O valor de 0.5 é utilizado como *threshold* por padrão, de forma que probabilidades acima de 0.5 indicam rótulo 1.

As redes foram treinadas com 1000 épocas, usando otimizador Adam (*learning rate* de 0.001), função de perda da família *Binary Cross Entropy* (BCELoss) e duas camadas de *dro-*

pout com probabilidades 0.8 e 0.6. A partir da definição da arquitetura de rede, tais parâmetros podem ser definidos por tentativa e erro ou com base em redes ajustadas com sucesso em problemas parecidos.

O custo computacional de treinar redes LSTM nas bases aumentadas é bastante alto. Por este motivo, instâncias do Google Colab Pro+ foram utilizadas. Google Colab é um ambiente de desenvolvimento em nuvem com os principais pacotes Python previamente instalados. Os tempos computacionais foram de 59 minutos, em média, considerando a aceleração de *hardware* usando TPUs V2 e RAM de 52gb. Na base original, este tempo é inferior a 10 minutos.

Os modelos *CatBoost* foram ajustados em linguagem Python, muito embora existam implementações em C, Java, Rust e `R`, além de uma API para Apache Spark. GPUs *premium* disponíveis no ambiente Google Colab Pro+ foram utilizadas e o tratamento de *features* categóricas e de texto foi realizado através dos métodos padrão da biblioteca. Os tempos computacionais são sensivelmente menores que os verificados com as redes neurais (43 minutos, em média). Os modelos *CatBoost* são muito versáteis e permitem utilizar apenas as últimas k árvores na etapa de predição, assim como existem módulos adaptando a implementação básica para tarefas como detecção de *outliers* e *feature importance*.

A abordagem baseada em similaridade é livre de algoritmos, ou seja, utiliza apenas os *embeddings* gerados via *doc2vec*. É o núcleo de muitas iniciativas de IA no judiciário brasileiro, como o projeto Athos do STJ. A similaridade por cosseno é utilizada no presente trabalho, mas existem módulos em Python com outras métricas de distância, tais como Mahalanobis, Metropolis, Euclidiana e etc. Muito embora a implementação de estratégias baseadas em similaridade sejam simples, o custo computacional tende a ser alto. Nesta dissertação, as matrizes de similaridade possuem mais de 4gb inicialmente e a escolha das etiquetas de ODS para um processo em avaliação é feita através da replicação das etiquetas do processo mais parecido. Um ponto positivo desta abordagem, em relação ao custo computacional, é que se trata de estratégia válida para todos os ODS, ao contrário das redes e dos modelos *CatBoost*, que precisam ser instancia-

dos e ajustados para cada ODS individualmente.

A utilização das redes LSTM, algoritmos *CatBoost* e estratégias de similaridade reúne modelos simples (similaridade e árvores de decisão), com memória para dependências de longo prazo (LSTM) e com combinação de *features* categóricas e em texto (*CatBoost*). Os modelos de *ensemble*, que combinam modelos em estratégias de votação, buscam reunir as melhores características de cada modelo individual, algo que o presente trabalho pretende fazer. O fluxo completo da metodologia proposta no trabalho é dado pela Figura 4.4.

Figura 4.4: Fluxograma

Fonte: Próprio autor

As etapas ilustradas na Figura 4.4 estão sintetizadas a seguir:

1. Limpeza dos textos - indica o pré processamento, limpeza, passo de *pos tag* nos textos e vetorização de textos via *doc2vec*.

2. Organização das bases - indica a separação das bases etiquetadas e não etiquetadas. 60%

da base etiquetada é utilizada para treinamento, 20% para validação dos hiperparâmetros e 20% para teste.

3. *Upsample* via sinônimos contextuais - indica aumento da base de treinamento, via *upsample* associado à sinônimos obtidos através de *transformers* BERT.

4. Clusterização - etapa de agrupamento dos textos, identificação de centróides e raio para propagação de etiquetas. Nesta etapa, que é realizada na base de treinamento pós *upsample* e na base não rotulada, é feita a validação dos parâmetros da própria clusterização.

5. Propagação de etiquetas - indica a etapa onde processos inicialmente não rotulados recebem etiquetas sintéticas relacionadas aos vizinhos rotulados originalmente. Nesta etapa, o conjunto de teste (formado por 20% da base etiquetada) é utilizado para avaliação da performance do agrupamento e da própria propagação de etiquetas.

6. Classificação - indica a etapa onde as novas bases, obtidas pelo passo de propagação de etiquetas, são utilizadas no treinamento de modelos de redes neurais e *CatBoost*, além de estratégias de similaridade, com objetivo de melhorar o fluxo de classificação processual.

As caixas em laranja indicam as etapas do fluxo completo onde todos os ODS são tratados de forma conjunta, limpeza e organização das bases. As demais caixas, que estão em azul, indicam etapas onde os ODS são tratados de forma separada. A principal ideia do próximo capítulo, dedicado aos resultados, é verificar se as bases aumentadas produzem melhorias de performances nas redes LSTM, modelos *CatBoost* e de similaridade utilizados para sugerir automaticamente etiquetas de ODS.

Capítulo 5

Resultados

5.1 Introdução

Este capítulo reúne informações sobre os resultados da clusterização e dos métodos de classificação. Para a clusterização são apresentados os resultados das etapas de treinamento, validação e testes. Os resultados da etapa de classificação envolvem métricas de acurácia e sensibilidade para os três métodos escolhidos - redes LSTM, algoritmos *CatBoost* e similaridade por cosseno. A similaridade é o único dos métodos que não possui etapa de treinamento, dado que não envolve estimação de parâmetros, sendo aplicado diretamente aos *embeddings* obtidos via *doc2vec*. A opção pelo *ensemble* dos três modelos se dá pela expectativa de melhorar o desempenho dos modelos isolados.

As seções a seguir se ocupam de mostrar a disposição original das etiquetas de ODS, discussões sobre resultados intermediários da etapa de clusterização, sobre a distribuição das etiquetas depois do aumento de dados e sobre a performance das redes LSTM com as bases original e aumentada, bem como comparações entre os resultados obtidos pelos três modelos de classificação, individualmente e em conjunto (estratégia *ensemble*). Ideias de melhoria e sobre trabalhos futuros são apresentadas no capítulo seguinte da dissertação, que ainda conta com dois anexos

dedicados aos principais *frameworks* para processamento de linguagem natural em `R` e Python e sobre a atividade de *pos tag*, utilizada na parte de limpeza de textos.

5.2 Resultados - Clusterização

A etapa de clusterização é a mais importante do presente trabalho e possui muitos resultados intermediários interessantes. Os ODS em análise possuem distribuição de etiquetas originais (propostas por funcionários do tribunal) apresentadas na Tabela 5.1, onde se verificam importantes desbalanceamentos entre as classes. O ODS 16 - Paz, Justiça e Instituições Eficazes é o único que possui mais etiquetas 1 do que etiquetas 0, visto que se trata de um objetivo de desenvolvimento sustentável que pode ser imediatamente aplicado em quase todas as peças de processos em cortes constitucionais, como é o caso do STF. O claro desbalanceamento entre as classes e o baixo número de etiquetas 1, exemplos positivos de ODS relacionados com as peças jurídicas, justificam a estratégia de *data augmentation*, tema principal desta dissertação.

