ROLF PAGOTTO VEIGA

DIRETRIZES PARA CLASSIFICAÇÃO AUTOMÁTICA DE MÚSICAS CRISTÃS: APLICAÇÃO DO BERT NA IDENTIFICAÇÃO DE ADORAÇÃO

LAVRAS-MG
2023

ROLF PAGOTTO VEIGA

DIRETRIZES PARA CLASSIFICAÇÃO AUTOMÁTICA DE MÚSICAS CRISTÃS:

APLICAÇÃO DO BERT NA IDENTIFICAÇÃO DE ADORAÇÃO

Dissertação apresentada à Universidade Federal de Lavras como parte das exigências do Programa de Pós-graduação em Ciência da Computação, área de concentração em Ciência da Computação, para a obtenção do título de Mestre.

Prof. DSc. Eric Fernandes de Mello Araújo

Orientador

LAVRAS – MG
2023

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Veiga, Rolf Pagotto.
Diretrizes para Classificação Automática de Músicas Cristãs : Aplicação do BERT na Identificação de Adoração / Rolf Pagotto Veiga. - 2024.
86 p. : il.

Orientador(a): Eric Fernandes de Mello Araújo.

Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2024.
Bibliografia.

1. Processamento de Linguagem Natural. 2. Análise de Sentimento. 3. Bert. I. Araújo, Eric Fernandes de Mello. II. Título.

ROLF PAGOTTO VEIGA

DIRETRIZES PARA CLASSIFICAÇÃO AUTOMÁTICA DE MÚSICAS CRISTÃS:

APLICAÇÃO DO BERT NA IDENTIFICAÇÃO DE ADORAÇÃO

GUIDELINES FOR AUTOMATIC CLASSIFICATION OF CHRISTIAN MUSICS:

APPLICATION OF BERT IN WORSHIP IDENTIFICATION

Dissertação apresentada à Universidade Federal de Lavras como parte das exigências do Programa de Pós-graduação em Ciência da Computação, área de concentração em Ciência da Computação, para a obtenção do título de Mestre.

APROVADA em 30 de agosto de 2024.
Dr. Mayron Cesar de Oliveira Moreira UFLA
Dr. Luiz Henrique de Campos Merschmann UFLA
Dr. Fernando Pasquini Santos CALVIN

Prof. DSc. Eric Fernandes de Mello Araújo
Orientador

LAVRAS – MG
2023

RESUMO

O estudo apresentado nesta dissertação explora a aplicação de algoritmos avançados de Processamento de Linguagem Natural (PLN) para examinar as letras de músicas evangélicas brasileiras. Utilizando o modelo BERT (Bidirectional *Encoder* Representations from *Transformer*s), desenvolvido por Devlin et al. em 2019, esta pesquisa se propõe a decifrar padrões linguísticos e temáticos nas composições, visando entender como essas refletem as crenças, valores e a identidade religiosa no contexto brasileiro. A música evangélica é um componente vital da cultura brasileira, servindo como uma expressão de fé e um veículo para a construção da identidade comunitária e individual. Este estudo foca particularmente na análise de sentimentos e na identificação de temas de adoração, usando o BERT para analisar a complexidade e a riqueza do conteúdo lírico das músicas. A dissertação detalha a metodologia empregada na coleta e análise dos dados, destacando o papel transformador do BERT no entendimento contextualizado de textos. Combinando técnicas quantitativas e qualitativas, a pesquisa não apenas fornece conhecimento sobre a evolução da música evangélica brasileira, mas também o potencial do uso de modelos de PLN no estudo detalhado de textos musicais. Este trabalho é significativo para os campos de estudos da música, religião e tecnologia da informação, oferecendo uma nova perspectiva sobre como a tecnologia pode ser utilizada para aprofundar a compreensão das dinâmicas culturais e espirituais contemporâneas.

Palavras-chave: BERT; processamento de linguagem natural; música evangélica; análise de sentimento; adoração.

ABSTRACT

The study presented in this dissertation explores the application of advanced Natural Language Processing (NLP) algorithms to examine the lyrics of brazilian Christian songs. Utilizing the BERT model (Bidirectional *Encoder* Representations from *Transformer*s), developed by Devlin et al. in 2019, this research aims to decipher linguistic and thematic patterns in the compositions, seeking to understand how these reflect beliefs, values, and religious identity in the brazilian context. Christian music is a vital component of brazilian culture, serving as an expression of faith and a way for the construction of both communal and individual identity. This study particularly focuses on sentiment analysis and the identification of worship themes, using BERT to analyze the complexity and richness of the lyrical content of the songs. The dissertation details the methodology employed in the data collection and analysis, highlighting the transformative role of BERT in the contextual understanding of texts. By combining quantitative and qualitative techniques, the research not only provides insights into the evolution of brazilian Christian music but also demonst*rate*s the potential of using NLP models in the detailed study of musical texts. This work is significant for the fields of music studies, religion, and information technology, offering a new perspective on how technology can be utilized to deepen the understanding of contemporary cultural and spiritual dynamics.

Keywords: BERT; natural language processing; evangelical music; sentiment analysis; worship.

INDICADORES DE IMPACTO

Esta pesquisa, ao aplicar o modelo BERT para a classificação de composições de música cristã no Brasil, destaca potenciais impactos culturais e tecnológicos. A análise identificou que aproximadamente **35% das composições analisadas não apresentam características de adoração**, indicando uma transformação no conteúdo lírico que prioriza reflexões espirituais, mensagens motivacionais ou valores morais, em detrimento de elementos tradicionais de louvor e adoração a Deus. Culturalmente, a investigação contribui para compreender essas mudanças nas composições evangélicas ao longo do tempo, promovendo diálogo entre diferentes gerações e contextos. No aspecto tecnológico, o uso do Processamento de Linguagem Natural (PLN) representa um avanço na aplicação de inteligência artificial para análise de dados textuais, demonstrando o potencial de ferramentas como o BERT na exploração de fenômenos culturais e religiosos. A pesquisa exemplifica como tecnologias emergentes podem inovar metodologias em estudos culturais, alinhando-se ao Objetivo de Desenvolvimento Sustentável 9 (Indústria, Inovação e Infraestrutura). Além disso, ao contribuir para a democratização do acesso a ferramentas tecnológicas e fomentar o engajamento crítico com conteúdos culturais, o estudo apoia o Objetivo de Desenvolvimento Sustentável 4 (Educação de Qualidade), permitindo um acesso mais amplo ao conhecimento e o desenvolvimento de abordagens interdisciplinares para compreender composições evangélicas.

IMPACT INDICATORS

This research, by applying the BERT model to classify Christian music compositions in Brazil, highlights potential cultural and technological impacts. The analysis identified that approximately **35% of the analyzed compositions do not exhibit characteristics of worship**, indicating a transformation in lyrical content that prioritizes spiritual reflections, motivational messages, or moral values over traditional elements of praise and worship to God. Culturally, the investigation contributes to understanding these changes in evangelical compositions over time, fostering dialogue between different generations and contexts. Technologically, the use of Natural Language Processing (NLP) represents an advancement in the application of artificial intelligence for textual data analysis, demonstrating the potential of tools like BERT in exploring cultural and religious phenomena. The research exemplifies how emerging technologies can innovate methodologies in cultural studies, aligning with Sustainable Development Goal 9 (Industry, Innovation, and Infrastructure). Furthermore, by contributing to the democratization of access to technological tools and fostering critical engagement with cultural content, the study supports Sustainable Development Goal 4 (Quality Education), enabling broader access to knowledge and the development of interdisciplinary approaches to understanding evangelical compositions.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo irá esclarecer as motivações e a história do tema estudado: a música cristã no Brasil, demonstrando os contextos do trabalho, sua justificativa e seus objetivos e perguntas norteadoras.

1.1 História e Evolução da Música Cristã

No período da Idade Média, a Igreja Católica demonstrou grande interesse pela música, por acreditar que poderia influenciar o s h omens, p or meio dela(SIMÕES, 2011).

Representantes da Igreja forneciam um apoio valioso para o estudo e o ensino musical. Fundaram então as capelas, escolas, academias, bibliotecas, orquestras polifônicas e instrumentais e promoveram a formação de compositores, cantores, concertistas e musicólogos. São Gregório Magno, Papa de 590 a 604, teve um papel crucial na organização e padronização da liturgia e do canto litúrgico. Ele Criou dois importantes livros intitulados Antiphonarium e Cantatorium, que continham canções e novos hinos, na ordem das festas e cerimônias, e que serviram de padrão para toda a Igreja Católica . As canções escritas no livro deixados por São Gregório Magno, que buscavam exaltar a figura papal e expressar à palavra de Deus, foram chamadas de canto gregoriano ou cantochão. Nesse sentido, um canto falado a uma só voz e composto por uma só melodia assume o carácter de uma oração cantada, símbolo da fé cristã (MARTINOFF, 2014).

Com o humanismo renascentista, que se iniciou no século XIV, a inspiração para a música era promover a liberdade do homem e a valorização da razão humana. Este movimento coincidiu com a expansão científica e geográfica, o crescimento do comércio e o fortalecimento da burguesia.Neste contexto de efervescência cultural, surgiram protestos dentro do mundo católico, perturbando a vida religiosa. A música religiosa experimentou um novo impulso com a Reforma Protestante, liderada por Martinho Lutero, e enraizada no humanismo. A Reforma enfatizou a responsabilidade individual na fé, defendendo a leitura da Bíblia como fonte primordial da mesma e a necessidade de educação para todos os fiéis (RESTREPO, 1965).

No Brasil a chegada do protestantismo se deu no início do século XIX, coincidindo com um período de transformações significativas n o p aís, c omo a abertura dos portos ao comércio inglês e o estímulo à imigração europeia. Essa conjuntura política e econômica, além da ratificação d a l iberdade r eligiosa pela Constituição de 1824, proporcionaram um ambiente

favorável para o estabelecimento de diversas denominações protestantes, incluindo anglicanos, episcopais (bispos) e luteranos. A chegada de missionários dos Estados Unidos da América e do Reino Unido foram fundamentais para a fundação das primeiras igrejas protestantes de língua portuguesa no Brasil, marcando o início de uma era de crescente influência protestante nas esferas social e político-brasileiras (HELGEN; HOEK, 2022).

Nos círculos das igrejas protestantes tradicionais, os representantes adotaram uma prática musical desvinculada da cultura local, pautada nos padrões europeus e norte-americanos, desencorajando assim o desenvolvimento de uma expressão religiosa brasileira. Conforme apontado por DOLGHIE (2004), a hinódia tradicional do protestantismo brasileiro era essencialmente uma importação de hinos folclóricos americanos, os quais eram considerados como sagrados e destinados à adoração divina.

Neste contexto é necessário exemplificar alguns termos c omo: a hinódia, que engloba toda a produção de hinos protestantes, e a hinologia oficial, que se refere aos hinos autorizados para uso nas igrejas protestantes. Nesse sentido, somente os hinos importados dos Estados Unidos eram reconhecidos como parte da hinologia oficial, marginalizando assim uma produção musical independente, como os corinhos que emergiram nos anos 50, posteriormente denominados cânticos. Contudo, estes cânticos eram restritos a encontros de jovens, acampamentos e Escolas Dominicais, não recebendo reconhecimento litúrgico e enfrentando resistência por parte da tradição hinódica conservadora (DOLGHIE, 2004).

A insatisfação com a prática musical e litúrgica nas igrejas protestantes tradicionais abriu espaço para o surgimento do movimento evangélico, especialmente impulsionado pela Igreja Renascer em Cristo, fundada em São Paulo em 1986 por Estevam Hernandes e sua esposa, Sônia Hernandes. A Igreja Renascer adotou estratégias de marketing eficazes, identificou um nicho mercadológico entre os jovens, valorizou a condição dos músicos ao oferecer liberdade estilística e incentivo ao crescimento profissional, e rompeu com a hinódia tradicional, adotando estilos musicais populares que se tornaram parte da liturgia oficial da igreja (DOLGHIE, 2004).

É relevante destacar que a música cristã no Brasil recebeu reconhecimento como expressão cultural por meio de uma emenda à Lei nº 8.313/91, a Lei de Incentivo à Cultura, em 9 de janeiro de 2012. Contudo, a aceitação dessa política pública pela sociedade foi controversa. Apesar da conquista política, o reconhecimento do status de arte para a música cristã não é uniforme, pois há uma tensão envolvendo a interação cultural, religiosa e política. SANT'ANA (2013) argumenta que a música cristã poderia ser considerada parte da cultura devido a sua

natureza musical, mas que sua associação com experiências religiosas pode ser vista como um obstáculo a essa categorização, uma "distorção da linguagem da cultura" que não inclui a religião para receber benefícios legais. A valorização da diversidade nos gêneros musicais é contrastada com a conotação de "catequese", que sugere uma resistência ao diferente e uma imposição religiosa (SANT'ANA, 2013).

A trajetória da música cristã no Brasil implica em um debate amplo e multidimensional, exigindo de quem empreende essa tarefa levar em consideração aspectos históricos, sociais, políticos, culturais percebendo as transformações da sociedade contemporânea. Percebemos, diante disso, que não é algo novo que surgiu nesse contexto de estudo, mas um refinamento para entender os impactos sociais e culturais dessas composições. Desta forma tomaremos como objeto de pesquisa a presença de elementos tradicionais de adoração em composições de música cristã, considerando suas influências e tendências (CUNHA, 2007).

Em muitas composições de música cristã contemporânea, é possível identificar uma continuidade com elementos tradicionais de adoração, com letras que exaltam a divindade, expressões de gratidão e louvor, e melodias que trazem uma atmosfera de espiritualidade. Esses elementos essenciais têm raízes profundas na história evangélica e continuam a ressoar com os fiéis, proporcionando uma conexão emocional e espiritual com sua fé (MENDONÇA, 2009).

É importante ressaltar que à medida que a música cristã evolui para acompanhar as mudanças culturais e sociais, esses elementos tradicionais muitas vezes são adaptados para se adequar a novos contextos e audiências. Por exemplo, as letras podem abordar questões contemporâneas, como desafios da vida moderna, questões sociais e temas relevantes para a juventude. As melodias também podem incorporar influências m usicais c ontemporâneas, c omo o p op, o r ock o u o hip-hop, para alcançar um público diversificado e engajar os ouvintes de maneiras inovadoras (MENDONÇA, 2009).

Nesse sentido, a tecnologia desempenha um papel importante na forma como os elementos tradicionais de adoração são apresentados na música cristã contemporânea. Gravações de alta qualidade, vídeos musicais elaborados e plataformas de streaming digital permitem que as composições alcancem um público global de maneiras que eram inimagináveis nas gerações passadas. Isso não apenas amplifica a mensagem das músicas, mas também influencia a maneira como são percebidas e interpretadas pelos ouvintes (WAINER, 2017).

1.2 Objetivos do Trabalho

Nessa perspectiva, indagamos: a criação e aplicação de um modelo classificador das composições de música cristã no Brasil podem evidenciar a transformação das músicas, antes de adoração a Deus, em abordagens espirituais, reflexões cristãs ou ensinamentos morais, mas que não têm a adoração a Deus como objetivo? E como criar método para suportar essa avaliação das músicas?

