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

Análise e Estudo Comparativo de Modelos Supervisionados no processo de Identificação de Risco de Conluio em Publicações de Órgãos Públicos

Thiago Luis Rodrigues Pinho

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Flávio de Barros Vidal

Brasília
2021

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

Análise e Estudo Comparativo de Modelos Supervisionados no processo de Identificação de Risco de Conluio em Publicações de Órgãos Públicos

Thiago Luis Rodrigues Pinho

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Flávio de Barros Vidal (Orientador)
CIC/UnB

Prof.a Dr.a Aleteia Patricia Favacho de Araujo
CIC/UnB

Prof.a Dr.a Maristela Terto de Holanda
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenadora do Curso de Engenharia da Computação

Brasília, 05 de Novembro de 2021

Dedicatória

Eu dedico essa pesquisa à minha família, os que me acompanham ainda hoje e em memória de meu avô José Ribamar Santos Rodrigues, vitimado pela pandemia.

Agradecimentos

Agradeço à Universidade de Brasília por possibilitar, com todas as suas limitações orçamentárias, um espaço para estudos e pesquisa de qualidade em tecnologias de ponta.

Ao Professor Dr. Flávio de Barros Vidal que de forma pedagógica potencializou a pesquisa e deu suporte para as dificuldades que surgiram dela.

À comunidade de software livre que incentiva um ecossistema de aprendizagem e compartilhamento de conhecimento permitindo práticas de autodidatismo e estudos coletivos de forma ampla que podem ter impacto social.

Ao Sistema Único de Saúde (SUS) que permitiu que essa pesquisa fosse completada durante a maior pandemia que vivenciei; Protegendo, ainda que com atrasos, a saúde de minha família e a minha.

Resumo

Nos últimos anos houve uma demanda crescente no combate à corrupção por parte da população, sendo necessário o suporte de técnicas computacionais para melhorar a eficiência de atividades direcionadas para essa questão. Atualmente os avanços no campo processamento de linguagem natural permitiram ganhos significativos na área de combate à corrupção. O uso de informações públicas é um dos cernes de trabalho que visam combater a corrupção, esta na forma de conluio, de forma que a análise nestes documentos públicos podem permitir melhorias significativas em todo o processo. Assim, neste trabalho, foi realizado o treinamento e comparação de um conjunto de classificadores a partir de modelos esparsos e conexionistas para detecção e análise de risco de conluio em publicações oficiais de licitações. Para isso, foi concebido um conjunto de modelos de avaliação, estes utilizando métodos esparsos e conexionistas. Os resultados obtidos pelo trabalho avaliaram diferentes versões dos modelos treinados e avaliados em diferentes partições das amostras, bem como sua precisão em relação aos modelos avaliados.

Palavras-chave: Classificação de Texto, Análise de Fraudes, Análise de Conluio, Processamento de Linguagem Natural, Aprendizagem Profunda, Validação Cruzada, Análise Semântica

Abstract

In recent years, there has been a growing demand on the part of the population to fight corruption, requiring the help of computational techniques that help in this task. Currently, the focus on the use of natural language processing has allowed significant advances in combating corruption. The use of public information is one of the cores of work aimed at combating corruption, this in the form of collusion. The analysis of these public documents can allow significant improvements in the entire process. This work performed training and comparison of a set of classifiers using sparse and connectionist models to detect and analyze the collusion risk in official bidding publications. A set of evaluation models was designed, these using sparse and connectionist methods. The results obtained by the work evaluated different versions of the models trained and evaluated in different sample partitions and their accuracy concerning the evaluated models.

Keywords: Text Classification, Fraud Analysis, Collusion Analysis, Natural Language Processing, Deep Learning, Cross Validation, Semantic Analysis

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Siglas

FN Falsos Negativos

FP Falsos Positivos

$HTML$ Linguagem de Marcação de Hipertexto

NLP Processamento de Linguagem Natural

VN Verdadeiros Negativos

VP Verdadeiros Positivos

Variáveis

w_t Pesos escolhidos para uma dada interação t em um *SGD*.

Z_i Amostra selecionada aleatoriamente para treinar um *SGD*.

γ Uma constante de ganho escolhido escolhido para um *SGD*.

C Conjunto de centroides do modelo *K-means*

c_J Um centroide do Conjunto de centroides C *K-means*.

D Conjunto de amostras avaliadas *K-means*.

d Distância entre um centroide *K-means*.

D' Conjunto de amostras tratadas e representadas em um espaço vetorial para o modelo *K-means*.

k Amostra selecionada para avaliação para o modelo *K-means*.

n Número de amostras total*SGD*.

p_J Uma probabilidade de pertencimento de uma dada amostra pertencer a um dado centroide no modelo *K-means* .

S_k Amostra selecionada para o modelo *K-means*.

Capítulo 1

Introdução

1.1 Motivação

Nos últimos anos com a ocorrência de uma demanda crescente no combate à corrupção por parte da população, além do aumento do questionamento do uso prática da ciência produzida pela academia, tomou o debate público de segundo apresentado na pesquisa de Lima [1]. Além disso, avanços no campo do processamento de linguagem natural (*NLP*, sigla em inglês para *Natural Language Processing*) têm oferecido uma ampla gama de soluções, sejam em técnicas ou em metodologias, que permitem seu emprego em diversas áreas do conhecimento, incluindo o combate à corrupção generalizada.

O processamento de linguagem natural [2]e a análise de indícios de fraudes, em conjunto com a área de modelagem estatística, possibilitaram soluções algorítmicas promissoras para o auxílio de profissionais trabalhando na investigação de possíveis conluios em diferentes campos [3].

O combate a fraudes em licitações, principalmente em licitações de obras, apresenta-se como um grande desafio para a sociedade ao envolver grandes somas em dinheiro e alta complexidade técnica e burocrática. Além disso, o desvio de recursos, objetivo destas fraudes, é precedido de ações que visam direcionar ou manipular o resultado das licitações [4].

Com essas questões apresentadas, bem como pela importância dos trabalhos já desenvolvidos, mostra-se necessário uma avaliação comparativa de métodos e técnicas de processamento de linguagem natural nesse tipo de documentos, de forma a permitir o seu uso no combate a fraudes em licitações, entre outras análises que auxiliam no combate à corrupção. Bem como, apoiar e melhor fundamentar modelos futuros que acrescentem na elaboração de fluxos mais eficientes na realização de tal tarefa.

1.2 Definição do Problema de Pesquisa e Justificativa

O desafio de determinação de todos os tipos de fraudes em processos públicos é mais evidente na detecção de conluio. Por ser feita a análise manual e supervisionada, sob a ótica da análise de dados, além a utilização dos textos das publicações do Diário Oficial da União como forma de enfrentar este desafio, existe a possibilidade de que técnicas de inteligência artificial se consolidem para tal fim.

Sendo assim, a questão a ser abordada neste trabalho é o treinamento e a comparação de um conjunto de classificadores a partir de modelos esparsos e conexionistas para detecção e análise de risco de conluio em publicações oficiais de licitações. E estas publicações são provenientes do sítio do Diário Oficial da União (DOU) [1].

1.3 Objetivos

1.3.1 Objetivos Gerais

Nesta pesquisa serão avaliar métodos de classificação para para modelar e identificar padrões para análise de risco de conluio. Essa avaliação será feita mediante o estudo de técnicas de pré-processamento e vetorização de termos. Esses diferentes métodos serão então avaliados comparativamente a partir de um conjunto de classificadores supervisionados, com o escopo de estimar seu desempenho para publicações oficiais.

1.3.2 Objetivos Específicos

Busca-se com esta pesquisa a elaboração de fluxos de processamento de linguagem natural para alcançar os seguintes quesitos:

1. Implementar diferentes técnicas de pré-processamento e suas diferentes combinações e parâmetros;
2. Calcular diferentes métricas de desempenho de modelos estatísticos;
3. Desenvolver um conjunto de modelos esparsos e conexionistas supervisionados utilizados para classificação de textos automáticos;
4. Aplicar técnicas de treinamento e avaliação desses modelos;
5. Avaliar o desempenho dos modelos de classificação para diferentes configurações e parâmetros;

[1] Acessado pelo endereço: `https://www.in.gov.br/acesso-a-informacao/dados-abertos/base-de-dados`.

Ainda que os modelos de classificação sejam de diferentes categorias, as métricas e técnicas usadas têm um objetivo prático em comum, que é a busca de modelos estáveis e de bom desempenho para investigações em possíveis conluios.

1.4 Organização do Manuscrito

O trabalho é estruturado da seguinte forma: O Capítulo 2 apresenta a Fundamentação Teórica que é a base deste trabalho, Capítulo 3 descreve brevemente o projeto no qual está inserido esta pesquisa e aborda os trabalhos relacionados recentes na determinação de conluio e na classificação de textos usando processamento de linguagem natural (*NLP*). O Capítulo 4 apresenta a metodologia de desenvolvimento deste trabalho. O Capítulo 5 apresenta os resultados alcançados pela pesquisa. Por último, o Capítulo 6 apresenta as conclusões derivadas dos resultados e as recomendações para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais fundamentos e procedimentos teóricos necessários para o entendimento das técnicas utilizadas na realização deste trabalho. Faz-se necessário observar que as informações apresentadas são consolidadas de forma resumida, e sempre indicando a fonte para maiores informações nas referências apresentadas em cada seção que compõe o capítulo.

2.1 Processo de Classificação e Reconhecimento de Textos

Seguindo Goldberg [5], uma das áreas da Inteligência Artificial é o processamento de linguagem natural (*NLP*, sigla em inglês para *Natural Language Processing*) no qual são desenvolvidas técnicas e modelos que usam textos como entrada.

Algumas soluções desenvolvidas neste campo podem ser encontradas em nosso dia-a-dia, tais como tradutores automatizados, *chatbots*, filtros de *spam* ou corretores gramaticais. As principais áreas de *NLP* descritas por Goldberg [5] são:

- **Análise de Sentimentos** (*Sentiment Analysis*): usada para interpretar e classificar o tipo de emoção tem um texto, frase ou documento;
- **Modelos de Linguagem** (*Language Modeling*): predição da próxima palavra ou letra de um texto. Modelos dessa área podem ser usados para conversas automatizadas, geração de texto e outras aplicações;
- **Tradução** (*Machine Translation*): aplicações de tradução automática e flexível a diferenças semânticas entre línguas;
- **Classificação de Textos** (*Text Classification*): procedimentos e modelos dessa área serão usadas nesta pesquisa. Isso será descrito em detalhes na Seção 2.4;

- **Respostas a Perguntas** (*Question Answering*): normalmente com base em um texto de referência um modelo é treinado para responder perguntas;
- **Outros**: reconhecimento de entidades nomeada, sumarização, extração de relacionamentos e aplicações diversas;

2.2 Extração de Características em Textos

Nesta seção serão explicados os fundamentos teóricos das técnicas utilizadas para tratamento do conteúdo textual a fim de maximizar a informação extraída desses dados.

2.2.1 Tokenização

A tokenização é um processo que consiste na separação do texto em frases, palavras, símbolos ou outros agrupamentos de caracteres e esse átomo é então chamada de *token*. O objetivo principal desse passo é extrair palavras individuais dentro de uma sentença [6].

Normalmente o critério de separação de cada átomo são os espaços e, portanto, não são permitidos dentro dos *tokens*. Alguns métodos, no entanto, permitem o uso de *token* de mais de uma palavra, mais adequados a algumas linguagens, como o alemão [7]

2.2.2 Palavras de Parada (*Stop words*)

Os *tokens* não necessariamente trarão a mesma quantidade de informação para uma análise. Alguns termos são normalmente pouco informativos em uma sentença e podem ser retirados sem haver perda de informações. Também há termos que são pouco informativos dependendo do problema estudado.

Além de diminuir a quantidade de índices usados futuramente para alocação dos *tokens*, a remoção desses termos pode diminuir o risco de sobre-ajuste em modelos [8].

2.2.3 Capitalização de Palavras

Normalmente em um texto, há frases com letras com caixa alta e baixa. Múltiplas frases podem compor um documento. Para reduzir o problema de espaço alocado para uma mesma palavra que é representada com diversos *tokens*, é possível reduzir a caixa um único formato em todo o texto. Porém, esse tipo alteração pode mudar de significado em algumas palavras e siglas. Além de dificultar a diferenciação de substantivos próprios e comuns. Uma forma de abordar esse problema é identificar e extrair informações semânticas e sintáticas juntos de cada *token* [9].

2.2.4 Palavras Fora do Vocabulário

Gírias e abreviações podem ser um problema em etapas de pré-processamento. Uma abreviação é a forma encurtada de uma palavra que compartilha o mesmo significado que sua versão expandida. Assim, uma gíria é uma ou mais palavras que tem um significado diferente para certos interlocutores em dado contexto.

Uma forma de lidar com esse tipo de palavras é considerá-lós palavras de parada, descartando-as; Outra forma é converter para um correspondente não abreviado ou uma frase em termos mais diretos e abrangentes, em sua forma normalizada [10].

2.2.5 Remoção de Ruído

Outra técnica que pode ser utilizada na etapa de pré-processamento é a remoção de ruído. Documentos textuais podem conter caracteres como pontuação e símbolos especiais que são resquícios da extração do texto, tanto o meio em que o texto foi gerado, como redes sociais, quanto processos de identificação ótica de caracteres (*OCR*, que vem do inglês *Optical Character Recognition*). Esses símbolos podem confundir os modelos de classificação, afetar o entendimento de onde começa e termina uma sentença, e consequentemente, funções semânticas.

2.2.6 Correção Textual

Uma parte que pode ou não ser incluída na etapa de pré-processamento é corrigir erros ortográficos. Há diferentes técnicas que podem ser aplicadas, como correções usando tabelas pré-escritas ou técnicas sensíveis ao contexto que podem auxiliar na correção. Também é possível usar comparações de distâncias damerau-levenshtein [11] para estimar a correção devida para um erro.

2.2.7 Stemização

Na stemização busca-se o caule léxico de uma palavra independente do contexto ou a função sintática dessa em uma frase. O caule é a parte da palavra que não é alterada mesmo se a palavra mudar morfologicamente. Essa partícula resultante não precisa ser uma palavra existente, mas todas as palavras criadas a partir desse caule devem poder ser mapeadas de volta para essa mesma partícula. É recomendado que a stemização seja feita com palavras que compartilham um mesmo significado semântico, que diferentes palavras podem ser reduzidas para um mesmo caule ainda que não necessariamente tenham o mesmo significado. Essa técnica pode ter efeitos diferentes dependendo da complexidade e variação morfológica de palavras em uma linguagem [12].

2.2.8 Lematização

A lematização é semelhante à stemização. Esse é o processo linguístico onde variações morfológicas de palavras são mapeadas em uma forma de base [13], na qual pode ou não ter significado, enquanto na lematização a palavra é acompanhada de um sufixo, caso necessário, para que sempre tenha um significado. Geralmente, é convertida em um verbo no infinitivo [14], por exemplo, nos textos foi possível encontrar as palavras trabalhar, trabalharam, trabalho e trabalhos. Após a lematização, elas foram reduzidas para o verbo trabalhar. Assim, plurais, flexões verbais são perdidas, assim como distinção de gênero e variações linguísticas que poderiam confundir os modelos de vetorização.

2.3 Vetorização de Palavras

Diferentes procedimentos de incorporação de palavras podem ser usados para transformar o texto pré-processado em *tokens* em uma entrada possível para modelos de aprendizado de máquina. Nessa seção serão apresentados as técnicas que serão utilizadas nessa pesquisa.

2.3.1 Word2Vec

Word2vec é um grupo de modelos usados para gerar vetores de palavras. Esses modelos são constituídos de redes neurais, normalmente não muito profundas, treinadas para reconstruir contextos linguísticos de palavras.

Esses modelos recebem como entrada grandes conjuntos de palavras e produzem vetores espaciais, que podem ter diversas centenas de dimensões. Nesses vetores cada palavra diferente é representada com um vetor correspondendo nesse espaço. Vetores de palavras são posicionadas no espaço vetorial de forma que palavras que compartilham um contexto semelhante estejam próximas uma das outras [15].

2.3.2 Frequência e Ponderação de Termos

Uma forma de vetorizar um texto é contar o número de aparições que cada palavra tem, e valorar ocorrências atribuindo um peso em um vetor para dado documento.

Pacote-de-Palavras

Pacote de palavras (*BOW*. A sigla vem do inglês *Bag of Words*), é uma representação que converte um texto em um vetor de palavras relacionando uma coluna para cada palavra, e o pontua de acordo com a contagem de palavras naquele documento. Nessa forma de

vetorização, a ordem das palavras não é armazenada, e frases com ordens diferentes das mesmas palavras são atribuídas ao mesmo vetor.

Frequência Inversa de Termos

Outra forma de pontuar os vetores de palavras é utilizando o inverso de sua frequência por documentos. Essa forma tende a dar mais valor para palavras específicas ou de nicho, dependendo da amostra. Palavras mais raras recebem um peso mais alto enquanto palavras mais comuns tem menor [15].

2.4 Técnicas de Classificação de Texto

A Classificação de Textos é a área do campo de *NLP* na qual se elabora modelos para a atribuição de etiquetas ou categorias a sequências textuais ou documentos. Segundo Bramer [16], a classificação, sendo uma atividade frequente, na maioria das vezes envolve dividir elementos em categorias mutualmente exaustiva e excludentes, ou seja, cada objeto só tem uma categoria ou etiqueta. As classificações podem ser binárias, de duas classes, ou de múltiplas classes [16].

