UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI

Christian Fernandes de Deus

Análise de textos de literatura ficcional por meio de métodos computacionais

São João Del Rei
2023

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI

Christian Fernandes de Deus

Avaliação de métodos computacionais para identificação de técnicas na literatura ficcional

Monografia apresentada como requisito da disciplina de Projeto orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Orientador: Flávio Luiz Schiavoni

Universidade Federal de São João del-Rei — UFSJ
Bacharelado em Ciência da Computação

São João Del Rei
2023

Christian Fernandes de Deus

Análise de textos de literatura ficcional por meio de métodos computacionais

Monografia apresentada como requisito da disciplina de Projeto orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Trabalho aprovado. São João Del Rei, 30 de junho de 2023:

Flávio Luiz Schiavoni

Edimilson Batista dos Santos

Convidado 1

Convidado 2

Convidado 2

São João Del Rei
2023

Dedico este trabalho à minha família, amigos e a todas as pessoas que me apoiaram nos momentos difíceis.

Agradecimentos

Agradeço imensamente à minha família que sempre contribuiu para o meu crescimento e me deu todo o apoio para que eu continuasse mesmo quando a desistência parecia o melhor caminho. Agradeço aos meus amigos, em especial ao Carlos Roberto e ao DCS, mais conhecido como Gustavo Silva, que sempre estiveram presentes nos momentos que eu mais precisei. Por fim, agradeço ao meu orientador Flávio Luiz pela paciência e orientação durante todo esse processo e ao professor Elverton Carvalho pelo incentivo.

Não quero ter a terrível limitação de quem vive apenas do que é passível de fazer sentido. Eu não: quero é uma verdade inventada.

(Clarice Lispector)

Resumo

A arte sempre foi uma maneira dos seres humanos se expressarem e a literatura como uma obra artística, trilha o mesmo caminho. Para tentar alcançar uma obra artística literária de qualidade, diversas técnicas de escrita foram desenvolvidas, mas existe mesmo uma maneira de qualificar uma obra? Utilizando ferramentas computacionais, como a análise de sentimentos e redes neurais, este estudo busca identificar padrões e técnicas distintivas que diferenciam obras literárias de alta qualidade de outras supostamente inferiores. Com o entendimento de que definições de arte e avaliações de sua qualidade são altamente subjetivas e dependem de fatores como percepção, emoção e contexto sociocultural, a literatura foi escolhida como objeto deste estudo devido à sua rica gama de expressões e possibilidades. Assim, foram usadas obras literárias clássicas, reconhecidas pela sua excelência, como exemplos de bons textos técnicos, e, em contraste, textos do próprio autor, cuja qualidade ainda não foi comprovada. Ao final do projeto, foi possível concluir que apesar de existirem algumas semelhanças, fica incerto o julgamento da qualidade por meio das técnicas agregadas. Além de deixar claro uma lacuna notável nos estudos de figuras de linguagem em português, reforçando a necessidade de pesquisa contínua nessa área.

Palavras-chave: literatura, redes neurais, arte, análise computacional, técnicas literárias

Abstract

Art has always been a way for human beings to express themselves, and literature, as an art form, follows the same path. In an attempt to achieve high-quality artistic work, various writing techniques have been developed, but is there truly a way to qualify a work? Utilizing computational tools, such as sentiment analysis and neural networks, this study seeks to identify distinctive patterns and techniques that differentiate high-quality literary works from others that are supposedly inferior. With the understanding that definitions of art and assessments of its quality are highly subjective and depend on factors such as perception, emotion, and sociocultural context, literature was chosen as the object of this study due to its rich range of expressions and possibilities. Thus, classic literary works, recognized for their excellence, were used as examples of good technical texts, and in contrast, texts by the author himself, whose quality has not yet been proven.

Keywords: literature, neural networks, art, computational analysis, literary techniques

Lista de abreviaturas e siglas

BERT	Bidirectional Encoder Representations from Transformers
CSV	Comma Separated Values
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
VADER	Valence Aware Dictionary and sEntiment Reasoner

Sumário

1. Introdução.....11
2. Conceitos Envolvidos.....14
2.1 Redes Neurais..... 14
2.2 Redes Neurais Profundas (Deep Learning).....15
2.3 BERT (Bidirectional Encoder Representations from Transformers)..... 16
2.4 Análise de Sentimentos.....17
2.5 Figuras de Linguagem.....17
3. Proposta..... 18
3.1 Análises Estatísticas.....18
3.2 Análises de Sentimentos..... 19
3.3 Rede Neural.....20
4. Implementação..... 22
4.1 Algoritmos Estatísticos..... 22
4.1.1 Algoritmo de Frequência Léxica Lematizada.....22
4.1.2 Algoritmo de Contagem de Palavras Únicas..... 25
4.1.3 Algoritmo de Análise do Tamanho Médio das Frases..... 27
4.2 Algoritmos de Análise de Sentimento..... 28
4.3 Rede Neural para Identificação de Figuras de Linguagem.....29
4.4 Treinamento do Modelo.....33
5. Resultados e Discussões..... 35
5.1 Algoritmo de Contagem de Palavras Únicas..... 35
5.2 Algoritmo de Frequência Léxica Lematizada.....39
5.3 Algoritmo de Frequência de Palavras na Frase.....42
5.4 Algoritmo de Análise de Sentimentos..... 44
5.5 Rede Neural para Identificação de Figuras de Linguagem.....48
6. Conclusões.....53
7. Referências.....55

1. Introdução

Desde os primórdios da humanidade a arte e o simbolismo sempre fizeram parte da cultura mundial no que parece uma necessidade incessante de se expressar. Começando nas pinturas rupestres e traçando uma jornada até o pós-modernismo, diversos autores nesse tempo tentaram sistematizar o formato e a estética da arte como, por exemplo, Aristóteles em seu livro Poética, que mesmo escrita aos moldes das peças gregas, continua sendo uma obra importante na literatura contemporânea.

Mesmo com todo o esforço e anos de pesquisas visando teorizar a arte, ainda não há consenso sobre sua estética e essência. Kant, por exemplo, nos diz que "aquilo que é puramente subjetivo na representação de um objeto, isto é, o que constitui a sua relação ao sujeito, e não ao objeto, é a sua qualidade estética" [1]. Enquanto outros trabalhos, como o do Adorno, focam na parte objetiva da arte, dizendo que "a arte responsável orienta-se por critérios que se aproximam muito dos do conhecimento: o lógico e o ilógico, o verdadeiro e o falso." [2].

Apesar das inúmeras definições propostas na literatura, é difícil afirmar categoricamente qual é a correta, uma vez que a arte é uma experiência multifacetada, complexa e profundamente individualizada, que envolve a interseção de inúmeros fatores como a percepção, a emoção e o contexto sociocultural. Essa pluralidade de visões sobre a arte é o que torna a sua análise tão rica e profundamente interessante. E ainda que fiquemos munidos de definições teóricas sobre as definições de arte, algumas perguntas ainda ficam permeando em nossa mente: O que é uma boa arte? O que é necessário para se criar uma boa arte?

A tentativa de responder estas perguntas pode nos levar a uma análise mais profunda da arte e seus valores. As artes visuais, por exemplo, podem ser julgadas por vários critérios e, historicamente, muitos padrões têm sido usados para avaliar a qualidade artística. No entanto, dois componentes emergem quando a discussão é qualidade artística: técnica e originalidade. Ambos os critérios têm desempenhado papéis importantes na história da estética, mas a questão é: quanto [3]? Este projeto foi feito baseado nessa premissa: como a tecnologia computacional pode ajudar a encontrar e mensurar técnicas na arte que podem auxiliar a pautar a qualidade e o impacto de uma obra artística.

Neste contexto, de todas as formas de arte, a literatura foi optada como objeto de nosso estudo. Como uma das minhas formas de arte preferidas, ela oferece um vasto campo

para a exploração da interseção entre técnica e originalidade. A profundidade e a complexidade da linguagem escrita, com suas infinitas combinações de palavras e ideias, fornecem um terreno fértil para o desenvolvimento de técnicas inovadoras e poderosas. Além de ser primorosa no sentido de que ela é capaz de capturar e comunicar a experiência humana de maneira particularmente profunda e matizada.

Em minha perspectiva pessoal como escritor, acredito que as técnicas literárias desempenham um papel crucial na criação de uma obra literária de qualidade. Elas oferecem ao escritor uma série de ferramentas que podem ser usadas para ajudá-lo a estruturar melhor o seu trabalho, permitindo que ele comunique suas ideias, emoções e observações de maneira clara, vívida e envolvente. E ser capaz de encontrar de maneira objetiva quais são estas técnicas, ou ainda, quais as técnicas mais efetivas, seria um grande passo para acelerar tanto o aprendizado, quanto para verificar o que a crítica especializada considera bom.

Essa tarefa de definir parâmetros para a avaliação de um texto literário é sem dúvida desafiadora, pois exige que lidemos com uma série de questões complexas. Por exemplo, como podemos definir e medir a "qualidade" literária de forma quantitativa e objetiva? Como podemos ensinar um programa de computador a entender e apreciar as nuances da linguagem humana?

Essas questões exigem uma abordagem interdisciplinar incluindo ciência da computação para o desenvolvimento da proposta de soluções computacionais, linguística para o entendimento das nuances da língua, literatura que envolve todo o objeto de estudo que são os textos ficcionais e filosofia para a concepção teórica dos conceitos de arte, beleza e qualidade.

Neste sentido, o potencial de um programa de computador para analisar a técnica literária é uma ideia intrigante, que pode abrir novos caminhos para a crítica literária e a teoria da arte, assim como provocar discussões significativas sobre o papel e o impacto da tecnologia em nossa cultura.

Sabendo da existência de diversas concepções teóricas para o que é arte de qualidade, será considerado como exemplo de textos com boas técnicas aqueles que são reconhecidos como clássicos, como por exemplo: Machado de Assis. Essas obras resistiram ao crivo do tempo e provaram seu valor artístico e literário por gerações. Seja pelas mensagens que carregam, pela originalidade de suas ideias, ou pelo domínio magistral da linguagem, estes textos foram saudados como marcos da literatura e continuam a influenciar escritores e

leitores até hoje. Sua reputação duradoura e sua aceitação pelos críticos literários os tornam exemplos ideais para a investigação da técnica literária.

Em contraponto aos textos de qualidade e excelência, também foi necessário a utilização de textos que não tem sua qualidade ainda comprovada. Para evitar qualquer tipo de conflito e para fim de comparação com os textos aclamados, foram considerados textos com baixo nível técnico: os meus próprios textos.

Portanto, tem-se que o objetivo deste trabalho foi pautado em testar diversos métodos para a avaliação, como análise de sentimentos, análises estatísticas e rede neural, de textos tecnicamente bons e que não estão presentes ou são diferentes nos textos de qualidade supostamente inferiores.

Para extrair e entender as técnicas utilizadas nesses textos, recorri a várias abordagens computacionais como: o uso da análise de sentimentos, técnica esta que permite extrair e quantificar as emoções expressas em cada texto, proporcionando uma compreensão mais profunda da atmosfera, do tom e da carga emocional de cada obra. A utilização de redes neurais; modelos de aprendizado de máquina inspirados na estrutura e funcionamento do cérebro humano que são capazes de aprender e identificar padrões complexos em grandes volumes de dados. E finalmente, também foi feita uma análise de caracteres em frases e parágrafos, bem como na identificação de palavras repetidas e outros padrões textuais. Essas abordagens permitem estudar aspectos mais sutis e específicos do estilo literário, como a ritmicidade e a densidade lexical.

2. Conceitos Envolvidos

A fim de padronizar alguns conhecimentos que serão utilizados ao longo deste trabalho, este capítulo traz os conceitos envolvidos no mesmo.

2.1 Redes Neurais

Redes neurais artificiais, ou simplesmente redes neurais, são um tipo de modelo de aprendizado de máquina inspirado pelo sistema nervoso biológico. Como mostra a Figura 1, a estrutura de uma rede neural é composta de nós, conhecidos como neurônios artificiais, organizados em camadas. Cada neurônio em uma rede neural está conectado a vários outros neurônios, e essas conexões têm pesos associados a elas [4].

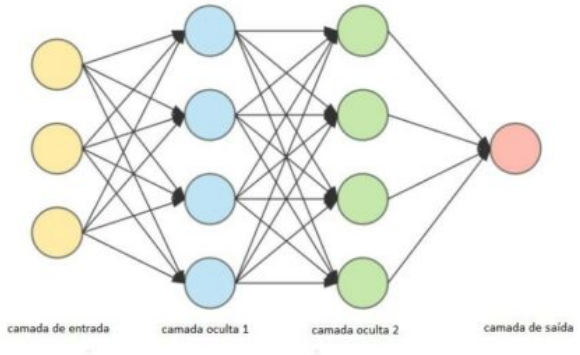

Figura 1: Modelo simplificado de rede neural

Os neurônios recebem entradas de outros neurônios ou, caso seja neurônios de entrada, recebem os dados fornecidos. Logo após, cada entrada é multiplicada pelo peso da conexão correspondente. Em seguida, esses resultados são somados e esse somatório é enviado para uma função de ativação para produzir a saída do neurônio [4].

Para que a rede neural seja capaz de receber entradas e retornar uma saída satisfatória, ela precisa ser treinada [5].