Isto posto, a presente pesquisa baseou-se em dois objetivos:

a) criar diretrizes para a classificação de músicas cristãs; e
b) investigar se as composições de música cristã, em comparação com as de décadas anteriores, apresentam tendências na relação de adoração.

Desta forma o trabalho se divide da seguinte forma: Introdução, Referencial teórico, Trabalhos Relacionados, Metodologia e Resultados e Discussões, onde:

a) Introdução - abordamos a motivação do problema e como isso é importante dentro da sociedade em que vivemos;
b) Referencial Teórico - faz uma contextualização do algoritmo utilizado e de temas relevantes para o entendimento da dissertação;
c) Trabalhos Relacionados - relaciona estudos na área de reconhecimento de sentimento e como temos diversas abordagens;
d) Metodologia - descreve como o algoritmo foi implementado e quais variâncias foram aplicas e;
e) Resultados e Discussões - onde se apresenta os resultados alcançados e como isso nos retorna para as perguntas norteadoras da nossa pesquisa e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, apresentaremos uma base conceitual necessária para a compreensão do trabalho desenvolvido. Pretendemos não apenas contextualizar nossa pesquisa dentro do campo acadêmico, mas também oferecer ao leitor uma visão clara dos assuntos teóricos que suportam nosso estudo.

2.1 Conceituação Para Estudos Envolvendo Base de Dados e Técnicas de Aprendizagem de Máquina

Esta seção tem como objetivo fornecer uma base compreensível para as nomenclaturas e classificações utilizadas ao longo desta dissertação. Será apresentada uma explicação de cada termo relevante, juntamente com exemplos elucidativos, a fim de facilitar a compreensão das seções subsequentes.

Corpus é uma coleção de textos escritos ou falados que são selecionados e organizados para fins de análise linguística ou de pesquisa. Esses textos podem ser de uma única língua ou de várias línguas, e são frequentemente utilizados para estudar diferentes aspectos da linguagem, como gramática, semântica, entre outros (AIJMER; RüHLEMANN, 2014).

O termo Corpora é o plural de corpus e pode ser utilizado para se referir a várias coleções de textos usados em análises linguísticas ou em diferentes áreas de pesquisa. Sua importância se caracteriza no entendimento de como a língua é utilizada em diferentes contextos. Portanto, os corpora linguísticos são coleções estruturadas de textos que são usadas para análise linguística e pesquisa em diversos campos, como linguística computacional, tradução automática, processamento de linguagem natural, entre outros. Os corpora são elaborados com diferentes propósitos e podem incluir uma variedade de gêneros textuais, como livros, artigos acadêmicos, transcrições de conversas, entre outros (AIJMER; RüHLEMANN, 2014).

Os *embeddings* de palavras são representações numéricas num espaço vetorial contínuo, utilizados em modelos de linguagem para que estes aprendam associar cada palavra a um vetor de números reais, indicando uma similaridade para palavras semanticamente semelhantes (VASWANI et al., 2017). Como exemplo, consideremos as seguintes sentenças:

a) "o filme foi excelente e emocionante".
b) "a atuação dos atores foi incrível".

c) "a trama do filme é interessante".

Tabela 2.1 - Tabulação do funcionamento do vetor de *embeddings*

Palavra	Composição Vetor
"O"	[0.1, -0.2, 0.3]
"filme"	[-0.2, 0.3, 0.1]
"foi"	[0.3, -0.1, 0.2]
"excelente"	[-0.1, 0.4, -0.2]
"e"	[0.2, 0.1, 0.3]
"emocionante"	[0.4, 0.2, 0.5]
"A"	[-0.1, 0.3, -0.2]
"atuação"	[0.3, 0.5, 0.1]
"dos"	[0.2, -0.1, 0.4]
"atores"	[0.5, 0.2, 0.3]
"incrível"	[0.1, 0.2, 0.4]
"trama"	[0.2, 0.1, 0.2]
"do"	[0.1, -0.3, 0.2]
"é"	[-0.2, 0.1, 0.3]
"interessante"	[0.3, -0.2, 0.4]

Fonte: O Autor (2024)

A Tabela 2.1 mostra de forma simplificada como os *embeddings* de palavras são construídos a partir das sentenças acima. Esses vetores representam as palavras em um espaço de três dimensões, onde a proximidade entre os vetores reflete a semelhança semântica entre elas. Por exemplo, os vetores para "excelente" e "incrível", quando usamos a similaridade do cosseno (VOS et al., 2022; STEININGER et al., 2021) - medida que calcula o ângulo entre dois vetores, indica que essas palavras não são semanticamente similares no contexto dos exemplos propostos.

A *token*ização é o processo que fragmenta textos de linguagem natural (dados não estruturados) em unidades de informação contáveis como elementos isolados. Isso transforma imediatamente um texto não estruturado em uma estrutura de dados numérica adequada para aprendizado de máquina. Essas frequências podem ser utilizadas diretamente por computadores para acionar ações e respostas úteis. Alternativamente, podem servir como

características em um processo de aprendizado de máquina, induzindo decisões ou comportamentos mais complexos (LANE; HOWARD; HAPKE, 2019).

Utilizando as mesmas frases usadas no exemplo de *embeddings*, a *token*ização das três frases se daria da seguinte forma:

a) "O filme foi excelente e emocionante".

   Tokens: [’o’, ’film’, ’##e’, ’f’, ’##oi’, ’excel’, ’##ente’, ’e’, ’em’, ’##oc’, ’##ion’, ’##ante’, ’.’]

   ***Token* IDs**: conversão da frase em *tokens* [1051, 2143, 2063, 1042, 10448, 24970, 15781, 1041, 7861, 10085, 3258, 12956, 1012]

   Input IDs (com *tokens* especiais): adiciona-se os *tokens* especiais de início e final de frase [101, 1051, 2143, 2063, 1042, 10448, 24970, 15781, 1041, 7861, 10085, 3258, 12956, 1012, 102]

   Texto decodificado: [CLS] "O filme foi excelente e emocionante". [SEP];

b) "A atuação dos atores foi incrível".

   Tokens: [’a’, ’at’, ’##ua’, ’##cao’, ’dos’, ’at’, ’##ores’, ’f’, ’##oi’, ’inc’, ’##ri’, ’##vel’, ’.’]

   ***Token* IDs**: - conversão da frase em *tokens* [1037, 2012, 6692, 20808, 9998, 2012, 16610, 1042, 10448, 4297, 3089, 15985, 1012]

   Input IDs (com *tokens* especiais): - adiciona-se os *tokens* especiais de início e final de frase [101, 1037, 2012, 6692, 20808, 9998, 2012, 16610, 1042, 10448, 4297, 3089, 15985, 1012, 102]

   Texto decodificado: [CLS] "A atuação dos atores foi incrível". [SEP];

c) "A trama do filme é interessante".

   Tokens: [’a’, ’tram’, ’##a’, ’do’, ’film’, ’##e’, ’e’, ’inter’, ’##ess’, ’##ante’, ’.’]

   ***Token* IDs**: conversão da frase em *tokens* [1037, 12517, 2050, 2079, 2143, 2063, 1041, 6970, 7971, 12956, 1012]

   Input IDs (com *tokens* especiais): adiciona-se os *tokens* especiais de início e final de frase [101, 1037, 12517, 2050, 2079, 2143, 2063, 1041, 6970, 7971, 12956, 1012, 102]

   Texto decodificado: [CLS] "A trama do filme e interessante". [SEP].

Os Modelos Recorrentes, especialmente no contexto do Processamento de Linguagem Natural, referem-se a uma classe de redes neurais projetadas para processar sequências de dados como textos ou séries temporais. Esses modelos são chamados de recorrentes porque realizam a mesma tarefa para cada elemento de uma sequência, com a saída dependendo não

apenas da entrada atual mas também das entradas anteriores. Isso é possível graças à manutenção de um dado (podemos chamar de memória) interno que carrega informações ao longo da sequência, permitindo que o modelo capture dependências temporais e contextuais. As Redes Neurais Recorrentes (RNNs), *Long Short-term Memory* (LSTM) e *Gated Recurrent Units* (GRU) são modelos recorrentes e, ao contrário das redes neurais tradicionais, que assumem que as instâncias são independentes e identicamente distribuídas, os modelos recorrentes tratam os dados como sequências que possuem uma ordem específica. Isso os torna ideais para tarefas em que a ordem dos dados é importante (SCHMIDT, 2019).

Inferência de Linguagem Natural (NLI) é o processo pelo qual um sistema computacional ou um método de processamento de linguagem natural tenta compreender e extrair significado de textos ou falas de linguagem humana. Isso envolve a capacidade de interpretar o contexto, entender o significado das palavras, identificar padrões e relações entre diferentes partes do texto. Para realizar inferência de linguagem natural, os sistemas geralmente empregam técnicas de Processamento de Linguagem Natural (PLN), que incluem análise sintática, análise semântica, extração de informações e outras abordagens para entender e processar a linguagem humana de forma eficaz (BIRD; KLEIN; LOPER, 2009; JURAFSKY; MARTIN, 2008).

O Mecanismo de Atenção permite que o modelo se concentre em diferentes partes dos dados de entrada ao executar uma determinada tarefa, ponderando uma importância relativa de diferentes palavras em uma frase. Isso é feito calculando pontuações de atenção que determinam quanto cada parte da entrada deve influenciar cada parte da saída. Em outras palavras, permite que o modelo decida onde "prestar atenção" ao processar os dados de entrada (VASWANI et al., 2017).

As Conexões Residuais no *Transformer*, embora conceitualmente possam parecer similares ao processo sequencial de uma Rede Neurais Convolucionais (CNN) (HE et al., 2016), servem a um propósito diferente (XIE et al., 2023; VASWANI et al., 2017):

a) facilitam o Treinamento de *Transformer* com muitas camadas: As conexões residuais ajudam a mitigar o problema do desaparecimento do gradiente (problema causado pela atualização dos pesos nas camadas, o que efetivamente impede a rede de aprender de maneira eficaz. Esse fenômeno é particularmente problemático em redes com muitas camadas, pois o gradiente pode se aproximar de zero antes de alcançar as camadas mais distantes do ponto de entrada dos dados, comprometendo a capacidade de ajuste fino dos parâmetros da rede durante o treinamento),

permitindo que o gradiente flua diretamente através das camadas sem ser diluído por uma longa sequência de transformações lineares e não lineares;

b) preservação da Informação: Elas permitem que a informação da entrada seja preservada até as camadas mais profundas da rede, garantindo que detalhes importantes não sejam perdidos após múltiplas transformações.

Em resumo, enquanto as recorrências em CNNs são mecanismos para capturar dependências temporais sequenciais, as conexões residuais no *Transformer* são uma técnica de arquitetura de rede para preservar informações e facilitar o treinamento de modelos profundos (HE et al., 2016). Elas não introduzem recorrência na forma como os dados são processados; em vez disso, elas ajudam a rede a aprender de forma mais eficaz (XIE et al., 2023).

2.2 Métricas de Avaliação

No contexto do desenvolvimento e validação de modelos de classificação, as métricas de avaliação desempenham um papel essencial ao fornecer uma análise quantitativa do desempenho do modelo. A seguir exemplificaremos algumas abordagens para avaliar os modelos de PLN.

A Matriz de Confusão consiste em uma tabela de contingência de dois por dois que representa as disposições de um conjunto de instâncias em um classificador. Esta matriz permite calcular várias métricas comuns de desempenho, incluindo a taxa de verdadeiros positivos (sensibilidade ou *recall*), a taxa de falsos positivos, revocação e acurácia (M.; M.N, 2015). A matriz de confusão é fundamental para entender o desempenho de um classificador em termos de suas decisões corretas e incorretas (FAWCETT, 2006). A matriz de confusão é composta por quatro classificações e exemplificada no Quadro 2.1.

Para interpretar a Quadro 2.1, consideramos as seguintes definições:

Quadro 2.1 - Matriz de Confusão

	Classe Verdadeira	
Classe Hipotetizada	**p**	**n**
Y	Verdadeiros Positivos	Falsos Positivos
N	Falsos Negativos	Verdadeiros Negativos

Fonte: Fawcett (2006)

a) verdadeiros Positivos (TP): Número de instâncias positivas corretamente classificadas como positivas;
b) falsos Negativos (FN): Número de instâncias positivas incorretamente classificadas como negativas;
c) verdadeiros Negativos (TN): Número de instâncias negativas corretamente classificadas como negativas;
d) falsos Positivos (FP): Número de instâncias negativas incorretamente classificadas como positivas.

A partir destes valores, podemos compreender a Revocação (ou *Recall*) como a proporção de verdadeiros positivos entre todas as instâncias positivas reais (Equação 2.1). A Acurácia (ou *Accuracy*) é a proporção de todas as instâncias (positivas e negativas) que foram corretamente classificadas (Equação 2.2).

$$recall = \frac{TP}{P} \tag{2.1}$$

$$accuracy = \frac{TP+TN}{P+N} \tag{2.2}$$

A Taxa de Falsos Positivos (FPR) é a relação entre a quantidade de FN e a quantidade total de instâncias negativas (Equação 2.3). A Taxa de Verdadeiros Positivos (TPR) é a relação entre a quantidade de TP e a quantidade total de instâncias positivas (Equação 2.4).

$$FPR = \frac{FP}{N} \tag{2.3}$$

$$TPR = \frac{TPR}{P} \tag{2.4}$$

A Precisão (ou *Precision*) pode ser definida como relação de TP sobre a soma de TP e FP (Equação 2.5).