Tabela 5.1: Distribuição de etiquetas dos processos etiquetados manualmente

ODS	Etiquetas 0	Etiquetas 1
ODS 3	1635	370
ODS 4	1877	128
ODS 8	1559	446
ODS 9	1937	68
ODS 10	1635	370
ODS 11	1914	91
ODS 15	1909	96
ODS 16	763	1242
ODS 17	1787	218

Pela Tabela 5.1 pode-se notar que os ODS 9 - Indústria, Inovação e Infraestrutura, ODS 11 - Cidades e Comunidades Sustentáveis e ODS 15 - Vida Terrestre são os que possuem menor número de etiquetas 1. São casos extremos, para os quais servidores que trabalham diariamente na autuação de processos dispõem de poucos exemplos para ajudar no processo de classificação.

Tratam-se, portanto, de etiquetas e processos de trabalho que podem ganhar em qualidade com a estratégia de aumento de dados. Outro ponto de interesse na estratégia de *data augmentation* tem relação com as etiquetas de difícil classificação, como no caso dos ODS 16 e 17. Tal dificuldade pode ocorrer pela forte aderência do ODS às peças processuais (ODS 16) ou pelo caráter genérico do objetivo de desenvolvimento (ODS 17).

O procedimento que será mostrado a seguir para o ODS 3 foi realizado para todos os tamanhos de *cluster* propostos e todos os ODS em análise, representando, de forma exemplar, a estratégia de clusterização adotada no trabalho. O fluxo é o mesmo descrito nos primeiros cinco itens da Figura 4.4 e inclui os passos 1. *upsample* para balanceamento da base de dados para cada ODS, 2. separação das bases de treinamento, validação e teste também para cada ODS, 3. agrupamento de textos, 4. escolha dos parâmetros ótimos na validação e 5. etapa de teste.

1. *Upsample* para balanceamento da base de dados para cada ODS. Como o ODS 3 - Saúde e Bem Estar possui originalmente 1622 etiquetas 0 e 370 etiquetas 1, o *upsample* via sinônimos contextuais aumenta a base em 1252 processos (de etiqueta 1), tornando a base original balanceada. A nova base rotulada, com 3244 entradas, é utilizada para agrupamento de textos.

2. Separação da base de treinamento, validação e teste para cada ODS. A base sem rótulos (42115 entradas) é adicionada à 60% da base etiquetada pós *upsample* para o ODS 3 (1947 entradas) e será utilizada na etapa de treinamento. São, ao todo, 44062 entradas na base de treinamento para clusterização, número que muda de ODS para ODS a depender do *upsample* necessário. 20% da base etiquetada pós *upsample* para o ODS 3 (649 entradas) será utilizada para etapa de validação e os outros 20% (649 entradas) será utilizado para etapa de teste.

3. Agrupamento de textos. A Tabela 5.2 mostra a quantidade de processos por *cluster* e o total de processos etiquetados por grupo, no ajuste realizado para 5 *clusters* no ODS 3.

Tabela 5.2: Agrupamento em 5 *clusters* para o ODS 3

Cluster	Tamanho do *Cluster*	Quantidade de Etiquetados
Cluster 1	12950	255
Cluster 2	8949	408
Cluster 3	7328	757
Cluster 4	10126	323
Cluster 5	4709	204

É possível perceber que os grupos estão bem divididos, não havendo *clusters* com poucas entradas e nem grupos muito grandes em relação aos demais. Também é possível verificar que a quantidade de processos etiquetados é bem distribuída entre os *clusters*.

4. Escolha dos parâmetros ótimos na validação. Em seguida, a etapa de validação avalia (através das métricas acurácia e sensibilidade), em 20% da base etiquetada pós *upsample*, todas as combinações de limiar de classificação (50%, 60% e 70%) e raio do centro dos *clusters* (5%, 10%, 25% e 100%). A Tabela 5.3 mostra os resultados reais para a etapa de validação no ODS 3 com 5 *clusters*.

Tabela 5.3: Etapa de validação para o ODS 3 em ajuste com 5 *clusters*

	Limiar de 50%	Limiar de 60%	Limiar de 70%
Raio de 5% (32 processos)	0.55	0.58	0.58
Raio de 10% (97 processos)	0.62	0.72	0.56
Raio de 25% (234 processos)	0.65	0.72	0.68
Raio de 100% (649 processos)	0.69	0.69	0.69

O raio de 100% usa todos os processos separados para validação (649 entradas), enquanto que o raio de 5% usa apenas 32 processos para o ODS 3. É possível notar que, para 5 *clusters*, a melhor combinação de raio e limiar de classificação é 25% de raio e 60% de limiar. Na presença de métricas parecidas, como nos raios de 10% e 25% neste ODS, a escolha é pelo raio com maior número de processos.

5. Etapa de teste. Utilizando os parâmetros escolhidos na validação para a etapa de teste, realizada também em 20% da base etiquetada pós *upsample* (649 processos), foram observados

os resultados de 70% de acurácia e 71% de sensibilidade, métricas que se mostram boas e validam os resultados obtidos na etapa anterior.

Os passos de 1 a 5 serão repetidos para todos os tamanhos de *clusters* considerados, a saber, 5, 10, 25, 50 e 100. A Tabela 5.4 mostra a quantidade de processos por grupo para 10 *clusters*, bem como a quantidade de processos etiquetados.

Tabela 5.4: Agrupamento em 10 *clusters* para o ODS 3

Cluster	Tamanho do *Cluster*	Quantidade de Etiquetados
Cluster 1	4060	124
Cluster 2	4002	245
Cluster 3	3154	65
Cluster 4	5317	360
Cluster 5	2950	219
Cluster 6	6822	183
Cluster 7	4623	251
Cluster 8	4213	146
Cluster 9	3509	161
Cluster 10	5412	193

Pela Tabela 5.4 é possível verificar que os *clusters* estão bem distribuídos, bem como a quantidade de processos originalmente etiquetados por grupo. Não se observam valores extremamente pequenos e nem grupos com muitos processos a mais do que outros. Este comportamento, registrado aqui com exemplos de 5 e 10 *clusters* para o ODS 3, se repete para todos os números de *clusters* candidatos e todos os ODS avaliados no trabalho e isso mostra que o algoritmo de clusterização (*k-means*) é capaz de dividir bem os textos da base de dados balanceada com *upsample*. A Tabela 5.5 mostra os resultados da métrica acurácia na etapa de validação para 10 *clusters*.

Pela Tabela 5.5 pode-se ver que, a exemplo do que ocorre com 5 *clusters*, a melhor combinação de raio e limiar é dada por 25% de raio e limiar de 60%. No conjunto de teste, as métricas de acurácia e sensibilidade foram, respectivamente, 0.72 e 0.74. Como a acurácia se mostra maior com 10 *clusters*, o ajuste feito com 5 grupos é eliminado. Repetindo este procedimento para

Tabela 5.5: Validação para o ODS 3 em 10 *clusters*

	Limiar de 50%	Limiar de 60%	Limiar de 70%
Raio de 5% (32 processos)	0.52	0.56	0.57
Raio de 10% (97 processos)	0.52	0.68	0.57
Raio de 25% (234 processos)	0.65	0.74	0.70
Raio de 100% (649 processos)	0.70	0.69	0.70

todos os tamanhos de *cluster* no ODS 3, observa-se que a melhor combinação de parâmetros é dada por 25 *clusters*, limiar de 60% e raio de 10% do centro, onde se verifica acurácia de 0.77 e sensibilidade de 0.78 na etapa de teste.