O treinamento de uma rede neural envolve ajustar seus pesos de forma a minimizar a diferença, ou erro, entre as previsões que a rede faz e os valores verdadeiros que ela deveria prever. Isso é feito através de um processo de otimização iterativa que procura encontrar os valores de peso que minimizam a função de perda, uma medida quantitativa do erro de previsão da rede [6].

Este processo de otimização é geralmente realizado utilizando um algoritmo chamado gradiente descendente. A ideia por trás do gradiente descendente é simples: iniciar com um conjunto de pesos aleatórios, calcular o gradiente da função de perda em relação a esses pesos, e então ajustar os pesos em uma quantidade proporcional ao negativo do gradiente. Isso é repetido até que a função de perda pare de diminuir significativamente, ou até que um número máximo de iterações seja atingido [7].

Na aprendizagem de máquina, o gradiente de uma função de perda em um dado ponto é um vetor que aponta na direção em que a função tem a maior taxa de aumento. Em outras palavras, o gradiente indica a "subida" mais íngreme da função de perda. Para minimizá-la, queremos ir na direção oposta, ou seja, a "descida" mais íngreme, daí o termo "gradiente descendente" [8].

O algoritmo de *backpropagation* (ou retropropagação) entra nesse processo como uma forma eficiente de calcular os gradientes necessários para a otimização por gradiente descendente em redes neurais. A retropropagação faz isso aplicando a regra da cadeia do cálculo, uma técnica para calcular a derivada de uma função composta, em várias etapas. Em particular, ela calcula o gradiente da função de perda em relação a cada peso na rede, começando com os pesos nas camadas mais próximas à saída e trabalhando para trás através da rede.

Depois que todos os gradientes são calculados pela retropropagação, eles são usados para atualizar os pesos da rede no passo do gradiente descendente. Este processo de calcular os gradientes via retropropagação e depois atualizar os pesos via gradiente descendente é repetido muitas vezes, a cada vez usando um diferente mini-lote dos dados de treinamento, até que a rede esteja adequadamente treinada.

2.2 Redes Neurais Profundas (*Deep Learning*)

Redes neurais profundas são uma extensão das redes neurais que contêm várias camadas ocultas entre a camada de entrada e a camada de saída, como pode ser observado na Figura 2. Essas camadas adicionais permitem que a rede aprenda características mais complexas e abstratas dos dados.

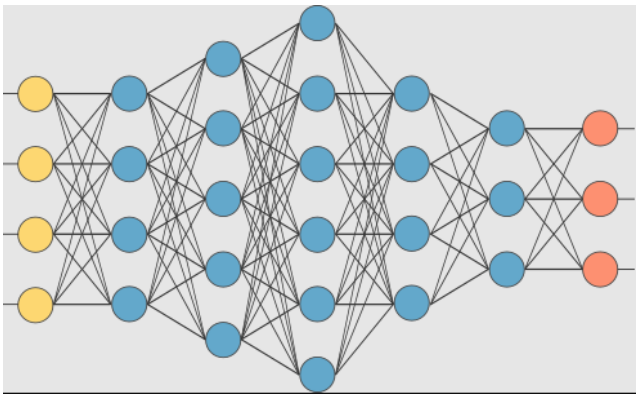

Figura 2: Modelo de rede neural profunda

Por exemplo, em uma rede neural profunda treinada para reconhecimento de imagens, as primeiras camadas podem aprender a reconhecer bordas, as camadas intermediárias podem aprender a reconhecer formas básicas combinando bordas, e as últimas camadas podem aprender a reconhecer partes de objetos combinando formas básicas [9].

As redes neurais profundas têm a capacidade de aprender automaticamente e melhorar com a experiência, e têm demonstrado um desempenho impressionante em uma variedade de tarefas, especialmente naquelas que envolvem dados não estruturados, como imagens ou texto.

2.3 BERT (Bidirectional Encoder Representations from Transformers)

BERT é um modelo de rede neural profunda especificamente projetado para processamento de linguagem natural, que é o campo de estudo focado na interação entre computadores e linguagem humana. Desenvolvido pela equipe do Google AI, o BERT é pré-treinado em um grande corpus de texto e depois pode ser afinado para uma variedade de tarefas específicas de processamento de linguagem natural [10].

A inovação chave do BERT é a maneira como ele lida com o contexto em linguagem. Em vez de analisar o texto linearmente (da esquerda para a direita ou da direita para a esquerda), o BERT é capaz de analisar o texto bidirecionalmente. Isso significa que para entender o significado de uma palavra específica, ele considera todas as palavras à sua volta [10].

2.4 Análise de Sentimentos

A Análise de Sentimentos é uma subárea da Inteligência Artificial (IA) que se concentra no estudo das opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação a entidades como produtos, serviços, organizações, indivíduos, questões, eventos, tópicos e seus atributos. Esta disciplina, também conhecida como Mineração de Opiniões, utiliza técnicas de NLP, aprendizado de máquina e análise estatística para extrair e identificar informações subjetivas dos dados [11].

Os métodos empregados podem ser divididos em dois grupos principais: métodos baseados em léxico e métodos baseados em aprendizado de máquina. Os primeiros aproveitam listas predefinidas de palavras associadas a sentimentos positivos ou negativos. Os segundos, por outro lado, aprendem essas associações a partir de dados previamente anotados. No entanto, ambos os métodos enfrentam desafios comuns no domínio da análise de sentimentos, como a detecção de ironia e sarcasmo, a compreensão da linguagem figurativa e a análise do sentimento em contextos específicos [11].

2.5 Figuras de Linguagem

Figuras de linguagem, também conhecidas como figuras de estilo, são expressões que transcendem o significado literal das palavras, mudam a ordem normal das palavras na frase ou são produzidas por padrões sonoros [12]. Elas são amplamente utilizadas na literatura, publicidade, discursos políticos e conversas cotidianas.

As figuras de linguagem incluem metáforas, simbolismos, hipérboles, personificações, entre outras, e são categorizadas em termos de semelhança, contraste ou ênfase [12]. Elas podem ser classificadas em grupos como, por exemplo: figuras de sintaxe e figuras de semântica.

As Figuras de Sintaxe ou de construção, se caracterizam por uma disposição especial das palavras, frases ou estruturas gramaticais para destacar uma mensagem. As principais figuras de sintaxe incluem anáfora, elipse, zeugma, silepse, polissíndeto, entre outras.

Por outro lado, as Figuras de Semântica ou de pensamento, estão relacionadas ao significado das palavras e frases. Elas utilizam palavras e frases de maneiras que vão além do seu sentido literal para criar uma linguagem mais evocativa e expressiva. Exemplos conhecidos de figuras de semântica são: metáfora, metonímia, sinédoque, antítese, hipérbole, eufemismo, entre outras.

3. Proposta

A proposta deste projeto ocorreu por meio de um processo iterativo e gradual, envolvendo a aplicação de diversas técnicas de NLP e aprendizado de máquina. As técnicas aplicadas foram selecionadas cuidadosamente com base em sua relevância para a análise literária e na capacidade de contribuir para o objetivo central deste trabalho: testar diversos métodos para a avaliação da qualidade literária.

3.1 Análises Estatísticas

O projeto foi iniciado com a implementação de algoritmos de contagem de palavras. As técnicas de contagem de palavras, embora relativamente simples em sua concepção, são fundamentais para a análise textual. Fornecem uma visão quantitativa da composição do texto e, quando usadas de maneira eficaz, podem revelar características cruciais que podem ser indicativas da qualidade literária.

A primeira abordagem consistiu na identificação de palavras repetidas no texto. Este é um elemento crucial na análise de texto, pois a frequência com que uma palavra é usada pode indicar sua relevância no contexto do texto. No entanto, esta abordagem inicial não considera a forma da palavra, isto é, se ela aparece em diferentes conjugações ou formas derivadas. Por isso, foi incorporada uma segunda abordagem de contagem de palavras: a lematização.

A lematização é um processo linguístico que reduz uma palavra à sua forma de base ou raiz, independentemente de sua conjugação ou forma derivada. Por exemplo, palavras como 'correndo', 'correu' e 'corre' seriam todas reduzidas à sua forma base 'correr'. Esta abordagem, portanto, oferece uma visão mais precisa da frequência das palavras, uma vez que agrupa todas as formas variantes de uma palavra sob a mesma raiz.

No entanto, a repetição de palavras e a lematização por si só não são suficientes para avaliar a complexidade do texto ou a diversidade linguística do autor, características que são frequentemente consideradas indicadores de alta qualidade literária. Portanto, foi implementado um algoritmo para calcular o número de palavras únicas em um texto. Esta

medida, conhecida como riqueza lexical, pode fornecer uma visão valiosa da diversidade de vocabulário do autor e da complexidade do texto.

A implementação também incluiu a análise da estrutura textual, como o comprimento médio das frases e dos parágrafos. O comprimento das frases pode ser um indicativo do ritmo e fluidez do texto, enquanto o comprimento dos parágrafos pode indicar a profundidade e complexidade do pensamento do autor. A análise destas características estruturais do texto fornece insights importantes sobre o estilo do autor e, portanto, contribui para a avaliação da qualidade literária.

3.2 Análises de Sentimentos

Nesta implementação, o foco principal foi na análise de sentimentos realizada em três dimensões principais: ao nível do parágrafo, ao nível do texto como um todo e em segmentos de texto. Cada uma dessas abordagens proporcionou reflexões exclusivas e valiosas sobre a qualidade literária.

A análise de sentimentos a nível de parágrafo procura discernir a carga emocional presente em cada unidade distinta de pensamento expressa pelo autor. Este tipo de análise é fundamental para compreender as nuances da escrita do autor e como ele articula e manipula a emoção para direcionar a narrativa. Autores profissionais geralmente empregam uma variedade de técnicas para orquestrar as emoções do leitor, incluindo o uso deliberado de parágrafos com sentimentos contrastantes para criar tensão, impulsionar o envolvimento ou enfatizar pontos importantes. Assim, a análise de sentimentos no nível do parágrafo pode revelar muito sobre a habilidade do autor em modular a emoção e direcionar a experiência do leitor.

Já a análise de sentimentos em nível de texto fornece uma visão macro do tom emocional geral do texto. Este nível de análise é particularmente útil para entender o estado de espírito predominante ou a atmosfera que o autor visava instilar na obra. Por exemplo, uma análise de sentimentos em um romance trágico pode revelar um tom negativo predominante, enquanto uma comédia provavelmente apresentará um tom positivo mais forte. A análise do sentimento geral também pode nos dar uma visão sobre as estratégias emocionais do autor e se elas foram eficazes em manter a consistência emocional ao longo do trabalho.

A análise de sentimentos de segmentos de texto envolve a divisão do texto em partes iguais e a análise do sentimento de cada segmento. Tal abordagem permite mapear a progressão emocional em diferentes estágios da narrativa. Por exemplo, um autor pode

começar a história com um tom positivo, mergulhar em sentimentos negativos no meio, e finalmente retornar a um tom positivo no final. A análise de sentimentos segmentada permite a visualização dessa progressão emocional, proporcionando uma compreensão mais profunda da estrutura narrativa e da habilidade do autor em desenvolver a trama e os personagens.

Portanto, o uso dessas três dimensões de análise de sentimentos oferece uma visão abrangente e holística da qualidade literária. Além disso, esta metodologia permite um exame mais completo das estratégias emocionais empregadas pelo autor, permitindo uma avaliação mais rica da qualidade literária.

3.3 Rede Neural

As redes neurais desempenham um papel fundamental no campo do aprendizado de máquina e do NLP (Natural Language Processing). São especialmente úteis quando se lida com problemas complexos de classificação e previsão de sequência, tais como a detecção de figuras de linguagem em textos. Este projeto emprega uma variante específica de redes neurais chamada Transformador, mais precisamente o modelo BERT. Utilizei o BERTimbau, uma versão pré-treinada do BERT otimizada para o português, para construir um modelo capaz de identificar várias figuras de linguagem em textos.

A implementação do modelo passou por diversas etapas, cada uma aumentando a complexidade da tarefa. Iniciei criando um conjunto de dados focado em figuras de linguagem de **comparação**. Os textos utilizados neste conjunto foram escritos por mim mesmo manualmente. A relevância de iniciar por aqui se deve à recorrência desta figura nos textos, proporcionando uma base sólida para o treinamento do modelo. Este conjunto de dados foi utilizado para treinar o modelo inicial, com o objetivo de identificar se uma frase contém ou não essa figura de linguagem.

Com o sucesso da etapa inicial, a implementação prosseguiu para a inclusão de uma segunda figura de linguagem, a **metáfora**. A inclusão da metáfora neste estágio apresentou uma nova camada de complexidade para o modelo, uma vez que agora deveria distinguir entre duas figuras de linguagem diferentes, bem como identificar se uma frase não continha nenhuma delas. Para atingir esse objetivo, um novo conjunto de dados foi construído, composto por frases com comparações, metáforas e frases sem nenhuma figura de linguagem. Este conjunto de dados ampliado forneceu ao modelo um ambiente mais diversificado para aprender e generalizar, aumentando sua capacidade de identificar corretamente essas duas figuras de linguagem.