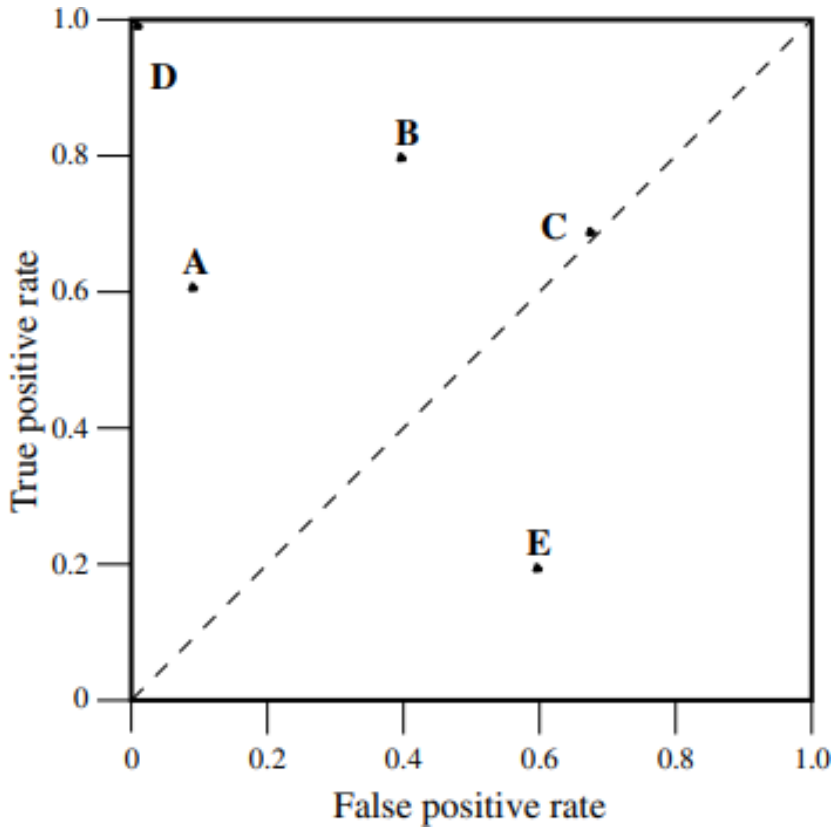
$$precision = \frac{TP}{TP+FP} \tag{2.5}$$

Por fim, a Medida F (ou F-measure) pode ser compreendida como média harmônica entre Precisão e Revocação (Equação 2.6).

$$f\_measure = \frac{2}{\frac{1}{precision}+\frac{1}{recall}} \quad (2.6)$$

A análise de Características Operacionais do Receptor (ou Receiver Operating Characteristics ROC) é uma técnica amplamente utilizada para visualizar, organizar e selecionar classificadores com base em seu desempenho. Esta abordagem iniciou-se na teoria da detecção de sinais, onde é usada para representar o trade-off entre as taxas de detecção corretas e as taxas de alarme falso dos classificadores. Posteriormente, passou-se a utilizar esta técnica para fins de aprendizado de máquina (FAWCETT, 2006). A base da análise ROC reside na matriz de confusão, onde cria-se um gráfico bidimensional com a taxa de TP representada no eixo Y (vertical) e a taxa de FP no eixo X (horizontal). Estes gráficos representam os trade-offs relativos entre benefícios e custos de um classificador (FAWCETT, 2006; DEVLIN et al., 2019).

Figura 2.1 - Curva Básica ROC mostrando 5 classificadores discretos.



Fonte: Fawcett (2006)

Para exemplificar a curva ROC a Figura 2.1 apresentada por FAWCETT, (2006) exemplifica 5 classificadores discretos onde cada classificador gera um par de TPR e FPR que corresponde a um único ponto no espaço ROC. Analisando a imagem podemos entender que o classificador D, localizado no canto superior esquerdo, tem uma taxa de verdadeiros positivos de 1,0 e uma taxa de falsos positivos de 0,0, indicando uma classificação perfeita. O classificador B, mais próximo do centro superior, possui uma alta taxa de verdadeiros positivos e uma taxa moderada de falsos positivos, sugerindo um bom desempenho com alguns erros. O classificador C, localizado mais à direita e acima da linha diagonal, possui uma taxa de verdadeiros positivos razoável, mas também uma alta taxa de falsos positivos, o que indica um desempenho menos ideal. O classificador A, situado ao lado esquerdo do gráfico, tem uma taxa de verdadeiros positivos moderada e uma baixa taxa de falsos positivos, mostrando um comportamento conservador. Por fim, o classificador E, posicionado na parte inferior do gráfico, apresenta uma baixa taxa de verdadeiros positivos e uma alta taxa de falsos positivos, refletindo um desempenho ruim. Esses pontos ilustram as diferentes características e trade-offs de cada classificador no espaço ROC.

A área abaixo da curva, ou Area Under the Curve (AUC) é uma métrica que pode ser interpretada como a probabilidade de uma classificação correta em que uma instância positiva aleatória seja mais alta que uma instância negativa aleatória. (FAWCETT, 2006; M.; M.N, 2015). A AUC varia de 0 a 1, onde :

a) AUC = 1: Indica um classificador que separa perfeitamente todas as instâncias positivas e negativas;
b) AUC = 0, 5: Indica um classificador que não tem poder discriminatório, equivalente a adivinhação aleatória;
c) AUC < 0, 5: Indica um classificador com desempenho pior do que o acaso, o que é raro e geralmente indica um erro no modelo ou nos dados.

A perda de entropia cruzada, ou Cross-Entropy Loss, é uma métrica de desempenho utilizada em modelos de classificação que quantifica a diferença entre a distribuição real dos dados e a distribuição prevista pelo modelo. Em termos simples, a Cross-Entropy Loss mede o quão bem o modelo está prevendo as classes corretas. Quanto menor o valor da Cross-Entropy Loss, melhor o desempenho do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016). Em uma classificação binária, a fórmula da Cross-Entropy Loss é dada pela Equação 2.7, onde y é o rótulo verdadeiro (0 ou 1) e p é a probabilidade prevista pelo modelo para a classe positiva.

$$Cross - Entropy\ Loss\ =\ -(y\ log(p)\ +\ (1\ -\ y)\ log(1\ -\ p)) \qquad (2.7)$$

A Equação 2.7 é um caso especial do cálculo do Cross-Entropy Loss multiclasse com C = 2. A Equação 2.8 é definida para casos com mais de duas classes.

Neste caso, (1) C é o número de classes, (2) yo,c é o rótulo binário (1 ou 0) indicando se a classe correta para a observação o é c, e (3) po,c é a probabilidade prevista para a observação o ser da classe c.

$$Cross - Entropy\ Loss\ =\ -\ \sum_{c=1}^{C} y_{o,c}\ log(p_{o,c}) \qquad (2.8)$$

2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é um campo interdisciplinar que se dedica à compreensão das linguagens naturais e à sua aplicação na interação entre humanos e computadores. As linguagens naturais são intrinsecamente complexas, e muitos problemas de PLN apresentam desafios que demandam soluções precisas. Assim, as pesquisas iniciais de PLN dependiam fortemente de regras e heurísticas definidas manualmente para realizar tarefas como análise sintática, semântica e pragmática (JURAFSKY; MARTIN, 2008).

Com o aumento dos dados textuais disponíveis na internet e em outras fontes digitais, foi necessária a aplicação de técnicas de aprendizado de máquina e processamento estatístico de linguagem natural em grande escala. Isso possibilitou que os sistemas de PLN fossem treinados em conjuntos de dados extensos, aprendendo padrões linguísticos complexos e construindo modelos probabilísticos que capturam a variabilidade e a ambiguidade da linguagem natural de maneira mais eficaz (GUDIVADA; ARBABIFARD, 2018; HALEVY; NORVIG; PEREIRA, 2009).

Essa mudança de paradigma, impulsionada por dados, permite lidar com a complexidade inerente às linguagens naturais, aproveitando a capacidade de aprendizado de máquina para extrair padrões e informações significativas a partir de grandes volumes de dados textuais. Assim, modelos baseados em dados tornam-se ferramentas valiosas para enfrentar desafios e m t arefas e específicas de PLN, promovendo avanços notáveis na compreensão e na geração de linguagem natural, além de abrir novas possibilidades para aplicações sofisticadas e interações mais naturais entre humanos e sistemas computacionais (GUDIVADA; ARBABIFARD, 2018; JORDAN; MITCHELL, 2015; BIRD; KLEIN; LOPER, 2009; INDURKHYA; DAMERAU,

2010). Neste contexto de dados e de forma abrangente, a análise de um problema pode ser dividida nos seguintes passos (que serão explicados mais detalhados posteriormente):

a) definição do Tipo de Tarefa: D efinir qual o ob jetivo do trabalho (classificação de texto, tradução automática, análise de sentimento) e escolher a abordagem apropriada com base na natureza da tarefa (supervisionada ou semi-supervisionada - esses tipos de tarefas serão explicados posteriormente);
b) pré-processamento de Dados: Coletar e limpar os dados, além de realizar tarefas de *token*ização, remoção de pontuação, etc;
c) pré-treinamento Modelos: A escolha do modelo a ser utilizado depende significativamente da disponibilidade e especificidade dos dados além dos recursos computacionais. Pode-se optar por utilizar modelos já treinados, conhecidos como modelos pré-treinados, que são ajustados (*fine-tuned*) para a tarefa específica, ou desenvolver um modelo completamente novo, treinando-o a partir do zero com os dados coletados. Essa decisão estratégica é crucial para o sucesso da aplicação de PLN;
d) extração de Recursos: A extração eficaz de características é fundamental para o desempenho do modelo de PLN. As técnicas incluem a transformação de texto bruto em *embeddings* de palavras, que capturam contextos semânticos e relacionamentos entre termos além da extração de características sintáticas e gramaticais. Essas características são então utilizadas para treinar o modelo, permitindo que ele interprete e processe a linguagem natural de maneira mais eficiente;
e) seleção e Treinamento do Modelo: Selecionar o modelo apropriado (de acordo com as etapas anteriores), além de se basear com o tamanho dos dados e recursos disponíveis. Nesta parte os dados também são divididos entre conjuntos de treinamento, validação e teste. Os Ajustes dos modelos também são feitos nessa parte do processo;
f) avaliação e Ajuste: Avaliar o desempenho e ajustar (se necessário) o modelo com métricas já conhecidas, como: precisão, *recall*, F1-score, etc.

2.4 Técnicas e Modelos no PLN

Dentro do universo do PLN existem maneiras de apresentar e abordar um problema. A escolha da técnica ideal depende das características dos dados e qual o tipo do problema será

endereçado. Dentre essas técnicas indicaremos algumas e sua utilização (INDURKHYA; DAMERAU, 2010):

a) modelos baseados em *Transformer*s: Os modelos baseados em *Transformer*s, como o BERT (Bidirectional *Encoder* Representations from *Transformer*s) e o GPT (Generative Pre-trained *Transformer*), são usados em tarefas de tradução automática, sumarização de texto, resposta a perguntas, entre outras (DEVLIN et al., 2019; RADFORD; NARASIMHAN, 2018). Eles são construídos sobre arquiteturas de transformadores e, por ser o foco deste trabalho, será mais detalhado à frente;
b) CNNs: Embora as CNNs sejam mais conhecidas por sua aplicação no processamento de imagens, elas também são aplicáveis ao PLN. As CNNs são capazes de identificar padrões locais em uma sequência de palavras, tornando-as úteis para tarefas como classificação de texto e análise de sentimentos (KIM,2014);
c) modelos de Memória de Curto e Longo Prazo (LSTM): As LSTMs são um tipo de rede neural recorrente que se destaca por sua capacidade de manter e acessar informações de longo prazo. Essa característica é especialmente útil em tarefas onde a dependência de contexto de longo prazo é relevante (HOCHREITER; SCHMIDHUBER, 1997);
d) modelos de Máquinas de Vetores de Suporte (SVM): Embora menos comuns em comparação com modelos baseados em redes neurais, as SVMs são empregadas em tarefas de classificação de texto. Sua eficácia na separação de classes lineares em um espaço vetorial torna-as adequadas para detecção de spam, análise de sentimentos e outras tarefas de classificação (JOACHIMS, 1998);
e) modelos de Máxima Entropia (MaxEnt): Também conhecidos como modelos de classificação de multinomial de máxima entropia, são empregados em tarefas de classificação de texto onde a distribuição de probabilidade é modelada para cada classe possível. Eles oferecem flexibilidade na modelagem da distribuição de probabilidade, tornando-os úteis em uma variedade de contextos (T; DELLAPIETRA; PIETRA, 2002).

Além dos modelos citados, a forma como o contexto é analisado também é crucial no uso dos métodos. Podemos ter o contexto analisado de forma unidirecional ou bidirecional. A diferença entre as duas abordagens reside no uso da frase: enquanto a análise unidirecional considera apenas o contexto anterior ao processar uma palavra específica (no contexto da língua portuguesa, da esquerda para a direita), o modelo bidirecional captura informações em ambas direções. Por exemplo, na frase "João comprou pão na padaria", se a análise for a palavra

"pão", o método unidirecional usaria as palavras "João" e "comprou" para entender o significado de "pão". Já o modelo bidirecional consideraria as palavras "João", "comprou", "na" e "padaria" para entender o significado de "pão". A análise unidirecional limita a compreensão, afetando o treinamento do modelo utilizado (DEVLIN et al., 2019).

Outro ponto fundamental é baseado no tipo de aprendizado que será empregado na tarefa e que pode ter as seguintes classificações:

a) tarefas Supervisionadas: o modelo é treinado usando pares de entrada e saída rotulados. Por exemplo, na classificação de texto, o modelo recebe um texto como entrada e uma categoria pré-definida como saída (normalmente definida por um especialista) (A; PULABAIGARI; B, 2018);
b) tarefas Não Supervisionadas: o modelo é treinado usando apenas os dados de entrada, sem rótulos de saída correspondentes. O objetivo é encontrar estrutura nos dados ou representações úteis sem orientação externa (A; PULABAIGARI; B, 2018);
c) tarefas Semi-Supervisionadas: o modelo é treinado usando uma combinação de dados rotulados e não rotulados. Isso é útil quando há uma quantidade limitada de dados rotulados disponíveis, mas uma grande quantidade de dados não rotulados. Exemplos incluem classificação de documentos com poucos rótulos disponíveis (A; PULABAIGARI; B, 2018);
d) tarefas de Aprendizado por Reforço: o modelo aprende a tomar ações sequenciais para maximizar uma recompensa acumulada. O modelo recebe uma resposta sobre a qualidade de suas ações num contexto de recompensa, mas não recebe instruções explícitas sobre quais ações tomar. Exemplos incluem sistemas de diálogo, onde o modelo aprende a responder às perguntas dos usuários interagindo com eles (SUTTON; BARTO, 2015);
e) tarefas de Aprendizado Ativo: o modelo pode ser simplesmente descrito como um método de aprendizado de máquina que aprende seletivamente a partir dos dados disponíveis e obtém maior precisão com menos dados rotulados (MA et al., 2023).

2.5 *Transformers*

Criado em 2017 por VASWANI et al., o conceito de *Transformer*s surgiu para desenvolver uma arquitetura que evitasse o uso de modelos recorrentes até então amplamente

utilizados. A ideia foi criar um modelo que confiasse inteiramente no mecanismo de atenção para estabelecer uma relação global entre os dados de entrada e saída.

Ao ser o primeiro modelo criado sem usar redes neurais sequenciais ou convolucionais e ao depender exclusivamente de mecanismos de autoatenção, os *Transformer*s têm a capacidade de visualizar todos os *tokens* de entrada simultaneamente. Isso permite uma modelagem eficaz de dependências em sequências longas em tempo constante, viabilizando uma paralelização substancialmente maior. A arquitetura dos *Transformer*s pode ser descrita resumidamente e visualizada na Figura 2.2 (VASWANI et al., 2017).

Para entender a Figura 2.2, é importante examinar a estrutura do modelo *Transformer*, que compreende várias camadas e mecanismos específicos para processar e gerar sequências de texto de maneira eficiente. A seguir, detalhamos os componentes principais dessa arquitetura e seu funcionamento passo a passo:

2.5.1 Codificador (*Encoder*)

O processo inicia-se com a entrada dos dados, que passam por uma camada de *embedding* combinada com codificação posicional, como mostrado na parte inferior esquerda da Figura 2.2. A codificação posicional é essencial, pois fornece informações sobre a ordem dos *tokens* na sequência de entrada, compensando a ausência de processamento sequencial.