Muitos processos decisórios podem ser formulados como problemas de classificação de acordo com Lima [1]. Esta pesquisa se propõe a avaliar essa capacidade de detecção de padrões que podem revelar indícios de conluio em diferentes modelos de classificação.

2.4.1 K-vizinhos mais Próximos

De acordo com Manning et al. [17], *K-means*, K-vizinhos mais próximos ou também o algoritmo de Lloyd, em termos básicos, pode ser modelado em três etapas:

1. A primeira etapa escolhe os centroides iniciais (representado por C.), com o método mais básico sendo escolher amostras do conjunto de dados. Após a inicialização, o *K-means* consiste em fazer um laço entre as duas outras etapas. A primeira etapa atribui cada amostra ao seu centroide mais próximo;

2. A segunda etapa cria novos centroide, tomando o valor médio de todas as amostras atribuídas a cada centroide anterior. Sendo as amostras representada por D. .

3. A diferença entre o antigo e o novo centroide é calculada e o algoritmo repete essas duas últimas etapas até que esse valor seja menor que um limite. Em outras palavras, ele se repete até que os centroide não se movam significativamente.

Esse processo pode ser modelado de acordo com o pseudocódigo descrito nas equações2.1 e 2.2:

$$\begin{aligned} &\text{ETAPA-DOIS-KNN(C, D):} \\ &D' \leftarrow \text{PREPROCESS}(D) \\ &k \leftarrow \text{ SELECT-K } (\mathbf{C}, \mathbf{D'}) \\ &\text{return } D', k \end{aligned} \tag{2.1}$$

$$\begin{aligned} &\text{ETAPA-TRES-KNN(C, D', k, d):} \\ &S_k \leftarrow \text{ COMPUTE } (D', k, d) \\ &\text{for each } c_j \in C \\ &\text{do } p_j \leftarrow |S_k \cap c_j| / k \\ &\text{return } \arg\max_j p_j \end{aligned} \tag{2.2}$$

Usa-se a distância de uma nova amostra em relação aos centroides para determinar a classificação dessa amostra.

2.4.2 Máquinas de Vetor de Suporte (*SVM*)

Segundo Manning et. al. [17], uma máquina de vetores de suporte (*SVM*, sigla em inglês para *Support Vector Machine*) constrói um hiperplano ou conjunto de hiperplanos em um espaço dimensional alto ou infinito, que pode ser usado para classificação, regressão ou outras tarefas. Intuitivamente, uma boa separação é alcançada pelo hiperplano que possui a maior distância aos pontos de dados de treinamento mais próximos de qualquer etiqueta, pois, em geral, quanto maior for a margem menor será o erro de generalização do classificador [18].

2.4.3 Classificador SVM com SGB

Gradiente Descendente Estocástico, (*SGD*, sigla em inglês para *Stochastic Gradient Descent*), é uma forma de otimização de modelos de classificação lineares como regressões logísticas e máquinas de vetor de suporte [19].

Um gradiente descendente é um método iterativo. A cada iteração ele atualiza os pesos w seguindo um gradiente $E_n(fw)$:

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(Z_i, w_t) \tag{2.3}$$

Onde γ é o ganho escolhido. Sob suposições de regularidade suficientes, quando a estimativa inicial w_0 está perto o suficiente do ótimo, e quando o ganho γ é suficientemente

pequeno, este algoritmo atinge convergência linear, ou seja, $-\log \rho \sim t$, onde ρ representa o resíduo erro. Um algoritmo de gradiente descendente estocástico em vez de computar o gradiente $E_n(fw)$ exatamente. Cada iteração estima o gradiente baseado em uma única amostra selecionada aleatoriamente. Para uma amostra Z_t, tem-se:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(Z_i, w_t) \tag{2.4}$$

Esse modelo é aplicado com maior eficácia a problemas de aprendizado de máquina esparsos e em grande escala, frequentemente encontrados na classificação de texto e no processamento de linguagem natural [20].

2.4.4 Árvores de Decisão

Uma árvore de decisão é um modelo de aprendizado de máquina supervisionado. Para encontrar uma solução, uma árvore de decisão utiliza decisões sequenciais e hierárquicas para determinar o resultado de saída. Os componentes de um modelo de uma árvore de decisão são os nós e suas ramificações. Além disso, para o treinamento do modelo existem as etapas de dividir, parar e podar [21].

Nós e Ramificações

Existem três tipos de nós. Um nó raiz, também chamado de nó de decisão, representa uma escolha que resultará na subdivisão de todos os registros em dois ou mais subconjuntos mutuamente exclusivos. Os nós internos, também chamados de nós aleatórios, representam uma das escolhas possíveis disponíveis naquele ponto da estrutura da árvore; a borda superior do nó é conectada a seu nó pai e a borda inferior é conectada a seus nós filhos ou nós folha. Nós folha, também chamados de nós finais, representam o resultado final de uma combinação de decisões ou eventos [21].

Ramificações representam resultados aleatórios ou ocorrências que emanam de nós raiz e nós internos. Um modelo de árvore de decisão é formado usando uma hierarquia de ramos. Cada caminho do nó raiz através dos nós internos para um nó folha representa uma regra de decisão de classificação. Esses caminhos da árvore de decisão também podem ser representados como regras "se-então". Por exemplo, “se a condição 1 e a condição 2 e a condição ... e a condição k ocorrerem, então o resultado j ocorre. ”

Divisão

Ao construir o modelo, deve-se primeiro identificar as variáveis de entrada mais importantes e, em seguida, dividir os registros no nó raiz e nos nós internos subsequentes em duas ou mais categorias ou ’caixas’ com base no status dessas variáveis. As características que

estão relacionadas ao grau de "pureza"dos nós filhos resultantes (isto é, a proporção com a condição alvo) são usadas para escolher entre diferentes variáveis de entrada potenciais; essas características incluem entropia, índice de Gini, erro de classificação, ganho de informação, razão de ganho e critério de dois.

Este procedimento de divisão continua até que a homogeneidade pré-determinada ou os critérios de parada sejam atendidos. Na maioria dos casos, nem todas as variáveis de entrada potenciais serão usadas para construir o modelo de árvore de decisão e, em alguns casos, uma variável de entrada específica pode ser usada várias vezes em diferentes níveis da árvore de decisão.

Parada

Complexidade e robustez são características concorrentes de modelos que precisam ser consideradas simultaneamente na construção de um modelo estatístico. Quanto mais complexo for um modelo, menos confiável será quando usado para prever registros futuros.

Uma situação extrema é construir um modelo de árvore de decisão muito complexo que se espalhe o suficiente para tornar os registros em cada nó folha 100% puros (ou seja, todos os registros têm o resultado de destino). Tal árvore de decisão seria excessivamente ajustada às observações existentes e teria poucos registros em cada folha, portanto, não poderia prever casos futuros de forma confiável e, portanto, teria baixa generalização (ou seja, falta de robustez) [21].

Para evitar que isso aconteça, regras de parada devem ser aplicadas ao construir uma árvore de decisão para evitar que o modelo se torne excessivamente complexo. Os parâmetros comuns usados nas regras de parada incluem: (a) o número mínimo de registros em uma folha; (b) o número mínimo de registros em um nó antes da divisão; e (c) a profundidade (isto é, número de etapas) de qualquer folha do nó raiz. Os parâmetros de parada devem ser selecionados com base no objetivo da análise e nas características do conjunto de dados que está sendo usado.

Poda

Em algumas situações, as regras de interrupção não funcionam bem. Uma maneira alternativa de construir um modelo de árvore de decisão é cultivar uma árvore grande primeiro e, em seguida, podá-la para o tamanho ideal removendo nós que fornecem menos informações adicionais.

Assim, um método comum de selecionar a melhor subárvore possível de vários candidatos é considerar a proporção de registros com previsão de erro (ou seja, a proporção em que a ocorrência prevista do alvo está incorreta).

Outros métodos para selecionar a melhor alternativa é usar um conjunto de dados de validação (ou seja, dividir a amostra em dois e testar o modelo desenvolvido no conjunto de dados de treinamento no conjunto de dados de validação), ou, para pequenas amostras, validação cruzada (ou seja, dividir a amostra em 10 grupos e testar o modelo desenvolvido a partir de 9 dobras na décima dobra, repetido para todas as dez combinações e calculando a média das taxas ou previsões erradas). Existem dois tipos de poda, pré-poda (poda para frente) e pós-poda (poda para trás) [22].

2.4.5 Florestas Aleatórias (*Random Forests*)

No modelo de florestas aleatórias (em inglês *Random Forests*) são construídas diversas árvores de decisão a partir de subamostras aleatórias das amostras de treinamento. Diferentes árvores recebem diferentes pesos na votação da classificação final de acordo com o treinamento.

A aleatoriedade nas florestas produz árvores de decisão com erros de previsão um tanto dissociados. Ao tirar uma média dessas previsões, alguns erros podem ser cancelados. As florestas aleatórias alcançam uma variação reduzida combinando diversas árvores, às vezes ao custo de um ligeiro aumento no viés. Na prática, a redução da variância é frequentemente significativa, resultando em um modelo geral melhor[23].

2.4.6 Classificador de Ridge

Um classificador de Ridge, segundo Rifkin et al. [24], é um modelo linear que semelhante a uma regressão linear ajusta coeficientes de forma a minimizar a soma residual dos quadrados entre os alvos observados nas amostras e os alvos preditos pela aproximação linear. De forma matemática, o modelo de regressão linear resolve o problema apresentado na Equação 2.5:

$$\min_{w} ||Xw - y||_2^2 \tag{2.5}$$

Uma regressão de Ridge ataca o mesmo problema acrescentando uma penalidade ao tamanho dos coeficientes. Esses coeficientes de Ridge minimizam a soma residual dos quadrados. Matematicamente a regressão de Ridge pode ser descrita como resolvendo esse problema:

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2 \tag{2.6}$$

Quanto maior o parâmetro α mais os coeficientes se tornam robustos a colinearidade. Como pode ser expressado na imagem abaixo.

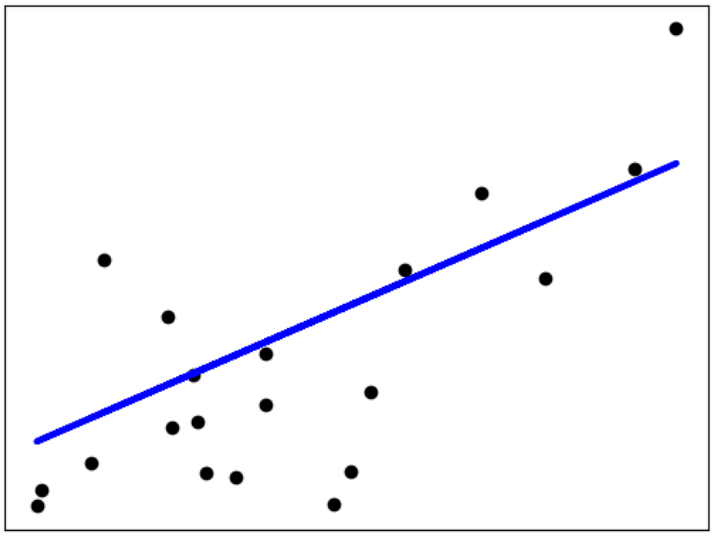

Figura 2.1:
Exemplo de Regressão Linear. Fonte [25]

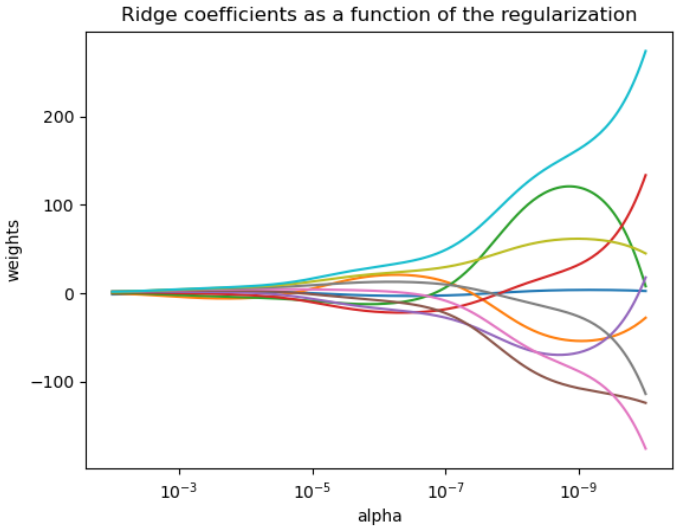

Figura 2.2:
Representando diferentes α na regressão de Ridge. Fonte [25].

2.4.7 Classificador Ingênuo Multinomial de Bayes

Os métodos de Bayes Ingênuo são uma série de algoritmos de aprendizagem supervisionada baseados na aplicação do teorema de Bayes que parte da premissa "ingênua"de

independência condicional entre cada par de entradas dado o valor de uma variável de etiqueta. O teorema de Bayes afirma a seguinte relação, dado uma variável de etiqueta y e um vetor de entradas dependentes x_1 até x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.7}$$

Usando a premissa ingênua de independência condicional tal como descrito na Equação 2.8:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y), \tag{2.8}$$

para todo i, esse relacionamento é simplificado para:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{2.9}$$

Já que $P(x_1, \ldots, x_n)$ é constante dado a entrada, nós podemos usar a seguinte regra de classificação:

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{gathered} \tag{2.10}$$

Os diferentes classificadores ingênuos de Bayes se diferem principalmente pelas premissas que tomam com relação a distribuição de $P(x_i \mid y)$.

Para o caso do classificador ingênuo multinomial de Bayes a distribuição é parametrizada por vetores $\theta_y = (\theta_{y_1}, ..., \theta_{y_n})$ para cada etiqueta y, no qual n é o tamanho do vocabulário e θ_{y_i} é a probabilidade $P(x_i \mid y)$ de uma palavra i aparecendo em uma amostra pertencer a etiqueta y. Os parâmetros θ_y são estimados pela versão amortecida de máxima semelhança descrita pela função:

$$\hat{\theta}_{y_i} = \frac{N_{y_i} + \alpha}{N_y + \alpha n} \tag{2.11}$$

Sendo $N_{y_i} = \sum_{x \in T} x_i$ é o número de vezes que a palavra i aparece na amostra da etiqueta y no conjunto de treino T e $N_y = \sum_{i=1}^{n} N_{y_i}$ é a contagem de todas as palavras que aparecem na etiqueta y.

Os αs de suavização são ajustes para as palavras não presentes no treinamento e previnem probabilidades nulas. $\alpha = 1$ é chamado de suavização de Laplace enquanto $\alpha < 1$ é chamado de suavização de Lindstone [26].

2.4.8 Classificador AdaBoost

O princípio central do AdaBoost é ajustar uma sequência de modelos auxiliares ligeiramente melhores que chutar aleatoriamente, como pequenas árvores de decisão, em versões diferentes da amostra de dados inicial. A predição de todos esses são combinadas por meio de uma soma ponderada para produzir uma predição final. As alterações nos dados a cada uma das iterações de impulsionamento (*boosting*) consistem em aplicar os pesos $w_1, w_2, ..., w_N$ para uma das amostras de treinamento. Inicialmente, esses pesos são calculados da forma $w_i = 1/N$, assim a primeira iteração treina um modelo mais fraco em relação aos dados originais. Para cada iteração posterior, os pesos das amostras são ajustados individualmente e o algoritmo de aprendizado é reaplicado para a amostra. As amostras de treinamento que foram incorretamente preditas na etapa anterior têm seus pesos aumentados, enquanto os pesos são diminuídos para aqueles que foram previstos corretamente. À medida que as iterações prosseguem, as amostras difíceis de prever recebem pesos cada vez maiores. Cada modelo auxiliar subsequente é, portanto, forçado a se concentrar nos exemplos que foram perdidos pelos anteriores na sequência [27].

2.4.9 Classificador Gradient Boosting

Classificadores Gradient Boosting (*GBRT*, sigla do inglês *Gradient Boosting Regresssion Trees*) são modelos aditivos em que a predição x_i para uma dada entrada y_i segue a forma da Equação 2.12.

$$\hat{y}_i = F_M(x_i) = \sum_{m=1}^{M} h_m(x_i), \tag{2.12}$$

onde h_m são os estimadores chamados aprendizes fracos. *Gradient Tree Boosting* usam árvores de decisão de tamanho fixo como aprendizes fracos. O M constante corresponde ao número de estimadores. Similar a outros algoritmos, a *GBRT* é construída sob o paradigma guloso como na Equação 2.13.

$$F_m(x) = F_{m-1}(x) + h_m(x), \tag{2.13}$$

onde uma nova árvore adicionada h_m é ajustada de forma a minimizar a soma das perdas l_m da montagem anterior F_{m-1}, vide Equação 2.14.

$$h_m = \arg\min_h L_m = \arg\min_h \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + h(x_i)), \tag{2.14}$$

onde $l(y_i, F(x_i))$ é definido como o parâmetro de perda. O modelo inicial F_0 é escolhido como a constante que minimiza as perdas. Para uma perda de mínimos quadrados, esse valor é a média empírica dos valores alvos. Usando uma aproximação de primeira ordem de Taylor, o valor pode ser aproximado como na equação 2.15.

$$l(y_i, F_{m-1}(x_i) + h_m(x_i)) \approx l(y_i, F_{m-1}(x_i)) + h_m(x_i) \left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)} \right]_{F=F_{m-1}}. \tag{2.15}$$

A quantidade $\left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}$ é a derivada da perda com relação a seu segundo parâmetro, avaliado em $F_{m-1}(x)$. Chamando esse termo como g_i e removendo os termos constantes, obtemos a equação 2.16.

$$h_m \approx \arg\min_h \sum_{i=1}^{n} h(x_i) g_i \tag{2.16}$$

Isso é minimizado se $h(x_i)$ for ajustado para predizer o valor que for proporcional ao gradiente negativo $-g_i$. Portanto, a cada iteração, o estimador h_m é ajustado para predizer o gradiente negativo das amostras. Os gradientes são então atualizados a cada iteração [28]

2.4.10 Classificador Extreme Gradient Boosting (XGB)

De acordo com Chen et al [29]., o *xgboost* ou *Extreme Gradient Boosting* é um implementação da estrutura de um *gradient boosting* mais eficiente e escalonável.