A terceira etapa representou um avanço significativo em termos de complexidade, com a inclusão de uma terceira figura de linguagem, a **hipérbole**, ao conjunto de dados. Isso significava que o modelo agora precisava ser capaz de distinguir entre três diferentes figuras de linguagem — **comparação**, **metáfora**, **hipérbole** — e ainda identificar frases que não continham nenhuma dessas figuras de linguagem. A adição da **hipérbole** ao conjunto de dados introduziu um novo nível de ambiguidade, exigindo um refinamento maior do modelo para alcançar um nível satisfatório de precisão.

Finalmente, a última etapa foi a inclusão de frases sem nenhuma figura de linguagem no conjunto de dados. A relevância dessa adição reside na capacidade do modelo de não apenas reconhecer as figuras de linguagem, mas também identificar corretamente quando elas não estão presentes. Este é um aspecto crucial para evitar falsos positivos e melhorar a eficiência geral do modelo.

4. Implementação

O escopo deste capítulo compreende a exposição detalhada das estratégias computacionais empregadas para análise textual nos experimentos conduzidos ao longo desta pesquisa que foram discutidos brevemente no capítulo de propostsa. As técnicas utilizadas para os algoritmos variam desde análises de frequência léxica e processos de lematização a treinamento de modelos de rede neurais.

4.1 Algoritmos Estatísticos

Algoritmos estatísticos, como os utilizados neste projeto, no contexto deste projeto, são um conjunto de procedimentos matemáticos que usam a teoria da probabilidade e a inferência estatística para extrair informações de dados numéricos. Eles foram usados para explorar e analisar as características dos dados textuais. Por exemplo, contagem de frequência de palavras, contagem de palavras únicas, contagem de palavras lematizadas, análise da distribuição de palavras e até cálculo da média de palavras por frase ou parágrafo.

4.1.1 Algoritmo de Frequência Léxica Lematizada

O primeiro algoritmo implementado, foca na quantificação da frequência léxica de palavras no texto. Ele processa um texto de entrada, divide-o em palavras individuais e conta a frequência de cada palavra. Sendo que a frequência de uma palavra é o número de vezes que ela aparece no texto. Este é um recurso útil na análise de texto, pois palavras que aparecem com mais frequência podem ter maior importância ou relevância.

Inicialmente, o texto é convertido para minúsculas e em seguida é realizado um processo de tokenização, segmentando o texto em unidades mínimas sem perda de sentido que são as palavras. Posteriormente, é efetuada a remoção de palavras consideradas de baixa relevância semântica para a análise, tais como artigos, pronomes e preposições, comumente conhecidas como *stop words*, como pode ser observado na Figura 3.

A remoção das *stop words* é um passo fundamental neste algoritmo já que o objetivo principal é verificar quais são as palavras mais utilizadas pelos autores. Essas palavras, caso não filtradas, acabam distorcendo as análises, pois o foco é encontrar palavras nos textos que possuem importância sintática e semântica relevantes.

```
# criar lista de palavras para remover
palavras_remover = ["um", "uma", "uns", "umas", "ele", "ela", "eles",
                    "elas", 'a','de', 'que', 'o', 'e', 'não', 'do', '–',
                    'da', 'é', 'os', 'se', 'em', 'com', 'as', 'para',
                    'mas', 'ao', 'por', 'no', 'era', 'na', 'à', 'como',
                    'mais', 'dos', 'ou', 'lhe', 'eu', 'me']
```

Figura 3: Lista de palavras removidas no algoritmo de frequência léxica

Finalmente, o algoritmo realiza a contagem de frequência das palavras remanescentes, retornando as mais frequentes. Este algoritmo de contagem de palavras pode ser particularmente útil no objetivo de comparar textos com diferentes qualidades técnicas. Ao comparar a frequência de palavras entre textos tecnicamente bons e não tão bons, é possível identificar padrões ou características que podem, talvez, indicar a qualidade técnica de um texto, como a maior ou menor utilização de palavras complexas.

Um exemplo hipotético pode ser que textos tecnicamente bons tendem a usar certas palavras com mais frequência, ou que a distribuição de palavras é mais equilibrada, indicando uma variedade maior de vocabulário e de estruturas de frases. Por outro lado, textos que não são tão tecnicamente bons podem mostrar uma super-representação de certas palavras ou uma distribuição de palavras mais desequilibrada.

O próximo passo é a lematização do algoritmo citado anteriormente. Um componente crucial na etapa de pré-processamento de dados de linguagem natural porque ajuda a reduzir o ruído nos dados e a consolidar diferentes formas de uma palavra em uma única representação, facilitando a análise subsequente. O uso da lematização foi especialmente útil na contagem de palavras para evitar a contagem de diferentes formas de uma palavra como entidades distintas.

O processo deste algoritmo foi realizado utilizando a biblioteca SpaCy[1], que é uma das várias bibliotecas para o processamento de NLP em Python. O SpaCy oferece suporte à lematização em português, que é essencial para este projeto.

Depois que o texto é dividido em palavras usando a função nlp do SpaCy, como pode ser observado na função *clean_text()*, disponível na Figura 4, as palavras são lematizadas utilizando a propriedade *lemma_* dos tokens gerados pelo SpaCy, como pode ser visto na mesma função. A função *nlp* divide uma string de texto em uma estrutura de dados especial

[1] Disponível em: https://spacy.io/

chamada Doc, composta por objetos do tipo *Token*, que correspondem às palavras individuais, o que é necessário para a lematização, pois esta função trabalha com uma palavra de cada vez.

Além disso, também são removidas as palavras que estão na lista de *stop words*, incluindo aquelas que a própria biblioteca *SpaCy* identifica como *stop words* através da propriedade *is_stop*. Estas palavras são normalmente preposições, artigos, entre outros, que são comuns na língua portuguesa, mas que normalmente não agregam muito valor à análise de texto.

```
def clean_text(text):
    text = text.lower().strip()
    doc = nlp(text)
    cleaned_text = [
        token.lemma_.strip() for token in doc
        if token.lemma_ not in STOP_WORDS and
        not token.is_stop and
        token.lemma_.strip() != ''
    ]
    return cleaned_text
```

Figura 4 : Utilização da função *clean_text()* para a limpeza do texto

Por fim, a função *analyze_vocabulary(),* como mostra a Figura 5, é usada para contar a frequência de cada palavra usando a classe *FreqDist* do NLTK (Natural Language Toolkit). As 10 palavras mais comuns são então visualizadas usando a biblioteca Matplotlib[2] para criar um gráfico de barras, e a biblioteca *WordCloud*[3] para criar uma nuvem de palavras. As 10 palavras mais frequentes também são exibidas no console para análise.

```
def analyze_vocabulary(cleaned_text):
    word_frequencies = nltk.FreqDist(cleaned_text)
    return word_frequencies
```

Figura 5: Função de análise do vocabulário

[2] Disponível em: https://matplotlib.org/
[3] Disponível em: http://amueller.github.io/word_cloud/.

A adição da lematização ao processo melhora a eficácia do algoritmo, pois ele agora leva em consideração as variações morfológicas das palavras e agrupa formas diferentes da mesma palavra. Isso fornece uma representação mais precisa da frequência de palavras, bem como melhora a qualidade dos dados que são alimentados nas etapas subsequentes de análise ou modelagem.

4.1.2 Algoritmo de Contagem de Palavras Únicas

Este algoritmo é um programa de análise completa do vocabulário de um conjunto de textos. Ele incorpora várias técnicas de pré-processamento e análise de texto, incluindo limpeza de texto, remoção de *stop words*, contagem de palavras, contagem de palavras únicas e visualização da frequência das palavras.

O algoritmo começa definindo um conjunto de *stop words* muito maior do que o algoritmo anterior, como pode ser visto na Figura 6, isso acontece por alguns motivos: na análise léxica lematizada, o objetivo é decompor o texto em seus componentes léxicos (ou tokens) e entender a estrutura gramatical. Portanto, é benéfico manter uma lista de *stop words* menor para evitar a remoção de palavras que possam ter relevância gramatical no texto.

```
STOP_WORDS = {'a', 'ao', 'aos', 'aquela', 'aquelas', 'aquele',
              'aqueles','aquilo', 'as', 'até', 'com', 'como',
              'da','das', 'de', 'dela', 'delas', 'dele', 'deles',
              'depois', 'do', 'dos', 'e', 'ela', 'elas', 'ele',
              'eles', 'em', 'entre', 'era', 'eram', 'essa', 'essas',
              'esse', 'esses', 'esta', 'estas', 'este', 'estes', 'eu',
             'foi', 'for', 'isso', 'isto', 'já', 'la', 'lá', 'lhe',
             'lhes', 'lo', 'mas', 'me', 'mesmo', 'meu', 'meus',
             'minha', 'minhas', 'muito', 'na', 'nas', 'nem', 'no',
             'nos', 'nós', 'num', 'numa', 'o', 'os', 'ou', 'para',
             'pela', 'pelas', 'pelo', 'pelos', 'por', 'qual', 'quando',
             'que', 'quem', 'se', 'sem', 'só', 'sua', 'suas','tal', 'te',
             'tem', 'têm', 'tinha', 'tinham', 'toda', 'todas', 'todo',
             'todos', 'tu', 'um', 'uma', 'é'}
```

Figura 6: Lista de palavras removidas no algoritmo de contagem de palavras

O algoritmo então define várias funções auxiliares. A primeira, *clean_text()*, mostrado na Figura 7, recebe uma lista de linhas de texto e realiza duas operações de limpeza: converte todas as letras para minúsculas e remove qualquer caractere que não seja uma letra ou um espaço. Esta é uma etapa de pré-processamento comum que ajuda a normalizar o texto e a reduzir o tamanho do vocabulário.

```
def clean_text(lines):
    cleaned_lines = []
    for line in lines:
        line = line.lower()
        line = re.sub(r"[^\w\s]", "", line)
        cleaned_lines.append(line)
    return cleaned_lines
```

Figura 7: Função que realiza a operação de limpeza

A função *analyze_vocabulary()*, evidenciado na Figura 8, recebe uma lista de linhas de texto e um conjunto de *stop words*, e retorna a contagem de palavras, a contagem de palavras únicas e as frequências de todas as palavras no texto. Primeiro, ele cria uma lista de todas as palavras no texto, em seguida, filtra essa lista para remover as *stop words*. Em seguida, calcula o número total de palavras, o número de palavras únicas (usando o operador set() para remover duplicatas) e as frequências de todas as palavras usando a classe *Counter* da biblioteca *collections*.

```
def analyze_vocabulary(lines, stop_words):
    words = []
    for line in lines:
        words.extend(line.split())

    filtered_words = [word for word in words if word not in stop_words]

    word_count = len(filtered_words)
    unique_word_count = len(set(filtered_words))
    word_frequencies = Counter(filtered_words)

    return word_count, unique_word_count, word_frequencies
```

Figura 8: Função de análise de vocabulário

As funções *visualize_word_frequencies()* e *create_word_cloud()* são usadas para visualizar as frequências das palavras, como mostra a Figura 9. A função *visualize_word_frequencies()* cria um gráfico de barras das n palavras mais comuns e suas frequências, enquanto *create_word_cloud()* cria uma nuvem de palavras, que é uma representação gráfica das palavras onde o tamanho de cada palavra é proporcional à sua frequência.

```
def visualize_word_frequencies(word_frequencies, n=10):
    common_words = word_frequencies.most_common(n)
    words, frequencies = zip(*common_words)

    plt.bar(words, frequencies)
    plt.xlabel("Palavras")
    plt.ylabel("Frequência")
    plt.title("Palavras mais frequentes")
    plt.show()

def create_word_cloud(word_frequencies):
    wordcloud = WordCloud(width=800, height=800, background_color="white").generate_from_frequencies(word_frequencies)

    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

Figura 9: Funções para visualizar as frequências de palavras e criação de gráficos

Por fim, a função *main()* itera sobre cada arquivo de texto, realiza a limpeza do texto, analisa o vocabulário e visualiza as frequências das palavras. Ele também calcula e imprime a "diversidade do vocabulário", que é o número de palavras únicas dividido pelo número total de palavras. Isso é uma medida de quão diversificado é o vocabulário do texto; um valor mais alto indica um vocabulário mais diversificado.

4.1.3 Algoritmo de Análise do Tamanho Médio das Frases

O algoritmo apresentado é um algoritmo simples para calcular a média de comprimento das frases em um dado texto. Ele também utiliza o pacote NLTK[4] para o processamento de linguagem natural. Logo após ele quebra o texto em frases utilizando a função *nltk.sent_tokenize()*. A função *sent_tokenize()* utiliza um algoritmo de tokenização de frases que segmenta um texto em frases.

[4] Disponível em: https://www.nltk.org/

Uma vez que o texto é dividido em frases, o próximo passo é calcular o comprimento de cada frase. Sendo que o comprimento de uma frase é medido pelo número de caracteres que ela contém. Este valor de comprimento é adicionado à lista de tamanhos.

Após isso, é calculada a média dos tamanhos de todas as frases do texto. Isso é feito somando todos os tamanhos na lista tamanhos e dividindo pelo número total de frases (que é o mesmo que o número de elementos na lista tamanhos). E ao final, a função retorna a média dos tamanhos das frases.

4.2 Algoritmos de Análise de Sentimento

A análise de sentimentos, como a utilizada neste projeto, é uma técnica de processamento de linguagem natural que usa algoritmos para identificar e extrair opiniões de um texto. Sendo útil para uma ampla gama de problemas que são de interesse para profissionais e pesquisadores de interação humano-computador, bem como para aqueles de áreas como sociologia, marketing e publicidade, psicologia, economia e ciência política [13].

Este projeto utiliza o VADER[5] (Valence Aware Dictionary and sEntiment Reasoner), um algoritmo de análise de sentimentos baseado em regras.

O algoritmo VADER é implementado no pacote NLTK e se destaca por utilizar um dicionário de sentimentos — ou "lexicon" — onde palavras são associadas a pontuações de intensidade de sentimentos. Além disso, o VADER é capaz de entender a intensificação de sentimentos por meio do uso de intensificadores (por exemplo, "muito bom" versus "bom") e do sentido de uma frase baseado na sua pontuação.

O algoritmo começa dividindo o texto em partes iguais, onde cada parte contém um número aproximadamente igual de frases. O objetivo dessa etapa é poder analisar a evolução do sentimento ao longo do texto. A função *dividir_texto*, como pode ser visto na Figura 10, realiza essa tarefa, usando a função *sent_tokenize* do NLTK para quebrar o texto em frases e, em seguida, agrupando as frases em partes.

[5] Disponível em: https://github.com/cjhutto/vaderSentiment