Os resultados finais das etapas de validação e teste na clusterização para todos os ODS avaliados no trabalho estão nas Tabelas 5.6 e 5.7. A Tabela 5.6 mostra apenas os resultados (acurácia e sensibilidade) das melhores configurações de números de *clusters*, raio e limiar de classificação obtidas na etapa de validação, enquanto que a Tabela 5.7 mostra os resultados obtidos na etapa de teste para a mesma configuração de parâmetros apresentada na tabela anterior. Apenas um ODS (17 - Parcerias e Meios de Implementação) tem métricas abaixo de 70%. O raio de 10% do centro se mantém como melhor em sete de nove ODS e o número ótimo de *clusters* se mantém em 25 para quase todos os ODS, com exceção do 17.

Tabela 5.6: Etapa de validação da clusterização para todos os ODS

ODS	*Clusters*	Distância	Limiar	Acurácia	Sensibilidade
ODS 3	25	10%	60%	0.78	0.78
ODS 4	25	10%	60%	0.79	0.75
ODS 8	25	10%	70%	0.78	0.74
ODS 9	25	10%	70%	0.66	0.67
ODS 10	25	10%	70%	0.81	0.82
ODS 11	25	10%	70%	0.70	0.70
ODS 15	25	10%	70%	0.81	0.78
ODS 16	25	25%	60%	0.79	0.82
ODS 17	50	25%	60%	0.65	0.62

Tabela 5.7: Etapa de teste da clusterização para todos os ODS

ODS	Acurácia	Sensibilidade
ODS 3	0.77	0.77
ODS 4	0.75	0.73
ODS 8	0.75	0.72
ODS 9	0.73	0.78
ODS 10	0.81	0.76
ODS 11	0.72	0.73
ODS 15	0.74	0.71
ODS 16	0.79	0.80
ODS 17	0.65	0.66

Apenas o ODS 17 se mantém abaixo de 70% nas métricas de acurácia e sensibilidade na etapa de testes da clusterização. Trata-se de um ODS genérico, que pode fazer mais sentido em contextos não jurídicos e até mesmo em outros países, dado que a Agenda 2030 da ONU é global e não se limita em Brasil. Este ODS permanece como ponto de interesse, em especial para a etapa de classificação.

Após os procedimentos de *upsample* e aumento da base de dados baseado em propagação de etiquetas via clusterização, a nova distribuição dos rótulos pode ser vista na Tabela 5.8, onde é possível notar que todos os ODS aumentaram de forma significativa o número de exemplos positivos, ou seja, com etiqueta 1. Isso é bastante interessante para aplicações de aprendizagem de máquina e substitui ferramentas de anotação rápida baseadas em Doccano, por exemplo. Os ODS 9 e 11 já mencionados, passaram a ter mais de 500 exemplos de etiquetas 1, quando originalmente tinham menos de 100. Trata-se de um aumento bastante relevante, pois, no geral, as base dobraram de tamanho. A exceção é o ODS 16, cujo aumento maior ocorre em razão do número original de etiquetas 1 ser sensivelmente maior do que de etiquetas 0. Com as bases aumentadas, é possível tratar da etapa de classificação, objeto da seção seguinte.

Tabela 5.8: Distribuição das etiquetas pós aumento de dados

ODS	Etiquetas 0	Etiquetas 1	Total
ODS 3	3590	590	4180
ODS 4	3908	509	4417
ODS 8	3453	654	4107
ODS 9	3964	548	4512
ODS 10	3604	642	4246
ODS 11	3953	535	4488
ODS 15	3934	529	4463
ODS 16	3957	6438	10395
ODS 17	3692	663	4355

5.3 Resultados - Classificação

O objetivo principal da clusterização é promover *data augmentation* para melhorar a classificação pelas redes neurais, *CatBoost* e similaridade, mas é verdade que a clusterização pode ser utilizada para classificação por si só. Os resultados da seção anterior, entretanto, mostram que somente processos próximos ao centróide podem ser classificados com acurácia ou sensibilidade razoável e que é preciso ajustar cuidadosamente, de acordo com cada *dataset* em estudo, os parâmetros utilizados pela clusterização. Apesar do claro potencial de aplicação, estudos mais profundos devem ser feitos para utilizar métodos de clusterização como classificadores.

Os primeiros testes da etapa de classificação foram realizados com redes LSTM ajustadas para as bases original e aumentada, através de uma estratégia de *bootstrap* (James et al., 2013) com 10 mil repetições, reamostrando com repetição 80% das bases para treino e usando os dados que não aparecem nas amostras de treino para testes. A comparação entre a performance das redes treinadas com a base original e com a base aumentada serve para verificar se o aumento da base melhora a capacidade preditiva do modelo atualmente utilizado pelo STF. Resultados da acurácia e sensibilidade médias estão descritos na Tabela 5.9.

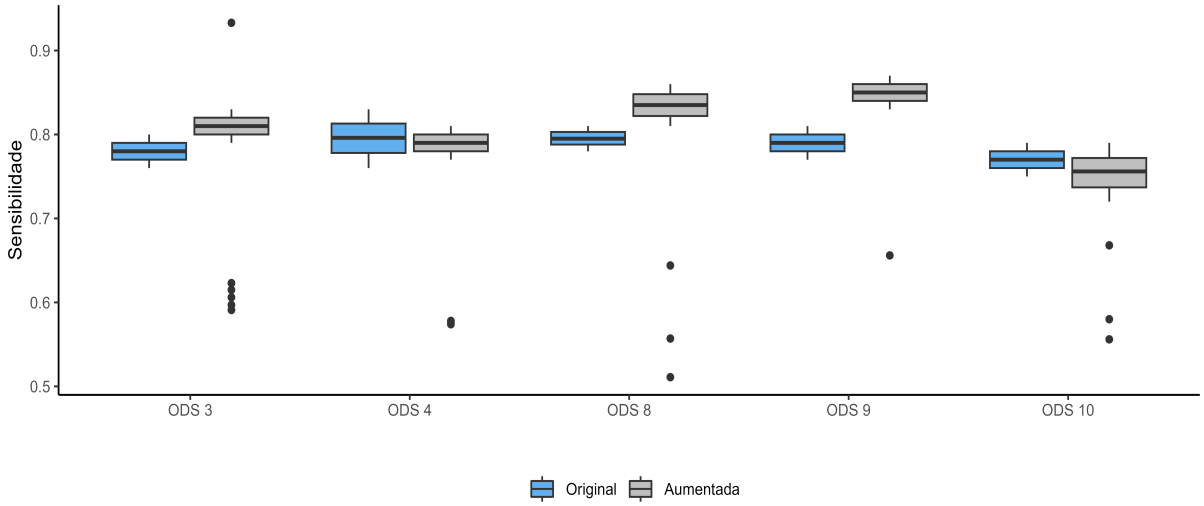
Tabela 5.9: Performance das redes neurais LSTM ajustadas para as bases original e aumentada

ODS	Base Original		Base Aumentada	
	Acurácia	Sensibilidade	Acurácia	Sensibilidade
ODS 3	0.86	0.78	0.91	0.80
ODS 4	0.78	0.79	0.84	0.79
ODS 8	0.87	0.79	0.88	0.83
ODS 9	0.81	0.79	0.92	0.85
ODS 10	0.84	0.76	0.84	0.75
ODS 11	0.75	0.74	0.84	0.80
ODS 15	0.71	0.69	0.83	0.83
ODS 16	0.88	0.80	0.92	0.83
ODS 17	0.70	0.74	0.74	0.75

É possível notar, pela Tabela 5.9, que todos os ODS apresentaram melhora de acurácia com os dados aumentados, com especial registro para o ODS 15 - Vida Terrestre, com 17% de aumento nesta métrica. Apenas o ODS 10 apresentou piora na métrica de sensibilidade. Todas as acurácias médias estão acima de 70%, algo que só não ocorre com a métrica de sensiblidade média em razão do ODS 15.