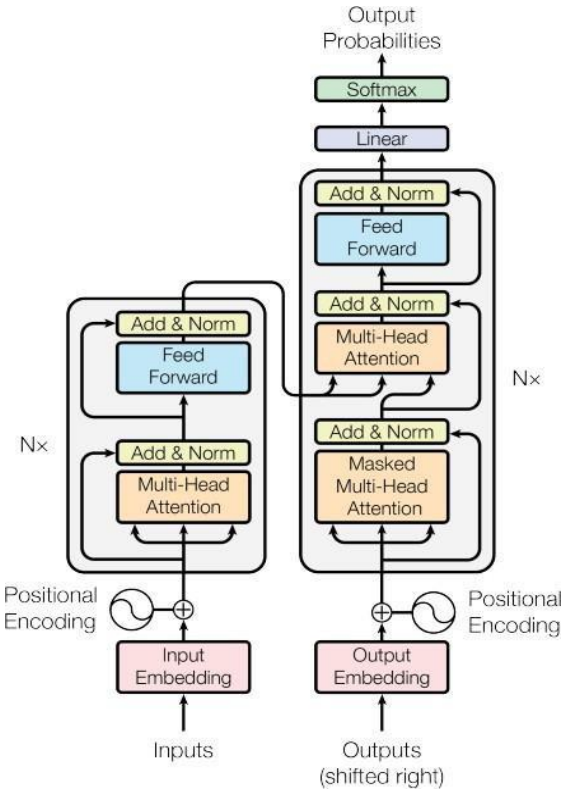
Usando a Tabela 2.1 como exemplo de entrada, usando a sequencia de *tokens* "O filme foi excelente" e supondo que as codificações posicionais para as posições de 1 a 4 no espaço de 3 dimensões sejam:

Tabela 2.2 - Codificações Posicionais

Posição	Codificação Vetorial
1	[0.0, 0.1, 0.2]
2	[0.1, 0.2, 0.3]
3	[0.2, 0.3, 0.4]
4	[0.3, 0.4, 0.5]

Fonte: O Autor (2024)

Figura 2.2 - Estrutura dos *Transformer*s.



Fonte: VASWANI et al. (2017)

A Combinação de *embeddings* e codificações posicionais (Tabela 2.2 é feita por soma de elemento a elemento, podendo ser visualizado na Tabela 2.3.

Tabela 2.3 - Combinação de *Embeddings* e Codificações Posicionais

Palavra	**Posição**	***Embedding***	**Codificação Posicional**	***Embedding* Combinado**
"O"	1	[0.1, -0.2, 0.3]	[0.0, 0.1, 0.2]	[0.1, -0.1, 0.5]
"filme"	2	[-0.2, 0.3, 0.1]	[0.1, 0.2, 0.3]	[-0.1, 0.5, 0.4]
"foi"	3	[0.3, -0.1, 0.2]	[0.2, 0.3, 0.4]	[0.5, 0.2, 0.6]
"excelente"	4	[-0.1, 0.4, -0.2]	[0.3, 0.4, 0.5]	[0.2, 0.8, 0.3]

Fonte: O Autor (2024)

Assim o resultado final da sequência de entrada de "O filme foi excelente" seria representada pelos vetores da Tabela 2.4.

Tabela 2.4 - *Embeddings* Combinados

Palavra	***Embedding* Combinado**
"O"	[0.1, -0.1, 0.5]
"filme"	[-0.1, 0.5, 0.4]
"foi"	[0.5, 0.2, 0.6]
"excelente"	[0.2, 0.8, 0.3]

Fonte: O Autor (2024)

Na sequência do decodificador encontramos N_x camadas idênticas, onde N_x geralmente é igual a 6. Cada uma dessas camadas contém duas subcamadas principais:

a) autoatenção com Múltiplas Cabeças (*Multi-Head Attention*): Esta subcamada permite que o modelo se concentre em diferentes partes da sequência de entrada simultaneamente, estabelecendo dependências de longo alcance entre os *tokens*;
b) rede *Feed-Forward*: Após a autoatenção, os dados são processados por uma rede *Feed-Forward* que aplica transformações não lineares para extrair características mais complexas. A rede *Feed-Forward* é composta por duas camadas lineares com uma função de ativação não linear (geralmente ReLU) entre elas(VASWANI et al., 2017). Esta estrutura permite ao modelo aprender e representar relações mais complexas entre os dados.

Cada uma dessas subcamadas é seguida por uma normalização de camada (*Add & Norm*), que estabiliza e acelera o treinamento do modelo. Conexões residuais são somadas à saída de cada subcamada para facilitar o fluxo do gradiente durante o treinamento.

2.5.2 Decodificador (Decoder)

A estrutura do decodificador, mostrada à direita na Figura 2.2, é similar a do codificador com algumas diferenças importantes. A entrada do decodificador é uma sequência de *embeddings* de saída deslocados para a direita, combinada com codificação posicional. Isso permite que o modelo gere a sequência de saída de maneira auto-regressiva, produzindo um *token* por vez (VASWANI et al., 2017).

Para exemplificar, consideremos a sequência "O filme foi excelente". Os *embeddings* dos *tokens* são apresentados na Tabela 2.1, enquanto as codificações posicionais são mostradas na Tabela 2.2. A combinação dessas informações gera a Tabela 2.5.

Tabela 2.5 - Combinação de *Embeddings* e Codificações Posicionais para o Decodificador

Palavra	**Passo 1**	**Passo 2**	**Passo 3**	**Passo 4**
"O"	[0.1, -0.1, 0.5]	[0.1, -0.1, 0.5]	[0.1, -0.1, 0.5]	[0.1, -0.1, 0.5]
"filme"	-	[-0.1, 0.5, 0.4]	[-0.1, 0.5, 0.4]	[-0.1, 0.5, 0.4]
"foi"	-	-	[0.5, 0.2, 0.6]	[0.5, 0.2, 0.6]
"excelente"	-	-	-	[0.2, 0.8, 0.3]

Fonte: O Autor (2024)

No decodificador, o processo de geração da sequência de saída ocorre de maneira auto-regressiva:

a) inicialização: O primeiro *token* de entrada é geralmente especial de início de sequência (por exemplo, <s>);
b) geração de *Tokens*: Em cada passo, o modelo gera um *token* da sequência de saída que é usado como parte da entrada no próximo passo;
c) deslocamento: A sequência de *tokens* de saída gerados até o momento é deslocada para a direita, e o próximo *token* é previsto usando essa sequência deslocada;
d) iteração: Este processo continua até que um *token* especial de fim de sequência (por exemplo, </s>) seja gerado ou até que o comprimento máximo da sequência seja atingido.

Esse mecanismo permite que o decodificador construa a sequência de saída *token* por *token*, utilizando a informação contextual de todos os *tokens* gerados anteriormente.

O decodificador também é composto por N_x camadas idênticas, cada uma contendo três subcamadas principais:

a) atenção Mascarada com Múltiplas Cabeças (*Masked Multi-Head Attention*): A primeira subcamada aplica uma máscara para impedir que o modelo acesse informações futuras na sequência de saída, garantindo que a previsão de um *token* dependa apenas dos *tokens* anteriores;
b) atenção com Múltiplas Cabeças (*Multi-Head Attention*): A segunda subcamada realiza uma atenção cruzada entre a saída do codificador e a entrada do decodificador, permitindo que o decodificador se concentre nas partes relevantes da sequência de entrada;
c) rede *Feed-Forward*: Semelhante ao codificador, a última subcamada é uma rede *Feed-Forward* que processa as informações transformadas.

Assim como no codificador, cada subcamada no decodificador é seguida por conexões residuais e normalização de camada (*Add & Norm*).

2.5.3 Transferência de Informação do Codificador para o Decodificador

A transferência de informação do codificador para o decodificador ocorre na subcamada de Atenção com Múltiplas Cabeças no decodificador. Especificamente, após a aplicação da atenção mascarada na entrada do decodificador, a segunda subcamada de atenção no decodificador utiliza a saída do codificador para realizar uma atenção cruzada.

A atenção cruzada não corrige diretamente os erros, mas fornece o contexto necessário para que o decodificador faça predições informadas em cada passo. Esse mecanismo assegura que cada *token* gerado na sequência de saída considere a sequência de entrada completa, permitindo ao modelo gerar textos mais coerentes e relevantes. A importância da atenção cruzada reside em sua capacidade de utilizar a informação da entrada para influenciar as predições de saída, garantindo uma integração contínua do contexto em todo o processo de geração de texto. Na Figura 2.2 isso é representado pelas setas que vão do bloco do codificador para o bloco do decodificador, indicando a passagem de informação entre os dois componentes.

2.5.4 Camadas de Saída e a Função da Camada Linear

Na etapa final, a saída das camadas do decodificador é transformada em logits por meio de uma camada linear (*Linear*), conforme mostrado na parte superior da Figura 2.2. Essa camada linear recebe os vetores de saída das sub-camadas do decodificador e os transforma em um novo espaço de vetores, onde a dimensionalidade do novo espaço corresponde ao tamanho do vocabulário do modelo. Basicamente, a camada linear realiza uma transformação linear, que pode ser representada matematicamente como:

$$Logits = W\ X\ Vetores + b \tag{2.9}$$

onde W é uma matriz de pesos e b é um vetor de *bias*.

Os valores resultantes dessa transformação são chamados de logits e representam pontuações não normalizadas para cada palavra possível no vocabulário do modelo. Eles indicam o quanto o modelo acredita que cada palavra do vocabulário poderia ser o próximo *token* na sequência de saída.

Após a camada linear, uma função Softmax é aplicada aos logits.convertendo esses logits em probabilidades, transformando as pontuações não normalizadas em valores entre 0 e 1 que

somam 1. Essas probabilidades são usadas para prever o próximo *token* na sequência de saída, escolhendo o *token* com a maior probabilidade.

2.5.5 Consistência Dimensional

Um aspecto crucial da arquitetura *Transformer* é a consistência dimensional. A dimensionalidade do modelo (d_{model} = 512) é mantida constante ao longo de todas as subcamadas e camadas de *embedding*, tanto no codificador quanto no decodificador. Isso padroniza o processamento dos dados em toda a arquitetura, facilitando a implementação e o treinamento do modelo.

2.6 Bidirectional Encoder Representations from Transformers – BERT

A relação entre o PLN e o modelo Bidirectional *Encoder* Representations from *Transformer*s (BERT) representa avanços notáveis na compreensão e geração de linguagem por sistemas computacionais. Esses avanços abrangem diversos aspectos, como a compreensão do contexto bidirecional em uma frase, a identificação de paráfrases (frases com o mesmo significado expressa de forma diferente), a reescrita de texto, entre outros. As melhorias do resultado apresentado pelo BERT foram (DEVLIN et al., 2019):

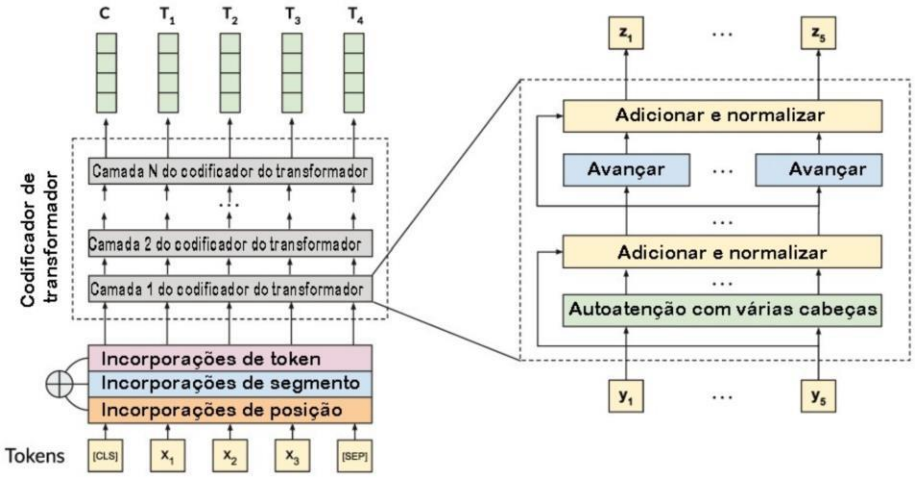
a) compreensão e interpretação de texto - Compreensão de leitura e *Question Answering* (QA): novos padrões no SQuAD (*Stanford Question Answering Dataset*) e outros benchmarks de QA, melhorando a precisão na localização de respostas em textos;
b) NLI - Melhorou o desempenho em tarefas de NLI, como no SNLI (*Stanford Natural Language Inference*) e *Multi*NLI, ajudando modelos a entender melhor a relação lógica entre sentenças (contradição, neutralidade, implicação);
c) reconhecimento de entidades nomeadas (NER) - O BERT mostrou melhorias notáveis no reconhecimento de entidades nomeadas em textos, como nomes de pessoas, locais, organizações, etc., aumentando a precisão na identificação e classificação dessas entidades;
d) previsão de próxima sentença - Uma das inovações do BERT foi o treinamento para prever se uma sentença é logicamente sequencial a outra, melhorando a compreensão do contexto e da coesão do texto;

e) preenchimento de lacunas (*Masked Language Model* ou Modelo de Linguagem Mascarada (MLM)) - O treinamento com mascaramento de palavras (onde algumas palavras do texto são ocultadas) melhorou a habilidade do modelo de inferir a palavra correta baseada no contexto, aprimorando a compreensão contextual e a geração de texto;

f) desambiguação de palavras - BERT aprimorou a capacidade de distinguir entre diferentes significados de uma palavra com base no contexto, crucial para muitas tarefas de NLP que dependem da compreensão precisa do significado das palavras.

O BERT é formado por um Codificador *Transformer* como descrito anteriormente e formulado por (VASWANI et al., 2017). Essas características abrem caminho para o treinamento de modelos mais profundos e sequências mais extensas. A Figura 2.3 ilustra um diagrama da arquitetura do BERT, e cada componente será explorado nas seções subsequentes.

Figura 2.3 - Diagrama do modelo BERT.



Fonte: SOUZA; NOGUEIRA; LOTUFO, (2020)

Resumidamente o funcionamento do BERT começa com a *token*ização do texto usando o método *WordPiece*, que quebra o texto em unidades menores, permitindo ao modelo gerenciar vocabulários desconhecidos eficazmente. Adicionalmente, *tokens* especiais são criados para processar as entradas ([CLS]: Um *token* especial adicionado no início de cada entrada, cuja representação final é usada como agregado da sequência para tarefas de classificação; [SEP]: Um *token* usado para separar sentenças ou segmentos dentro da entrada; [MASK]: Um *token* usado para substituir as palavras mascaradas durante o treinamento MLM. A Figura 2.4 mostra o uso desses *tokens*)(DEVLIN et al., 2019)

Após a *token*ização, os *tokens* são transformados em *embeddings* (incorporações) de entrada, estes são então processados por múltiplas camadas de *transformer*, no coração do BERT, onde operações de autoatenção permitem que cada *token* considere todos os outros na sentença para uma compreensão contextual completa. Esta capacidade de capturar contexto bidirecional distingue o BERT de modelos anteriores que analisavam o texto linearmente. (DEVLIN et al., 2019)

O treinamento do BERT é composto por duas partes principais: o MLM e a Previsão de Próxima Sentença (NSP). No MLM, uma fração dos *tokens* de entrada é aleatoriamente mascarada, e o modelo aprende a prevê-los com base no contexto fornecido pelos *tokens* visíveis. Isso ensina ao BERT uma compreensão profunda do contexto linguístico. Paralelamente, na tarefa NSP, o modelo é treinado para discernir se uma sentença segue logicamente outra, aprimorando sua habilidade de compreender relações entre sentenças. (DEVLIN et al., 2019)

Finalmente, a aplicabilidade prática do BERT é realizada através do ajuste fino, onde o modelo pré-treinado é adaptado para tarefas específicas de NLP adicionando-se camadas de saída específicas e ajustando-se em conjuntos de dados particulares.