```python
def dividir_texto(texto, num_partes):
    sentencas = sent_tokenize(texto, language="portuguese")
    total_sentencas = len(sentencas)
    sentencas_por_parte = math.ceil(total_sentencas / num_partes)
    partes = []

    for i in range(0, total_sentencas, sentencas_por_parte):
        partes.append(" ".join(sentencas[i : i + sentencas_por_parte]))

    return partes
```

Figura 10: Função para quebrar o texto em frases

Em seguida, para cada parte do texto, uma função é usada para calcular a pontuação de polaridade de sentimento. A pontuação de polaridade é um número entre -1 (sentimento muito negativo) e +1 (sentimento muito positivo). Essas pontuações são, então, mapeadas para um rótulo de sentimento (por exemplo, "Muito negativo", "Negativo", "Neutro", "Positivo", "Muito positivo"), como pode-se observar na Figura 11.

```python
vader = SentimentIntensityAnalyzer()
sentimentos_label = [
    "Muito negativo",
    "Negativo",
    "Neutro",
    "Positivo",
    "Muito positivo",
]
```

Figura 11: Função de análise de sentimento e mapeamento

O código é finalizado com o algoritmo plotando a evolução do sentimento ao longo do texto, criando um gráfico de linha onde o eixo x representa a progressão ao longo do texto (em porcentagem) e o eixo y representa o sentimento.

4.3 Rede Neural para Identificação de Figuras de Linguagem

A classificação de texto é um aspecto importante do processamento de linguagem natural NLP que envolve a categorização de textos em grupos predeterminados. Para isso, um

algoritmo de aprendizado de máquina foi utilizado para realizar a classificação de figuras de linguagem nesses textos.

Este projeto implementa uma solução de classificação de texto com o modelo BERT[6]. A implementação começa definindo uma classe *FiguresOfSpeechDataset* que pode ser observada na Figura 12, que herda do Dataset do *PyTorch*[7], um framework popular de aprendizado de máquina.

Esta classe herda de *torch.utils.data.Dataset*, que é uma classe abstrata que representa um conjunto de dados. Ela implementa os métodos *__len__* e *__getitem__*, que são necessários para qualquer classe que herda do *Dataset* do *PyTorch*. O método *__init__* inicializa o objeto da classe com os dados, o tokenizador e o tamanho máximo de sequência. O método *__getitem__* é responsável por tokenizar o texto e formatá-lo adequadamente para ser alimentado ao modelo BERT.

```python
class FiguresOfSpeechDataset(Dataset):
    def __init__(self, data, tokenizer, max_len):
        self.data = data
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        row = self.data.loc[idx]
        text = row["frase"]
        label = row["label"]
        inputs = self.tokenizer(
            text, return_tensors="pt", padding="max_length", truncation=True, max_length=self.max_len)
        input_ids = inputs["input_ids"].squeeze()
        attention_mask = inputs["attention_mask"].squeeze()
        return {"input_ids": input_ids, "attention_mask": attention_mask, "label": label}
```

Figura 12: Classe responsável pela formatação e tokenização do texto

Logo após foi criada uma função responsável por criar um *DataLoader*, que é uma estrutura do *PyTorch* usada para carregar os dados em lotes, uma técnica comum em aprendizado de máquina para treinar modelos de maneira mais eficiente, vide Figura 13.

[6] Disponível em: https://github.com/google-research/bert
[7] Disponível em: https://pytorch.org/

```python
def create_data_loader(data, tokenizer, max_len, batch_size):
    dataset = FiguresOfSpeechDataset(data, tokenizer, max_len)
    return DataLoader(dataset, batch_size=batch_size, num_workers=0)
```

Figura 13: Função que cria o DataLoader

Logo após, o treinamento do modelo é iniciado. Ele é feito a partir da função *train_epoch*, mostrada na Figura 14. Como ela é a parte central do processo de treinamento do modelo de aprendizado profundo, vou falar dela mais detalhadamente.

```python
def train_epoch(model, data_loader, optimizer, scheduler, device):
    model = model.train()
    losses = []

    for batch in data_loader:
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["label"].to(device)

        optimizer.zero_grad()
        outputs = model(input_ids=input_ids,
                        attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        scheduler.step()

        losses.append(loss.item())

    return sum(losses) / len(losses)
```

Figura 14: Função de treinamento do modelo

A primeira linha, model = model.train(), coloca o modelo em modo de treinamento. Isso é importante porque certas camadas de um modelo, como Dropout e BatchNorm, se comportam de maneira diferente durante o treinamento e a inferência.

Em *losses = []*, inicializamos uma lista vazia para armazenar as perdas calculadas para cada lote de dados.

O próximo passo é a iteração através do *DataLoader*. Em cada iteração, um lote de dados é carregado.

Para a preparação dos dados para o modelo, temos a linha: *input_ids = batch["input_ids"].to(device)*. *input_ids* são essencialmente representações numéricas das palavras no texto. Eles são movidos para a GPU (se disponível) para acelerar o cálculo.

Em cada iteração, é importante zerar os gradientes dos pesos do modelo. E é isso que a linha *optimizer.zero_grad()* faz. Isso acontece porque o *PyTorch* acumula gradientes por padrão, ou seja, a cada vez que chamamos *.backward()*, os gradientes são somados em vez de substituídos.

Na linha *outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels),* os dados de entrada são alimentados através do modelo. A máscara de atenção é usada para ignorar os tokens de preenchimento na entrada.

O modelo retorna os outputs que contêm a perda com o seguinte código: *loss = outputs.loss*, que é calculada comparando as previsões do modelo com as verdadeiras etiquetas.

Na linha *loss.backward()* é o ponto onde a mágica do aprendizado profundo acontece. Este comando calcula os gradientes de perda em relação a todos os parâmetros do modelo que têm *requires_grad=True*. Ou seja, esta etapa é onde o *backpropagation* acontece.

O código *optimizer.step()* atualiza os parâmetros do modelo com base nos gradientes calculados na etapa anterior. O *scheduler.step()* ajusta a taxa de aprendizado do otimizador conforme definido durante a configuração do escalonador. E o *losses.append(loss.item())* adiciona a perda atual à lista de perdas.

No final da função, é retornada a perda média para a época, que é calculada somando todas as perdas e dividindo pelo número de lotes, ou seja, o comprimento de losses.

Continuando o código, tem-se logo após a função *evaluate*, como pode ser visto na Figura 15. Ela é usada para avaliar o desempenho do modelo treinado em um conjunto de dados de teste, calculando a proporção de previsões corretas em relação ao total de previsões.

```python
def evaluate(model, data_loader, device):
    model = model.eval()
    correct_predictions = 0
    total_predictions = 0

    with torch.no_grad():
        for batch in data_loader:
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["label"].to(device)

            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            _, preds = torch.max(outputs.logits, dim=1)

            correct_predictions += torch.sum(preds == labels)
            total_predictions += labels.size(0)