2.4.11 Classificador Passivo Agressivo

Algoritmos passivo-agressivos são um tanto semelhantes a um modelo Perceptron, no sentido de que não requerem uma taxa de aprendizado. No entanto, eles incluem um parâmetro de regularização [30].

Os algoritmos passivo-agressivos são definidos como:

- *Passivo*: se a previsão estiver correta, mantenha o modelo e não faça nenhuma alteração. ou seja, os dados no exemplo não são suficientes para causar qualquer mudança no modelo.
- *Agressivo*: se a previsão estiver incorreta, faça alterações no modelo. ou seja, alguma mudança no modelo pode corrigi-lo [30].

2.4.12 Deep Learning

Aprendizagem Profunda, em inglês *Deep learning* é uma das bases da inteligência artificial (IA). Faz-se o uso de modelos de redes neurais com diferentes arquiteturas. Esses modelos utilizam uma hierarquia de recursos ou conceitos onde as representações de nível superior de abstração são definidas a partir de outros em menor nível de abstração e onde as mesmas representações de nível inferior ajudam a definir as de nível superior [31].

Por exemplo, uma rede neural convolucional treinada para solucionar uma questão de identificação de objetos, como um carro. Durante seu treinamento, seus neurônios mais próximos a entrada podem se tornar mais sensível para padrões específicos como luminosidade, cores ou formas enquanto seus neurônios mais próximos à saída podem compor esses diferentes conceitos em padrões mais abstratos como movimento ou tipificação partes do objeto - como uma porta ou pneu.

Técnicas de *Deep Learning* têm aprimorado a capacidade dos computadores em classificar, reconhecer, detectar e descrever [31].

Classificador de Perceptron de Multicamadas (MLP)

O *perceptron* de multicamadas (*MLP*, a sigla vem do inglês *Multi-layer Perceptron*) é um algoritmo de aprendizado supervisionado que, de acordo com Goodfellow et. al. [32], aprende uma função $f(\cdot) : R^m \rightarrow R^o$ por treinamento utilizando um conjunto de dados, onde m é o número de dimensões para entrada e o é o número de dimensões para saída. Dado um conjunto de recursos $X = x_1, x_2, ..., x_m$ e y um alvo, ele pode aprender um aproximador de função não linear para classificação ou regressão. É diferente da regressão logística, pois entre a camada de entrada e a de saída pode haver uma ou mais camadas não lineares, chamadas camadas ocultas. A Figura 2.3 mostra um MLP de uma camada oculta com saída escalar.

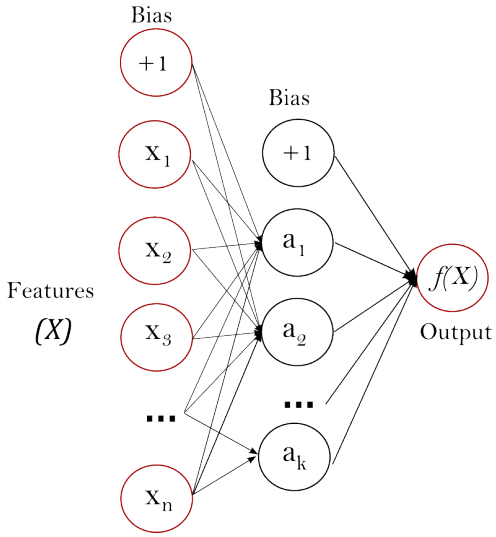

Figura 2.3:
Representação de um MLP, fonte [25]

A camada mais à esquerda, conhecida como camada de entrada, consiste em um conjunto de neurônios $\{x_i|x_1, x_2, ..., x_m\}$ representando os recursos de entrada. Cada neurônio na camada oculta transforma os valores da camada anterior com uma soma linear ponderada $w_1x_1 + w_2x_2 + ... + w_mx_m$, seguida por uma função de ativação não linear $g(\cdot) : R \rightarrow R$ - como a função tangente hiperbólica. A camada de saída recebe os valores da última camada oculta e os transforma em valores de saída [32].

Rede Neural Profunda (DNN)

Uma rede neural que consiste em mais de três camadas - incluindo as entradas e a saída - pode ser considerada um algoritmo de aprendizado profundo. Isso geralmente é representado usando o diagrama 2.5.

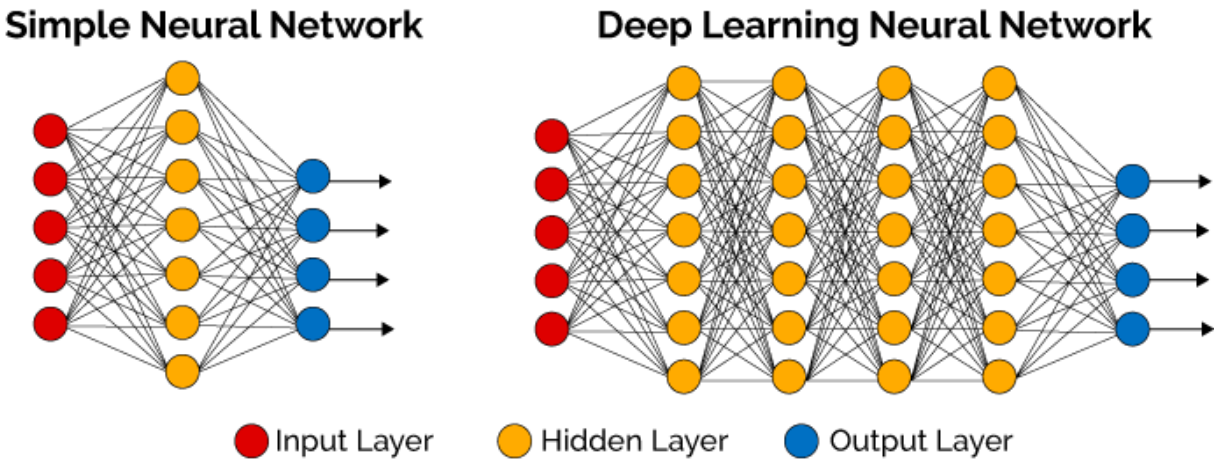

Figura 2.4:
Comparação de uma rede neural simples e uma DNN, fonte [33].

A maioria das redes neurais profundas é *feed-forward*, significando que fluem em uma direção apenas, da entrada para a saída. No entanto, você também pode treinar seu modelo por meio de retropropagação, ou seja, mova-se na direção oposta da saída para a entrada. A retropropagação nos permite calcular e atribuir o erro associado a cada neurônio, permitindo ajustar e ajustar o algoritmo de forma adequada [32].

Rede Neural Recorrente (*RNN*)

Redes Neurais Recorrentes (*RNN*, a sigla vem do inglês *Recurrent Neural Networks*) são um tipo poderoso e robusto de rede neural e pertencem aos algoritmos mais promissores em uso, porque é o único com uma memória interna.

Como muitos outros algoritmos de aprendizado profundo, as redes neurais recorrentes são relativamente antigas. Eles foram criados inicialmente na década de 1980, mas apenas nos últimos anos tem-se pesquisado seu verdadeiro potencial. Um aumento no poder computacional, junto com as enormes quantidades de dados com os quais temos que trabalhar agora, e a invenção da memória longa de curto prazo (*LSTM*) na década de 1990, realmente trouxe os *RNNs* para o primeiro plano.

Por causa de sua memória interna, os *RNNs* podem se lembrar de coisas importantes sobre a entrada que receberam, o que lhes permite ser muito precisos ao prever o que está por vir. É por isso que eles são o algoritmo preferido para dados sequenciais, como séries temporais, fala, texto, dados financeiros, áudio, vídeo, clima e muito mais. As redes neurais recorrentes podem formar uma compreensão muito mais profunda de uma sequência e seu contexto em comparação com outros algoritmos.

Uma rede neural recorrente é capaz de lembrar entradas interiores por causa de sua memória interna. Ele produz uma saída, copia essa saída e a envia de volta para a rede.

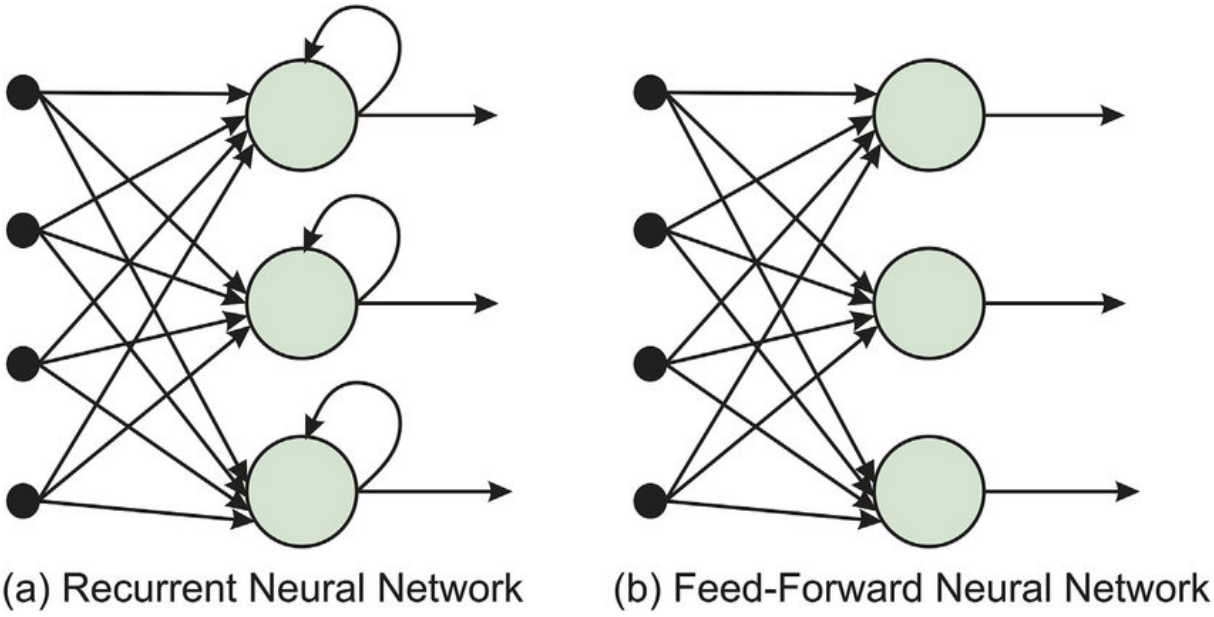

Figura 2.5:
Comparação de uma rede neural recorrente(a) e uma rede neural tradicional(b), fonte [34].

Portanto, uma *RNN* tem duas entradas: o presente e o passado recente. Isso é importante porque a sequência de dados contém informações cruciais sobre o que está por vir, razão pela qual uma *RNN* pode fazer coisas que outros algoritmos não podem [35].

Unidades Recorrentes com Portas (GRU)

Unidades Recorrentes com Portas (*RNN*, a sigla vem do inglês *Gated Recurrent Units*) é uma variante da é semelhante aos *LSTMs*, pois também funciona para resolver o problema de memória de curto prazo dos modelos *RNN*. Em vez de usar informações de regulação de "estado da célula", ele usa estados ocultos e, em vez de três portas, tem duas - uma porta de redefinição e uma porta de atualização. Semelhante às portas nos *LSTMs*, as portas de redefinição e atualização controlam quanto e quais informações reter [36].

Para resolver o problema da dissipação do gradiente de uma *RNN* padrão, a GRU usa dois portões, *reset* e *update gate*. Basicamente, eles são dois vetores que decidem quais informações devem ser passadas para a saída. O que há de especial neles é que eles podem ser treinados para manter informações por mais tempo, sem dissipá-las ou remover informações irrelevantes para a previsão.

A estrutura da GRU, representada na Figura 2.6, permite capturar adaptativamente dependências de grandes sequências de dados sem descartar informações de partes anteriores da sequência. Isso é alcançado através de suas unidades de portões, semelhantes às das *LSTMs*. Esses portões são responsáveis por regular as informações a serem mantidas ou descartadas a cada etapa do tempo.

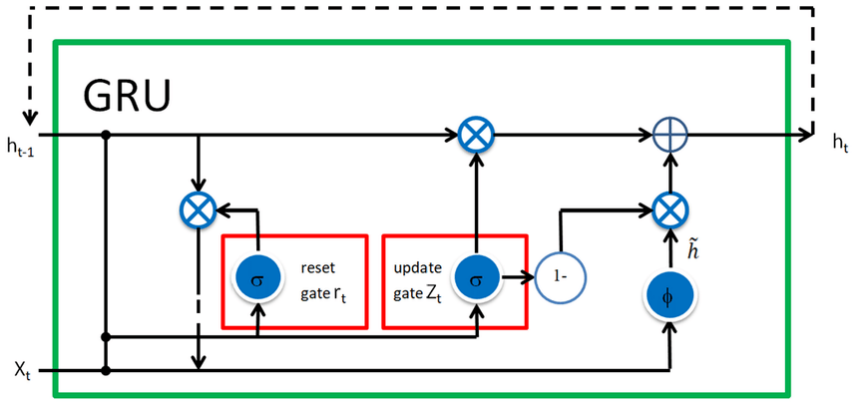

Figura 2.6:
Representação de uma GRU, fonte [37].

A capacidade da GRU de manter dependências ou memória de longo prazo decorre dos cálculos na célula da GRU para produzir o estado oculto. Enquanto as *LSTMs* têm dois estados diferentes passados entre as células – o estado da célula e o estado oculto, que carregam a memória de longo e curto prazo, respectivamente – as *GRUs* têm apenas um estado oculto transferido entre as etapas do tempo. Esse estado oculto é capaz de manter as dependências de longo e curto prazo ao mesmo tempo, devido aos mecanismos de restrição e cálculos pelos quais o estado oculto e os dados de entrada passam.

Memória de Curto Prazo Longa (*LSTM*)

A *LSTM*, segundo Goodfellow et. al. [32] é uma arquitetura de rede neural recorrente (*RNN*) que armazena valores em intervalos arbitrários. A *LSTM* é adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. A insensibilidade relativa ao comprimento da entrada dá uma vantagem à *LSTM* em relação a *RNNs* tradicionais (também chamadas "vanilla"), Modelos Ocultos de Markov (MOM) e outros métodos de aprendizado de sequências.

A estrutura de uma *RNN* é muito semelhante ao Modelo Oculto de Markov. No entanto, a principal diferença é como os parâmetros são calculados e construídos. Uma das vantagens da *LSTM* é a insensibilidade ao comprimento da entrada. *RNN* e MOM dependem do estado oculto antes da emissão/sequência. Se quisermos prever a sequência

após 1.000 intervalos em vez de 10, o modelo esqueceu o ponto de partida até então. Mas um modelo *LSTM* é capaz de armazenar essa informação por conta de sua estrutura de células, o diferencial da arquitetura *LSTM*.

A *LSTM* possui uma estrutura em cadeia que contém quatro redes neurais e diferentes blocos de memória chamados células. Como mostrado na Figura 2.7.