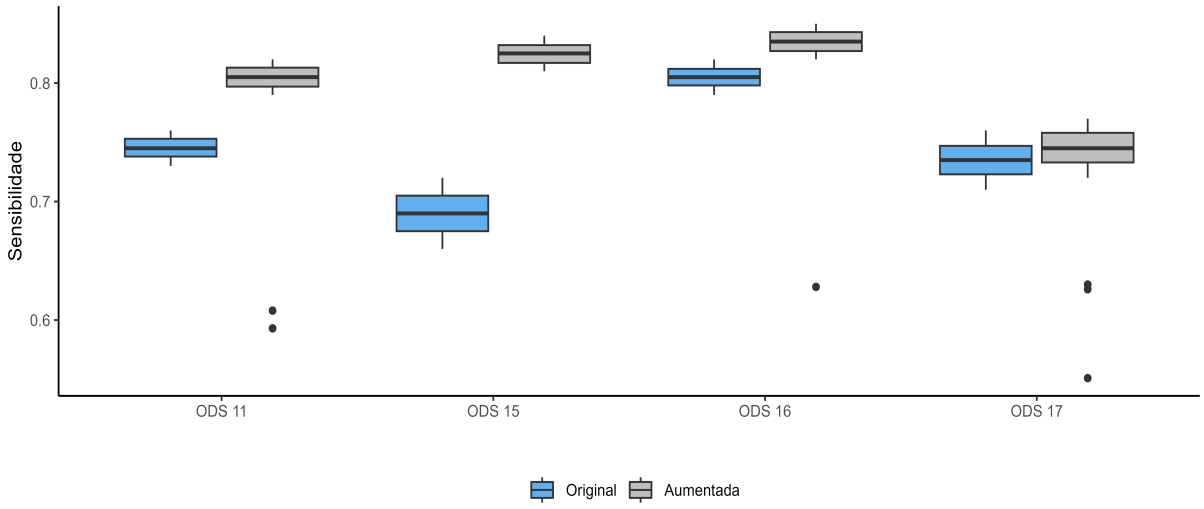
As Figuras 5.1 e 5.2 mostram os *boxplots* obtidos para medidas de sensibilidade observadas nas 10 mil iterações do método *bootstrap*, com as bases original e aumentada. A métrica de sensibilidade é especialmente importante, uma vez que um dos objetivos deste trabalho é aumentar a quantidade de textos relacionados aos ODS que possuem poucos exemplos naturalmente. Quase todas as medianas estão acima de 70% (exceto pelo ODS 15), e, além disso, nota-se que os *boxplots* para a base aumentada estão posicionados acima dos relacionados à base original. Testes-t para comparação de médias indicam significativa melhora, ao nível de 5%, entre as medidas de acurácia e sensibilidade obtidas com as bases original e aumentada para todos os ODS, com exceção dos 4 e 10 (cujas sensibilidades médias foram ligeiramente menores na base aumentada). Este resultado, junto da análise visual e dos valores médios observados, justificam a escolha da base aumentada para ajustes da rede neural, em detrimento da base original. Tais testes foram realizados com o pacote *SciPy* e seu módulo *stats*.

Figura 5.1: *Boxplot* para sensibilidades obtidas via *bootstrap* para os ODS 3, 4, 8, 9 e 10 para as redes neurais ajustadas com as bases original e aumentada



Fonte: Próprio autor

Figura 5.2: *Boxplot* para sensibilidades obtidas via *bootstrap* para os ODS 11, 15, 16, e 17 para as redes neurais ajustadas com as bases original e aumentada



Fonte: Próprio autor

Apesar das redes neurais possuírem boa performance, principalmente com uso da base aumentada, os motivos expostos na Seção 4.4 indicam que a performance combinada de vários modelos especialistas pode ser melhor que o ajuste de um único modelo generalista. Por este motivo, modelos diferentes das redes LSTM foram testados *CatBoost* e similaridade por cosseno) no presente trabalho.

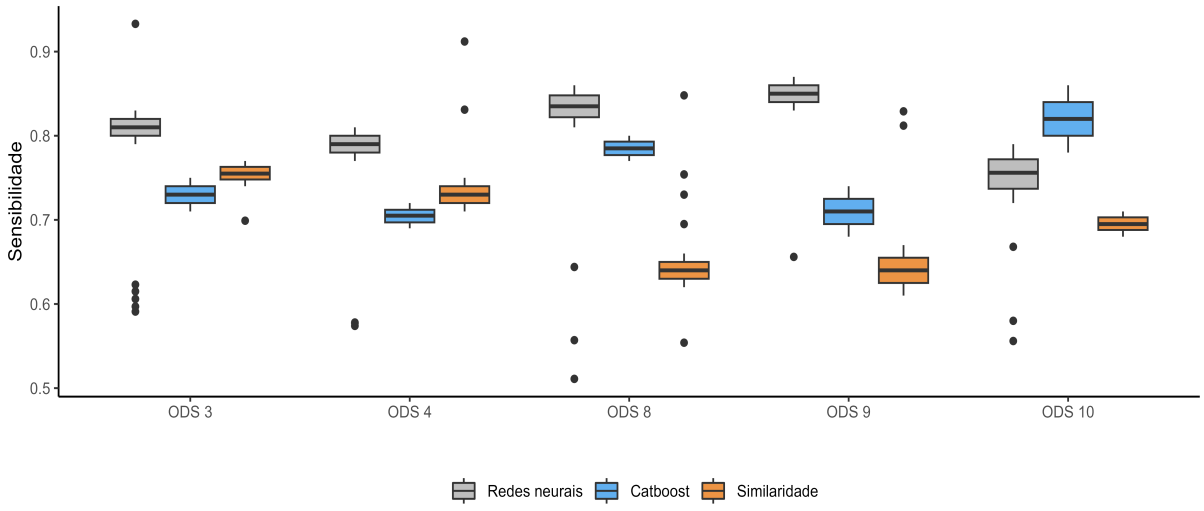
Os resultados do algoritmo *CatBoost* foram obtidos com os metadados classe processual, ramo do direito e contagem de palavras-chave de cada ODS. Os algoritmos *CatBoost* e a similaridade por cosseno foram ajustados nas mesmas bases de treinamento e teste obtidas via *bootstrap* para os modelos de redes neurais avaliados na base aumentada via clusterização. Uma das ideias do trabalho é avaliar métodos alternativos às redes neurais, com vistas a capturar aspectos diferentes de cada peça processual, tais como metadados associados (*CatBoost*) e uso simples de *embeddings* (similaridade). Os resultados médios obtidos para acurácia e sensibilidade, em comparação com os resultados das redes LSTM ajustadas na base aumentada com a estratégia de *bootstrap*, estão apresentados na Tabela 5.10.