    return correct_predictions.double() / total_predictions
```

Figura 15: Função de avaliação de desempenho do modelo

O algoritmo então lê os dados de um arquivo CSV (Comma Separated Values) e os divide em um conjunto de treinamento e um conjunto de testes. Em seguida, ele define a configuração do dispositivo, carrega o tokenizador e o modelo BERT, cria os DataLoaders para os dados de treinamento e de teste, define o otimizador e o escalonador, treina o modelo e, finalmente, avalia o modelo treinado.

4.4 Treinamento do Modelo

Para treinar uma rede neural, é muito importante ter um conjunto de dados suficientemente grande. Para o treinamento do modelo de rede neural que fosse capaz de identificar figuras de linguagem, foi necessário a construção de uma base de dados com este propósito. A primeira base de dados criada foi uma base de dados com o objetivo de fazer a rede neural encontrar figuras de linguagem de **comparação**.

Para tal, foi necessário a criação de uma variedade de frases diferentes entre si que precisavam possuir uma diversidade de palavras utilizadas para comparar. Por exemplo, a frase “o amor é quente como o fogo” utiliza a palavra “como” para fazer a comparação, enquanto a frase “a gentileza dela era refrescante feito uma brisa suave em um dia quente”, utiliza a palavra “feito”. Essa diversidade pode ser estendida para diversas outras frases. Além

disso, frases que não são figuras de linguagem de comparação e utilizam palavras tipicamente comparativas também precisaram ser adicionadas no conjunto de dados, um exemplo é a frase: “como eu gostaria de ser aprovado!”

Garantir essa diversidade é fundamental para o aprendizado da rede neural, ajudando-a a reconhecer as diferentes maneiras pelas quais esses padrões de fala são expressos em texto natural.

Para uma boa base de dados, também é necessário que o conjunto de dados esteja balanceado, ou seja, que haja exemplos suficientes de cada tipo de frases que são figuras de linguagem de comparação e exemplos de frases que não são comparações. Ao final da construção, o conjunto de dados finalizou com um total de 851 frases, onde 421 frases não são figuras de linguagem e as outras 430 são figuras de linguagem de comparação.

Além da base de dados de **comparação**, também foi construída uma base de dados exclusivamente de **metáforas** que possui um total de 249 frases. Uma base de dados apenas com frases que não possui nenhum tipo de figura de linguagem com um total de 349 frases. E uma terceira base de dados com **hipérboles, comparações, metáforas** e frases sem nenhum tipo de figura de linguagem. Sendo ela a seguinte junção: frases exclusivamentes de **comparação** + base de dados com frases sem figuras de linguagem + base de dados de **metáforas** + frases de **hipérbole** + acréscimos de novas frases em todas as categorias desta base de dados. Totalizando assim 1.626 frases.

5. Resultados e Discussões

A análise linguística realizada teve como base um conjunto de textos composto pelos seguintes romances do renomado autor brasileiro Machado de Assis: Casa velha, Dom Casmurro, Esaú e Jacó, Helena, Iaiá Garcia, A Mão e a Luva, Memorial de Aires, Memórias Póstumas de Brás Cubas, Quincas Borba, Ressurreição. Já os textos de minha autoria, consistem em onze contos e um romance.

5.1 Algoritmo de Contagem de Palavras Únicas

Para os textos do Machado de Assis, os resultados indicaram que a soma total de todas as palavras, desconsiderando as *stop words*, foi de 295.138 palavras. Além disso, a soma total de todas as palavras únicas, também desconsiderando as *stop words*, foi de 79.002 palavras. Isso indica uma diversidade no vocabulário de aproximadamente 0,2676, quando se considera a razão entre a quantidade de palavras únicas e a quantidade total de palavras, sendo que 1 significa que todas as palavras são únicas.

Além disso, as palavras mais frequentes utilizadas foram, em ordem de maior utilização para menor, foram: ('disse', 1.430), ('olhos', 1.237), ('casa', 1.236), ('tudo', 1.146), ('ser', 1.138), ('ainda', 1.135), ('nada', 1.080), ('tempo', 1.064), ('também', 1.063) e ('outra', 1.031). O gráfico pode ser observado no Gráfico de barras 1. Pode-se observar também as palavras mais utilizadas e outras palavras também frequentes no gráfico de nuvem de frequência de palavras disponível no Gráfico 2.

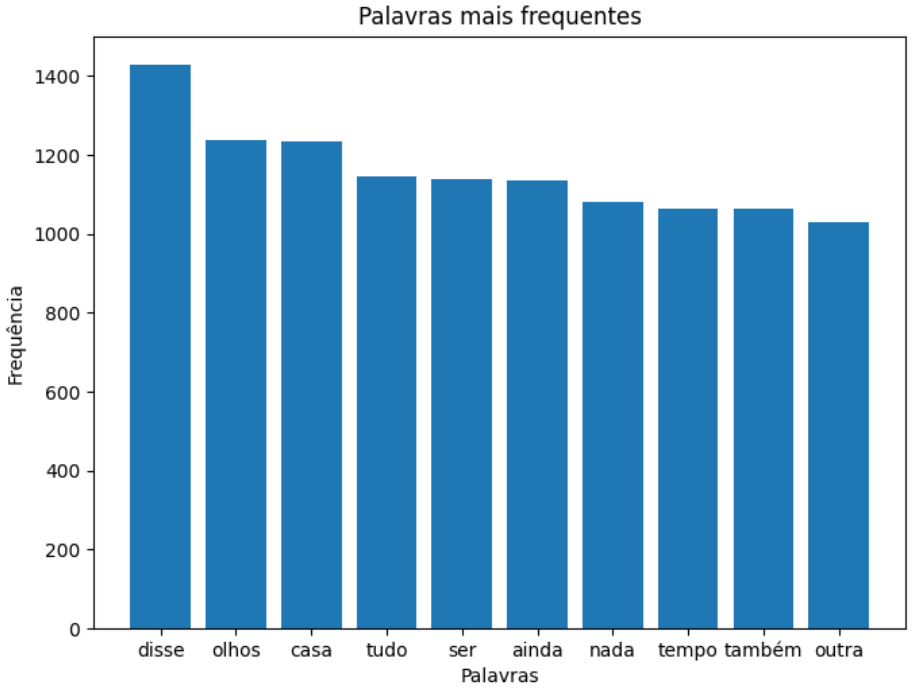

Gráfico 1: Gráfico de barras de palavras mais frequentes dos romances de Machado de Assis

Gráfico 2: Gráfico de nuvem de frequência de palavras dos Romances de Machado de Assis

Nos meus próprios textos, os resultados indicaram que a soma total de todas as palavras, desconsiderando as *stop words*, foi de 30.871 palavras. Já a soma total de todas as palavras únicas, também desconsiderando as *stop words*, foi de 10.519 palavras. O que indica uma diversidade no vocabulário de aproximadamente 0,3407.

Já as palavras mais frequentemente utilizadas foram: ('estava', 452), ('ryan', 200), ('você', 166), ('pra', 161), ('olhos', 159), ('pai', 155), ('iblis', 143), ('cabeça', 136), ('então', 130), ('queria', 127), como pode-se observar no Gráfico 3. O gráfico de nuvem de frequência de palavras também pode ser observado no Gráfico 4.

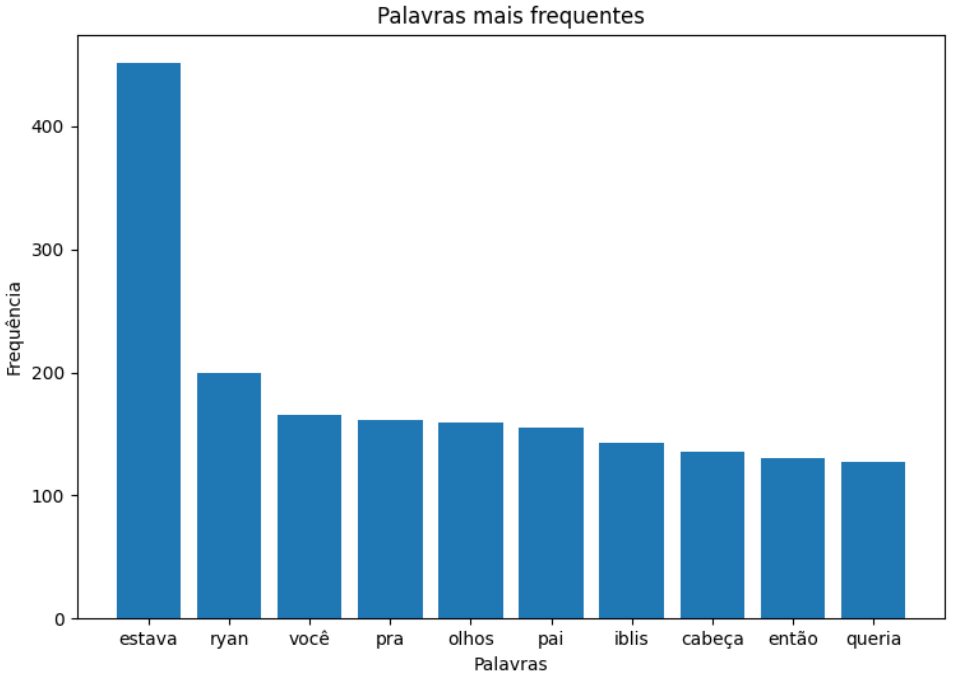

Gráfico 3: Gráfico de barras de palavras mais frequentes dos meus próprios textos

Gráfico 4: Gráfico de nuvem de frequência de palavras dos meus próprios textos

Com base nesta análise é possível tirar algumas conclusões. As palavras mais utilizadas por Machado de Assis são, em números, muito mais equilibradas que as minhas próprias palavras, muito provavelmente pela quantidade de dados disponíveis. A quantidade de dados quase 10x maior do Machado de Assis pode trazer palavras diferentes por alguns motivos: o meu maior texto, o romance, é narrado em primeira pessoa do passado, fazendo com que muitas vezes os verbos dicendi como, por exemplo, o “disse”, podem estar sendo substituído pelo “estava”. O que reflete também na diversidade do vocabulário, que no meu caso, é aproximadamente 27% maior. O que pode ser um indício de que mesmo o número de palavras do texto crescendo significativamente, a quantidade de vocábulos diferentes não necessariamente precisa acompanhar este crescimento.

Outra diferença considerável é no fato dos nomes dos personagens serem muito mais destacados nos meus textos do que nos textos do Machado. Isso, provavelmente, se deve ao fato da natureza dos textos serem diferentes. Os dois nomes que estão entre as dez palavras mais citadas são Ryan e Iblis, ambos os nomes vem dos contos. Ou seja, talvez a dinâmica se alteraria caso textos dos mesmos gêneros literários fossem comparados.

Uma outra possível causa das dessemelhanças pode ser interpretada na temática dos textos em si. Podemos observar nos textos do Machado de Assis o grande volume da palavra casa, sugerindo um local ou uma temática recorrente em seus textos, enquanto nos meus próprios textos a palavra pai foi uma das mais utilizadas, grande parte delas vindo do romance no qual o personagem principal interage frequentemente com o seu pai.

Sobre as semelhanças é interessante destacar uma palavra em específico que se repete com bastante frequência em ambos autores analisados: olhos. Como na vida cotidiana, os olhos, o olhar, parece uma ferramenta poderosa para descrever expressões dos personagens. Tanto que, uma das frases mais famosas do próprio Machado de Assis, é focada inteiramente na expressão, no olhar da personagem Capitu do livro Dom Casmurro: olhos oblíquos de cigana dissimulada.