Esse processo permite que o BERT demonstre sua versatilidade e eficácia em uma ampla gama de aplicações, desde a classificação de texto até o reconhecimento de entidades nomeadas (DEVLIN et al., 2019).

2.6.1 Entrada

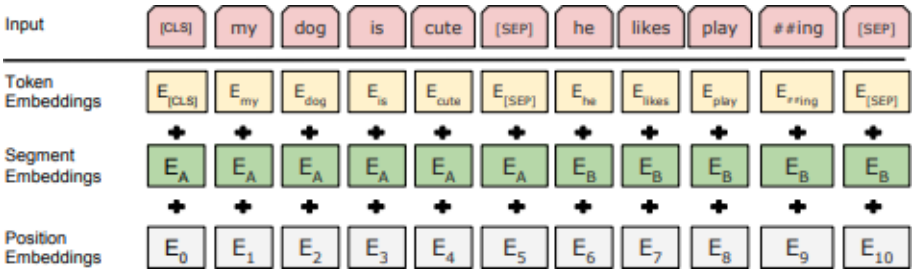
O BERT pode aceitar como entrada uma única sentença ou um par de sentenças. O termo refere-se à qualquer sequência de texto contínuo, não necessariamente uma sentença linguística específica. Por sua vez, "sequência" diz respeito à sequência de *tokens* de entrada, que pode ser composta por uma ou duas sentenças. Uma sequência de entrada é formada ao agrupar as sentenças com a inclusão de dois *tokens* especiais: [CLS], que é adicionado no início, e [SEP], usado para delimitar as sentenças. Uma única sentença é representada como [CLS] $x_1 \cdot \cdot \cdot x_n$ [SEP], enquanto um par de sentenças é representado como [CLS] $x_1 \cdot \cdot \cdot x_n$ [SEP] $y_1 \cdot \cdot \cdot y_m$ [SEP]. O *token* [SEP] funciona como um simples separador para indicar o final de uma sentença (DEVLIN et al., 2019).

Antes de ser inserida no modelo, uma sequência de *tokens* precisa ser convertida em representações vetoriais. Além dos *embeddings* simples de *token*, que mapeiam cada *token* do vocabulário para um vetor de *embedding* correspondente, o BERT incorpora cada *token* usando

dois tipos adicionais de *embeddings*: *embeddings* de posição e de segmento (DEVLIN et al., 2019).

A inclusão de *embeddings* de posição está diretamente ligada à natureza não sequencial da arquitetura *Transformer*. Enquanto modelos recorrentes levam em consideração a ordem sequencial ao processar uma palavra por vez, o *Transformer* opera em conjuntos de vetores e não possui uma noção intrínseca da ordem sequencial. Portanto, é necessário incluir informações sobre a posição relativa ou absoluta dos *tokens* na sequência na entrada. Este *embedding* de posição codifica informações de posição absoluta associando cada posição i (onde i pertence ao conjunto i ∈ 1, · · · , S) a um vetor de *embedding* com aprendizado já fixado, substituindo a codificação posicional do *Transformer*. A Figura 2.4 exemplifica este processo. O comprimento máximo da sequência S é um hiperparâmetro e foi definido com tamanho de 512 no trabalho original (DEVLIN et al., 2019).

Figura 2.4 – *Embeddings* de *Tokens*, Segmento e Posição.



Fonte: DEVLIN et al. (2019)

Os *embeddings* de segmento, por sua vez, estão relacionados à representação de entrada e são usados para distinguir duas sentenças, A e B, dentro de uma sequência. Cada *token* da sentença A é incorporado usando o *embedding* de segmento A, e cada *token* da sentença B é incorporado usando o *embedding* de segmento B. Para ilustrar o processo de incorporação para um par de sentenças a Tabela 2.6 exemplifica seu funcionamento. Formalmente, o vetor de incorporação para um *token* x_i pode ser expresso por:

$$E(x_i) = LayerNorm(E_{token}(x_i) + E_{pos}(i) + E_{seg}(A/B)) \in R^H \qquad (2.10)$$

onde LayerNorm representa a normalização de camada , E*token* ∈ RVxH esta é a matriz de *embeddings* de *tokens* (BA; KIROS; HINTON, 2016). Ela associa cada *token* no vocabulário V a um vetor de *embedding* H − dimensional. Portanto, para um vocabulário de

tamanho V , esta matriz terá V linhas e H colunas. Já $E_{pos} \in R^{SxH}$ é a matriz de *embeddings* de posição. Cada linha representa um vetor de posição para um *token* em uma sequência de comprimento máximo S. A dimensão H é a mesma que a dos *embeddings* de *token*, e E_{seg} refere-se aos *embeddings* de segmento A/B. Eles são utilizados para distinguir *tokens* pertencentes a diferentes sentenças dentro de uma sequência. Cada *token* de uma sentença A é incorporado usando o *embedding* de segmento A, e o mesmo se aplica à sentença B (SOUZA; NOGUEIRA; LOTUFO, 2020) (BA; KIROS; HINTON, 2016) (DEVLIN et al., 2019).

Tabela 2.6 - Tabulação do processo de incorporação de *embeddings* para um modelo BERT

Token	***Token Embedding* (E_*token*)**	***Embedding* de Segmento (E_A/E_B)**	***Embedding* de Posição (E_pos)**	***Embedding* Incorporado**
[CLS]	E_[CLS]	E_A	E_0	E_[CLS] + E_A + E_0
my	E_my	E_A	E_1	E_my + E_A + E_1
dog	E_dog	E_A	E_2	E_dog + E_A + E_2
is	E_is	E_A	E_3	E_is + E_A + E_3
cute	E_cute	E_A	E_4	E_cute + E_A + E_4
[SEP]	E_[SEP]	E_B	E_5	E_[SEP] + E_B + E_5
he	E_he	E_B	E_6	E_he + E_B + E_6
likes	E_likes	E_B	E_7	E_likes + E_B + E_7
play	E_play	E_B	E_8	E_play + E_B + E_8
##ing	E_##ing	E_B	E_9	E_##ing + E_B + E_9
[SEP]	E_[SEP]	E_B	E_10	E_[SEP] + E_B + E_10

Fonte: DEVLIN et al. (2019)

2.6.2 *Encoder*

O *encoder* pode ser dividido em cinco partes principais:

a) estrutura da camada: Cada camada de codificação no *Transformer* é formada por duas subcamadas essenciais: Autoatenção *Multi-Head* e *Feed-Forward*, como descrito em (VASWANI et al., 2017). A matriz $U \in R^{n \times H}$, que representa a

sequência intermediária de saída da subcamada anterior, é composta por n, o número de *tokens*, e H, a dimensão do modelo. Para manter as conexões residuais (método que auxilia no aprendizado e ajuda a preservar a informação entre as camadas) e garantir a estabilidade do treinamento, as dimensões de entrada e saída de cada subcamada são mantidas em H;

b) autoatenção *Multi-Head*: Essa camada processa todos os *tokens* da saída da camada anterior, permitindo que cada posição na sequência influencie todas as outras, o que enriquece a representação com informações contextuais (VASWANI et al., 2017);

c) camada de *Feed-Forward*: Composta por duas camadas totalmente conectadas, a camada de *Feed-Forward* é definida pela função $FNN(u) = Act(uW_1 + b_1)W_2 + b_2$, onde $u \in R^H$, $W_1 \in R^{H \times dff}$, $b_1 \in R^{dff}$, $W_2 \in R^{dff \times H}$, e $b_2 \in R^H$. A função de ativação Act é normalmente a ReLU, mas pode ser substituída pela GeLU para suavizar a não-linearidade, como em modelos como o BERT (HENDRYCKS; GIMPEL, 2016).

Essa função de ativação Act(x) é responsável por introduzir não linearidades na rede, permitindo que ela aprenda padrões mais complexos nos dados. Originalmente o *Transformer* tinha como função de ativação a ReLU (*Rectified Linear Unit*), definida como Act(x) = max(0, x), retirando valores negativos e mantendo somente os positivos. (VASWANI et al., 2017)

Já no BERT, a função de ativação utilizada é a GeLU (*Gaussian Error Linear Unit*), apresentada por Hendrycks e Gimpel em 2016. Ela é definida como:

$$GeLU(x) = 0{,}5 \,.\, x \,.\left(1 + tanh\left(\sqrt{\frac{\pi}{2}}.\left(x + 0{,}044715 \,.\, x^3\right)\right)\right) \qquad (2.11)$$

A GeLU serve como uma alternativa à função ReLU, mas com uma característica distinta: ela permite uma transição mais suave na ativação em resposta às variações dos valores de entrada. Em vez de uma alteração abrupta, a GeLU modula a ativação de forma mais gradual, proporcionando uma resposta mais refinada da função de ativação (HENDRYCKS; GIMPEL, 2016);

d) processamento da sequência: A entrada u é um vetor, e a camada é aplicada a cada posição da sequência de maneira separada e idêntica, ao contrário da atenção, que opera em toda a sequência simultaneamente (VASWANI et al., 2017);
e) camada de codificação: a saída da camada de codificação é uma sequência de vetores representando a informação contextualizada de cada posição na entrada original, incorporando informações de toda a sequência. Essa saída é então utilizada como entrada para a próxima camada ou etapa do modelo *Transformer* e pode ser descrito nos passos abaixo: (VASWANI et al., 2017)
   - A saída Z(i) da i-ésima camada de codificação é definida por:
   - Z(0) = E(x) (onde E(x) ∈ RnxH representa a sequência de *tokens* de entrada incorporados),
   - Y (i) = SubLayer(MultiHead(Z(i ⊗ 1), Z(i ⊗ 1), Z(i ⊗ 1)), Z(i ⊗ 1)),
   - Z(i) = SubLayer([FFN(y(i)1), ..., FFN(y(i)n)],Y (i)),
   - Z = Z(L) (onde L é o número total de camadas)

2.6.3 Autoatenção *Multi-Head*

A Autoatenção *Multi-Head* é um componente crucial na arquitetura *Transformer*, desempenhando um papel fundamental na modelagem de dependências de longo alcance em sequências de dados. Esta abordagem inovadora foi introduzida por Vaswani et al. (2017) como uma solução eficaz para lidar com a dependência contextual em tarefas complexas, como tradução automática e processamento de linguagem natural.

Antes da Autoatenção *Multi-Head*, modelos baseados em sequências, como LSTMs (*Long Short-Term Memory*) e GRUs (*Gated Recurrent Units*), eram amplamente utilizados para capturar relações sequenciais. No entanto, esses modelos enfrentavam desafios em lidar eficientemente com dependências extensas, devido à sua estrutura recorrente na qual os dados são processados de forma sequencial. A informação é passada de um passo de tempo para o próximo e pode sofrer uma degradação do gradiente. Este fenômeno se manifesta de duas formas principais: o desaparecimento do gradiente, onde os valores do gradiente se tornam progressivamente menores até se tornarem insignificantes, impedindo a rede de aprender correlações entre eventos distantes na sequência; e a explosão do gradiente, que envolve um aumento excessivo nos valores do gradiente, resultando em instabilidades durante o

aprendizado. Ambos problemas complicam o ajuste eficaz do modelo e sua capacidade de generalizar a partir de dados de treinamento (CHO et al., 2014; CHUNG et al., 2014)

A Autoatenção *Multi-Head* surge como uma alternativa revolucionária, especialmente destacada no *Transformer*, introduzindo um mecanismo flexível e paralelo para considerar diferentes partes da sequência simultaneamente. Ao dividir a atenção em múltiplas cabeças, o modelo pode focar em diferentes partes da entrada, permitindo a captura de informações contextuais mais ricas e complexas (SHAW; USZKOREIT; VASWANI, 2018; VASWANI et al., 2017).

2.6.3.1 Mecanismo de Atenção

O cerne da Autoatenção *Multi-Head* é o mecanismo de atenção, que originalmente surgiu em contextos de tradução automática neural (BAHDANAU; CHO; BENGIO, 2016). Esse mecanismo permite que o modelo relacione informações de diferentes posições em sequências longas, superando as limitações de modelos sequenciais tradicionais.

Na atenção do *Transformer*, o cálculo é realizado através de uma combinação linear ponderada dos vetores de valor. Essa ponderação é determinada pela compatibilidade entre o vetor de consulta e os vetores de chave. A fórmula geral para calcular a atenção é:

$$attn(Q, K, V) = \ so\ f\ tmax\ \left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

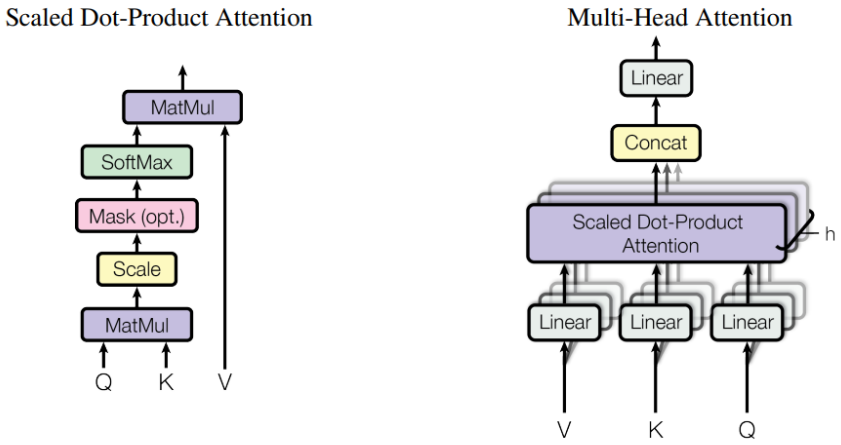
Nesta equação, Q,K e V representam as matrizes de consulta, chave e valor, respectivamente, e dk é a dimensão dos vetores de chave. A função softmax é aplicada para obter pesos normalizados.

Essencialmente, a atenção é calculada tomando o produto ponto entre os vetores de consulta e chave, dividindo pelo fator de escala $\sqrt{d_k}$ para lidar com problemas relacionados a gradientes pequenos. A função softmax é então aplicada para obter pesos normalizados, que são usados para ponderar os vetores de valor.

Este mecanismo de atenção, introduzido por CHO et al. (2014) e amplamente utilizado no *Transformer* (VASWANI et al., 2017), permite ao modelo focar em diferentes partes da sequência, proporcionando uma abordagem eficiente para capturar informações contextuais em dados sequenciais.

A Figura 2.5 representa o funcionamento do mecanismo de atenção.

Figura 2.5 - Mecanismo de Atenção e Autoatenção *Multi-Head*.