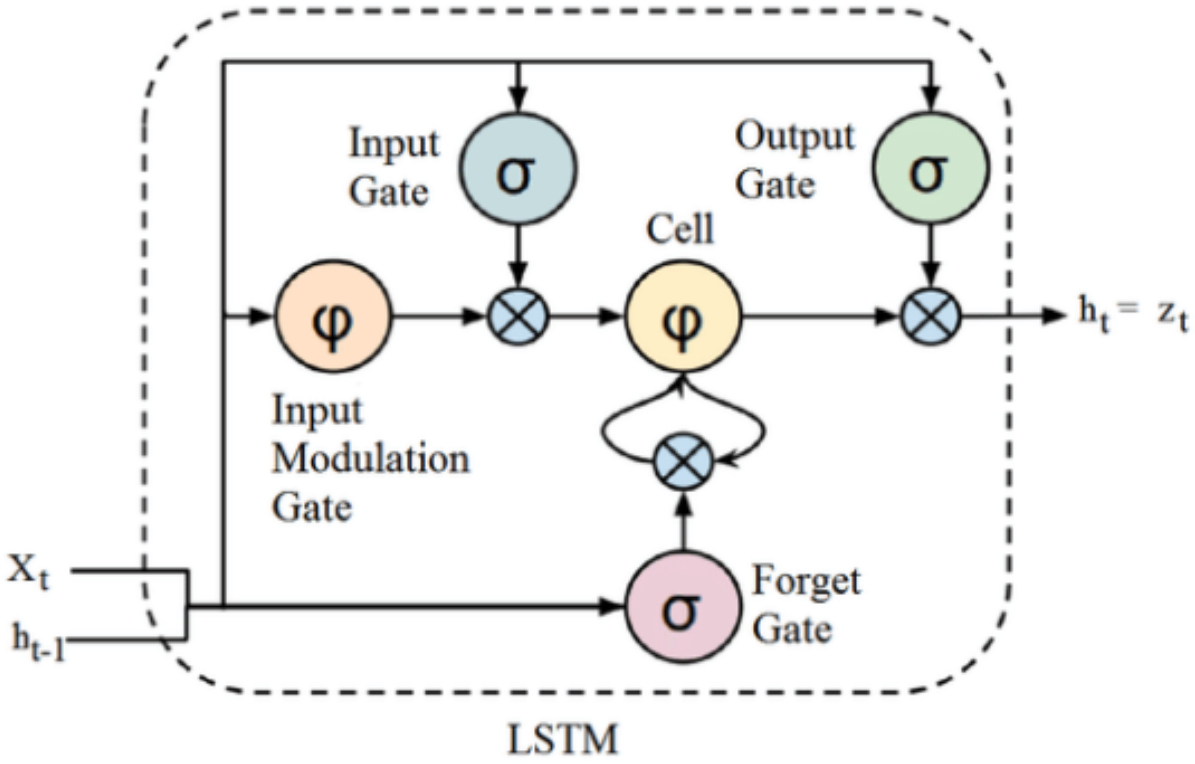

Figura 2.7:
Representação de uma rede neural de memória de curto prazo longa(*LSTM*), fonte [33].

A informação é retida pelas células e as manipulações de memória são feitas pelos portões (gates). Assim, existem três portões [32]:

- Função de esquecimento: as informações que não são mais úteis no estado da célula são removidas com o função de esquecimento. Duas entradas: x_t (entrada no momento específico) e $h_t - 1$ (saída de célula anterior) são alimentadas ao gate e multiplicadas por matrizes de peso, seguidas pela adição do bias. O resultante é passado por uma função de ativação que fornece uma saída binária. Se para um determinado estado de célula a saída for 0, a informação é esquecida e para a saída 1, a informação é retida para uso futuro;

- Função de entrada: a adição de informações úteis ao estado da célula é feita pelo função de entrada. Primeiro, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados de forma similar ao função de esquecimento usando as entradas $h_t - 1$ e x_t. Então, um vetor é criado usando a função $tanh$ que dá saída de -1 a $+1$, que contém todos os valores possíveis de $h_t - 1$ e x_t. Os valores do vetor e os valores regulados são multiplicados para obter as informações úteis;
- Função de saída: a tarefa de extrair informações úteis do estado da célula atual para ser apresentadas como uma saída é feita pelo função de saída. Primeiro, um vetor é gerado aplicando a função $tanh$ na célula. Então, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados usando as entradas $h_t - 1$ e x_t. Os valores do vetor e os valores regulados são multiplicados para serem enviados como uma saída e entrada para a próxima célula.

A célula *RNN* recebe duas entradas, a saída do último estado oculto e a observação no tempo $= t$. Além do estado oculto, não há informações sobre o passado para se lembrar. A memória de longo prazo é geralmente chamada de estado da célula. As setas em laço indicam a natureza recursiva da célula. Isso permite que as informações dos intervalos anteriores sejam armazenadas na célula *LSTM*. O estado da célula é modificado pelo função de esquecimento colocado abaixo do estado da célula e também ajustado pela porta de modulação de entrada. Da equação, o estado da célula anterior esquece, multiplica-se com a porta do esquecimento e adiciona novas informações através da saída das portas de entrada.

Redes Neurais Convolucionais (*CNN*)

De acordo com Goodfellow et al. [32], redes convolucionais (*CNN*, a sigla vem do inglês *Convolutional Neural Network*) são um tipo especializado de rede neural para processamento de dados que tem uma topologia conhecida, semelhante a uma grade. As redes convolucionais têm sido extremamente bem-sucedidas em aplicações práticas. O nome "rede neural convolucional" indica que a rede emprega uma operação matemática chamada convolução. A convolução é um tipo especializado de operação linear. As redes convolucionais são simplesmente redes neurais que usam convolução no lugar da multiplicação geral da matriz em pelo menos uma de suas camadas.

Como representado na Figura 2.8, as características das entradas são extraídas em uma ou mais camadas de convolução e depois tratadas em uma rede interna de neurônios a fim de relacionar essas características com saídas esperadas.

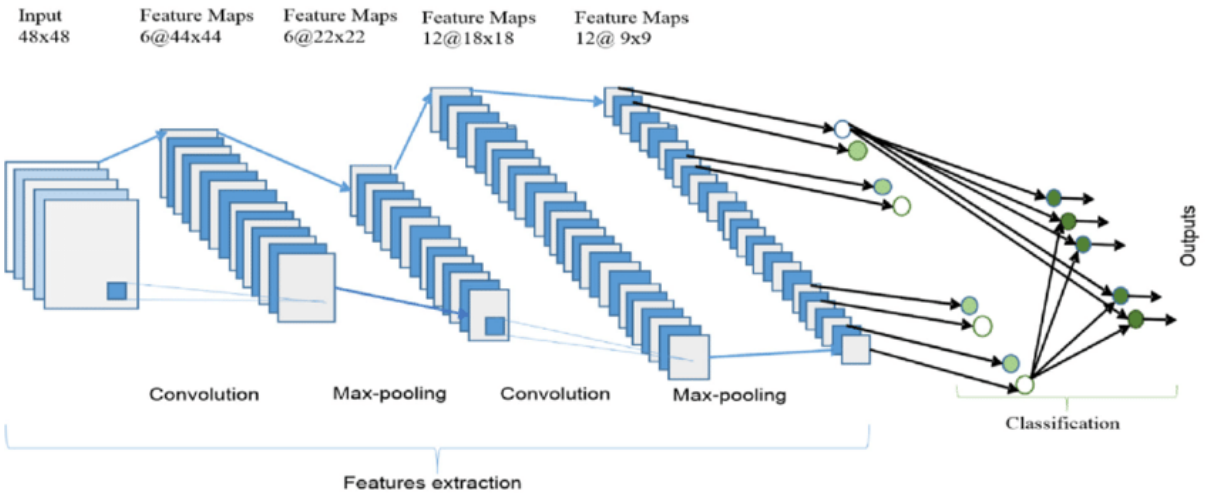

Figura 2.8:
Representação de uma rede neural convolucional (*CNN*), fonte [38].

2.5 Treinamento e Avaliação de Modelos

Nesta Seção serão fundamentadas os diferentes procedimentos de treinamento e avaliação de modelos de classificação.

2.5.1 Treinamento e Validação Cruzada

A validação cruzada, em inglês *cross-validation*, é usada principalmente no aprendizado de máquina para estimar a habilidade de um modelo de aprendizado de máquina em dados escassos ou para avaliar um fenômeno de baixa ocorrência. A técnica consiste em usar diferentes amostras limitadas para estimar como um modelo deve funcionar em geral quando usado para fazer previsões sobre dados não usados durante o treinamento do modelo. Como explicado por Browne [39], o procedimento geral pode ser abstraído para o algoritmo:

1. Misture o conjunto de dados aleatoriamente.
2. Divida o conjunto de dados em k grupos aleatoriamente
3. Para cada partição:
   (a) Separe essa partição como amostra de teste
   (b) Agrupe as $k-1$ partições restantes como amostra de treinamento
   (c) Treine um modelo com esse conjunto e o avalie utilizando o conjunto de teste

(d) Retenha as métricas de cada modelo

(e) Pontue o desempenho do modelo usando utilizando as métricas retidas

É importante ressaltar que cada observação na amostra de dados é atribuída a uma partição e permanece nesse grupo durante o procedimento. Isso significa que cada amostra individual tem a oportunidade de ser usada no conjunto de sustentação 1 vez e usada para treinar o modelo $k - 1$ vezes.

Essa abordagem envolve dividir aleatoriamente o conjunto de observações em k partições, ou dobras, de tamanho aproximadamente igual. A primeira partição é tratada como um conjunto de validação e o método é ajustado nas $k - 1$ partições restantes.

2.5.2 Métricas Estatísticas

Em problemas de classificação é necessário estabelecer métricas para avaliar um modelo. Existem diferentes técnicas utilizadas para levantar essas medidas. Como explicado por Power [40].

Como elas focam diferentes comportamentos de cada modelo, essas métricas podem ser utilizadas em conjunto para um entendimento geral do modelo e evitar vícios. Para entender melhor cada métrica, é necessário entender alguns conceitos.

Matriz de Confusão

Uma matriz de confusão é uma tabela que indica os erros e os acertos de um modelo comparando com o resultado esperado em amostras previamente categorizadas. Essas categorias podem ser chamadas de etiquetas, em inglês *labels*. A Figura 2.9 abaixo demonstra um exemplo de uma matriz de confusão.

	Categoria Predita	
Categoria Esperada	**Positivo**	**Negativo**
Positivo	**Verdadeiro Positivo (VP)**	**Falso Negativo (FN)**
Negativo	**Falso Positivo (FP)**	**Verdadeiro Negativo (VN)**

Figura 2.9:
Representação de uma matriz de confusão.

Explicando os componentes da matriz de confusão, tem-se:

- Verdadeiros Positivos (VP): o modelo previu positivo e o resultado esperado era positivo. A previsão foi correta.

- Falsos Negativos (FN - Erro Tipo II): erro em que o modelo previu a categoria Negativo quando o valor esperado era categoria Positivo.
- Falsos Positivos (FP - Erro Tipo I): erro em que o modelo previu a categoria Positivo quando o valor esperado era categoria Negativo.
- Verdadeiros Negativos (VN): classificação correta da categoria Negativo.

Ao ser feita a contagem de todos esses termos e se obter a matriz de confusão, é possível calcular métricas de avaliação. Algumas dessas métricas são:

Precisão

Dentre todas as classificações de etiqueta Positivo que o modelo fez, quantas estão corretas. A Equação 2.17 descreve o cálculo dessa métrica. A precisão pode ser usada em uma situação em que os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos. Por exemplo, ao classificar uma ação como um bom investimento, é necessário que o modelo esteja correto, mesmo que acabe classificando bons investimentos como maus investimentos (situação de Falso Negativo) no processo.

$$Precisão = \frac{VP}{VP + FP} \tag{2.17}$$

Acurácia

Indica uma performance geral do modelo. Dentre todas as classificações, quantas o modelo classificou corretamente. Ela é calculada de acordo com a Equação 2.18. A acurácia é uma boa indicação geral de como o modelo performou. Porém, pode haver situações em que ela é enganosa. Por exemplo, na criação de um modelo de identificação de fraudes em cartões de crédito, o número de casos considerados como fraude pode ser bem pequeno em relação ao número de casos considerados legais.

$$Acurácia = \frac{VP + VN}{Total} \tag{2.18}$$

Sensibilidade/Revocação/*Recall*

Esta técnica indica, dentre todas as situações de etiqueta Positivo como valor esperado, quantas estão corretas. A sensibilidade pode ser usada em uma situação em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos. Por exemplo, o modelo deve de qualquer maneira encontrar todos os pacientes doentes, mesmo que classifique alguns saudáveis como doentes (situação de Falso Positivo) no processo. A Equação 2.19 descreve o cálculo para encontrar a sensibilidade.

$$Sensibilidade = \frac{VP}{VP + FN} \tag{2.19}$$

F1-score

Média harmônica entre precisão e sensibilidade. O F1-Score é uma maneira de observar somente 1 métrica ao invés de duas (precisão e sensibilidade) em alguma situação. É uma média harmônica entre as duas, que está muito mais próxima dos menores valores do que uma média aritmética simples. Ver Equação 2.20.

$$F1 = \frac{2 \times Precisão \times Sensibilidade}{Precisão + Sensibilidade} \tag{2.20}$$

Curva Característica de Operação do Receptor (Curva *ROC*)

A Curva Característica de Operação do Receptor, em inglês *Receiver Operating Characteristic Curve (ROC curve)*, é uma medida de desempenho para os problemas de classificação em vários cenários. Representada de forma gráfica em dois eixos:

- Taxa de Verdadeiros Positivos, que é dado pelo número de Verdadeiros Positivos dividido pelo total de amostras que eram esperadas como Verdadeiras;
- Taxa de Falsos Positivos que é dado pelo valor de Falsos Positivos pelo total de amostras esperadas como Negativas;

Na Figura 2.10 foi representada uma curva *ROC*. A linha diagonal tracejada representa a curva esperada para um modelo aleatório, ou seja, com taxa de verdadeiros positivos e taxa de falsos positivos semelhantes. Quanto maior a área da curva *ROC* de um modelo acima da diagonal principal, melhor a avaliação do modelo.

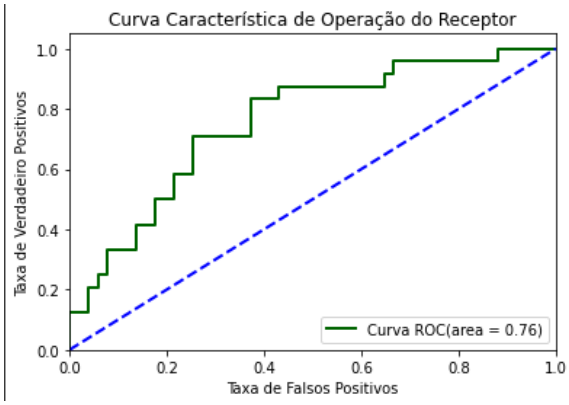

Figura 2.10:
Exemplo de curva ROC.

Capítulo 3

Trabalhos Relacionados

Neste capítulo serão abordados o projeto de pesquisa motivador para o trabalho realizado, e também os principais trabalhos relacionados ao tema discutido e apresentado neste manuscrito.

3.1 Classificação de Textos

Como explicado na Seção 2.4, os algoritmos de classificação de texto são, em geral uma função que pondera os dados de entrada para que a saída separe uma ou mais classes ou categorias.

A Figura 3.1 apresenta o grafo de conexões entre artigos produzido pelo aplicativo *connected papers* [1]. Os nós indicam os artigos citados pela revisão sistemática de Kowari [41], quanto mais escuro a representação do nó, mais recente são os trabalhos. Os tamanhos dos nós representam a quantidade de citações. Como referência, a revisão citada possui 295 citações, segundo o sistema do *connected papers*.

Outra conexão entre os trabalhos apresentados é o histórico de citações, começando pelo artigo *Bag of Tricks for Efficient Text Classification* com participação de Tomas Mikolov [42], do centro de pesquisa de inteligência artificial do Facebook. Neste artigo, a implementação de modelo SVM de Joachims de 1998, apresentada ao final deste capítulo, é citada como um poderoso modelo base para a classificação textual. Joachims [43], por sua vez, cita o trabalho de 1997 de Yang e Pedersen [44], ao confirmar a boa performance do modelo kNN na classificação de um mesmo corpus. Por fim, Yang e Pedersen citam o artigo de 1994 de Lewis [45], ao referenciar artigos que utilizam técnicas de redução de dimensionalidade com a extração e seleção de características.

[1] Maiores info: `https://www.connectedpapers.com/`

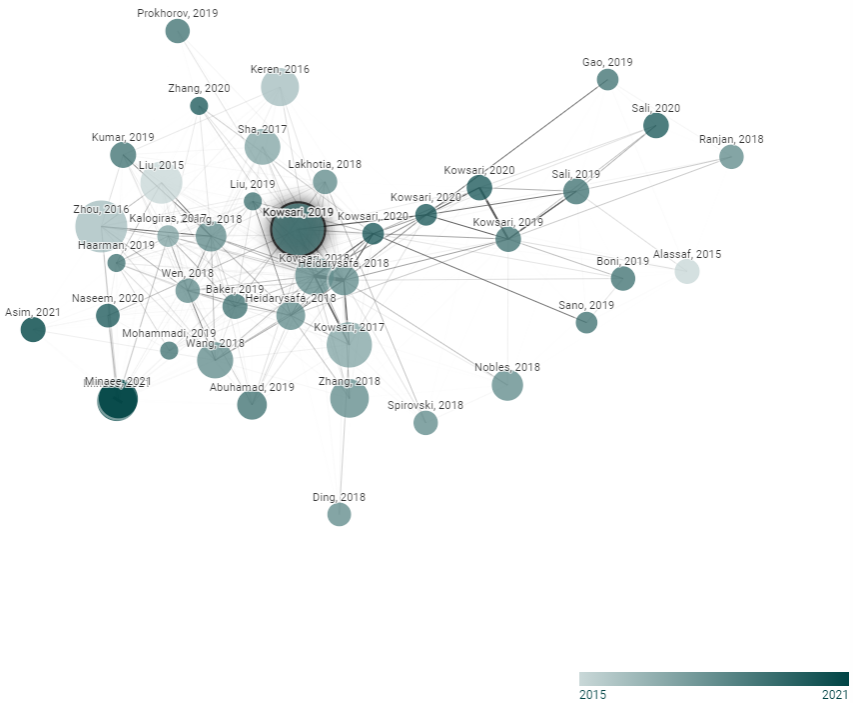

Figura 3.1: Grafo das conexões entre os trabalhos descritos neste capítulo.