Apesar desta análise permitir uma apreciação quantitativa da diversidade do vocabulário e das palavras mais comuns nos romances de Machado de Assis e nos meus próprios textos, é importante destacar que este é apenas um vislumbre da complexidade desses textos. A frequência das palavras não leva em consideração o contexto ou a semântica, e, portanto, apesar de informativa, esta análise oferece apenas uma visão parcial da riqueza dos textos.

5.2 Algoritmo de Frequência Léxica Lematizada

Para os romances do Machado de Assis, os resultados mostraram que as dez palavras lematizadas mais frequentes, desconsiderando as *stop words*, como pode ser observado no Gráfico 5, foram: ('dizer', 2.266), ('haver', 1.895), ('ser', 1.839), ('ter', 1.823), ('ir', 1.567), ('dia', 1.484), ('casa', 1.289), ('olho', 1.261), ('algum', 1.192), ('ficar', 989). Pode-se observar também as palavras mais utilizadas e outras palavras também frequentes no gráfico de nuvem de frequência de palavras disponível no Gráfico 6.

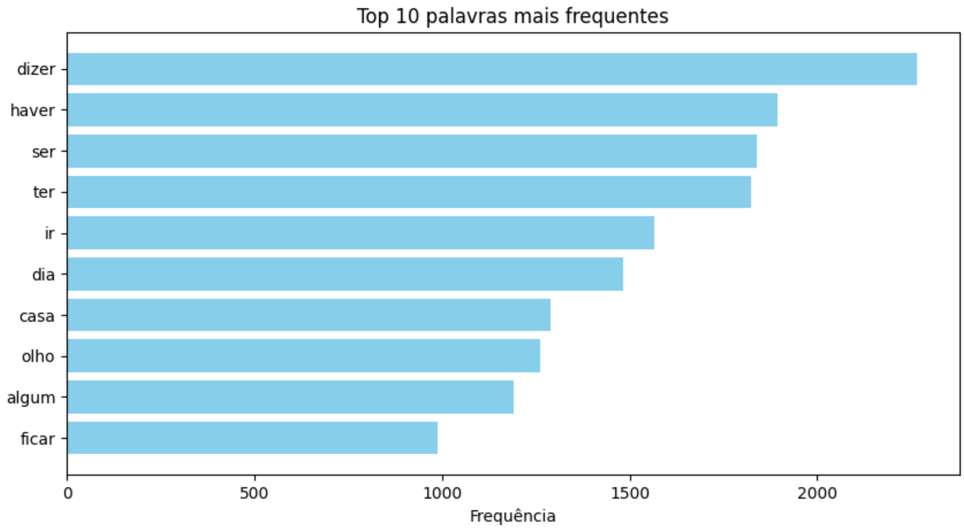

Gráfico 5: Gráfico das palavras mais frequentes lematizadas

Gráfico 6: Gráfico de nuvens de frequência de palavras lematizadas

Podemos observar que o gráfico das palavras frequentes e das palavras frequentes lematizadas são ligeiramente diferentes. Uma das principais diferenças fica na palavra "olho" que saiu da segunda posição com "olhos" e desceu até a oitava. Sendo substituída por alguns verbos como haver, ser, ter e ir.

Já no caso dos meus próprios textos, as palavras lematizadas mais frequentes, como pode ser observado no Gráfico 7, foram: ('ter', 358), ('falar', 263), ('ficar', 229), ('olhar', 205), ('ryan', 200), ('ir', 200), ('querer', 189), ('ser', 173), ('pai', 167), ('olho', 163). Além disso, pode-se também observar no Gráfico 8, o gráfico de nuvens de frequência de palavras lematizadas que pode dar uma visualização mais ampla de outras palavras também frequentes.

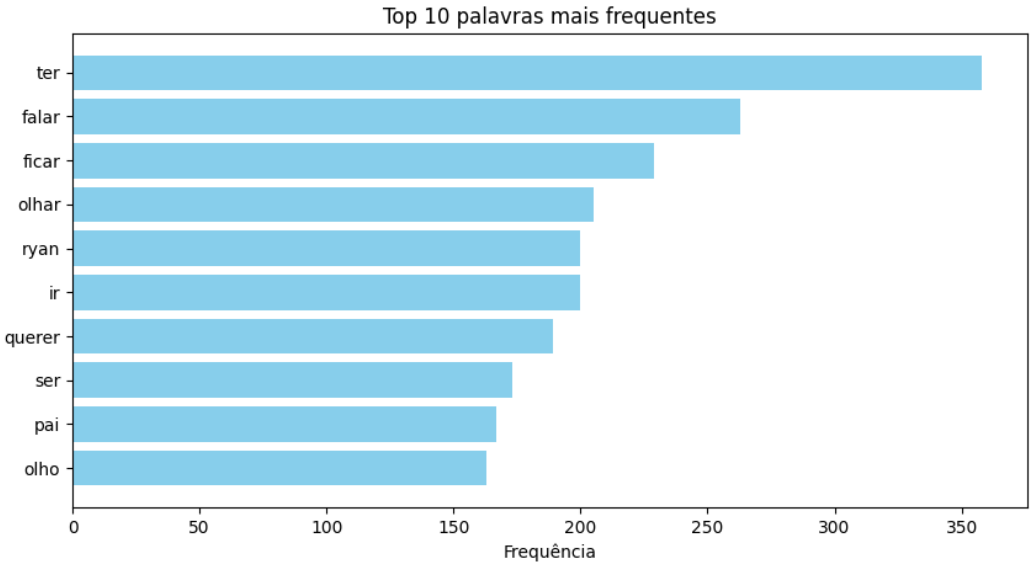

Gráfico 7: Gráfico de barras da frequência de palavras lematizadas nos meus próprios textos

Gráfico 8: Gráfico de nuvens da frequência de palavras lematizadas nos meus próprios textos

Pode-se observar que algo parecido com os textos do Machado de Assis também acontece aos meus: palavras como ter, ficar, ser e ir apareceram na lista de palavras mais comuns. Uma possível explicação disso é pela maneira que as frases em português são construídas. Pela diversidade de tempos verbais, frases como: “Eu estava triste.” e “Agora eu estou feliz” não entram na contagem de palavras únicas, mas ao lematizar as frases, o verbo “estar” acaba se repetindo.

Outra diferença interessante é a aparição de “olhar” na quarta posição e de “olhos” na décima. Uma possível explicação é a diferenciação de substantivo e verbo. Durante a fase de lematização, as conjugações do verbo “olhar”, como por exemplo, “Eu olhei para cima”, podem estar sendo convertidas para olhar. Já em frases como “Seus olhos eram lindos”, a palavra pode estar sendo convertida para “olhos”, já que a palavra nessa frase faz o papel de substantivo.

5.3 Algoritmo de Frequência de Palavras na Frase

Nos textos do Machado de Assis, para cada documento, três métricas foram calculadas: a média do tamanho das frases (em caracteres), a média de palavras por frase e a média de caracteres por palavra. Estas métricas nos fornecem uma visão geral da densidade do texto, que podem ser indicativos do nível de leitura requerido para compreendê-lo.

A média total do tamanho das frases de seus textos ficou em 89,4495 caracteres. Já o total de palavras por frase ficou em: 18,7993 palavras. Por fim, a média total de caracteres por palavra foi de 4,8148 caracteres. Esses resultados e os resultados divididos por texto podem ser observados no Gráfico 9.

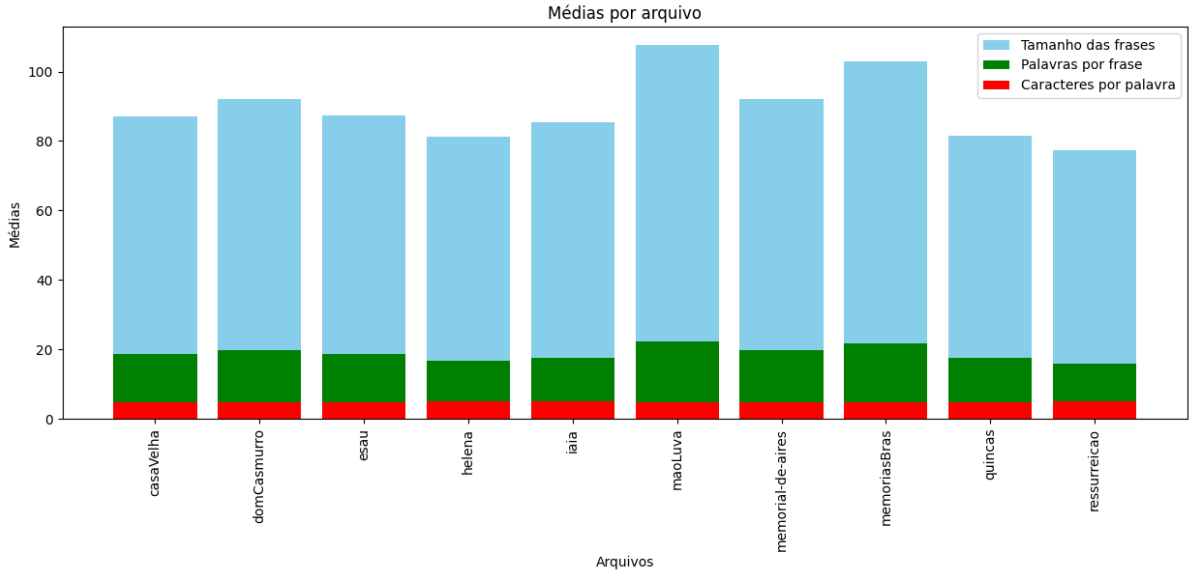

Gráfico 9: Gráfico de barras com as métricas das frases dos textos de Machado de Assis

Já nos meus próprios textos ficou em 64,4754 caracteres. Já o total de palavras por frase ficou em: 13,4277 palavras. Por fim, a média total de caracteres por palavra foi de 4,8891 caracteres. Esses resultados e os resultados divididos por texto podem ser observados no Gráfico 10.

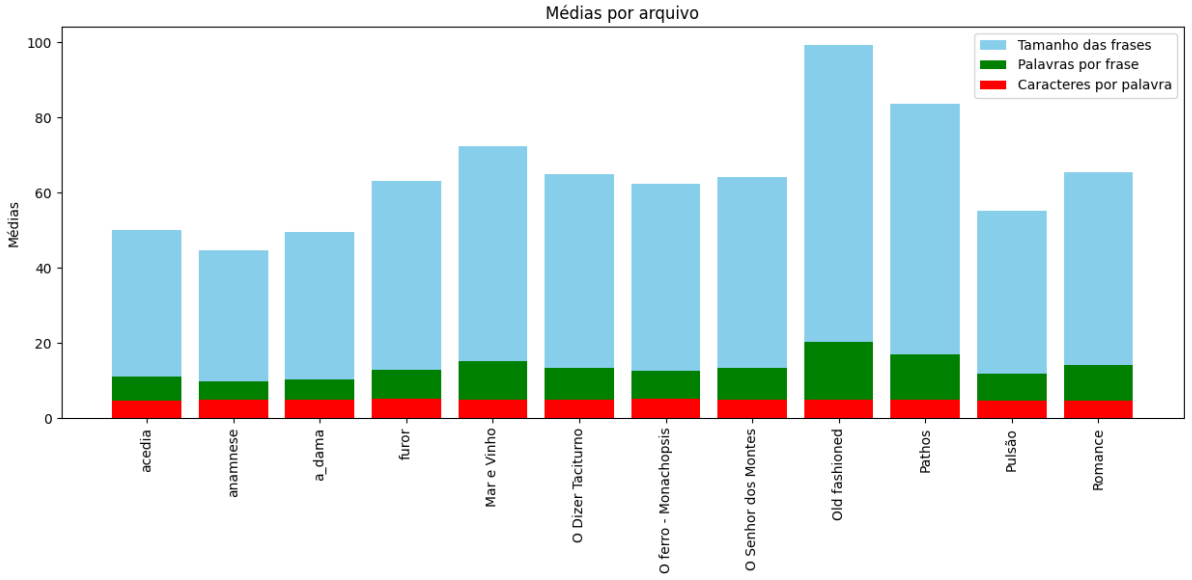

Gráfico 10: Gráfico de barras com as métricas das frases dos meus próprios textos

A média de caracteres por frase fornece uma ideia do comprimento das frases. Textos com uma média maior podem indicar um estilo de escrita mais complexo e detalhado. No caso dos textos de Machado de Assis, a média mais alta sugere que ele tende a usar frases mais longas, o que pode ser indicativo de uma característica típica de seu estilo de escrita sofisticado e detalhado.

Já a média de palavras por frase pode indicar a densidade da informação. Uma média mais alta sugere frases mais carregadas de informação, enquanto uma média mais baixa pode indicar um estilo de escrita mais direto e conciso. Esta métrica novamente foi mais alta nos textos de Machado de Assis, o que sugere que ele tende a embalar mais ideias e informações em suas frases.

Por fim, a média de caracteres por palavra pode nos dar uma ideia do uso de vocabulário. Uma média mais alta pode indicar o uso de palavras mais longas ou complexas. Embora a diferença nas médias entre os textos de Machado de Assis e dos meus próprios textos, como pode ser observado no Quadro 1, elas parecem ser pequenas demais para sugerirem qualquer diferença no tamanho médio do vocabulário utilizado.

Métricas	**Machado de Assis**	**Textos Próprios**
Média tamanho das frases	89,4495	64,4754
Total de palavras por frase	18,7993	13,4277
Média total de caracteres	4,8148	4,8891

Quadro 1: Comparação entre as métricas do Machado e as minhas próprias

5.4 Algoritmo de Análise de Sentimentos

Para esta análise, decidi começar executando o algoritmo nos textos do Machado de Assis. O primeiro texto a ter a análise de sentimentos medida foi o romance Casa Velha. Como é possível ver no Gráfico 11, o sentimento permanece oscilando entre positivo e muito negativo de maneira quase padronizada.

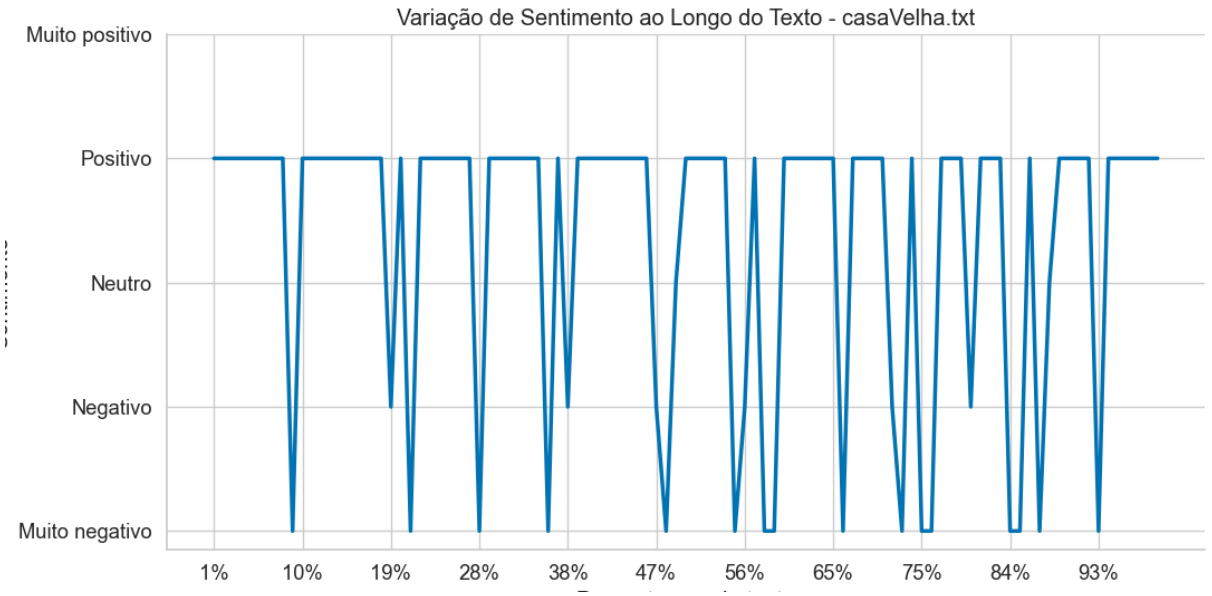

Gráfico 11: Gráfico de análise de sentimentos do romance Casa Velha de Machado de Assis

No meu próprio romance a análise de sentimentos retornou um resultado bastante semelhante, como pode ser observado no Gráfico 12. O gráfico também oscila entre o positivo e o muito negativo, mas diferente do texto do Machado de Assis que permanece padronizado do início ao fim, o meu texto tem alguns momentos de duração positivas maiores.

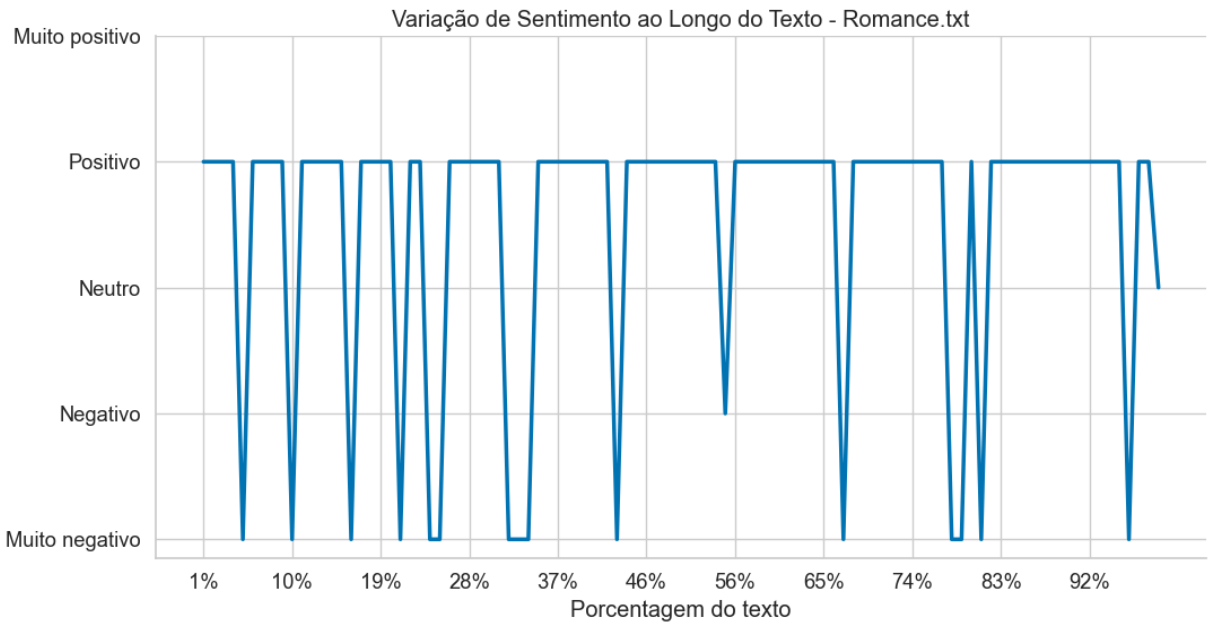

Gráfico 12: Gráfico de análise de sentimentos do meu próprio romance

Analisando estes dois textos individualmente, uma das possíveis hipóteses a ser levantada é que romances, em geral, variam o sentimento entre positivo e muito negativo recorrentemente. Mas ao adicionarmos novos dados na análise, percebemos que não necessariamente precisa ser assim. O romance Dom Casmurro, apesar de oscilar ao muito negativo em determinados momentos, como pode ser observado no Gráfico 13, tem sua maior parte avaliada com sentimentos positivos.

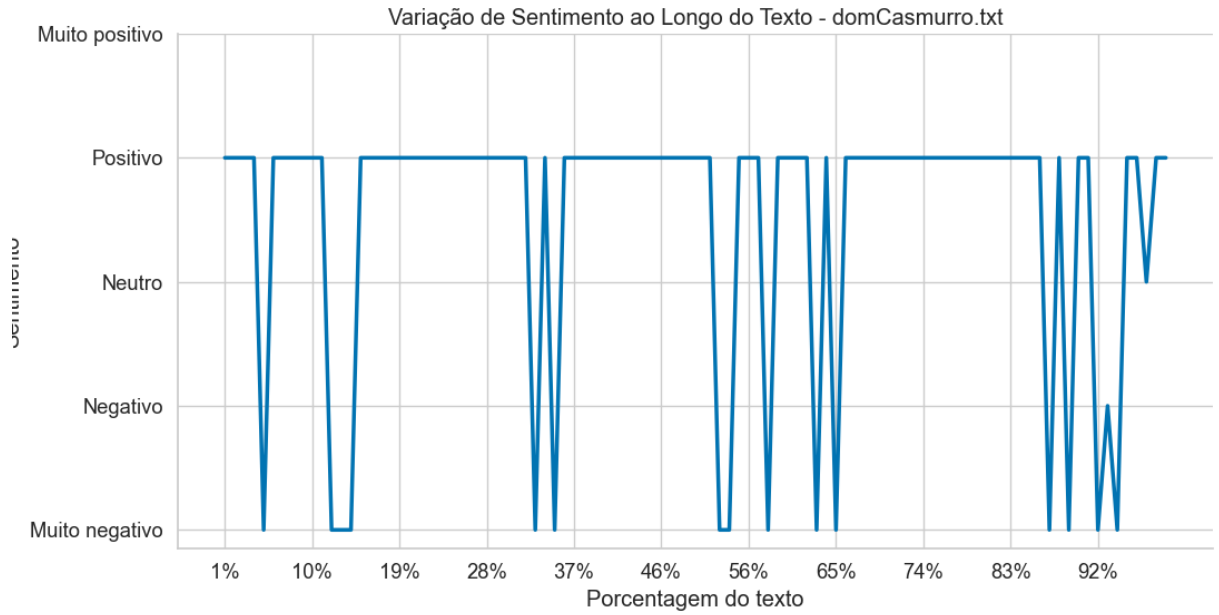

Gráfico 13: Gráfico de análise de sentimentos do romance Dom Casmurro de Machado de Assis

Outra hipótese que pode ser considerada é que a oscilação de sentimentos talvez reflita a dinâmica do enredo nos romances. É comum que um romance envolva conflitos e resoluções, momentos de tensão e alívio, que podem ser refletidos na variação de sentimentos positivos e negativos. No entanto, a amplitude e a frequência dessas oscilações podem ser influenciadas pelo estilo de escrita do autor, bem como pelo tema e contexto do romance.