Fonte: VASWANI et al. (2017)

2.6.3.2 *Multi-Head* e Projeções Lineares

Para aprimorar a eficiência computacional da Autoatenção *Multi-Head*, é introduzida a ideia de realizar várias funções de atenção em paralelo (cabeças). Cada cabeça opera com projeções lineares das consultas, chaves e valores, permitindo uma representação mais rica e abrangente.

A Autoatenção *Multi-Head* proporciona uma capacidade única de modelar relações complexas em dados sequenciais, tornando-se um pilar fundamental em arquiteturas de aprendizado profundo, como o BERT (DEVLIN et al., 2019).

2.6.3.3 Saída

O *token* [CLS] é incorporado como o primeiro elemento em cada sequência de entrada, com o intuito de criar uma representação codificada agregada c para ser empregada em tarefas de nível de sequência. Considerando a saída do bloco *Encoder* do *Transformer* Z = ($z_1, \cdots, z_n$), a representação codificada do [CLS] é agrupada usando uma camada linear com ativação tangente hiperbólica:

$$c = tanh(z_1 W_c + b_c) \tag{2.13}$$

Onde, c, z_1, b_c pertencem a R^H e W_c é uma matriz em R^{HxH}.

Em tarefas específicas relacionadas a *tokens*, a representação codificada T_i de cada *token* é obtida diretamente de sua posição correspondente em Z.

2.6.4 Pré Treinamento

Na etapa de pré-treinamento, o BERT passa por duas tarefas de autosupervisão: MLM e Predição da Próxima Sentença (NSP). Cada exemplo de pré-treinamento é criado pela concatenação de duas sentenças, A, com *tokens* (a_1, ..., a_m), e B, com *tokens* (b_1, · · · , b_0), da seguinte forma:

$$x = ([CLS]a_1 \cdots a_m[SEP]b_1 \cdots b_o[SEP]) \quad (2.14)$$

Dado um exemplo de pré-treinamento x gerado a partir da concatenação de duas sentenças A e B, onde A possui *tokens* (a_1, · · · , a_m) e B possui *tokens*(b_1, · · · , b_o), a tarefa NSP é definida da seguinte maneira: em 50% dos casos, B é a sentença que segue imediatamente A, formando um trecho contínuo de texto; nos outros 50%, B é uma sentença aleatória amostrada de um documento distinto do corpus (conjunto de textos de treinamento - como descrito anteriormente) (DEVLIN et al., 2019). Essa decisão é codificada no rótulo de verdade para a tarefa NSP:

yNSP = ⊮ (B é a continuação de A), onde ⊮(.) é a função indicadora.

Quanto à tarefa de MLM, cada exemplo é corrompido ao selecionar aleatoriamente um conjunto de posições (inteiros de 1 a n =| x |), representado por m = [m_1, · · · , m_k] onde k = ⌈0.15n⌈. Os *tokens* nessas posições são substituídos por uma das três opções: um *token* especial [MASK] com 80% de probabilidade, um *token* aleatório do vocabulário com 10% de probabilidade ou, caso contrário, mantendo o *token* original. Esse processo de substituição utiliza o mascaramento de palavra inteira, o que implica que, se um *token* pertencente a uma palavra composta por várias unidades de subpalavras for escolhido para ser corrompido, todas as outras unidades de subpalavras também serão corrompidas. Os *tokens* originais das posições mi são preservados e atuam como rótulos para a tarefa MLM:

$$yMLM = [x_{*m1}, \cdots, x_{*mk}] \quad (2.15)$$

A representação final do exemplo de pré-treinamento é expressa como uma tupla ($x_{corrupt}$, m, yMLM, yNSP), onde as sequências corrompidas $x_{corrupt}$ são utilizadas como entradas para o BERT, e as representações codificadas resultantes são empregadas como entradas para as cabeças de tarefas de pré-treinamento, anexadas durante esta fase inicial de treinamento (DEVLIN et al., 2019).

2.6.5 MLM e Função de Perda

A tarefa de MLM tem como objetivo prever o *token* original $x * i$ para cada posição corrompida $i \in m$, utilizando uma abordagem de classificação em todo o vocabulário V, que possui um tamanho V. Para cada posição corrompida *i*, a cabeça de MLM realiza os seguintes passos:

a) aplicação de normalização por GeLU:

$$h_{Mi} = LayerNorm(GeLU\ (T_i W_M + b_m)) \quad (2.16)$$

b) cálculo do softmax para obter a distribuição de probabilidade:

$$p_{Mi} = Softmax(h_{Mi} E_V^T + b_v) \quad (2.17)$$

Onde, $W_M \in R^{HH}$, $b_m \in R^H$, $b_v \in R^V$, e $E_V \in R^{VxH}$ representa a matriz de *embeddings* de *tokens* de entrada. Os *embeddings* de *tokens* de entrada compartilham pesos com a camada densa.

A função de perda (*loss function*) associada à tarefa de MLM é projetada para avaliar quão bem o modelo está realizando a predição dos *tokens* originais nas posições mascaradas. A função de perda de MLM, frequentemente denotada como L_{MLM} e utiliza a entropia cruzada média como métrica para calcular a discrepância entre as previsões do modelo e os *tokens* reais nos locais mascarados. Essa perda é fundamental durante o treinamento do BERT para ajustar os pesos do modelo de maneira aprimorar suas capacidades de representação de linguagem:

$$L_{MLM}(x_{Corrupt}, \theta) = \frac{1}{k}\sum_{i \in m} - log\, p_{Mi}(x_i^* | x_{corrupt}) \quad (2.18)$$

Onde, $p_{Mi}(x_i^* | x_{corrupt})$ representa a probabilidade estimada pelo modelo para o *token* original x_i^* na posição *i*, e *k* é o número de *tokens* mascarados (DEVLIN et al., 2019).

2.7 Técnicas de Avaliação

A utilização de técnicas de avaliação adequadas é fundamental para garantir a eficácia e a precisão dos modelos de aprendizado de máquina, especialmente ao comparar abordagens como aprendizado supervisionado (como no caso do BERT), *few-shot* e *zero-shot learning*. No caso do BERT, que utiliza aprendizado supervisionado, o modelo é pré-treinado em grandes volumes de dados não rotulados e, em seguida, ajustado (*fine-tuned*) com um conjunto de dados rotulados (RADFORD et al., 2018).

No contexto do *few-shot learning* o modelo é desafiado a aprender a partir de um número extremamente reduzido de exemplos por classe. Nesse caso, a avaliação deve focar em garantir que o modelo esteja realmente generalizando de maneira eficaz, em vez de simplesmente memorizar os poucos exemplos disponíveis. Portanto, a utilização de técnicas de avaliação rigorosas é essencial para medir com precisão essa capacidade de generalização e assegurar que o modelo pode ser aplicado de forma confiável em situações com dados limitados (CHEN et al., 2019; LUO et al., 2023).

Já no *zero-shot learning*, o modelo é desafiado a reconhecer ou classificar dados de classes que não foram vistas durante o treinamento. Nessa abordagem, o modelo deve extrapolar o conhecimento adquirido para novas categorias, permitindo realizar tarefas em classes desconhecidas sem exemplos explícitos (WANG et al., 2019).

3 TRABALHOS RELACIONADOS

Nos últimos anos, a classificação de texto tem ganhado destaque em áreas como marketing, política, psicologia e música (EDMONDS; SEDOC, 2021). As análises podem ser exclusivamente textuais, incluindo a classificação semântica, de texto e de sentimentos/emoções (REVATHY; PILLAI; DANESHFAR, 2023; AGRAWAL; SHANKER; ALLURI, 2021), ou integradas com áudio (MALHEIRO et al., 2013). Análises semânticas são particularmente eficientes para reconhecer sentimentos (BARTHET; FAZEKAS; SANDLER, 2013), e a análise de texto por si só é frequentemente suficiente para identificar características das letras, uma vez que o áudio não oferece uma predição significativamente melhor (EDMONDS; SEDOC, 2021).

No entanto, predições utilizando classificadores tradicionais como SVM, Naive Bayes e KNN não alcançaram alta acurácia na classificação de sentimentos em letras de músicas, obtendo resultados melhores apenas quando o áudio foi incluído na análise (MALHEIRO et al., 2013). Uma abordagem multi-classe, com emoções previamente classificadas no banco de dados e utilizadas na fase de treinamento, permitiu que o classificador Naive Bayes apresentasse resultados superiores em comparação ao modelo BERT (EDMONDS; SEDOC, 2021).

Em contraste, o uso do BERT e suas variações (BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), LyEmoBERT (REVATHY; PILLAI; DANESHFAR, 2023), DistilBERT (SANH et al., 2020) e RoBERTa (LIU et al., 2019)) tem se mostrado eficaz em diversas aplicações. Cada variação ajusta o algoritmo original para focar em problemas específicos ou otimizar a execução. Como o BERT suporta trabalhar com um conjunto de dados multi-idiomas, é possível otimizar seu funcionamento para que ajustes sejam feitos baseados nos dados que serão trabalhados. Camadas adicionais dentro do algoritmo são possíveis, além de utilizarmos algoritmos já disponibilizados por outros estudos (JAIN, 2022).

Neste contexto, onde o uso do BERT é diversificado e com atribuições diferentes, ressaltamos os seguintes estudos:

a) análise de relatórios de acidentes de carros: O BERT foi utilizado nas narrativas de acidentes para indicar a severidade do mesmo e identificar áreas com acidentes mais graves, visando melhorias de tráfego ou medidas de segurança (OLIAEE et al., 2023);
b) análise de satisfação de clientes: O BERT foi aplicado em pedidos de comida/serviço para direcionar os donos dos estabelecimentos a traçar planos de

melhorias baseados no retorno dos usuários em redes sociais (BISWAS et al., 2021);

c) análise médica: O uso do BERT em análise de câncer, baseado em relatórios clínicos de um hospital nos EUA, impulsionou o desempenho de modelos de linguagem para aplicações médicas na detecção de sintomas de câncer (Symptom-BERT) (ZEINALI et al., 2024). O AD-BERT utiliza notas clínicas não estruturadas de registros eletrônicos de saúde para prever o risco de progressão da Doença de Alzheimer (MAO et al., 2023);

d) análise de músicas e sentimentos: O LyEmoBERT investiga a classificação das emoções das letras de músicas e a recomendação de músicas baseadas nessas emoções. Utilizando o dataset Music4All, o estudo avalia características líricas relevantes para identificar emoções humanas como felicidade, raiva, relaxamento e tristeza. O estudo implementa várias técnicas de aprendizado de máquina e um modelo semântico psicométrico, observando uma melhora significativa na acurácia com o uso do modelo BERT (92%). Além disso, desenvolve um sistema de recomendação de letras simples usando o modelo Sentence *Transformer* (REVATHY; PILLAI; DANESHFAR, 2023);

e) camadas intermediárias: Análise com camadas intermediárias (SONG et al., 2020), onde, ao invés de analisar a camada final, a proposta é usar as informações das camadas intermediárias para melhorar a performance do resultado da camada final.

Esses estudos mostram a aplicabilidade e a versatilidade do BERT em diferentes contextos, desde a análise de satisfação de clientes até a detecção de sintomas médicos. Nos trabalhos relacionados às técnicas de avaliação para *few-shot learning*, estudos como o de RADFORD et al. 2018 destacam a importância de refletir com precisão tanto a eficácia do pré-treinamento quanto a capacidade de generalização após o ajuste fino. A ausência de avaliação adequada pode levar à subestimação ou superestimação da verdadeira capacidade do modelo, o que é particularmente crítico em cenários onde a coleta de dados é limitada e cada exemplo tem um impacto significativo no desempenho do modelo.

Técnicas como as redes prototípicas, que criam protótipos para cada classe e classificam novos exemplos com base em sua proximidade com esses protótipos, têm mostrado resultados promissores nesse contexto (SNELL; SWERSKY; ZEMEL, 2017). Essas abordagens são complementadas pela capacidade de generalização amplamente explorada no PLN, como demonstrado por modelos generativos, como o Generative Pre-trained

Transformer (GPT), que têm se mostrado eficazes na categorização de textos em novas classes de forma *zero-shot* (PURI; CATANZARO, 2019).

Além disso, aplicações em tradução automática, que permitem a tradução direta entre pares de idiomas nunca vistos, são viabilizadas pelo pré-treinamento multilíngue unificado (LAUSCHER et al., 2020). A técnica de *zero-shot learning* também se mostra promissora na extração de relações sem a necessidade de exemplos anotados, permitindo ao modelo compreender e extrair novas informações contextuais a partir de texto não treinado (LEVY et al., 2017). Contudo, a avaliação dessa técnica deve se concentrar em métricas específicas que possam medir com precisão essa habilidade de generalização, assegurando que o modelo possa identificar novas classes de forma robusta e sem introduzir viés indesejado (BLODGETT et al., 2020). Dessa forma, o *zero-shot learning* emerge como uma abordagem poderosa e flexível, cuja evolução contínua promete expandir ainda mais suas aplicações no PLN (WANG et al., 2019).

Outras abordagens relevantes incluem as redes de correspondência, que utilizam a atenção para comparar novas amostras com exemplos existentes e decidir a classe com base na similaridade (MENG et al., 2020). Adicionalmente, a literatura discute o uso de meta-aprendizado, onde o modelo aprende generalizando rapidamente a partir de poucos exemplos em diversas tarefas (GHAROUN et al., 2024). Análises comparativas em *few-shot learning* também enfatizam a necessidade de uma avaliação rigorosa e detalhada, garantindo que esses modelos possam ser aplicados com confiança em situações com dados escassos (CHEN et al., 2019).

No entanto, nosso trabalho difere desses estudos em vários aspectos:

a) foco no Contexto Religioso: Nosso estudo é específico para a análise de letras de músicas Cristãs no Brasil, investigando a presença e a adaptação de elementos tradicionais de adoração ao longo do tempo;
b) abordagem de Classificação: Adotamos abordagem binária de classificação focada em distinguir entre "Adoração" e "Não Adoração". Essa abordagem é aplicada a um corpus construído de músicas Cristãs, que contrasta com as abordagens multi-classes comuns em outros estudos, geralmente concentrados em emoções gerais como tristeza, alegria, nervosismo ou relaxamento. Essa especificidade permite análise mais direcionada e relevante para o contexto religioso;
c) configuração e Treinamento do Modelo: Embora utilizemos o BERT, nossas configurações de treinamento são ajustadas especificamente para o contexto de

músicas Cristãs, considerando parâmetros como tamanho dos *tokens* e balanceamento de classes de autores.

Essas diferenças destacam a singularidade do nosso estudo e sua contribuição específica para a análise de músicas Cristãs no Brasil.

4 METODOLOGIA

Este trabalho interdisciplinar investiga a composição das letras das músicas Cristãs no Brasil, com o objetivo de compreender como elementos tradicionais de adoração são mantidos ou adaptados ao longo do tempo. Conforme discutido no Capítulo 1, a música Cristã no Brasil vem passando por diversas transformações históricas, culturais e sociais que influenciam significativamente sua evolução.