3.1.1 Avaliação de Modelos de Classificação de Textos

Nessa subseção serão abordados os trabalhos relacionados à questão de classificação, como diferentes modelos se comportam e como foram avaliados os seus desempenhos.

Redes neurais recorrentes e convolucionais (CNN, RNN, LSTM)

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao e Bo Xu [46] compararam modelos de classificação como CNN, RNN, LSTMs e diversas variações dessas concepções centrais. Usaram seis bases textuais diferentes com diferente número de categorias, usos e tamanhos. Esses foram:

- MR[2] - Base textual de avaliações de fimes, podem ser positivas ou negativas;
- SST-1[3]. - *Stanford Sentiment Treebank* é uma extensão do MR de Socher et al; [47]. Contém avaliações de diferentes filmes em cinco categorias(muito negativo, negativo, neutro, positivo, muito positivo);
- SST-2[4] - Igual ao SST-1, mas com comentários neutros removidos e usa rótulos binários (negativo, positivo);

[2]https://www.cs.cornell.edu/people/pabo/movie-review-data/
[3]https://nlp.stanford.edu/sentiment/
[4]https://nlp.stanford.edu/sentiment/

- Subj[5] - Conjunto de dados com frases usados por Pang e Lee em 2004. [48]. Criados com o objetivo de classificar uma frase como sendo subjetiva ou objetiva;
- TREC[6] - Conjunto de dados de classificação de questões usados por Li e Roth, 2002. [49]. As questões podem ser classificadas em 6 tipos de perguntas (abreviatura, descrição, entidade, humano, localização, valor numérico);
- 20Newsgroups[7] - O conjunto de dados 20Ng contém mensagens de vinte grupos de notícias. Foram selecionadas quatro categorias principais(computação, política, recreação e religião) seguindo o exemplo de Hingmare et al. [50].

Após comparar os modelos envolvidos, os autores concluíram que o modelo de sua autoria, uma BLSTM-2DCNN (LTSTM Bidirecional acoplada a uma CNN com convoluções em 2 dimensões), tendo resultados de acurácia variando de 52.4% até 96.5% dependendo da base utilizada.

Máquinas de Vetores de Suporte (SVMs), KNN e Rochio

Hyunsoo Kim, Peg Howland e Haesun Park [51] comparam algumas abordagens utilizando uma SVM e um classificador usando KNN. Eles utilizaram como base de dados, o *Reuters-21578*. O qual possui 9603 documentos no conjunto de treinamento, e 3299 no conjunto de teste. Com cerca de 13 categorias para suas amostras textuais. Os resultados obtidos foram de acurácia média de 87%.

Além disso, Joachims [43], conduz experimentos realizando a classificação em dois corpora, com o objetivo de comparar os resultados de classificação de modelos SVM utilizando *kernels* polinomiais e RFB com os seguintes modelos convencionais de classificação: Naive Bayes, k-NN, o algoritmo Rocchio utilizado em sistemas de recuperação de informação (em inglês, *information retrieval* (IR)) e a árvore de decisão C4.5.

O primeiro conjunto textual, Reuters-21578, após pré-processamento possui 9962 termos distintos no conjunto de treinamento. O segundo, Ohsumed, possui 10000 documentos para treinamento e 10000 para testes. Após o pré-processamento, esse conjunto de documentos possui 15561 termos distintos no conjunto de treinamento.

Sobre os resultados observados para os métodos convencionais:

- O modelo k-NN apresentou a melhor resultado entre os modelos convencionais;
- Após o k-NN, o algoritmo Rochio apresentou o melhor resultado, seguido do modelo C4.5;

[5] http://cogcomp.cs.illinois.edu/Data/QA/QC/
[6] https://cogcomp.seas.upenn.edu/Data/QA/QC/
[7] https://nlp.stanford.edu/projects/glove//

- k-NN, Rochio e C4.5 obtiveram a maior performance utilizando o conjunto de 1000 atributos;
- A melhor performance do modelo Naive Bayes foi utilizando todos os atributos. Todavia, este foi o pior resultado entre as melhores performances dos outros modelos;

O artigo de Yang e Pedersen [44] apresenta métodos probabilísticos automáticos que analisam o texto quanto a frequência de seus termos para executar a importante tarefa de redução de dimensionalidade do espaço de atributos dos dados. Utilizando esta lógica de avaliação da importância dos atributos, os modelos implementados neste trabalho são baseados nos valores de frequência dos termos da base textual do Diário Oficial da União.

Em comparação com os estudos apresentados neste trabalho serão avaliadas a performance de diversos modelos de classificação já listados no Capítulo 2.

3.2 Projeto Deep Vacuity

Nesta seção são apresentadas as principais características do *framework* computacional do projeto *Deep Vacuity*[8], estes discutidos e apresentados em [52].

O projeto Deep-Vacuity tem como objetivo principal o desenvolvimento de metodologias para identificação de comportamento de cartéis de empresas em obras públicas utilizando técnicas de aprendizado de máquina e inteligência artificial.

Dentre as atividades relacionadas ao projeto, tem-se como atividades a serem desenvolvidas:

- Desenvolvimento e implantação de um *framework* com infraestrutura computacional de alto desempenho distribuído, para dar suporte às atividades de monitoramento e detecção de atividades suspeitas em licitações públicas. Estas realizadas em bases públicas disponíveis em sítios da web;
- Propor metodologias de captação de dados públicos nas esferas federais, estaduais e municipais;
- Incorporar no *framework* desenvolvido a base de aprendizado e expertise do corpo pericial da Polícia Federal no processo de detecção de fraudes em obras públicas para os ambientes computacionais;
- Permitir ações que abranjam outras esferas do poder público, tanto regulatório, quanto de fiscalização, incluindo forças policiais e órgãos de controle.

[8]Maiores info: `https://deepvacuity.cic.unb.br/`.

O projeto Deep-Vacuity visa auxiliar tarefas de fiscalização, auditoria e investigação, estas atualmente realizadas em sua maioria por análise humana. Desta forma, é de grande utilidade que o sistema tenha um canal de interação com um tipo de usuário final. Espera-se então que o sistema completo tenha a capacidade de acoplar interfaces úteis e de *frontend* com fácil usabilidade, como seguem alguns exemplos:

- Visualização de dados, metadados, agrupamentos e seleção de entidades;
- Painel de controle com gráficos e informações em escala macro e micro de determinados campos de conhecimento;
- Visualização de grafos e correlações entre dados, documentos e entidades;
- Representação espacial bidimensional ou tridimensional das relações entre dados;
- Visualização dos mesmos dados em diferentes escalas e tipos de intervalos temporais
- Representações dos dados por sua geo-localização;
- Interação com o usuário para validação e refinamento de resultados dos modelos inteligentes;
- Capacidade de retroalimentação de dados no sistema para retreinamento de modelos existentes ou treinamento de novos modelos inteligentes.

Este desenvolvimento é fruto da parceria com o Instituto Nacional de Criminalística (INC), do Serviço de Perícia em Engenharia (SEPENG) do Departamento de Polícia Federal (PF).

3.2.1 Histórico e Motivação

De acordo com [52], em meados de 2014, realizou-se uma operação anticorrupção, a Operação Lava-jato. Essa alcançou extraordinária magnitude em valores financeiros movimentados, além do número de agentes públicos envolvidos, de várias naturezas e espectros políticos.

Durante a operação, constatou-se que a capacidade de investigação, totalmente limitada e manual, dificultou o desenvolvimento e o surgimento de inúmeras linhas de trabalho científico e de esforço público de apoio ao combate aos crimes de corrupção (também conhecido por “crime do colarinho branco").

As investigações, segundo [52], resultaram na coleta de novos dados a serem processados e a partir daí surgiram vários movimentos na área de tecnologia da informação com o objetivo explícito de auxiliar no combate à corrupção, a saber:

1. Operação Serenata de Amor A proposta da Operação Serenata de Amor [53] é atuar no monitoramento dos gastos referentes à atividade parlamentar, principalmente monitorando de forma automatizada, com auxílio de tecnologia, os reembolsos efetuados pela Cota para Exercício da Atividade Parlamentar (CEAP) – verba que custeia alimentação, transporte, hospedagem e até despesas com cultura e assinaturas de TV dos parlamentares;

2. Operação Política Supervisionada A Operação Política Supervisionada [54] fiscaliza de forma detalhada os gastos realizados via CEAP (ou CEAPS); A OPS conta com a ajuda de seus colaboradores, espalhados pelo Brasil, para o levantamento de informações necessárias para a conclusão de fiscalizações, como por exemplo, o envio de fotos de endereços suspeitos em diversas cidades do país. Além disso, qualquer um pode ser um fiscal dos gastos públicos e este site oferece dados suficientes para isso.

3. Observatório da Despesa Pública O Observatório da Despesa Pública (ODP) [55] é uma unidade permanente do Ministério da Transparência e Controladoria-Geral da União (CGU) voltada à aplicação de metodologia científica, apoiada em tecnologia da informação de ponta, para a produção de informações que visam a subsidiar e a acelerar a tomada de decisões estratégicas por meio do monitoramento dos gastos públicos. O objetivo do ODP é contribuir para o aprimoramento do controle interno e funcionar como ferramenta de apoio à gestão pública, os resultados gerados pela unidade servem como insumo para realização de auditorias e fiscalizações conduzidas pela CGU, bem como para informar aos gestores sobre indicadores gerenciais relativos à realização de gastos públicos, de modo a permitir análises comparativas, subsidiando a tomada de decisões para melhoria da aplicação dos recursos públicos.

3.2.2 Desenvolvimento do Projeto do Sistema Deep Vacuity

Paralelamente ao desenvolvimento desta pesquisa, é desenvolvido o sistema que fará gestão do banco de dados, aplicação dos modelos desenvolvidos e possibilitará a interface a usuário destes dados. Para isso, o projeto emprega metodologia de gerenciamento de projetos baseadas nos princípios preconizados por métodos ágeis, com adoção de ferramenta própria para suporte das atividades de gerenciamento.

A Figura 3.2 apresenta a proposta de arquitetura inicial do Deep Vacuity, na qual são apresentados os principais aspectos da parte de infraestrutura e sua organização para o funcionamento esperado. No caso, o funcionamento é voltado para dar suporte à utilização de técnicas de aprendizagem de máquina profundo na atividade pericial. Nessa figura, o diagrama 3, a esquerda e em amarelo, representa as fontes públicas de dados disponíveis

em, por exemplo, o Diário Oficial da União. A fonte primária de dados. Esses dados são coletados pelos rastejadores representados logo ao lado no diagrama 2. O diagrama 1 de cor rosada ao centro e embaixo representada o organizador. Esse atribui e direciona os dados para outros módulos de armazenamento, que se comunicam com as interfaces com os usuários do sistema.

O trabalho realizado nessa pesquisa diz respeito ao diagrama 5, em amarelo e na direita embaixo, que recebe dados do organizados de licitações para avaliação de indícios de conluio.

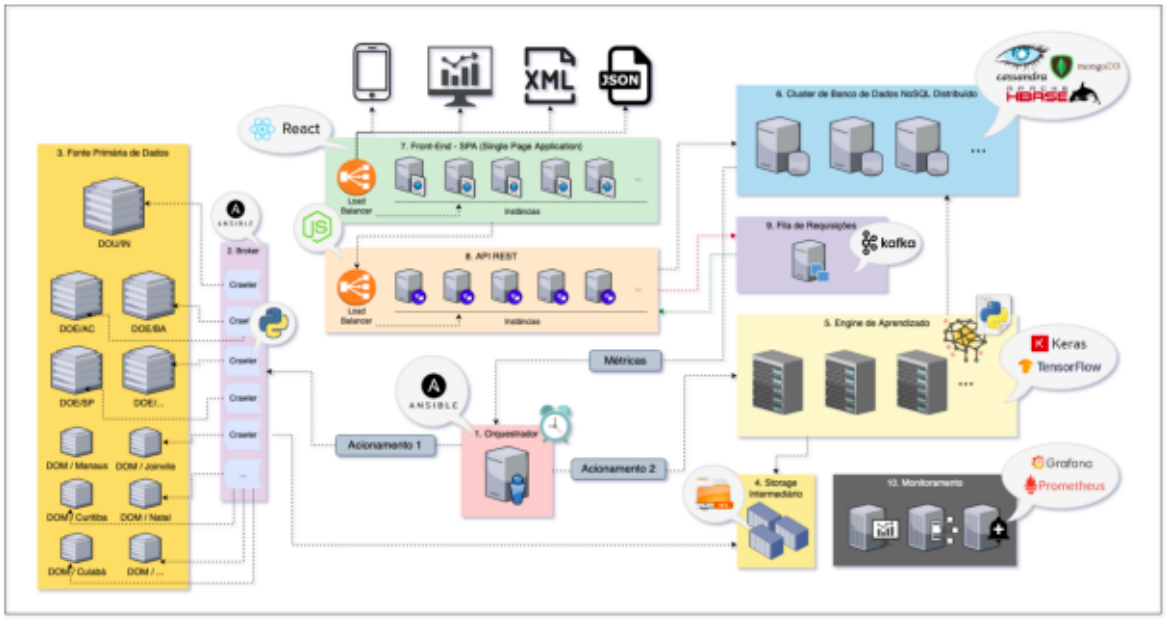

Figura 3.2:
Arquitetura inicial proposta para o Sistema Deep Vacuity.[9]

A aplicação de metodologia de gestão de projetos baseada na visão do SCRUM deve criar um equilíbrio entre as demandas de escopo, tempo, custo, qualidade e bom relacionamento entre os diversos atores do projeto. O sucesso dessa gestão estará relacionado ao alcance dos objetivos de: entrega dentro do prazo previsto, dentro do custo orçado, com nível de desempenho adequado, com plena aceitação pelo cliente e seus representantes (usuários finais), com atendimento de forma controlada às mudanças de escopo e em respeito à cultura da organização.

Por meio de um trabalho coordenado e interdependente entre as equipes do Instituto Nacional de Criminalística da polícia Federal e da Universidade de Brasília, as etapas de

[9]Esta figura foi elaborada por Leonardo Carvalho como parte do Relatório Descritivo de Atividades e Produtos Desenvolvidos (documento não publicado).

cada fase serão planejadas, discutidas, executadas e documentadas. As tarefas e atividades do convênio são sempre supervisionadas pelos coordenadores das duas instituições.

As equipes operacionais responsáveis pelos esforços técnicos serão lideradas por pesquisador sênior e por um gerente operacional. Uma equipe de gestão e qualidade será responsável pelos processos de gestão, pelo aceite dos relatórios do projeto e pelo acompanhamento do projeto. As equipes operacionais serão formadas por profissionais com diferentes experiências e qualificações.

3.3 Detecção de Conluio em Licitações

Nesta seção serão abordadas as iniciativas e técnicas atuais utilizadas na detecção de fraudes e conluio em licitações, especialmente aquelas realizadas pela Polícia Federal e a CGU, assim como a utilização de técnicas de inteligência artificial para este fim.

A Polícia Federal tem envidado esforços no desenvolvimento de mecanismos de detecção e comprovação de conluio em licitações. Neste intuito, Vallim [56] desenvolveu um modelo baseado em casos (*CBR*, sigla do inglês *Case-based reasoning*) abordando as licitações de pavimentação. Estes serviços, por serem grandes demandantes de recursos públicos em todas as esferas de governo, são constantemente alvo de de ações criminais. O modelo baseou-se em uma estrutura de dados que considerava: o tipo de licitação, as empresas envolvidas, os contratos e dados georeferenciados (localização) focando na classificação de casos de conluio e levantamento manual das informações.

De acordo com [52], várias iniciativas da Controladoria Geral da União, o órgão de controle interno do Governo Federal, também desenvolveram pesquisas no sentido de alcançar um classificador confiável para a detecção de fraudes em licitação.

Assim, foi desenvolvida a técnica de mineração de dados multi-agente (MAS) e descoberta de conhecimento em banco de dados (*database knowledge discovery*) com o intuito de detectar a cartelização de mercados públicos se baseando no sistema de compras governamentais, o ComprasNet. [57]. Foi proposta solução utilizando mineração com regra de associação (ARM) sob a base de compras do ComprasNet do período de 2005 a 2008, num total de 26.615 entradas, sendo 2.701 licitações. Segundo os autores, houve uma correta identificação de formação de cartel em 90% dos casos.

Balaniuk et al. [58] focou na avaliação de risco de fraude em agências governamentais usando classificadores do tipo Naïve Bayes para o planejamento de auditoria. Foram utilizados dados estruturados e busca por padrões de atividade fraudulenta. O processo separou 2.560 pares de entidades públicas e privadas dentro de um universo de 795.954 pares com alta probabilidade de risco.

Sun and Sales [59] utilizaram redes neurais tradicionais e redes neurais profundas (*DNN*, sigla em inglês para *Deep Neural Networks* -) para elaborar um sistema de alarme antecipado. Estes estudos usaram, de maneira geral, como características (*features*) e indicativos de fraude: o número de propostas, a relação entre preço e custo estimado, a relação entre os agentes públicos e privados, vinculação a partidos políticos etc.

Capítulo 4

Metodologia Proposta

Neste capítulo serão tratados os métodos utilizados para análise dos modelos relacionados à detecção de possíveis conluios em obras públicas. Seguindo o exposto no Capítulo 2, o fluxo proposto na Figura 4.1 descreve as etapas como caixas maiores à esquerda e cada uma de suas subetapas expandindo horizontalmente à direita. Serão explicadas nas próximas seções cada um dos tópicos ilustrados Figura 4.1.