Já os meus contos, por serem menores, o que consequentemente faz os fragmentos de texto submetidos a análise de sentimentos serem menores também, apresentam nos gráficos uma diferença notável: a taxa de oscilação. O conto Acedia, como mostra o Gráfico 14, apesar de ainda oscilar majoritariamente entre o positivo e o muito negativo, ainda tem alguns picos no neutro que ainda não havia sido observado antes.

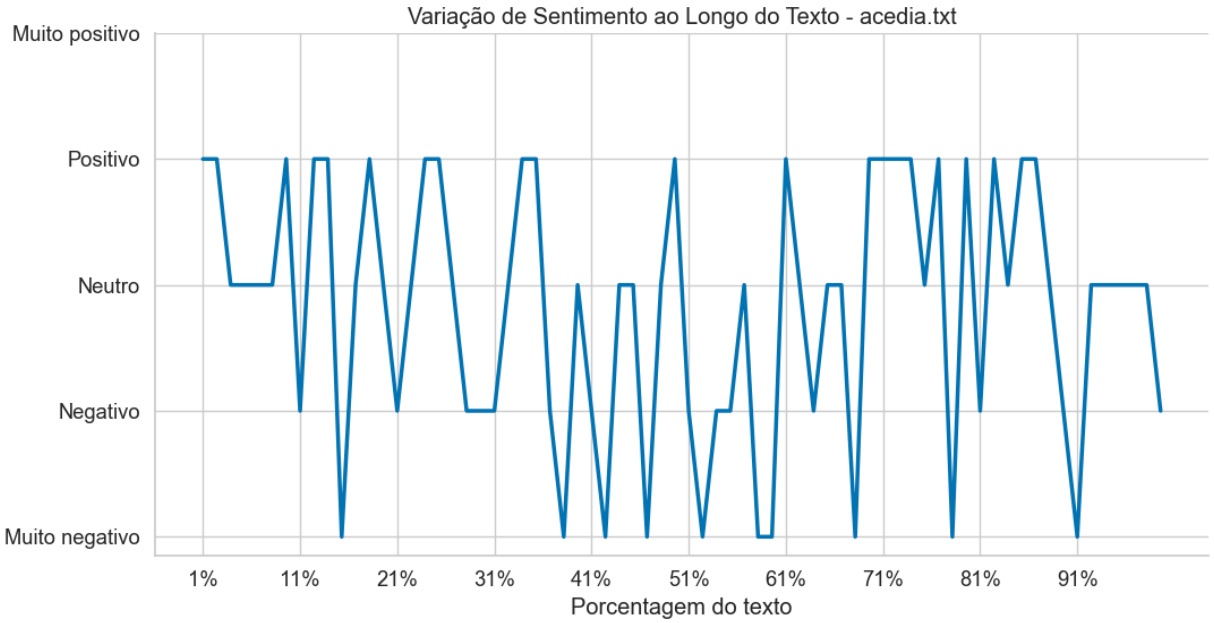

Gráfico 14: Gráfico de análise de sentimentos do conto Acedia

Por fim, ainda foi encontrado um último caso que se difere dos demais: contos em que a oscilação é menos frequente e muito mais presente no meio do gráfico. Ou seja, contos onde a maior taxa de oscilação ficou entre o neutro e o negativo. Um exemplo deste caso é o meu conto Old Fashioned. Caso este que pode ser observado no Gráfico 15.

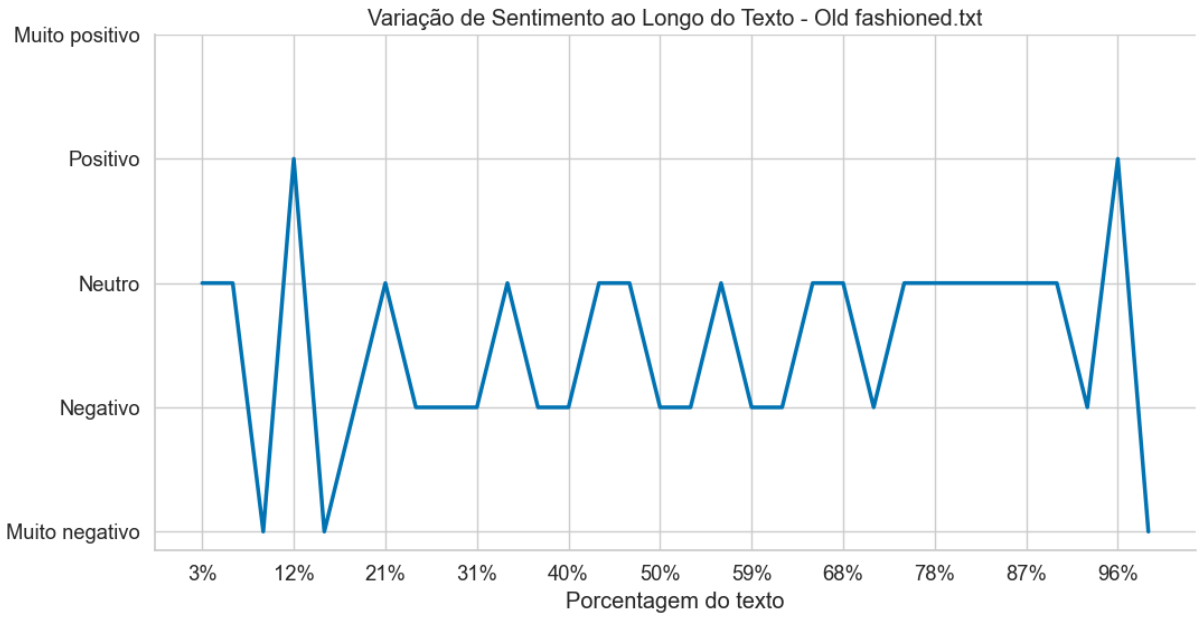

Gráfico 15: Gráfico de análise de sentimentos do conto Old Fashioned

Uma hipótese para isso é que os contos, por serem mais curtos, podem incluir partes de transição ou desenvolvimento que não provocam sentimentos fortes, mas são importantes para a estrutura da história. Ou ainda que a própria estrutura narrativa do texto pode ser pautada nesse aspecto. Contos onde o desfecho acontece apenas no final como Old Fashioned, podem ter esse aspecto neutro até o ápice dramático. No caso deste conto em específico, além dele ter o ápice dramático nas últimas linhas do texto, ele termina com um tom amargo, o que pode justificar a queda brusca de sentimento no final do gráfico.

5.5 Rede Neural para Identificação de Figuras de Linguagem

Para o treinamento da rede neural, o modelo percorreu quatro épocas[8], cada uma delas consistindo em uma passagem completa pelos dados de treinamento. A cada época, foi calculada a perda, que é uma medida de quão bem o modelo está se saindo. Quanto menor a perda, melhor o modelo está em fazer previsões precisas. Com a base de dados de figuras de linguagem de comparação, o modelo conseguiu as seguintes métricas: Época 1/4, Perda: 0,6300; Época 2/4, Perda: 0,2241; Época 3/4, Perda: 0,0588; Época 4/4, Perda: 0,0281.

Ao final do treinamento, o modelo foi avaliado em termos de acurácia no conjunto de testes. A acurácia é a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. Uma acurácia mais alta indica um modelo de melhor desempenho. Este modelo teve a métrica de 0,9941 no conjunto de teste.

Logo após, o modelo treinado foi aplicado no intuito de encontrar as figuras de linguagens nos textos do Machado de Assis. Como pode-se ver também no Gráfico 16, as seguintes comparações foram encontradas:

Comparações em Casa Velha: 24

Comparações em Dom Casmurro: 115

Comparações em Esaú e Jacó: 128

Comparações em Helena: 144

Comparações em A Mão e a Luva: 84

Comparações em Memorial de Aires: 41

Comparações em Memórias Póstumas de Brás Cubas: 173

Comparações em Quincas Borba: 140

[8] Época em redes neurais refere-se ao número de vezes que todo o conjunto de dados de treinamento é usado uma vez para atualizar os pesos na rede neural. Durante uma época, a rede neural passa por um processo de aprendizado em que tenta fazer previsões, comparar suas previsões com os resultados reais e, em seguida, ajustar seus pesos e vieses com base nos erros que fez. O objetivo é minimizar a diferença entre as previsões e os valores reais ao longo de múltiplas épocas.

Comparações em Ressurreição: 67

Total de comparações em todos os arquivos: 1.031

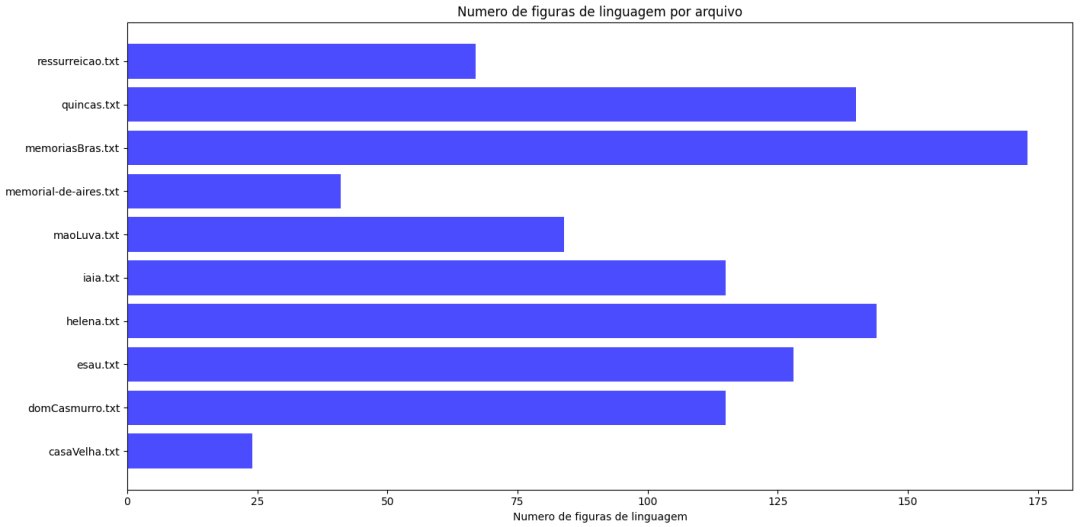

Gráfico 16: Gráfico de barras com o total de comparações encontradas nos textos de Machado de Assis

Todas as figuras de linguagem foram salvas em um arquivo JSON (JavaScript Object Notation) para consulta. Devido ao grande número de frases, conferir cada uma delas individualmente para analisar a assertividade do modelo tornou-se um pouco inviável, mas uma análise de algumas amostras, mostrou que o modelo, como esperado pela acurácia, foi bem preciso.

Alguns exemplos de frases encontradas pelo modelo foram: "O que aqui está é, mal comparando, semelhante à pintura que se põe na barba e nos cabelos, e que apenas conserva o hábito externo, como se diz nas autópsias; o interno não agüenta tinta", "São como fotografias instantâneas da felicidade", "A voz da mãe era agora mais perto, como se viesse já da porta dos fundos".

Pesquisando um pouco é possível de achar em meio as comparações alguns poucos falsos positivos, um exemplo é a seguinte frase: "A boca podia ser o cálice, os lábios a patena", "Não era o mesmo homem que estragava o chapéu em cortejar a vizinhança, risonho, olhos no ar, antes mesmo da administração interina".

Devido à natureza das redes neurais serem uma caixa preta, ou seja, impossível de acessar o percurso das decisões tomadas por ela, não é possível saber ao certo o motivo dos

erros do modelo, já que as frases citadas acima nem mesmo possuem palavras típicas de figuras de linguagem de comparação.

Posteriormente o modelo treinado foi aplicado nos meus próprios textos. Como pode-se ver no Gráfico 17, as seguintes figuras de linguagem foram encontradas:

Comparações em Acedia: 7

Comparações em Anamnese: 1

Comparações em A dama: 7

Comparações em Furor: 2

Comparações em Mar e Vinho: 5

Comparações em O Dizer Taciturno: 5

Comparações em O ferro - Monachopsis: 8

Comparações em O Senhor dos Montes: 21

Comparações em Old Fashioned: 6

Comparações em Pathos: 7

Comparações em Pulsão: 5

Comparações em Romance: 26

Número total de figuras encontradas: 100

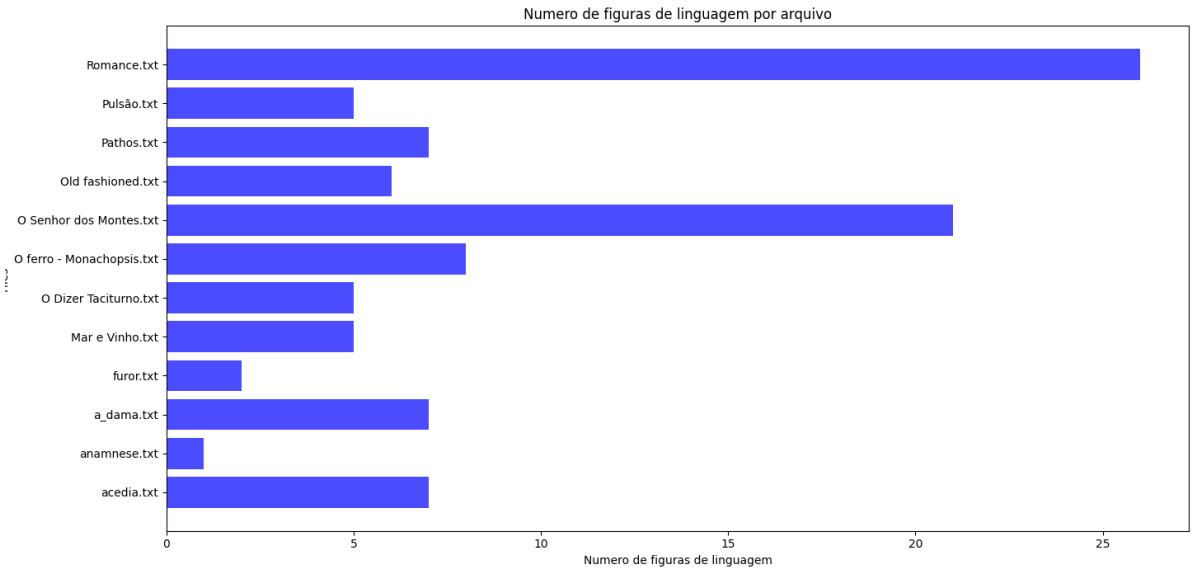

Gráfico 17: Gráfico de barras com o total de comparações encontradas nos meus próprios textos

Alguns exemplos de frases de comparações encontradas pela rede neural nos meus textos foram: "Seu rosto inerte, dissonante como um violino tocado com uma vara de

cascalho", "Como uma pintura em aquarela, as nuvens pareciam pinceladas em meio aos tons celestes que se misturavam em um degradê cinza-alaranjado", "Iblis subiu aos céus, cabelo esvoaçante tão dourado quanto os olhos", "Ele, ao seu lado, era menor que uma formiga".

Alguns falsos positivos também foram encontrados. Seguem alguns exemplos: "Ele era tão simples em sua complexidade", "O abismo entre eles era tão grande que, para Ryan, ela não era mais que uma pequena estrela perdida no céu: inalcançável".

Um detalhe interessante é que, depois da análise do arquivo JSON, foram encontrados muito menos falsos positivos nos meus textos do que nos textos do Machado de Assis. Uma possível explicação é que, como as frases do conjunto de dados de treinamento foram escritas por mim, talvez elas possam estar enviesadas para uma maior acurácia nos meus próprios textos. Outra explicação poderia estar na linguagem. A linguagem da base de dados é muito mais atual do que a linguagem utilizada nos textos do Machado de Assis, o que poderia aumentar a taxa de erro.

Por mais que o número de figuras de linguagem encontradas nos textos do Machado e nos meus textos fossem muito discrepantes, para se fazer uma avaliação mais precisa dos resultados é necessário levar em consideração o tamanho dos textos que foram analisados. A soma da quantidade total de palavras nos textos do Machado de Assis é de 535.224 palavras enquanto a soma de palavras dos meus textos são de 57.478 palavras, ou seja, um pouco mais de 9 vezes maior. Ao compararmos com a quantidade de figuras de linguagem encontradas pela rede neural, vemos que o número chega próximo a esse mesmo valor. Foram encontradas 1.031 figuras de linguagem de comparação nos textos do Machado de Assis e 100 figuras nos meus textos.

O próximo passo foi treinar um modelo de treinamento para a identificação de metáforas. A base de dados também foi construída por mim, mas diferentemente da anterior, dessa vez, para fins de teste, construí a base de maneira diferente. A base de dados era composta por aproximadamente 300 frases que eram metáforas e 300 frases que não eram nenhuma figura de linguagem, que é a maior diferença para a base de dados da figura de linguagem de comparação, já que as frases que não eram catalogadas como comparação, poderiam também ser outra figura de linguagem distinta.