Tabela 5.10: Performance das redes neurais ajustadas com a base aumentada, *CatBoost* e similaridade

ODS	Redes LSTM		*CatBoost*		Similaridade - Cosseno	
	Acurácia	Sensibilidade	Acurácia	Sensibilidade	Acurácia	Sensibilidade
ODS 3	0.91	0.80	0.78	0.72	0.90	0.76
ODS 4	0.84	0.79	0.74	0.71	0.96	0.75
ODS 8	0.88	0.83	0.79	0.78	0.84	0.66
ODS 9	0.92	0.85	0.69	0.69	0.98	0.63
ODS 10	0.84	0.75	0.77	0.79	0.86	0.69
ODS 11	0.84	0.80	0.73	0.76	0.96	0.67
ODS 15	0.83	0.83	0.72	0.72	0.97	0.77
ODS 16	0.92	0.83	0.79	0.84	0.81	0.85
ODS 17	0.74	0.75	0.75	0.73	0.89	0.54

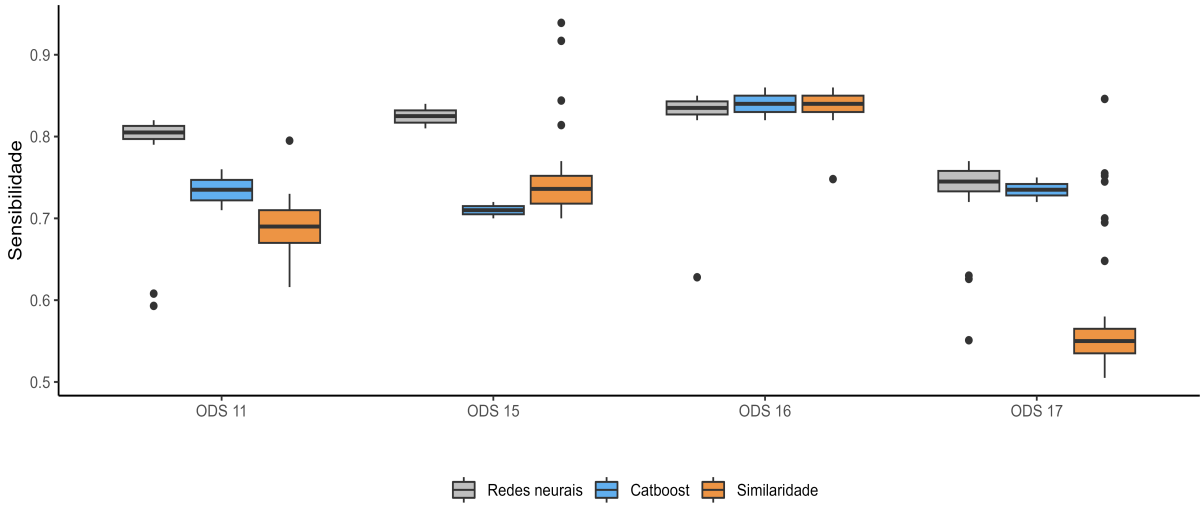
Pode-se verificar, pela Tabela 5.10, que as redes neurais possuem melhor performance em relação aos outros dois métodos. A similaridade por cosseno possui boas medidas de acurácia, mas baixas sensibilidades. As métricas de acurácia e sensibilidade do algoritmo *CatBoost* não ultrapassam 80% (com exceção do ODS 16), mas há uso de metadados associados às peças e menor tempo computacional em relação às redes neurais. As Figuras 5.3 e 5.4 mostram *boxplots* obtidos para as medidas de sensibilidade entre os modelos de redes neurais (ajustadas com a base aumentada), *CatBoost* e similaridade por cosseno. A abordagem de similaridade possui acurácia média acima dos demais modelos, mas sensibilidades bem abaixo das redes neurais e *CatBoost*, com maioria de valores menores do que 70%. Para o ODS 17, por exemplo, a métrica de sensibilidade média em 10 mil iterações de *boostrap* fica abaixo de 55%, indicando uma performance ruim. No geral, as redes neurais ajustadas com as bases aumentadas apresentam melhor performance individual em relação aos modelos *CatBoost* e de similaridade por cosseno.

Figura 5.3: *Boxplot* para sensibilidades obtidas via *bootstrap* para os ODS 3, 4, 8, 9 e 10 para as redes neurais, algoritmo *CatBoost* e similaridade ajustados com a base aumentada



Fonte: Próprio autor

Figura 5.4: *Boxplot* para sensibilidades obtidas via *bootstrap* para os ODS 11, 15, 16, e 17 para as redes neurais, algoritmo *CatBoost* e similaridade ajustados com a base aumentada



Fonte: Próprio autor

Como cada um dos modelos é bom em capturar aspectos específicos dos textos e os resultados, na fase de testes, foram obtidos com a mesma base, é possível usar um *ensemble* envolvendo as redes neurais, os modelos *CatBoost* e a similaridade por cosseno. A estratégia de votação é maioria simples, ou seja, vale o rótulo que mais aparece entre as predições dos três métodos. Os resultados de acurácia e sensibilidade médias podem ser vistos na Tabela 5.11.

Tabela 5.11: Performance do *ensemble* formado pelos modelos de redes neurais, *CatBoost* e similaridade.

ODS	Acurácia	Sensibilidade
ODS 3	0.92	0.83
ODS 4	0.93	0.79
ODS 8	0.84	0.81
ODS 9	0.84	0.82
ODS 10	0.88	0.76
ODS 11	0.89	0.82
ODS 15	0.87	0.82
ODS 16	0.92	0.85
ODS 17	0.91	0.79

O método de *ensemble* melhora a performance para todos os ODS, em relação aos modelos atualmente em produção - redes neurais LSTM. Um aspecto interessante da reunião de modelos é o equilíbrio entre as medidas de acurácia e sensibilidade, sendo que todos os ODS possuem medidas de acurácia acima de 85%. A maioria dos ODS apresenta aumento de sensibilidade com a abordagem *ensemble*, em relação ao melhor modelo individual (rede neural). Reduzir a diferença entre as medidas de acurácia e sensibilidade, algo que também ocorre na reunião de modelos, é algo importante para fluxos de classificação textual.

Os modelos de *ensemble* ganharam destaque por apresentarem bons resultados em competições sobre *data science*, como as realizadas na plataforma *Kaggle*. O presente trabalho utiliza a estratégia de *ensemble* para combinar métodos centrados em metadados (*CatBoost*) com métodos focados em texto e entradas antigas de exemplo (redes neurais e similaridade). A ideia é replicar o nível de informação e consultas que os analistas do tribunal podem fazer antes de classificar um processo recentemente autuado. O trabalho segue com a apresentação das conclusões e registros para trabalhos futuros.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

O presente trabalho propõe uma abordagem de *data augmentation* baseada em agrupamento de textos e propagação de rótulos, com aplicação imediata em contextos jurídicos. A ideia principal é melhorar fluxos de classificação de textos utilizando bases aumentadas via clusterização de documentos não rotulados inicialmente. Tal abordagem, apesar do custo computacional, se mostra melhor do que estratégias de anotação rápida, baseadas em ferramentas como Doccano e Brat (*Browser-Based Rapid Annotation Tool*). Este tipo de ferramente envolve análise humana, o que pode causar sobreposição de trabalho para os funcionários que, no poder judiciário, atuam em setores de autuação e classificação de processos.