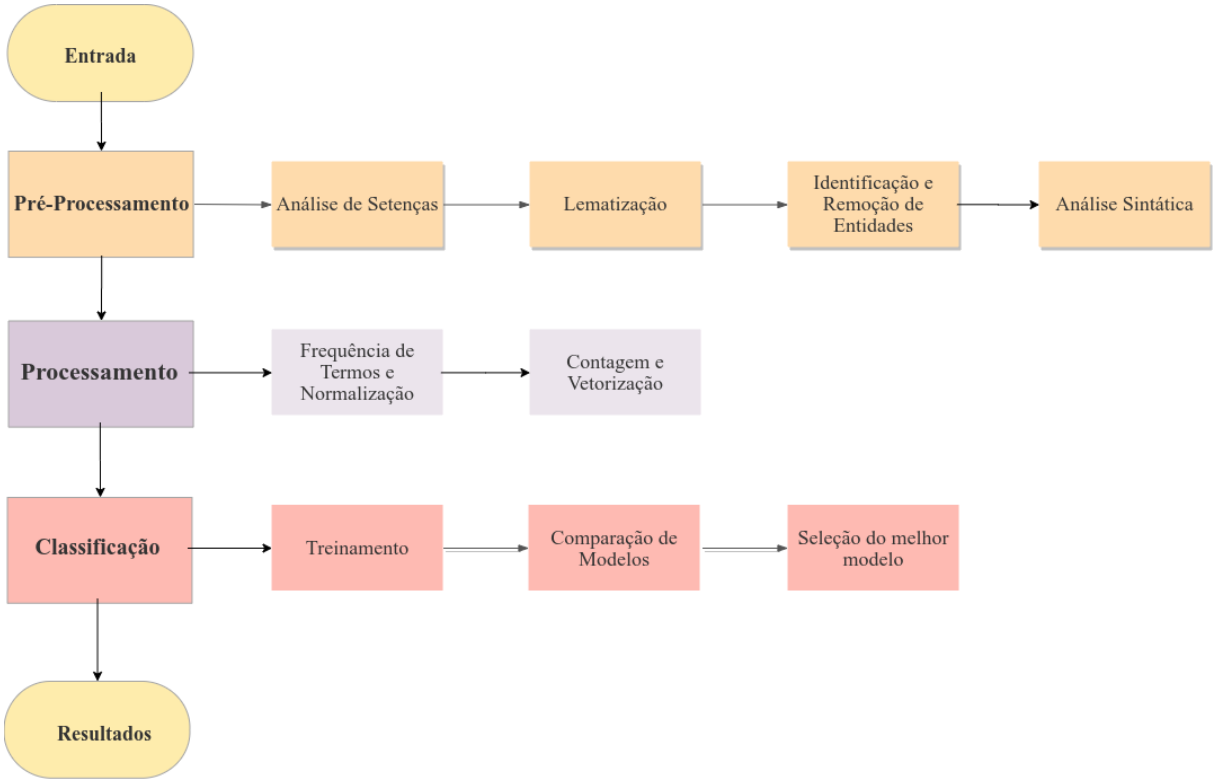

Figura 4.1: Fluxo da metodologia proposta.

4.1 Pré-Processamento

As amostras selecionadas inicialmente disponibilizadas foram de cerca de 15 mil documentos textuais contendo diversas páginas de texto. Espera-se que esses documentos sejam fornecidos no formato JSON, e previamente classificados com relação ao risco de conluio de forma binária, tendo indícios de conluio ou não.

Esses arquivos serão carregados para memória principal. Tendo seus corpos textuais tratado pela remoção de símbolos e outros caracteres especiais, considerados pouco explicativos ou provenientes de possíveis ruídos no processo de reconhecimento ótico de caracteres (*OCR*). Também se pretende remover marcações disponíveis nos textos originais, que serão provenientes de páginas virtuais, relacionadas a HTML. Mais informações sobre palavras normalmente com pouco poder explicativo, palavras de parada, foram dadas na Subseção 2.2.2, e sobre remoção de ruído na Subseção 2.2.5.

4.1.1 Análise de Sentenças

Nesta sub-etapa será feita uma análise do conteúdos dos textos utilizando o modelo de processamento de linguagem natural da biblioteca spacy [60]. Esse modelo é utilizado para fazer a demarcação de sentenças para melhor identificação de palavras além de melhorar a extração de seu significado semântico e sintático nas orações que fazem parte.

Durante essa análise, fazer-se-á os procedimentos de tokenização previamente detalhado na Subseção 2.2.1, identificação de sentenças e análise da capitalização das palavras como explicado na Subseção 2.2.3. Esses procedimentos prepararão o texto para o próxima subetapa, a lematização.

4.1.2 Lematização

Com as sentenças analisadas e as palavras atomizadas, pretende-se aplicar a lematização a cada uma desses *tokens*. A lematização é semelhante à *stemização*. Para mais informações sobre a lematização, busque a Subseção 2.2.8. Para a *stemização*, vá a Subseção 2.2.7.

4.1.3 Identificação e Remoção de Entidades

Nessa subetapa, será feita a classificação de cada palavra como pertencentes ou não a um dos seguintes grupos:

- PER - para nomes de pessoas ou famílias;
- LOC - para nome de uma localidade geográfica ou política como cidades, países, regiões, montanhas ou corpos de água;

- ORG - para entidades corporativas governamentais ou outros tipos de organizações;
- MISC - para eventos, nacionalidades, produtos ou obras de arte.

Uma palavra não necessariamente precisa fazer parte de um desses quatro tipos, mas é atribuída a no máximo um deles. Assim, depois de analisada a disposição dessas entidades nos corpos textuais, serão removidas as entidades de tipo PER e ORG a fim de evitar sobre-ajustes ligados a pessoas, famílias ou empresas que possam viciar os classificadores. Como exemplo, uma mesma empresa ou família pode aparecer em documentos diferentes majoritariamente com uma das categorias e viciar os modelos de classificação.

4.1.4 Análise Sintática

A próxima etapa é a de filtragem de palavras de acordo com seu papel sintático. Manter-se-ão nomes próprios, substantivos, verbos, advérbios e adjetivos. As demais funções sintáticas são descartadas. Para finalizar a etapa de preprocessamento, remover-se-ão acentos e pontuação de palavras. Assim como textos de peças que possam ter sido esvaziadas após as filtragens anteriores.

4.2 Processamento

Nessa seção serão tratados os modelos de vetorização dos textos. Como os classificadores não trabalham com texto puro, mas com representações numéricas desse texto, é necessário um método eficiente para melhor descrever esses textos na forma de um vetor.

4.2.1 Frequência de Termos e Normalização

Nessa subetapa será feito um agrupamento em termos (N-gramas) contendo de uma a três palavras. Esses termos terão sua correspondência em cada texto contada em valores absolutos e essa informação será armazenada na forma de um vetor, veja a Seção 2.3.

4.2.2 Contagem e Vetorização

Esses vetores serão utilizados para gerar uma frequência relativa a cada documento de forma a normalizar a análise independente do comprimento do texto. Aqui serão atribuídos pesos de forma a priorizar termos com menor repetição a fim de estimular os classificadores no reconhecimento de padrões em palavras-chave ou combinações de palavras-chave em vez de termos que se repetem mais. Para mais detalhes, consultar Subseção 2.3.2.

Após essa vetorização, utilizar-se-á uma bateria de modelos de classificação para avaliar o desempenho de cada um.

4.3 Classificação

Nesta seção serão explicados os modelos a serem utilizados para classificação das amostras categorizadas de forma binária com relação à possibilidade de conluio. Suas peculiaridades, desempenho e resultados. Para isso, foram utilizados os seguintes modelos:

1. K-vizinhos mais próximos;
2. Máquinas de Vetor de Suporte (SVM);
3. Classificador SVM com SGB;
4. Árvores de Decisão;
5. Florestas Aleatórias (Random Forests);
6. Classificador de Ridge;
7. Classificador Ingênuo Multinomial de Bayes;
8. Classificador AdaBoost;
9. Classificador Gradient Boosting;
10. Classificador Extreme Gradient Boosting (XGB);
11. Classificador Passivo Agressivo;
12. Classificador de Perceptron de Multicamadas (MLP);
13. Rede Neural Profunda (DNN);
14. Rede Neural Recorrente (RNN);
15. Unidades Recorrentes com Portas (GRU);
16. Memória de Curto Prazo Longa (LSTM);
17. Convolutional Neural Networks (CNN);

4.3.1 Treinamento e Teste

A fim de garantir que os modelos serão avaliados de forma exaustiva e evitar possíveis sobre-ajustes, será utilizada uma técnica de validação cruzada das amostras, subseção 2.5.1.

Elas serão previamente divididas em dez lotes com diferentes seleções da amostra inicial. Devido a escassez de casos categorizados como positivos para a possibilidade de conluio poderá haver repetição desses casos nos diferentes lotes. Para cada um desses

dez diferentes lotes, será feita a subdivisão em duas frações: para treinamento e para testes. O percentual de treinamento e teste escolhido será entre 70-90% para o primeiro e 10-30% para o segundo grupo. As amostras de cada sublote serão reorganizadas em dez combinações de treinamento e teste diferentes. Assim, haverá dez reorganizações em treinamento e teste para cada um dos lotes, totalizando 100 diferentes conjuntos para treinamento e teste.

Assim, cada um dos modelos será avaliado em 100 cenários, permitindo uma comparação de comportamentos mais geral de cada modelo com o problema estudado.

4.3.2 Comparação de Modelos

Os modelos serão avaliados utilizando métricas estatísticas como acurácia, sensibilidade, precisão e F1-score. Para cada um dos cenários serão coletadas essas medidas, e tanto os valores individuais dos melhores cenários como as médias dos modelos serão utilizadas para ranquear os modelos utilizados. Priorizando o desempenho na métricas F1-score. Como explicado na Subseção 2.5.2.

Assim, o modelo com melhor desempenho será então avaliado em detalhes com diferentes técnicas de visualização dos dados.

4.3.3 Seleção do Melhor Modelo

Após a seleção do melhor modelo, será usada uma matriz de confusão para detalhar o melhor modelo. Identificando os pontos que o modelo pode ter tido maior sucesso e as partes de maior falha.

Capítulo 5

Resultados

O fluxo de solução desenvolvido para esta pesquisa seguiu as etapas definidas na Figura 4.1 no capítulo anterior. Os resultados de cada etapa e suas respectivas sub-etapas serão discutidos abaixo.

5.1 Pré-Processamento

As amostras selecionadas inicialmente disponibilizadas tiveram cerca de 15 milhões de documentos textuais contendo diversas páginas de texto. Os documentos foram fornecidos no formato JSON, e previamente classificados com relação ao risco de conluio de forma binária, tendo indícios de conluio ou não tendo.

Esses arquivos então foram carregados para memória principal. Os termos mais frequentes encontrados antes do pré-processamento podem ser vistos na Figura 5.1.

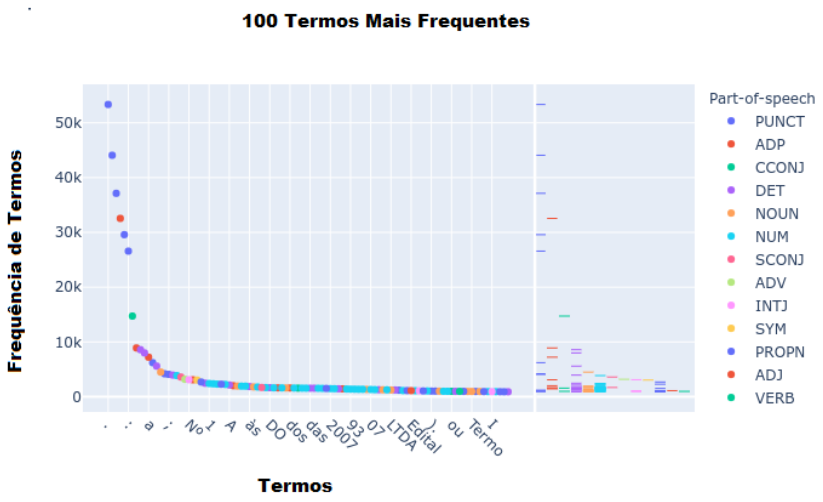

Figura 5.1: 100 Termos Mais Frequentes Antes do Pré-processamento

Seus corpos textuais foram inicialmente tratados removendo símbolos e outros caracteres especiais considerados pouco explicativos ou provenientes de possíveis ruídos no processo de reconhecimento ótico de caracteres(OCR). Também foram removidos marcações disponíveis nos textos originais relacionadas a *HTML*.

5.1.1 Análise de Sentenças

Nesta sub-etapa o texto foi subdividido em sentenças e cada sentença foi atomizada em *tokens*. Na Figura 5.2 é possível identificar os 100 *tokens* de maior frequência antes da lematização.

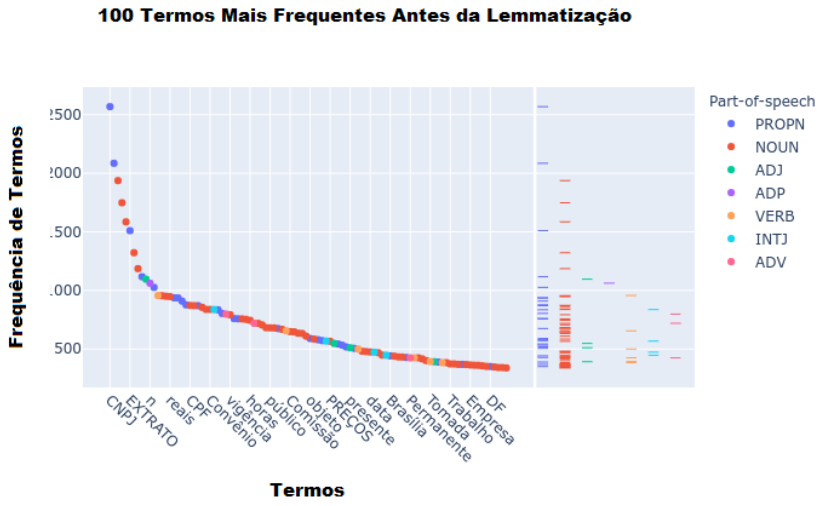

Figura 5.2: 100 Termos Mais Frequentes Antes Da Lematização

5.1.2 Lematização

Nesta etapa, os *tokens* foram normalizadas de acordo com suas variações morfológicas para uma forma base. Para isso, diferentes técnicas foram testadas até obter uma operação que normalizasse variações julgadas menos importantes, conservando as variações que continham mais informações. Assim, plurais e função sintática foram em sua maioria mantidos enquanto flexões verbais, distinções de gênero e a diferença entre letras minúsculas e maiúsculas foram majoritariamente alteradas. Na Figura 5.3 é possível visualizar os termos mais frequentes após a lematização.

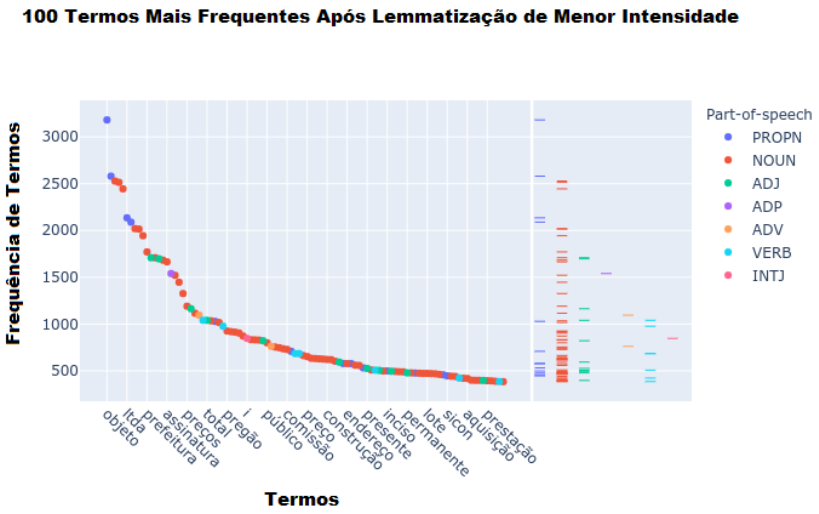

Figura 5.3: 100 termos mais frequentes após lematização.

5.1.3 Identificação e Remoção de Entidades

Nesta sub-etapa, as palavras foram avaliadas se seriam pertencentes a um tipos de entidade. A maior parte dos *tokens* não foi entendida como uma entidade. Dos que foram, o percentual encontrado de cada tipo de entidade pode ser visto na Figura 5.4. Como descrito na Seção 4.1.3, as entidades são agrupadas em quatro, os quais são: PER - para nomes de pessoas ou famílias; LOC - para nome de uma localidade geográfica ou política como cidades, países, regiões, montanhas ou corpos de água; ORG - para entidades corporativas, governamentais ou outros tipos de organizações; MISC - para eventos, nacionalidades, produtos ou obras de arte.