O treino também foi composto por 4 épocas e surtiu em um resultado ainda melhor que a de comparação: Época 1/4, Perda: 0,4376; Época 2/4, Perda: 0,0549; Época 3/4 Perda: 0,0155; Época 4/4, Perda: 0,0114; Acurácia: 1,0000.

Por mais que no treino o modelo tenha tido um resultado excepcional, na hora de aplicar o modelo nos textos para encontrar as figuras de linguagem, seu desempenho foi péssimo. Ele não conseguiu encontrar quase nenhuma metáfora. A mesma dificuldade em encontrar falso positivos no modelo anterior, foi a dificuldade de encontrar exemplos verdadeiramente positivos neste modelo.

Algumas hipóteses podem ser levantadas para tentar especular o motivo do fracasso. A primeira hipótese é de que o modelo aprendeu a identificar apenas metáforas em meio a frases que não contém nenhuma figura de linguagem. Dado a diversidade de composições e figuras de linguagem que existem na língua portuguesa, esse cenário é quase improvável. A segunda hipótese é que o conjunto de dados é demasiadamente pequeno para este tipo de figura de linguagem. Como a metáfora é uma figura de semântica, ela é mais complexa do que a de comparação que é de natureza sintática, logo pode ser necessário criar uma base de dados ainda maior para que a rede neural consiga realmente compreender em como identificá-las. Ou então, talvez fazer mais épocas de treinamento ou variar os hiperparâmetros da rede neural.

O mesmo ocorreu ao tentar treinar um modelo com três figuras de linguagem diferentes: hipérbole, metáfora e comparação. Mas, além de ter fracassado no momento de encontrar figuras de linguagem, sua acurácia no treinamento ficou em aproximadamente 0,65. Isso ocorre, provavelmente, pelo fato de que é mais simples fazer uma decisão binária do que encaixar uma frase em alguma das figuras pré-estabelecidas. Tornando a tarefa de classificação muito mais difícil.

6. Conclusões

A arte, em suas múltiplas formas e manifestações, permanece como um campo inesgotável de expressão e interpretação, com a literatura representando, em minha opinião, uma de suas vertentes mais ricas e complexas. Este estudo procurou identificar padrões e técnicas literárias que se mostrassem distintivas entre obras reconhecidamente de alta qualidade e aquelas de qualidade ainda não comprovada. Por meio do uso de ferramentas computacionais avançadas, como a análise de sentimentos e redes neurais, foi possível explorar as complexidades da escrita.

Entretanto, sabendo que a subjetividade é inerente à arte, e consequentemente à literatura, é necessário cautela ao interpretar esses resultados. A qualidade literária não é um atributo que pode ser facilmente mensurado ou categorizado e os padrões identificados neste estudo não devem ser vistos como determinantes absolutos de excelência literária. Além disso, a diversidade de percepções, emoções e contextos socioculturais que informam nossa interpretação da literatura significa que uma obra pode ser apreciada e valorizada de muitas maneiras diferentes.

Um ponto importante a se ressaltar é que foi particularmente notável a carência de recursos e estudos no campo das figuras de linguagem na língua portuguesa. Durante este estudo, foram realizadas extensas pesquisas, e ainda assim, não foi possível encontrar um conjunto de dados adequado com figuras de linguagem, o que levou à necessidade de construção de um próprio. Esta experiência enfatiza a importância da pesquisa contínua e do desenvolvimento neste campo.

Ao abrir novos caminhos para a análise da técnica literária, este estudo pode ser um pontapé na reflexão sobre o papel e o impacto da tecnologia em nossa relação com a arte. À medida que continuamos a explorar e a desenvolver ferramentas que nos permitam entender melhor a arte em suas várias formas, também devemos considerar as maneiras como essas ferramentas podem influenciar e transformar nossa experiência artística. Neste sentido, a junção entre as ciências da computação e a literatura se mostra como um campo promissor de estudo, desafiando-nos a reavaliar e a expandir continuamente nossas compreensões da arte, da beleza e da qualidade literária.

Apesar de incipiente, este trabalho já trouxe importantes contribuições para a compreensão da arte literária através da lente da ciência da computação. Entre os principais

resultados, podemos destacar o desenvolvimento de um conjunto de dados inédito sobre figuras de linguagem na língua portuguesa, criado devido à ausência de recursos pré-existentes nesta área. Esta base de dados não apenas foi essencial para a realização do presente estudo, mas também constitui uma valiosa contribuição para futuras pesquisas na intersecção entre a ciência da computação e os estudos literários.

Além disso, foi proposto um algoritmo para a análise de textos literários, utilizando técnicas avançadas como redes neurais. Este algoritmo permitiu a identificação de padrões e técnicas distintivas nas obras analisadas, contribuindo para aprofundar nossa compreensão da complexidade da escrita literária.

Por fim, a análise realizada neste estudo ressaltou a vasta diversidade de interpretações possíveis para uma obra literária, refletindo a rica complexidade de percepções, emoções e contextos socioculturais que influenciam nossa experiência com a arte.

Embora este trabalho seja apenas um primeiro passo, é esperado que suas contribuições possam servir como base para futuras pesquisas, estimulando um diálogo cada vez mais profundo e produtivo entre as ciências da computação e a literatura. Ao desbravar este novo território, temos a oportunidade de expandir continuamente nossas compreensões da arte, da beleza e da qualidade literária, e de explorar as inúmeras possibilidades que se abrem quando a tecnologia se encontra com a arte.

7. Referências

[1] KANT, Immanuel. Crítica da faculdade do juízo. Tradução de Valério Rohden e Antônio Marques. 2. ed. Rio de Janeiro: Forense Universitária, 1995

[2] Adorno, T. W. Über den Fetischcharakter in der Musik und die Regression des Hörens. Studies in Philosophy and Social Science. 1938

[3] Kozbelt, Aaron. Originality and technical skill as components of artistic quality. Empirical studies of the arts. 2004

[4] Kovács, Zsolt László. Redes neurais artificiais. Editora Livraria da Física, 2002.

[5] Barca, Maria Carolina Stockler, Tiago Redondo de Siqueira SILVEIRA, and Marcio Magini. Treinamento de redes neurais artificiais: o algoritmo backpropagation. 2005

[6] Tissot, Hegler C., Luiz C. Camargo, and Aurora TR Pozo. Treinamento de redes neurais feedforward: comparativo dos algoritmos backpropagation e differential evolution. 2012.

[7] Cong, G., Bhardwaj, O., & Feng, M. An Efficient, Distributed Stochastic Gradient Descent Algorithm for Deep-Learning Applications. 2017 46th International Conference on Parallel Processing (ICPP). 2017.

[8] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep Learning. Massachusetts: MIT Press, 2016.

[9] Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. 2016

[10] SUN, Fei et al. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM international conference on information and knowledge management. 2019. p. 1441-1450

[11] Zhang, L., & Liu, B. Sentiment Analysis and Opinion Mining. Encyclopedia of Machine Learning and Data Mining, 1–10. 2016

[12] Baldick, Chris. The Oxford Dictionary of Literary Terms." Oxford University Press. 2008.

[13] Hutto, C.J. & Gilbert, E.E. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). 2014