Utilizando um conjunto de dados rotulados em ODS da Agenda 2030 por servidores do Supremo Tribunal Federal (STF) e textos de peças jurídicas sem rótulo foi possível estabelecer uma estratégia satisfatória para aumento da base originalmente rotulada, com vistas a melhorar os fluxos de classificação processual, atividade que já ocorre no dia a dia da corte com apoio de aprendizagem de máquina e *deep learning* através das ferramentas Victor e RAFA 2030. Alguns ODS passaram de menos de 100 exemplos para cerca de 500 etiquetas, considerando

rótulos originais e sintéticos (ver Tabela 5.8). Isso é de grande utilidade para o balanceamento de classes e ajuste de modelos estatísticos e de aprendizagem de máquina.

No caso em tela, a estratégia de aumento de dados, por si só, melhorou a performance dos modelos utilizados atualmente na classificação processual em todos os ODS avaliados (redes neurais LSTM). Além disso, também foi observada relevante melhora na atividade de classificação de peças com a associação da estratégia de aumento de dados e modelos *ensemble* com algoritmos *CatBoost* e de similaridade por cosseno. Tais resultados podem ser aplicados em outros contextos e com etiquetas de naturezas diversas, sendo um método de aplicação simples e intermediário custo computacional. Passos intermediários utilizados nesta dissertação também são interessantes para adoção em fluxos de NLP, com especial registro para a técnica *upsample* com sinônimos contextuais via BERT, a etapa de limpeza com passo de *pos tag* (*part of speech*) e para a ideia de fugir das bordas em tarefas de clusterização textual.

6.2 Trabalhos Futuros

Naturalmente, nem todos os modelos possíveis para agrupamento e classificação de textos foram explorados no presente trabalho, especialmente os baseados em atenção, como os da família *transformers* e os LLM (*Large Language Models*), tais como os da família GPT. A ideia de usar métodos simples de agrupamento (*k-means*) se baseia na importância de validar o método proposto antes de sofisticar as etapas de agrupamento, por exemplo, com os modelos mais sofisticados de NLP disponíveis atualmente. As principais atividades que podem ser desenvolvidas em trabalhos futuros envolvem novos modelos, novas métricas de distância, novas heurísticas para propagação de rótulos e melhorias de performance computacional. Exemplos de melhoria são dados por:

- Utilização de grafos (modelos gráficos) para propagação de rótulos na etapa de *data augmentation* e para tratamento de metadados processuais.
- Determinação dos *clusters* via *transformers* da família BERTopic ou via abordagem *WE-Clustering*.
- Utilização de *transformers* treinados em contextos legais (LegalBERT) para etapas de *embedding* e similaridade.
- Utilização de outras métricas para similaridade, tais como Mahalanobis e distância Euclidiana, disponíveis em pacotes da linguagem Python.
- Melhoramento da limpeza de textos com abordagens baseadas em frequência de palavras, para eliminar lixos oriundos do processo do OCR - transformação de imagens em textos.
- Ajuste de modelos de clusterização em linguagem C, para melhorar a performance desta etapa em especial.
- Utilização de outra estrutura de árvore nos modelos *CatBoost*.

Apêndice A

Frameworks para Processamento de Linguagem Natural

Os principais *frameworks* para processamento de linguagem natural são desenvolvidos em linguagem Python, mas existem bons pacotes em linguagem `R`, centrados em tarefas de menor complexidade computacional. As tarefas básicas de NLP - limpeza de textos, *pos tag*, classificação, Q&A (*Question and Answer*), *chatbots* e agrupamento de documentos - normalmente são desenvolvidas com apoio de pacotes de *deep learning*. O apêndice em tela traz uma lista com pacotes e *frameworks* de NLP, com breves explicações sobre o funcionamento dos principais.

Spacy: Biblioteca mais utilizada em tarefas de NLP atualmente. É desenvolvido em linguagem Python (especialmente em Cython) e tem foco na construção de produtos reais e de larga escala envolvendo textos de modo geral. Tem implementações pré treinadas para *transformers* (BERT), vetores de palavras, *pos tag*, reconhecimento de entidades nomeadas (NER), tokenização em diversos idiomas e suporte nativo para construção de modelos em Pytorch e Tensorflow. Tem uma vasta documentação e bastante apoio da comunidade para usuários iniciantes. Foi largamente utilizado neste trabalho para a tarefa de *pos tag*.

HuggingFace: Maior referência técnica em modelos de NLP. Companhia norte americana que desenvolve produtos de aprendizagem de máquina e *deep learning*, é responsável pela curadoria de imensas bibliotecas e pacotes, especialmente de *transformers*. Os modelos gratuitos do repositório HuggingFace são compatíveis com Pytorch, Tensorflow e JAX (Python). Tem um *blog* muito ativo, onde informações sobre as últimas tendências em NLP são compartilhadas diariamente. Mantém cursos gratuitos sobre *transformers*, além de um canal do Discord para dúvidas e participação da comunidade. Os modelos BERT utilizados pelo presente trabalho (BERTimbau) foram retirados do repositório HuggingFace.

Gensim: Pacote voltado para construção de *embeddings*, vetores semânticos para representação de textos. É compatível com todos os sistemas operacionais que suportam Python e Numpy. Utiliza rotinas em linguagem C para processamentos de dados estáticos e em *streaming*. Se destaca pela implementação de modelos *word2vec* e *doc2vec*, utilizados no presente trabalho na etapa de vetorização dos textos. Acumula mais de 2600 citações acadêmicas atualmente.

Em linguagem `R` também é possível fazer NLP em bom nível. No presente trabalho, no entanto, os pacotes em `R` foram utilizados na etapa inicial de processamento de textos, dadas as já conhecidas facilidades de sintaxe para manipulação de dados em geral. Os principais pacotes utilizados em tarefas intermediárias de NLP foram:

Tidyverse: Coleção de pacotes com funcionalidades para *data science*. Todos os pacotes tidy possuem mesma gramática e filosofia, facilitando, assim, fluxos de ciência de dados e modelos de caráter geral. O núcleo do pacote Tidyverse é formado pelos pacotes ggplot2 (gráficos), dplyr (manipulação de dados), readr (leitura de dados tabulares), purrr (programação funcional) e stringr (manipulação de *strings*), entre outros.