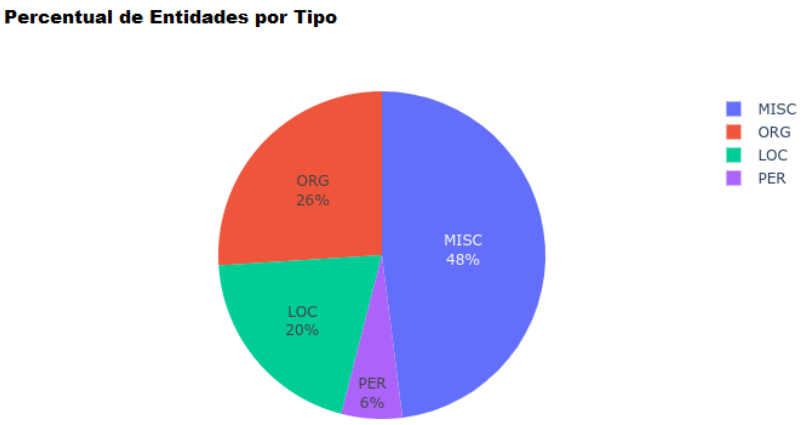

Figura 5.4: Percentual de entidades encontradas.

Depois de analisada a disposição dessas entidades nos corpos textuais, foram removidas as entidades de tipo PER e ORG a fim de evitar sobre-ajustes ligados a pessoas, famílias ou empresas que possam viciar os classificadores. Essas alterações podem ser vistas na Figura 5.5, apresentada essa nova disposição de entidades.

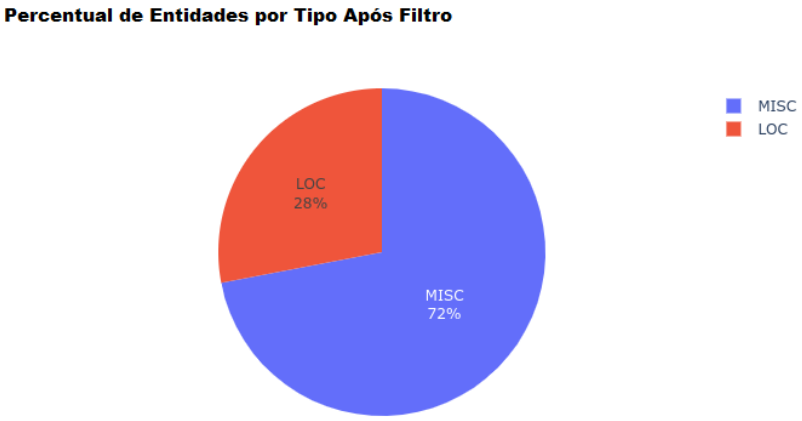

Figura 5.5: Percentual de entidades encontradas após filtragem.

5.1.4 Análise Sintática

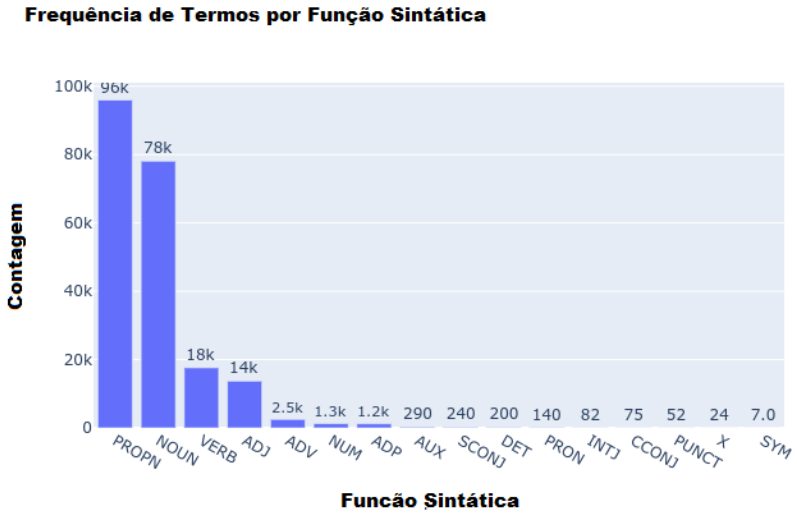

Figura 5.6: Percentual de funções sintáticas.

A próxima etapa é a de filtragem de palavras de acordo com seu papel sintático. Mantém-se nomes próprios, substantivos, verbos, advérbios e adjetivos. As demais funções sintáticas são descartadas. Para finalizar a etapa de pré-processamento, removem-se das palavras acentos e pontuação. Além disso, retira-se também textos de peças que possam ter sido esvaziadas após as filtragens anteriores.

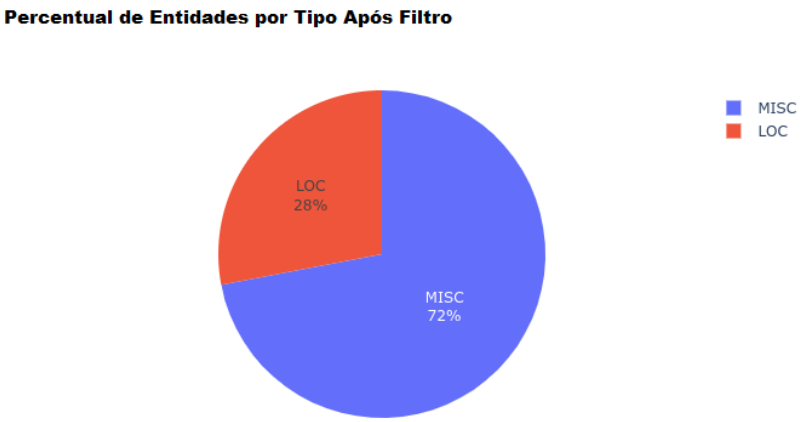

Figura 5.7: Percentual de funções sintáticas após filtragem.

5.2 Processamento

Nesta seção serão expostos resultados referentes aos modelos de vetorização dos textos. Como os classificadores não trabalham com texto puro, mas com representações numéricas desse texto, foi necessário um método eficiente para melhor descrever esses textos na forma de um vetor.

5.2.1 Contagem e Vetorização

Inicialmente foram feitos agrupamentos em termos (N-gramas) contendo de uma a três palavras. Esses termos tiveram sua correspondência em cada texto contada em valores absolutos e essa informação é armazenada na forma de um vetor. Na Figura 5.8 foram listadas na primeira coluna alguns dos *N-gramas* mais comuns. Na segunda coluna o seu código número correspondente dentro do modelo e na terceira coluna a sua frequência percentual com relação a uma das amostras de texto.

	Termo	Código do Termo	Frequência Percentual
0	acordo	214	9.72%
1	acordo lei	226	13.68%
2	cidade	3073	9.11%
3	conhecimento	3893	9.22%
4	conhecimento licitantes	3914	16.97%
5	conhecimento licitantes interessar	3915	17.47%
6	fica	8772	8.94%
7	fica revogado	8802	17.47%
8	informacoes	10470	7.22%
9	informacoes poderao	10502	11.73%
10	informacoes poderao obtidas	10504	12.35%
11	interessar	10873	13.78%
12	interessar possa	10874	15.03%
13	junto	11337	9.91%
14	km	11430	9.51%
15	km lote	11458	15.22%
16	lei informacoes	11756	16.57%
17	lei informacoes poderao	11757	17.47%
18	licitantes	12077	12.19%
19	licitantes interessar	12079	17.47%
20	licitantes interessar possa	12080	17.47%

Figura 5.8: Exemplos de N-gramas.

5.2.2 Frequência de Termos e Normalização

Esses vetores foram então usados em modelos para medir uma frequência relativa a cada documento de forma a normalizar a análise independente do comprimento do texto. Assim, foram atribuídos pesos de forma a priorizar termos com menor repetição a fim de estimular os classificadores a darem preferências para termos com maior poder de delimitação.

Após essa vetorização, utilizou-se modelos de classificação descritos no capítulo anterior.

5.3 Classificação

Nesta seção serão explicados os modelos que foram utilizados para classificação das amostras categorizadas de forma binária com relação à possibilidade de conluio.

5.3.1 Treinamento e Teste

Nesta subetapa os modelos foram divididos em dois grupos, os quais foram os neurais e os não-neurais. Sendo os não-neurais:

1. K-vizinhos mais próximos;
2. Máquinas de Vetor de Suporte (SVM);
3. Classificador SVM com SGB;
4. Árvores de Decisão;
5. Florestas Aleatórias (Random Forests);
6. Classificador de Ridge;
7. Classificador Ingênuo Multinomial de Bayes;
8. Classificador AdaBoost;
9. Classificador Gradient Boosting;
10. Classificador Extreme Gradient Boosting (XGB);
11. Classificador Passivo Agressivo.

E os neurais são:

1. Classificador de Perceptron de Multicamadas (MLP);

2. Rede Neural Profunda (DNN);
3. Rede Neural Recorrente (RNN);
4. Unidades Recorrentes com Portas (GRU);
5. Memória de Curto Prazo Longa (LSTM);
6. Convolutional Neural Networks (CNN).

Mais informações sobre cada modelo disponíveis na Seção 2.4.

Cada grupo recebeu um pré-processamento semelhante, exceto pela vetorização dos termos. Os modelos do grupo não-neural receberam dados tratados utilizando vetorização por frequência inversa (detalhes na Subseção 2.3.2). Enquanto os modelos do grupo neural foram alimentados com dados tratados por modelos de vetorização previamente treinados em amostras de caráter geral(veja Subseção 2.3.1).

Essa decisão foi tomada estudando as complexidades e as arquiteturas de cada modelo a fim de maximizar seus desempenhos.

Depois de pré-processados os dados, ambos os grupos tiveram os mesmos procedimentos. Cada modelo foi treinado com amostras separadas em amostras de treinamento e amostras de testes utilizando validação cruzada (visto na seção 2.5.1). Esta subetapa foi executada como especificada na Subseção 5.3.1

Depois de treinado cada modelo, eles foram então avaliados como descritos na próxima seção.

5.3.2 Comparação de Modelos

Os modelos foram avaliados utilizando métricas como precisão, acurácia, sensibilidade e *f1-score*. 2.5. Para cada lote da validação cruzada, uma nova versão de cada modelo foi treinada e avaliada. Assim, as métricas de cada modelo foram construídas agregando os resultados de cada versão em cada lote. Na Figura 5.9 foram utilizados as médias simples de cada uma das versões dos modelos para cada uma das respectivas métricas listadas. O eixo horizontal descreve os valores de precisão, e o eixo vertical relata os valores de sensibilidade(*Recall*). Cortes transversais pontilhados representam valores de F1-Score, métrica obtida pela média harmônica entre precisão e sensibilidade. Cada ponto colorido representa uma média de modelos e as hastes em formato de cruz medem a variância entre os resultados dos modelos. Na legenda estão listados todos os modelos. Os doze melhores modelos foram representados na Figura 5.9.

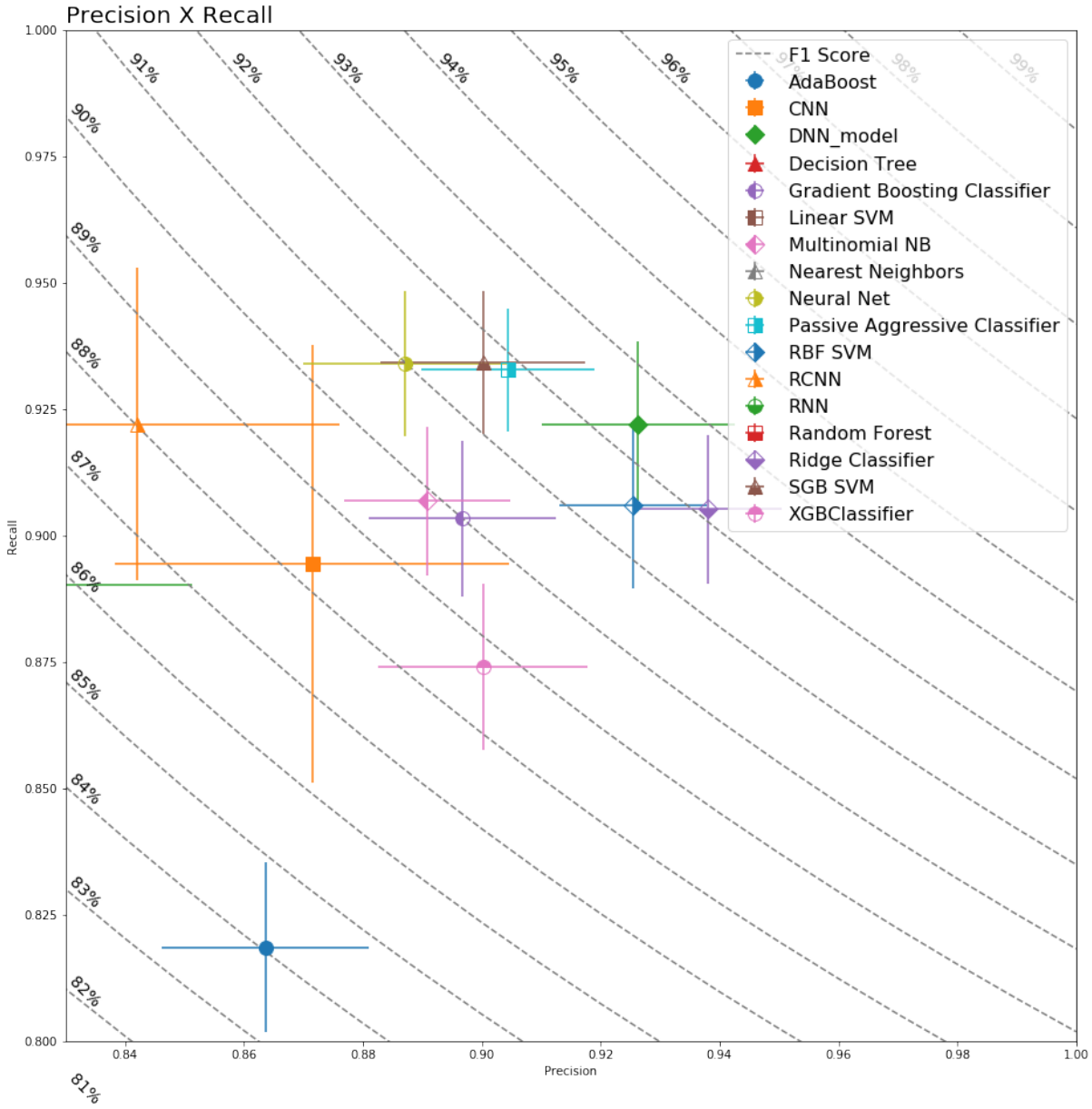

Figura 5.9: Gráfico de Precisão (eixo vertical) por Sensibilidade (eixo horizontal) com cortes transversais de *F1 Score*.

Na Tabela 5.1, o Macro *F1-Score* é obtido a partir da média dos desempenhos de cada modelo. A partir dessa métrica, é possível observar como alguns modelos conseguiram ter bons desempenhos em diferentes lotes e amostras enquanto outros tiveram menor desempenho.

Tabela 5.1: Tabela com as métricas agregadas de cada modelo

Modelo	**Macro F1-Score**
DNN	0.918195
Classificador Ridge	0,915813
Classificador Agressivo Passivo	0,910638
SGB SVM	0,909085
RBF SVM	0,906654
Rede Neural	0,905330
Multinomial NB	0,895415
Classificador Gradient Boosting	0,886195
RCNN	0,877314
CNN	0,872369
XGBClassifier	0,870653
RNN	0,847796
AdaBoost	0,833190
SVM linear	0,816911
Árvore de Decisão	0,737690
Vizinhos mais próximos	0,603052
Floresta aleatória	0,302980

A rede neural profunda (*DNN*) teve o melhor resultado em relação aos demais. Por isso, na Seção 5.3.3 será avaliado em maior detalhes o desempenho desse modelo.

5.3.3 Seleção do Melhor Modelo

Para avaliação dos modelos, foram comparadas diferentes métricas: Precisão, Sensibilidade, F1-Score e a curva *ROC* de cada versão. Essas métricas foram agregadas por modelo, calculadas métricas compostas como variância e o desvio padrão. O modelo que obteve melhores resultados foi o de redes neurais profundas. Descritos na Seção 2.4.12.

Analisando detalhadamente esses resultados, é possível ver a curva *ROC* desse modelo na Figura 5.10. A curva em azul escuro representa a média das curvas de cada uma das versões enquanto o degradê representa o desvio padrão da curva média.

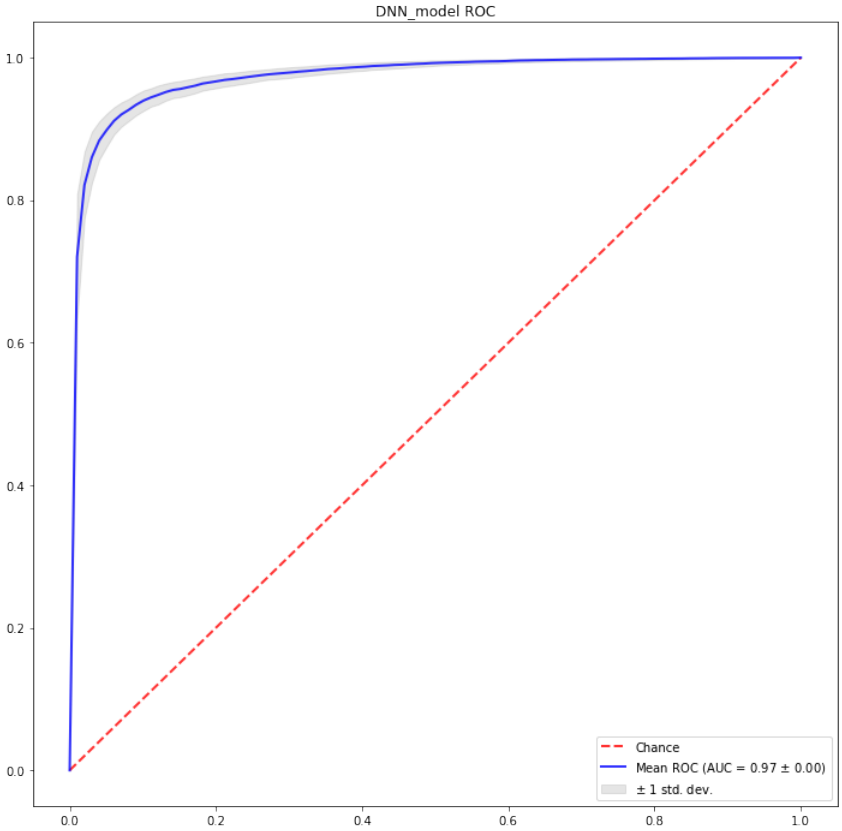

Figura 5.10: Curvas ROC para as versões do modelo DNN.