A qualidade da base de dados é crucial para a relevância dos resultados, tornando esta etapa fundamental para o sucesso do projeto. A abordagem para interpretar a mudança de foco nas letras das músicas cristãs é estruturada em quatro partes principais.

4.1 Coleta de Dados

Para realizar a análise da presença de elementos tradicionais de adoração em composições de músicas cristãs, os dados das músicas foram inicialmente coletados. Utilizamos um script para acessar o banco de dados do Spotify[1]. A coleta foi direcionada especificamente para o gênero de música cristã, obtendo-se informações como os nomes das músicas, os autores e os anos de lançamento. O Spotify, sendo uma das maiores plataformas de streaming musical do mundo, oferece um vasto catálogo de músicas cristãs, permitindo a coleta de dados para esta pesquisa.

Contudo, o Spotify não disponibiliza as letras das músicas diretamente em sua plataforma. Para contornar essa limitação e obter as letras necessárias para a análise, recorreu-se ao aplicativo Vagalume[2]. O site Vagalume contém um extenso banco de dados de letras de músicas, abrangendo diversos gêneros e permitindo aos usuários acessar gratuitamente as letras de suas canções. Utilizando as informações das músicas encontradas no Spotify, foi possível compilar as letras correspondentes no Vagalume para análise.

[1] Disponível:<https://developer.spotify.com/documentation/web-pi/reference/get-several-audio-features>

[2] disponível em: <https://api.vagalume.com.br/docs/>

4.2 Anotação dos Dados

Para exemplificar e fornecer critérios claros para a classificação de músicas cristãs, foram estabelecidos parâmetros que diferenciam entre "Adoração" e "Não Adoração". Essa classificação foi baseada na análise do conteúdo temático das letras e na intenção expressa nelas. O objetivo foi identificar músicas centradas na exaltação e devoção a Deus, distinguindo-as daquelas que, embora abordassem temas espirituais ou cristãos, não tinham a adoração como foco principal.

A classificação de "Adoração" foi definida como músicas cujo tema central é a exaltação, louvor e devoção a Deus. Essas músicas são geralmente utilizadas em contextos litúrgicos ou de culto, com um foco claro na relação direta entre o fiel e Deus. Por outro lado, "Não Adoração" inclui músicas que podem abordar temas espirituais, reflexões cristãs ou ensinamentos morais, mas que não têm a adoração a Deus como foco principal, abrangendo também músicas gerais sem contexto cristão.

O processo de anotação começou com a análise do tema central da música. A avaliação concentrou-se no tema predominante da letra. Se a exaltação de Deus fosse o tema central, a música seria classificada como "Adoração". Foi importante verificar se esse tema foi mantido de forma consistente ao longo de toda a música, especialmente por meio da repetição de refrões de louvor ou orações, o que indicaria fortemente que a música pertencia à categoria de "Adoração".

Em seguida, foram procuradas palavras ou frases que indicassem adoração, como "Senhor", "Deus", "louvor", "glorificar", "adoração". A presença frequente dessas palavras ao longo da música sugeriu sua classificação como "Adoração". Além disso, foi crucial considerar o contexto em que essas palavras foram utilizadas. Se uma palavra como "Deus" não estava associada a expressões de louvor ou devoção, a música poderia ser classificada como "Não Adoração". Por exemplo, músicas onde a palavra "Deus" é utilizada em uma exclamação de desespero, como "Oh meu Deus, porque isso aconteceu comigo?", seriam classificadas como "Não Adoração".

Os anotadores, em vez de tentarem deduzir a intenção geral da música, focaram nas estruturas frasais e no uso de palavras-chave emocionais. Frases exclamativas e imperativas, por exemplo, indicavam uma intenção de adoração, enquanto frases declarativas sugeriam uma reflexão pessoal ou um ensinamento moral. Essa análise ajudou a captar a intenção expressa ou implícita na letra.

A sensibilidade emocional da música também foi avaliada, observando-se a repetição de palavras ou frases que evocassem sentimentos específicos. A repetição de termos relacionados a louvor e adoração indicou uma emoção positiva e devocional, enquanto a repetição de termos introspectivos indicou reflexão. Essa análise foi crucial para determinar a natureza emocional da música e, consequentemente, sua classificação.

A Figura 4.1, apresenta um fluxograma que visualiza o processo descrito, facilitando a compreensão das etapas de classificação entre "Adoração" e "Não Adoração". Em algumas situações, a música combinou temas de adoração com outros temas, como analogias pessoais, situações cotidianas e expressões variadas. Nesses casos, foi determinado qual tema era predominante. Se mais de 50% da música estava focada em adoração, ela foi classificada como "Adoração". Em situações ambíguas, foi adicionado um comentário explicando a decisão tomada, detalhando quais elementos da música influenciaram a classificação.

Para garantir a eficácia da anotação, foi realizado um processo de anotação piloto. Foram selecionadas 12 músicas que representaram uma variedade de casos, incluindo músicas claramente de "Adoração", músicas claramente de "Não Adoração" e casos ambíguos. Os anotadores participaram de uma sessão de treinamento onde os critérios de anotação foram apresentados e algumas músicas usadas como exemplo para alinhar os entendimentos.

Figura 4.1 - Diretivas para Classificação

Fonte: O Autor (2024)

Cada anotador anotou as músicas do conjunto piloto de forma independente, registrando suas classificações e quaisquer dúvidas ou ambiguidades encontradas durante o processo. Após a anotação, as classificações dos diferentes anotadores foram comparadas, a

concordância foi avaliada e as discrepâncias foram discutidas em grupo para ajustar e refinar o guia exemplificado nesta sessão.

Durante as discussões definiu-se que um exemplo de "Adoração" seria uma música que repete várias vezes as expressões "Glória ao Senhor", "Rei dos reis", "exaltado seja Teu nome" ao longo de seus versos e refrão. Já um exemplo de "Não Adoração" seria uma música que fala sobre viver uma vida de fé e esperança, mas sem expressar diretamente louvor ou devoção a Deus.

Para elaborar este guia baseou-se em obras como as de Artstein e Poesio (2008) sobre concordância entre anotadores, e outros estudos sobre anotação de linguagem natural. Esses fundamentos teóricos ajudaram a assegurar que o processo de anotação fosse tanto rigoroso quanto replicável. (PUSTEJOVSKY; STUBBS, 2012; HOVY; SPRUIT, 2016; SNOW et al., 2008; BONTCHEVA; ROUT, 2014)

Para garantir que os anotadores classificassem as músicas de maneira consistente, foram utilizadas métricas de concordância, como o coeficiente Kappa (ARTSTEIN; POESIO, 2008). Todo o processo, desde a seleção das músicas até as discussões e revisões do guia, foi cuidadosamente documentado. O coeficiente Kappa é definido com a equação 4.1 e tem seu valor definido entre −1 e 1.

$$k = \frac{P_0 - P_e}{1 - P_e} \tag{4.1}$$

Onde:

a) P_o: Proporção de concordância observada entre os anotadores;

b) P_e: Proporção de concordância esperada ao acaso.

O resultado desse coeficiente pode ser exemplificado conforme a Tabela 4.1:

Tabela 4.1 - Exemplo uso Kappa

	Anotador 2: Adoração	**Anotador 2: Não Adoração**	**Total**
Anotador 1: Adoração	40	10	50
Anotador 1: Não Adoração	5	45	50
Total	45	55	100

Fonte: adaptado de ARTSTEIN; POESIO (2008)

A proporção de concordância observada (P_0) é a soma das concordâncias nas diagonais (40 + 45 = 85) dividida pelo total (100), ou seja:

$$k = \frac{85}{100} = 0{,}85 \quad (4.2)$$

A proporção de concordância esperada (P_e) é calculada considerando a probabilidade de cada categoria. A probabilidade de ambos os anotadores escolherem "Adoração" ao acaso seria:

$$k = \frac{50}{100}\ X\ \frac{45}{100} = 0{,}225 \quad (4.3)$$

Similarmente, a probabilidade de ambos escolherem "Não Adoração" ao acaso seria:

$$k = \frac{50}{100}\ X\ \frac{55}{100} = 0{,}275 \quad (4.4)$$

A soma dessas probabilidades é:

$$P_e\ =\ 0{,}225\ +\ 0{,}275\ =\ 0{,}50 \quad (4.5)$$

Substituindo na fórmula do Kappa:

$$P_e\ =\ \frac{0{,}85-0{,}5}{1-0{,}50} = \frac{0{,}35}{0{,}50} = 0{,}70 \quad (4.6)$$

Com κ = 0,70, podemos dizer que há uma concordância substancial entre os anotadores, considerando que valores entre 0,61 e 0,80 indicam uma concordância substancial garantindo a confiabilidade dos dados anotados e assegurando que o processo de anotação é consistente entre os diferentes anotadores (LANDIS; KOCH, 1977).

4.3 Pré-Processamento dos Dados

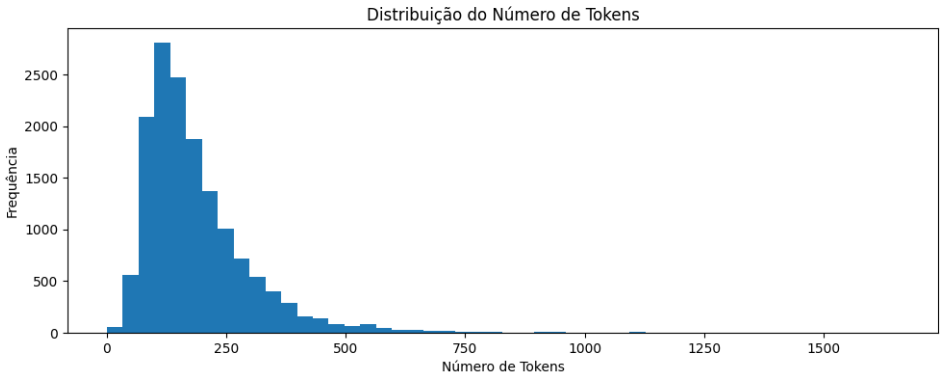
Após a coleta de dados, eliminamos eventuais duplicidades como versões com o título da música contendo as palavras "Ao Vivo" e "Acústico" , garantindo que não tivéssemos

músicas repetidas dentro da base. Com isso, as entradas foram reduzidas de 21.597 para 14.935 músicas, totalizando 2.808.971 palavras.

Uma das dificuldades na hora de eliminar ruídos dentro do corpus foram as repetições dentro das músicas (como o coro), para evitar refrões repetidos se buscou classificadores presentes em algumas músicas como "refrão" e "coro" e se todas as palavras fossem iguais dentro das outras partes da música, também foram retirados.

Após a remoção das redundâncias expostas acima e reconhecendo a importância do tamanho das sentenças para o funcionamento do BERT, fizemos a distribuição da quantidade de *tokens* dentro da base. A Figura 4.2 mostra a distribuição dos tamanhos das letras das músicas na base de dados e suas incidências. Com isso, o tamanho do texto, como parâmetro de entrada, foi parametrizado de três modos:

Figura 4.2 - Distribuição de *Tokens*



count	mean	std	min	25%	50%	75%	max
14935.0	188.29949782390358	114.25944216985057	1.0	112.0	159.0	231.0	1656.0

Fonte: O Autor (2024)

a) padrão - usar a entrada padrão do algoritmo desenvolvido pelos seus idealizadores de 512 *tokens* (DEVLIN et al., 2019);
b) mediana - é uma medida estatística de tendência central que divide um conjunto de dados ordenado em duas partes iguais. Na análise dos dados o tamanho de entrada para o BERT será de 150 *tokens* e;
c) 3º Quartil - Quartis são estatísticas que dividem um conjunto de dados ordenado em quatro partes iguais, cada uma representando um quarto da distribuição. Esses pontos são úteis para entender a dispersão e a centralidade dos dados, além de fornecer uma visão sobre a simetria e a presença de dados com valores discrepantes

do restante do conjunto (outliers). Na análise dos dados o tamanho do 3o quartil (que engloba 75% dos dados) para a entrada do BERT foi de 231 *tokens*[3]

Além do tamanho do texto de entrada, nosso corpus foi dividido em dois grupos distintos de dados: dados de treinamento e dados de teste. Os dados de treinamento compreendem aproximadamente 20% das músicas da estrutura de dados e foi classificado de duas formas: "Adoração" e "Não Adoração". Desta forma, o arquivo de treinamento possui 3.077 músicas, enquanto o arquivo de dados de teste possui 11.858 músicas.

Como não encontramos um conjunto de dados previamente classificado entre "adoração" e "não adoração", a classificação da base de treinamento foi realizada pelo autor desta dissertação, partindo do modelo de anotação exposto na Sessão 4.2.

A divisão da base de teste foi realizada de forma aleatória, entretanto, para realizar um treinamento preciso, considerando que muitos autores possuem uma quantidade maior de músicas, foi desenvolvido um algoritmo para mapear todas as incidências de autores nas músicas. Com base nesse mapeamento, as músicas selecionadas para o treinamento tiveram um peso distribuído de acordo com a quantidade de incidências do mesmo autor.

Dessa forma, autores com poucas músicas tiveram uma menor incidência na base de treinamento, enquanto que autores com mais produções musicais tiveram um número maior e proporcional de músicas incluídas na base de teste (STEININGER et al., 2021; KIMURA; HINO, 2024).

Para mapear os autores das músicas, utilizou-se uma abordagem probabilística, onde Q foi definida como a razão entre o número de incidências de um autor ($N_{deIncidencia}$) e o número total de músicas (N_{Total}) (DEGROOT; SCHERVISH, 2002; KIMURA; HINO, 2024):

$$Q = \frac{N_{deIncidencia}}{N_{Total}} \tag{4.7}$$

Depois da divisão entre base de teste e de treinamento é necessário *token*izar o conjunto de dados carregado. Para esta tarefa usamos um *token*izador Bert*Token*izer da biblioteca *transformer*s da *Hugging Face*. O *token*izador é inicializado com o modelo neuralmind/ bert-base-portuguese-cased, que é um BERT treinado especificamente para o idioma português[4].

[3] Disponível: <https://uq.pressbooks.pub/portable-introduction-data-analysis/chapter/quantiles/>
[4] Disponível: <https://huggingface.co/neuralmind/bert-base-portuguese-cased>

Com todos os dados de treino e teste carregados, a partir de planilhas, utilizou-se a biblioteca Pandas para estruturarmos os dados, também foi necessário traduzir na base de treinamento feita manualmente entre "Adoração" e "Não Adoração". Para isso, cada música avaliada e classificada como "Adoração" foram mapeadas para o valor 1, enquanto as classificadas como "Não Adoração" foram mapeadas para o valor 0. Essa conversão para valores inteiros foi necessária para que os rótulos pudessem ser utilizados pelo algoritmo BERT durante o treinamento.

Após a preparação inicial, seguimos para a etapa de *token*ização, onde os textos são convertidos em *tokens* utilizando o *token*izer do BERT. Este processo inclui adicionar *tokens* especiais necessários para o modelo BERT (como explicado, o [CLS] para o início e [SEP] para o final da sequência), em seguida aplicamos o preenchimento (*padding*) e truncamento para garantir que todas as sequências tenham o mesmo comprimento, necessário para o processamento em lote pelo modelo BERT. Com os textos *token*izados e padronizados, criou-se então máscaras de atenção, que ajudam o modelo BERT a distinguir entre *tokens* reais e *tokens* de preenchimento. Essas máscaras são essenciais para o funcionamento correto do modelo.