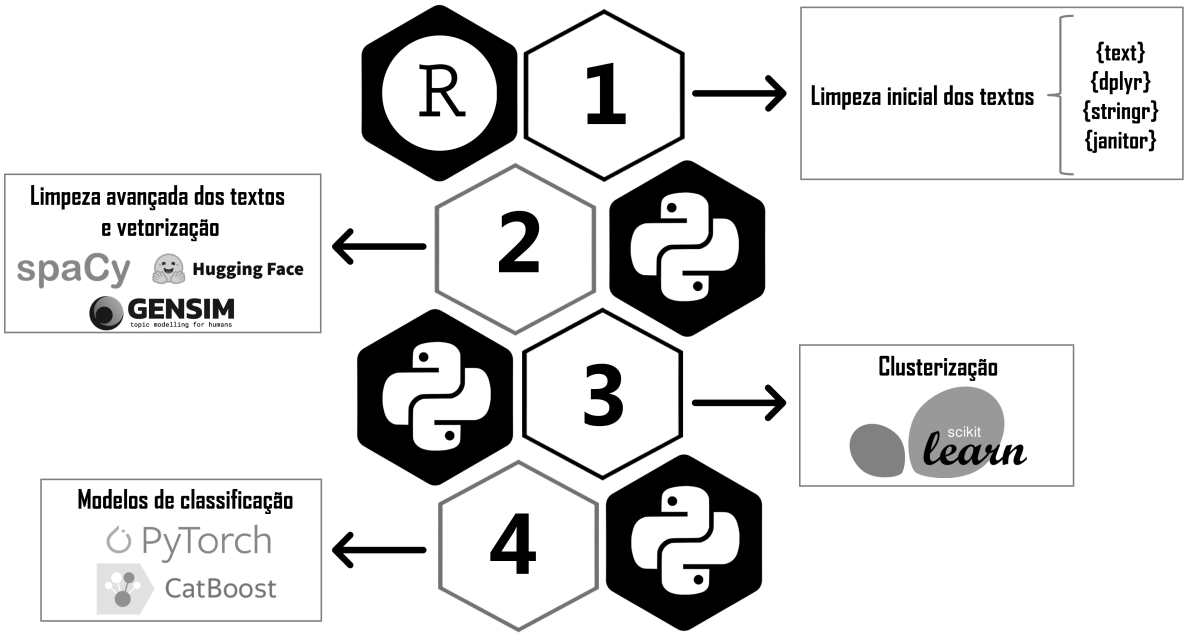
Text: Pacote que reúne implementações de *transformers* em linguagem R. Além desta família de modelos que podem ser utilizados em tarefas de vetorização, o pacote traz ferramentas gráficas e de análise textual, tornando-o uma biblioteca ponta a ponta.

Quanteda: Família de pacotes focada em manipulação geral de textos. Possui também ferramentas para visualização (quanteda.textplots), modelos estatísticos para NLP (quanteda.textmodels) e estatística descritivas (quanteda.textstats).

SpacyR: Implementação, em `R`, do pacote Spacy da linguagem Python. Facilita a interoperabilidade entre as linguagens, mantendo muitas das funcionalidades do pacote original.

Os modelos para classificação dos textos foram construídos com apoio das bibliotecas Pytorch e *Catboos*t, ambos em linguagem Python. Também desta linguagem, houve utilização do pacote Scikit-learn, especialmente em tarefas de clusterização. A Figura A.1 mostra o fluxo de utilização de cada pacote no trabalho.

Figura A.1: *Fluxo de pacotes*



Fonte: Compilação do autor [1]

Apêndice B

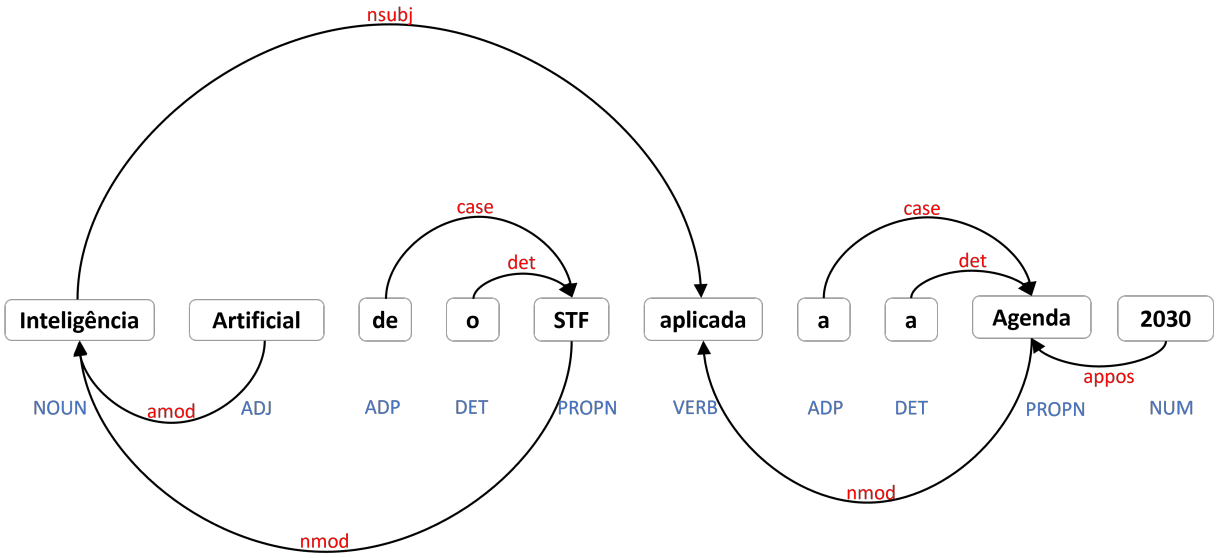
Tarefa *Part of Speech - Pos Tag*

A atividade de pos tag (*part of speech*) começou a ser feita quando o principal *corpus* de língua inglesa da época (*Brown corpus* - década de 60) começou a ser etiquetado, inicialmente por humanos, e depois por programas de computador. Trata-se de marcar palavras em um texto, como partes específicas do discurso. Mais especificamente, é processo de etiquetar cada palavra de uma frase, sentença ou texto com sua classe sintática (morfossintática). Em língua portuguesa, destacam-se os etiquetadores baseados no *corpus* Tycho Brahe de Português Histórico, tais como Finger, 2000 e Santos, Milidiú e Rentería, 2008, que usam regras de associação entre palavras determinantes e cadeias de Markov. O *corpus* Mac-Morpho, desenvolvido no âmbito do projeto Lacio-Web Project foi um marco da atividade de *pos tag* em português (Aluísio et al., 2003). Em 2013, Fonseca e Rosa, 2013 propuseram uma nova versão do *corpus* anotado Mac-Morpho usando redes neurais *perceptron* multicamadas. As demais modificações no *corpus* Mac-Morpho foram feitas com apoio de features categóricas associadas à *embeddings* de palavras.

As classes de palavras para atividades *pos tag* seguem um padrão, estabelecido por uma convenção seguida internacionalmente. As principais classes são ADJ (adjetivo), ADP (adpo-

sição), ADV (advérbio), CONJ (conjunção), INTJ (interjeição), NOUN (nome), NUM (numeral), PROPN (pronome), PUNCT (pontuação), PROPN (nome próprio) e VERB (verbo). O *framework* Spacy tem um bom módulo de *pos tag* em português do Brasil, que pode ser utilizado para atividades de extração de palavras-chave (*keywords*) e limpeza de textos, como no caso do presente trabalho. Em linguagem `R`, o pacote UDPipe, que é agnóstico em relação à linguagem e pode ser utilizado para *pos tag*, lematização, tokenização e outras tarefas de NLP, é uma excelente opção. Trata-se de um pacote com fácil instalação e dependências simples. A Figura B.1 mostra um exemplo de atividade de *pos tag*.

Figura B.1: *Exemplo - Pos Tag*



Fonte: Compilação do autor [1]

O ajuste *pos tag* feito na frase "Inteligência artificial do STF aplicada à Agenda 2030" mostra corretamente a relação entre as palavras do texto. É possível aplicar tal metodologia para escolher as palavras "inteligência", "STF" e "agenda" como *keywords*, uma vez que possuem maior número de conexões com outras partes do texto. Análises similares são possíveis em textos maiores e isso ajuda no resumo e sumarização de peças jurídicos, por exemplo. Em fluxos de limpeza de textos, a frase "Inteligência artificial do STF aplicada à Agenda 2030" pode ser

substituída pela *string* "inteligência STF agenda", diminuindo-se, assim, a complexidade dos textos e o custo computacional das tarefas de classificação.