Na Tabela 5.2 estão disponíveis algumas métricas agregadas das versões desse modelo. Para o F1-Score, Precisão e Sensibilidade foram demonstrados o menor, o maior, a mediana, a média e o desvio padrão. É possível verificar que a Sensibilidade teve um desvio padrão maior do que os demais métricas, assim como o menor valor.

Tabela 5.2: Tabela com as métricas agregadas das versões do modelo DNN

Métricas	**F1-Score**	**Precisão**	**Sensibilidade**
Menor Valor	0.901750	0.910000	0.900000
Maior Valor	0.931759	0.930000	0.930000
Mediana	0.917688	0.920000	0.915000
Média	0.918195	0.919500	0.918000
Desvio Padrão	0.008582	0.008659	0.009813

Capítulo 6

Conclusões e Trabalhos futuros

Nesta pesquisa foi proposta a avaliação de performance de diferentes modelos de processamento de linguagem natural para identificação de risco de conluio em publicações de órgãos públicos. Para isso, foram fundamentadas diferentes técnicas e conceitos do campo do processamento de linguagens naturais (*NLP*), diferentes formas de tratamento de texto como tokenização (*tokenizing*), stemização (*stemming*) e lemanitzação (*lematization*). Além disso, foram implementadas também formas de vetorização e contagem de termos como *Word2Vec* e frequência relativa de termos. Para isto, foram avaliados 17 modelos de classificação entre eles: Máquinas de Vetor de Suporte (*SVM*), *Gradiendt Boosting*, Rede Neural Recorrente (*RNN*), Memória de Curto Prazo Longa (*LSTM*) e Rede Neural Profunda (*DNN*).

Durante o desenvolvimento da pesquisa, problemas relacionados a escala das amostras e a quantidade de modelos foram apresentados. Além de complicações de deslocamento e comunicação devido à pandemia de Covid-19. Essas questões foram resolvidas com otimização dos processos e uso de máquinas com mais recursos. Também houve complicação com a vetorização do fluxo de solução dos modelos de redes neurais. Um novo fluxo foi construído usando o modelo *Word2Vec* para vetorizar as amostras de texto durante o treinamento e avaliação desses.

A análise dos modelos de classificação demonstrou que, apesar de alguns modelos estatísticos lineares terem desempenho razoáveis, tendem a serem superados por redes neurais profundas para esse problema. Além disso, eles apresentaram a maior estabilidade para diferentes cenários, tendo variância menor. O modelo em que as versões em média tiveram melhor desempenho foi a Rede Neural Profunda(*DNN*). As amostras estudadas tendem a ter precisão maior do que a sensibilidade, o que pode levar a situações de mais Falsos Negativos - não detectar um risco de conluio - do que Falsos Positivos - sugerir um indício de conluio não presente naquela amostra.

6.1 Trabalhos Futuros

Para trabalhos futuros é sugerido a experimentação de diferentes formas de vetorização, o *Word2Vec* é um modelo semântico e há outros disponíveis na comunidade acadêmica, assim como sendo produzidos a cada ano. Também é recomendado experimentar diferentes formas de pré-processamento. Das diversas técnicas que foram aplicadas nesta pesquisa, não foram exauridas as diferentes combinações e experimentações devido ao tamanho da amostras, e nuances que podem ser referentes a particularidades dos documentos estudados. Diferentes modelos de classificação também podem ser avaliados, arquiteturas como a de transformadores como o *BERT*. É possível que novas amostras sejam classificadas a medida que mais interesse for demonstrado nesse tipo de pesquisa, aumentando o escasso número de amostras classificadas como possuindo risco de conluio com relação ao total.

Referências

[1] Lima, Marcos Cavalcanti: *Deep vacuity: detecção e classificação automática de padrões com risco de conluio em dados públicos de licitações de obras.* 2021. 3, 10

[2] Jurafsky, Dan e James H. Martin: *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition.* Pearson Prentice Hall, Upper Saddle River, N.J., 2009, ISBN 9780131873216 0131873210. http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y. 3

[3] Lima, Marcos, Roberta Silva, Felipe Lopes de Souza Mendes, Leonardo R. de Carvalho, Aleteia Araujo e Flavio de Barros Vidal: *Inferring about fraudulent collusion risk on Brazilian public works contracts in official texts using a Bi-LSTM approach.* Em *Findings of the Association for Computational Linguistics: EMNLP 2020*, páginas 1580–1588, Online, novembro 2020. Association for Computational Linguistics. https://aclanthology.org/2020.findings-emnlp.143. 3

[4] CADE: *Combate a cartéis em licitações. guia prático para pregoeiros e membros de comissões de licitação*, 2008. 3

[5] Goldberg, Yoav: *Neural network methods for natural language processing.* Synthesis lectures on human language technologies, 10(1):1–309, 2017. 6

[6] Gupta, Gaurav e Sumit Malhotra: *Text Document Tokenization for Word Frequency Count using Rapid Miner (Taking Resume as an Example).* março 2015. 7

[7] Straka, Milan e Jana Strakova: *Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe.* páginas 88–99, janeiro 2017. 7

[8] Wilbur, W. John e Karl Sirotkin: *The automatic identification of stop words.* Journal of Information Science, 18(1):45–55, 1992. 7

[9] Brown, Eric W. e Anni R. Coden: *Capitalization Recovery for Text.* Em Goos, Gerhard, Juris Hartmanis, Jan van Leeuwen, Anni R. Coden, Eric W. Brown e Savitha Srinivasan (editores): *Information Retrieval Techniques for Speech Applications*, volume 2273, páginas 11–22. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, ISBN 978-3-540-43156-5 978-3-540-45637-7. http://link.springer.com/10.1007/3-540-45637-6_2, acesso em 2021-03-03, Series Title: Lecture Notes in Computer Science. 7

[10] Imran, Muhammad, Prasenjit Mitra e Carlos Castillo: *Twitter as a Lifeline: Human-annotated Twitter Corpora for NLP of Crisis-related Messages*. arXiv:1605.05894 [cs], maio 2016. http://arxiv.org/abs/1605.05894, acesso em 2021-03-03, arXiv: 1605.05894. 8

[11] Bard, Gregory V: *Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric*. Cryptology ePrint Archive, 2006. 8

[12] Korenius, Tuomo, Jorma Laurikkala, Kalervo Järvelin e Martti Juhola: *Stemming and lemmatization in the clustering of finnish text documents*. Em *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, páginas 625–633, 2004. 8

[13] Singh, Jasmeet e Vishal Gupta: *Text stemming: Approaches, applications, and challenges*. ACM Comput. Surv., 49(3), setembro 2016, ISSN 0360-0300. https://doi.org/10.1145/2975608. 9

[14] Singh, Jasmeet e Vishal Gupta: *Text Stemming: Approaches, Applications, and Challenges*. ACM Computing Surveys, 49(3):1–46, dezembro 2016, ISSN 0360-0300, 1557-7341. https://dl.acm.org/doi/10.1145/2975608, acesso em 2021-01-09. 9

[15] Mikolov, Tomas, Kai Chen, Greg Corrado e Jeffrey Dean: *Efficient estimation of word representations in vector space*, 2013. 9, 10

[16] Bramer, Max: *Principles of data mining*, volume 180. Springer, 2007. 10

[17] Manning, Christopher D., Prabhakar Raghavan e Hinrich Schutze: *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008, ISBN 0521865719. 10, 11

[18] Smola, Alex J. e Bernhard Schölkopf: *A tutorial on support vector regression*, 2004. 11

[19] Bottou, Léon: *Stochastic gradient descent tricks*. Em *Neural networks: Tricks of the trade*, páginas 421–436. Springer, 2012. 11

[20] Bottou, Léon: *Large-Scale Machine Learning with Stochastic Gradient Descent*. Em Lechevallier, Yves e Gilbert Saporta (editores): *Proceedings of COMPSTAT'2010*, páginas 177–186. Physica-Verlag HD, Heidelberg, 2010, ISBN 978-3-7908-2603-6 978-3-7908-2604-3. http://link.springer.com/10.1007/978-3-7908-2604-3_16, acesso em 2021-04-12. 12

[21] Loh, Wei-Yin: *Classification and regression trees*. WIREs Data Mining and Knowledge Discovery, 1(1):14–23, janeiro 2011, ISSN 1942-4787, 1942-4795. https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.8, acesso em 2021-04-12. 12, 13

[22] Song, Yan Yan e Ying Lu: *Decision tree methods: applications for classification and prediction*. Shanghai Archives of Psychiatry, 27(2):130–135, abril 2015, ISSN 1002-0829. 14

[23] Cutler, Adele, D Richard Cutler e John R Stevens: *Random forests*. Em *Ensemble machine learning*, páginas 157–175. Springer, 2012. 14

[24] Rifkin, Ryan M e Ross A Lippert: *Notes on regularized least squares*. 2007. 14

[25] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011. 15, 20

[26] Rennie, Jason D, Lawrence Shih, Jaime Teevan e David R Karger: *Tackling the poor assumptions of naive bayes text classifiers*. Em *Proceedings of the 20th international conference on machine learning (ICML-03)*, páginas 616–623, 2003. 17

[27] Hastie, Trevor, Saharon Rosset, Ji Zhu e Hui Zou: *Multi-class adaboost*. Statistics and its Interface, 2(3):349–360, 2009. 17

[28] Hastie, Trevor, Robert Tibshirani e Jerome Friedman: *Boosting and additive trees*. Em *The elements of statistical learning*, páginas 337–387. Springer, 2009. 18

[29] Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho *et al.*: *Xgboost: extreme gradient boosting*. R package version 0.4-2, 1(4):1–4, 2015. 18

[30] Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz e Yoram Singer: *Online passive aggressive algorithms*. 2006. 18

[31] Deng, Li e Dong Yu: *Deep learning: Methods and applications*. Relatório Técnico MSR-TR-2014-21, Microsoft, May 2014. `https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/`. 19

[32] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep learning*. MIT press, 2016. 19, 20, 21, 23, 24, 25

[33] Vázquez, Favio: *Deep Learning made easy with Deep Cognition*. `https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351`, acesso em 2021-02-08. 21, 24

[34] Eliasy, Ashkan e Justyna Przychodzen: *The role of ai in capital structure to enhance corporate funding strategies*. Array, 6:100017, julho 2020. 22

[35] Liu, Shujie, Nan Yang, Mu Li e Ming Zhou: *A recursive recurrent neural network for statistical machine translation*. Em *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 1491–1500, Baltimore, Maryland, junho 2014. Association for Computational Linguistics. `https://aclanthology.org/P14-1140`. 22

[36] Dey, Rahul e Fathi M. Salem: *Gate-variants of gated recurrent unit (gru) neural networks*. Em *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, páginas 1597–1600, Aug 2017. 22

[37] su, Yuanhang e C. Kuo: *On extended long short-term memory and dependent bidirectional recurrent neural network*. Neurocomputing, 356, fevereiro 2018. 23

[38] Alom, Md. Zahangir, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal e Vijayan Asari: *A state-of-the-art survey on deep learning theory and architectures*. Electronics, 8:292, março 2019. 26

[39] Browne, Michael W: *Cross-validation methods*. Journal of Mathematical Psychology, 44(1):108–132, 2000, ISSN 0022-2496. `https://www.sciencedirect.com/science/article/pii/S0022249699912798`. 26

[40] Powers, David MW: *Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation*. arXiv preprint arXiv:2010.16061, 2020. 27

[41] Kowsari, Kamran, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes e Donald Brown: *Text classification algorithms: A survey*. Information, 10(4), 2019, ISSN 2078-2489. `https://www.mdpi.com/2078-2489/10/4/150`. 30

[42] Joulin, Armand, Edouard Grave, Piotr Bojanowski e Tomás Mikolov: *Bag of tricks for efficient text classification*. CoRR, abs/1607.01759, 2016. `http://arxiv.org/abs/1607.01759`. 30

[43] Claire Nédellec, Céline Rouveirol (editor): *Machine Learning: ECML-98, 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*. 1998. `https://app.dimensions.ai/details/publication/pub.1108495092`. 30, 32

[44] Yang, Yiming e Jan O Pedersen: *A comparative study on feature selection in text categorization*. Em *Icml*, volume 97, páginas 412–420. Nashville, TN, USA, 1997. 30, 33

[45] Lewis, David D e Marc Ringuette: *A comparison of two learning algorithms for text categorization*. Em *Third annual symposium on document analysis and information retrieval*, volume 33, páginas 81–93, 1994. 30

[46] Zhou, Peng, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao e Bo Xu: *Text classification improved by integrating bidirectional lstm with two-dimensional max pooling*, 2016. 31

[47] Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng e Christopher Potts: *Recursive deep models for semantic compositionality over a sentiment treebank*. Em *Proceedings of the 2013 conference on empirical methods in natural language processing*, páginas 1631–1642, 2013. 31

[48] Pang, Bo e Lillian Lee: *A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts*. arXiv preprint cs/0409058, 2004. 32

[49] Li, Xin e Dan Roth: *Learning question classifiers.* Em *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. 32

[50] Hingmire, Swapnil, Sandeep Chougule, Girish K Palshikar e Sutanu Chakraborti: *Document classification by topic labeling.* Em *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, páginas 877–880, 2013. 32

[51] Kim, Hyunsoo, Peg Howland, Haesun Park e Nello Christianini: *Dimension reduction in text classification with support vector machines.* Journal of machine learning research, 6(1), 2005. 32

[52] Lima, Marcos Cavalcanti: *Deep Vacuity: Detecção e Classificação Automática de Padrões com Risco de Conluio em Dados Públicos de Licitações de Obras.* página 126. 33, 34, 37

[53] Musskopf, Irio: *Operação Serenata de Amor*, 2018. `https://serenata.ai/about/`, acesso em 2021-08-19. 35

[54] Big, Lucio: *OPS :: Operação Política Supervisionada*, 2018. `https://www.ops.net.br/`, acesso em 2021-08-19. 35

[55] *Observatório da Despesa Pública.* `https://www.gov.br/cgu/pt-br/assuntos/informacoes-estrategicas/observatorio-da-despesa-publica/observatorio-da-despesa-publica`, acesso em 2021-08-19. 35

[56] Vallim, Joao José de Castro Baptista: *Uso do modelo de raciocínio baseado em casos para monitoramento de conluio em licitações de obras de pavimentação urbana.* 2020. 37

[57] Ghedini Ralha, Célia e Carlos Vinícius Sarmento Silva: *A multi-agent data mining system for cartel detection in brazilian government procurement.* Expert Systems with Applications, 39(14):11642–11656, 2012, ISSN 0957-4174. `https://www.sciencedirect.com/science/article/pii/S0957417412006343`. 37

[58] Balaniuk, Remis, Pierre Bessiere, Emmanuel Mazer e Paulo Cobbe: *Risk based Government Audit Planning using Naïve Bayes Classifiers.* Em *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, Spain, 2012. `https://hal.archives-ouvertes.fr/hal-00746198`. 37

[59] Sun, Ting e Leonardo J Sales: *Predicting public procurement irregularity: An application of neural networks.* Journal of Emerging Technologies in Accounting, 15(1):141–154, 2018. 38

[60] Honnibal, Matthew e Ines Montani: *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.* To appear, 2017. 40

Anexo I

Métricas das Melhores Versões dos Modelos de Classificação

Modelo	F1 Score	Média F1 Score	Desvio Padrão
DNN	0.931759	0.918195	0.008582
Ridge Classifier	0.927419	0.915813	0.006002
Passive Aggressive Classifier	0.928947	0.910638	0.009429
SGB SVM	0.932115	0.909085	0.009629
RBF SVM	0.918121	0.906654	0.006078
Neural Net	0.920882	0.905330	0.008396
Multinomial NB	0.906736	0.895415	0.006212
Gradient Boosting Classifier	0.909814	0.886195	0.010339
RCNN	0.893035	0.877314	0.010745
CNN	0.904701	0.872369	0.022620
XGBClassifier	0.886212	0.870653	0.008085
RNN	0.878788	0.847796	0.010885
AdaBoost	0.850129	0.833190	0.007469
Linear SVM	0.852459	0.816911	0.013189
Decision Tree	0.783311	0.737690	0.028552
Nearest Neighbors	0.612319	0.603052	0.005698
Random Forest	0.670203	0.302980	0.249705