Referências Bibliográficas

Aggarwal, Charu C. e Reddy, Chandan K. (2014). *Data Clustering: Algorithms and Applications*. CRC Press. ISBN: 978-1-46-655821-2. URL: `http://www.charuaggarwal.net/clusterbook.pdf`.

Aljarah, I., Faris, H. e Mirjalili, S. (2020). *Evolutionary Machine Learning Techniques: Algorithms and Applications*. Springer. ISBN: 978-9813299894.

Aluísio, S. et al. (2003). "An account of the challenge of tagging a reference corpus for brazilian portuguese." *Proceedings of the 6th International Conference on Computational Processing of the Portuguese Language. PROPOR 2003.* 1.1, pp. 141–154.

Amilar, A. D. M. (2018). "Agrupamento Automático de Documentos Jurídicos com uso de Inteligência Artificial". Diss. de mestr. Brasília: Instituto Brasiliense de Direito Público, Escola de Administração de Brasília, Mestrado Profissional em Administração Pública.

Bengio, Y. et al. (2003). "A Neural Probabilistic Language Model". *Journal of Machine Learning Research* 2.3, pp. 1137–1155.

Finger, M. (2000). "Técnicas de otimização da precisão empregadas no etiquetador tycho brahe". *Proceedings of PROPOR* 1.1, pp. 141–154.

Fonseca, E. R e Rosa, J. L. G. (2013). "Mac-morpho revisited: Towards robust part-of-speech tagging." *Proceedings of The 9th Brazilian Sympsoium in Information and Human Language Technology* 1.1.

Geron, Aurelien (2021). *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media. ISBN: 978-1491962299.

Gu, Quanquan e Han, Jiawei (2013). "Clustered Support Vector Machines". *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. Ed. por Carlos M. Carvalho e Pradeep Ravikumar. Vol. 31. Proceedings of Machine Learning Research. Scottsdale, Arizona, USA: PMLR, pp. 307–315.

Hausladen, C. I., Schubert, M. H. e Elliot, A. (2020). "Text classification of ideological direction in judicial opinions". *International Review of Law and Economics* 62.

Hochreiter, Sepp e Schmidhuber, Jürgen (1997). "Long Short-Term Memory". *Neural Computation* 9.8, pp. 1735–1780.

Hvitfeldt, E. e Silge, J. (2021). *Supervised Machine Learning for Text Analysis in R*. A Chapman & Hall book. CRC Press. ISBN: 9780367554194. URL: https://books.google.com.br/books?id=WQ1qzgEACAAJ.

James, Gareth et al. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

Junior, A. P. C., Calixto, W. P. e Castro, C. H. A. (2020). "Aplicação da inteligência artificial na identificação de conexões pelo fato e tese nas petições iniciais e integração com o sistema de processo eletrônico". *Revista Eletrônica do CNJ* 4.

Katz, D. M., Bommarito, M. J. e Blackman, J. (2017). "A general approach for predicting the behavior of the Supreme Court of the United States". *PLoS ONE* 12, pp. 485–498.

Lantz, Brett (2013). *Machine Learning with R*. Packt Publishing. ISBN: 1782162143.

McCarthy, John e Feigenbaum, Edward A. (1990). "In Memoriam: Arthur Samuel - Pioneer in Machine Learning." *AI Magazine* 11.3, pp. 10–11. URL: http://dblp.uni-trier.de/db/journals/aim/aim11.html#McCarthyF90.

Medvedeva, M., Vols, M. e Wieling, M. (2020). "Using machine learning to predict decisions of the European Court of Human Rights". *Artificial Intelligence and Law* 28, pp. 236–267.

Menezes, E. J. N. e Clementino, M. B. M. (2022). "Using deep learning to predict outcomes of legal appeals better than humans experts: A study with data from brazilian federal courts". *PLoS ONE* 17.

Mikolov, Tomas et al. (2010). "Recurrent neural network based language model". *INTERSPEECH*. Ed. por Takao Kobayashi, Keikichi Hirose e Satoshi Nakamura. ISCA, pp. 1045–1048.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". *Advances in Neural Information Processing Systems*. Ed. por C.J. Burges et al. Vol. 26. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Müller, A.C. e Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media. ISBN: 9781449369897. URL: https://books.google.com.br/books?id=vbQlDQAAQBAJ.

Nascimento, E. G. S. e Oliveira, R. S. (2022). *Data Clustering*. Intechopen.

Nunes, I. O., Carvalho, D. B. F. e Lucena, C. J. P. (2009). "Estudo sobre algoritmos de classificação para o referenciamento de gestantes de alto risco". *Monografias em Ciência da Computação* 38, pp. 1–11.

ONU (2022). *Objetivos de Desenvolvimento Sustentável*. https://brasil.un.org/pt-br/sdgs.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". Em: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Prokhorenkova, Liudmila Ostroumova et al. (2018). "CatBoost: unbiased boosting with categorical features." *NeurIPS*. Ed. por Samy Bengio et al., pp. 6639–6649. URL: http://dblp.uni-trier.de/db/conf/nips/nips2018.html#ProkhorenkovaGV18.

Radygin, V. Y. et al. (2021). “Application of text mining technologies in Russian language for solving the problems of primary financial monitoring”. *Procedia Computer Science* 190, pp. 678–683.

Rosenblatt, F. (1958). “The perceptron: A probabilistic model for information storage and organization in the brain.” *Psychological Review* 65.6, pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

Samuel, A. L. (1959). “Some Studies in Machine Learning Using the Game of Checkers”. *IBM Journal of Research and Development* 3.3, pp. 210–229. DOI: 10.1147/rd.33.0210.

Santos, C. N. dos, Milidiú, R. L. e Rentería, R. P. (2008). *Portuguese part-of-speech tagging using entropy guided transformation learning.* 1st. Springer.

Scitovski, R. et al. (2022). *Cluster Analysis and Applications*. Springer. ISBN: 978-3030745516.

Selvaraj, S. e Choi, E. (2021). “Swarm Intelligence Algorithms in Text Document Clustering with Various Benchmarks”. *Sensors* 21.

Silva, I. N., Spatti, D. H. e Flauzino, R. A. (2016). *Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas. Fundamentos Teóricos e Aspectos Práticos*. Artliber. ISBN: 978-8588098879.

Souza, Fábio, Nogueira, Rodrigo e Lotufo, Roberto (2020). “BERTimbau: pretrained BERT models for Brazilian Portuguese”. *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*.

STF (2020). *Agenda 2030 e o STF*. https://portal.stf.jus.br/hotsites/agenda-2030/.

Turing, A. M. (out. de 1950). “Computing Machinery and Intelligence”. *Mind* LIX.236, pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433.

Vaswani, Ashish et al. (2017). “Attention is all you need”. *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Weizenbaum, Joseph (jan. de 1966). “ELIZA a Computer Program for the Study of Natural Language Communication Between Man and Machine”. *Commun. ACM* 9.1, pp. 36–45.

ISSN: 0001-0782. DOI: 10.1145/365153.365168. URL: http://doi.acm.org/10.1145/365153.365168.

Zanuz, L. e Rigo, S. J. (2022). "Fostering Judiciary Applications with New Fine-Tuned Models for Legal Named Entity Recognition in Portuguese". *Computational Processing of the Portuguese Language* 1, pp. 219–229.