Após essa etapa, os dados *token*izados e as máscaras de atenção foram convertidos em tensores que foram combinados em uma base de dados do *PyTorch*[5]. Esta base de dados é utilizada para criar um *DataLoader*[6] que é uma classe disponível na biblioteca *PyTorch*, que facilita o manuseio dos dados durante o treinamento e facilita a avaliação do modelo BERT, permitindo uma interação eficiente sobre os dados em lotes. Durante a manipulação da base de dados maiores, enfrentou-se problemas de falta de memória. Ao utilizar o *DataLoader* para carregar os dados em lotes menores, conseguimos evitar esses problemas. Esse método permitiu que o algoritmo fosse executado de maneira eficiente, dividindo os dados em partes menores e mais gerenciáveis.

Em resumo o processo até essa etapa se deu da seguinte forma:

a) leitura dos dados: Carregar os dados em planilhas em formato tabela na biblioteca do Pandas;
b) pré-Processamento: Limpar e mapear rótulos para valores inteiros;
c) *token*ização: Converter textos em *tokens* utilizando um *token*izer do BERT;

[5] Disponível em: <https://pytorch.org/hub/huggingface_pytorch-*transformer*s/>
[6] Disponível em: <https://pytorch.org/tutorials/beginner/basics/data_tutorial.html>

d) *padding* e truncamento: Garantir que todas as sequências tenham o mesmo comprimento;

e) criação de máscaras de atenção: Distinguir entre *tokens* reais e de preenchimento;

f) conversão para tensores: Converter os dados *token*izados, máscaras de atenção e rótulos em tensores do *PyTorch*;

g) criação do conjunto de dados: Combinar os tensores em um conjunto de dados do *PyTorch*;

h) criação do *DataLoader*: Facilitar a interação dos dados em lotes durante o treinamento e avaliação do BERT.

4.4 Pré-treinamento

O pré-treinamento é dividido em duas fases principais: pré-treinamento de palavras e pré-treinamento de sentenças. Após essas fases, ocorre o ajuste fino (*fine-tuning*), que pode ser direcionado para a análise desejada, como, neste caso, a análise de entendimento entre "adoração" e "não adoração".

Durante o treinamento, diversas métricas de avaliação são utilizadas para monitorar o desempenho do modelo: acurácia (*accuracy*), precisão (*precision*), revocação (*recall)* e F1-score. Essas métricas ajudam a entender a eficácia do modelo em diferentes aspectos da classificação, além de validar os resultados finais.

Configurando os argumentos de treinamento, especificando parâmetros como a taxa de aprendizado (*learning rate*), o tamanho dos lotes de treino e validação (per_device_train_batch_size e per_device_eval_batch_size), o número de épocas (num_train_epochs) e a taxa de decaimento do peso (*weight decay*), parâmetros críticos para garantir que o modelo treine de forma eficaz e que a convergência ocorra de maneira estável. A Tabela 4.2 mostra os parâmetros configurados.

Definir os argumentos de treinamento envolve decisões críticas que afetam diretamente o desempenho do modelo. A taxa de aprendizado (*learning rate*) é essencial para garantir que o modelo convirja para uma solução ótima. Taxas de aprendizado muito altas podem causar instabilidade no treinamento, enquanto taxas muito baixas podem tornar o treinamento excessivamente lento. O tamanho dos lotes (*batch size*) de treino e validação afetam diretamente a eficiência do treinamento e o uso de memória. Lotes maiores podem acelerar o treinamento,

mas exigem mais memória, enquanto lotes menores podem ser mais precisos, porém mais lentos.

O número de épocas, (num_train_epochs), define quantas vezes o modelo verá o conjunto de dados completo durante o treinamento. É necessário encontrar um equilíbrio para garantir que o modelo aprenda o suficiente sem sobreajustar os dados de treinamento. A taxa de decaimento do peso, (*weight decay*), ajuda a prevenir o *overfitting* (que é um desafio comum em aprendizado de máquina e tem como objetivo encontrar um equilíbrio entre ajustar-se bem aos dados de treinamento e manter a capacidade de generalização para novos dados), aplicando uma penalidade aos pesos do modelo, o que favorece soluções mais simples e generalizáveis.

Tabela 4.2 - Configuração dos Argumentos de Treinamento

Parâmetro	**Valor**
Taxa de Aprendizado (*learning rate*)	$2 \times 10{-}5$
Tamanho do Lote de Treinamento (per_device_train_batch_size)	8
Tamanho do Lote de Validação (per_device_eval_batch_size)	8
Número de Épocas (num_train_epochs)	5
Taxa de Decaimento do Peso (weight decay)	0.01
Estratégia de Avaliação (eval_st*rate*gy)	epoch

Fonte: O Autor (2024)

Utilizou-se a classe *Trainer*[7] da biblioteca *transformer*s para gerenciar o processo de treinamento. Esta classe facilita o treinamento do modelo ao lidar com muitos aspectos complexos do aprendizado profundo, como:

a) otimização: Inclui a escolha do algoritmo de otimização, ajuste da taxa de aprendizado, aplicação de gradientes, clipping de gradientes e decaimento de peso, garantindo que o modelo treine de forma eficaz e estável. Para atualizar os pesos do modelo a classe usa o algoritmo de otimização Adaptive Moment Estimation with *Weight Decay* (AdamW), ajusta também a taxa de aprendizado de forma automática e aplica técnicas de regularização (*weight_decay*) para evitar *overfitting*. No algoritmo implementado o parâmetro *weight_decay* está configurado em 0.01, que indica o decaimento de peso que será aplicado durante o treinamento. Além disso, a classe também permite o ajuste da taxa de aprendizado (*schedulers*), no nosso caso

[7] Disponível em: <https://huggingface.co/docs/*transformer*s/pt/training#trainer>

usamos uma convergência linear que diminuir a taxa de aprendizado ao longo do treinamento;

b) avaliação: Abrange a definição de métricas de desempenho, a separação adequada dos dados em conjuntos de treino, validação e teste, o monitoramento contínuo do desempenho do modelo e a implementação de *early stopping* para evitar *overfitting*.

c) armazenamento de *Checkpoints*: Salva automaticamente o estado do modelo, incluindo os pesos dos parâmetros e o estado do otimizador, permitindo retomar o treinamento a partir de um ponto salvo em caso de interrupções. Também são utilizados para selecionar o melhor modelo se baseando no desempenho de validação, garantindo que o modelo final seja o mais eficaz possível.

Configuramos a classe *Trainer* com o modelo, os argumentos de treinamento, os conjuntos de dados de treino e validação, e a função de cálculo de métricas. Adicionalmente, incluímos o callback de *Early Stopping* que é uma técnica usada para interromper o treinamento quando o desempenho no conjunto de validação não melhora após um certo número de épocas. Isso ajuda a evitar o *overfitting* e economiza tempo e recursos de computação ao interromper o treinamento de forma antecipada quando é improvável que ocorra uma melhoria significativa, para interromper o treinamento caso o desempenho no conjunto de validação não melhore após um determinado número de épocas.

4.5 Uso do *Transformer* para Análise Contextual de Letras Músicais

Para analisar as relações entre todas as palavras nas letras das músicas, utilizou-se a arquitetura *Transformer*. Esta arquitetura é composta por várias camadas de codificação, cada uma usando mecanismos de autoatenção e redes neurais *Feed-Forward*.

a) autoatenção *Multi-Head:*

- permite que a entrada do modelo seja examinada simultaneamente em diferentes partes;
- executa a atenção múltiplas vezes em paralelo (*Multi-Head*) obtendo dependências contextuais e melhorando as representações de cada palavra.

b) rede Neural *Feed-Forward:*

- as saídas são processadas por redes neurais totalmente conectadas que operam independentemente em cada posição da sequência;

- as representações intermediárias são transformadas em formas mais úteis para as camadas subsequentes. O intuito é refinar as características contextuais extraídas anteriormente.

c) conexões residuais e normalização - As saídas das subcamadas são combinadas através de conexões residuais e normalização, preservando a informação e facilitando o treinamento da rede;

d) saída - Com a implementação das características anteriores (brevemente descritas), a saída da última camada de codificação são usadas para a tarefa de análise de sentimento, onde uma camada adicional específica para classificação é adicionada. Esta interpreta as representações contextuais e atribui rótulos de sentimento, como "adoração" ou "não adoração".

4.6 Validadores

Como explicado anteriormente no Capítulo 2, utilizamos três ferramentas para validar os resultados obtidos: Matriz de Confusão, Curva ROC e AUC. A Matriz de Confusão serve como base para calcular as taxas de TP e FP, que são usadas para traçar a Curva ROC. A Curva ROC permite avaliar o desempenho do classificador em diferentes limiares (valores usados para decidir se a classe de uma instância é positiva ou negativa), enquanto a AUC (Área Sob a Curva) resume o desempenho da Curva ROC em um único valor. Essas três ferramentas estão intimamente relacionadas e, juntas, fornecem uma visão completa do desempenho de um classificador.

Utilizamos a função roc_curve(y_true, y_probs) para calcular os valores de FP e TP para todos os limiares possíveis, permitindo uma avaliação abrangente do desempenho do classificador. O código usa a função softmax do *PyTorch* para calcular as probabilidades previstas para a classe positiva, que são armazenadas como y_probs. A função roc_auc_score calcula a AUC, que resume o desempenho da Curva ROC em um único valor. Em seguida, a função roc_curve gera as FPR e TPR para todos os limiares possíveis, permitindo a plotagem da Curva ROC e a visualização do trade-off entre sensibilidade e especificidade do classificador.

4.7 Técnicas de Avaliação

Após a execução do algoritmo, criou-se duas variações:

a) *zero-shot learning*: com a implementação original, retirou-se o *fine-tuning* do algoritmo. Desta forma o algoritmo foi inicializado com o BERT treinado com a base em português e não teve o treinamento para definir "adoração" e "não adoração". Após isso, usou-se as mesmas métricas no algoritmo desenvolvido para essa dissertação.

b) *few-shot learning*: na implementação original, gerou-se um arquivo com um conjunto de treinamento para ser utilizado parcialmente no treinamento na abordagem *few-shot learning*. O tamanho pra treinamento usou-se a técnica 5-way 5-shot, onde pegou-se 5 exemplos pre-treinados de cada classe, no nosso caso, 10 músicas.

4.8 Ajustes Finais

Após a etapa de treinamento e avaliação, utilizou-se o modelo treinado para fazer previsões no conjunto de teste. As letras das músicas no conjunto de teste são *token*izadas, e as previsões do modelo são mapeadas para os rótulos de "adoração" ou "não adoração".

Executou-se o algoritmo com 3 parâmetros de taxa aprendizado (5e-5, 3e- 5 e 2e-5) referenciamos o estudo de (REVATHY; PILLAI; DANESHFAR. 2023) e 3 tamanhos de entrada de texto (Descrito no início desse capítulo). Os resultados são apresentados no Capítulo 5.

5 RESULTADOS E DISCUSSÕES

Nesse capítulo vamos mostrar os resultados obtidos e como isso influencia no uso do algoritmo.

5.1 Anotação dos Dados

Para demonstrar a importância da anotação dos dados, apresento os resultados obtidos em duas abordagens distintas. Na primeira abordagem, as músicas foram enviadas aos avaliadores sem nenhum treinamento ou explicação sobre como realizar a anotação, com o objetivo de validar como a falta de orientação influenciaria o resultado e qual o resultado do coeficiente Kappa nessa situação. Na segunda abordagem, as mesmas músicas foram submetidas aos avaliadores após um treinamento detalhado e uma explicação dos critérios descritos no 4, para evidenciar a variação do coeficiente nesses dois cenários.

Na primeira abordagem, o resultado das anotações pode ser visualizado na Tabela 5.1, onde a contagem em cada coluna reflete a soma das classificações idênticas feitas por todos os anotadores para cada música. Por exemplo, para a Música 1, quatro anotadores classificaram a música como "Adoração" e dois anotadores classificaram-na como "Não Adoração".

Para calcular o Coeficiente Kappa de Fleiss (CKF) da Tabela 5.1, utilizou-se a biblioteca *Statsmodels*[1], criando-se um vetor de entrada composto por tuplas que representam a somatória das classificações de cada música, realizadas por todos os anotadores.

O resultado do CKF nesse cenário foi de $CKF = -0.12427807486630993$, indicando um problema significativo na forma como os dados foram anotados (LANDIS; KOCH, 1977). Esse resultado pode ser atribuído a critérios de classificação mal definidos, categorias ambíguas ou inconsistências no processo de anotação, demonstrando a diversidade de interpretação entre os anotadores na tentativa de capturar o sentimento de adoração nas músicas.

[1] Disponível em: <https://www.statsmodels.org/stable/index.html>

Tabela 5.1 - Contagem das classificações de "Adoração" e "Não adoração" por música sem anotação.

Música	Adoração	Não adoração	Posição Vetor
Música 1	4	2	0
Música 2	4	2	1
Música 3	6	0	2
Música 4	5	1	3
Música 5	5	1	4
Música 6	5	1	5
Música 7	4	2	6
Música 8	5	1	7
Música 9	4	2	8
Música 10	4	2	9
Música 11	4	2	10
Música 12	5	1	11

Fonte: O Autor (2024)

Após essa classificação inicial, a metodologia de anotação foi ensinada e os critérios para identificar o sentimento de "Adoração" foram explicados e exemplificados para cada tipo de classe, incluindo músicas que poderiam ser ambíguas entre as categorias. Dessa forma, as músicas foram reanalisadas, e os resultados podem ser vistos na Tabela 5.2

O resultado do CKF após o treinamento foi de CKF = 0.4250221827861579, o que, segundo LANDIS; KOCH 1977, indica uma concordância moderada entre os anotadores. Durante as discussões com os anotadores para revisar o modelo de anotação, notou-se que a concordância sobre os parâmetros relacionados ao tema de adoração é subjetiva e varia de acordo com a vivência e a experiência individual de cada anotador. Essa subjetividade é particularmente evidente em músicas ambíguas, que apresentam uma variação maior na interpretação, dificultando uma divisão mais precisa e clara. Essa dificuldade evidencia a subjetividade inerente à análise de PLN, especialmente em temas complexos como adoração (MONTOYO; MARTINEZ-BARCO; BALAHUR, 2012).

5.2 BERT com Treinamento

Após a anotação dos dados executou-se o algoritmo de classificação BERT descrito no Capítulo 4, concatenamos todas as músicas, dividindo-as por ano e mostrando o resultado da classificação realizada.

Tabela 5.2 - Contagem das classificações de "Adoração" e "Não adoração" por música com o treinamento.

Música	Adoração	Não adoração
Música 1	6	0
Música 2	5	1
Música 3	6	0
Música 4	5	1
Música 5	5	1
Música 6	5	1
Música 7	5	1
Música 8	5	1
Música 9	0	6
Música 10	1	5
Música 11	1	5
Música 12	5	1

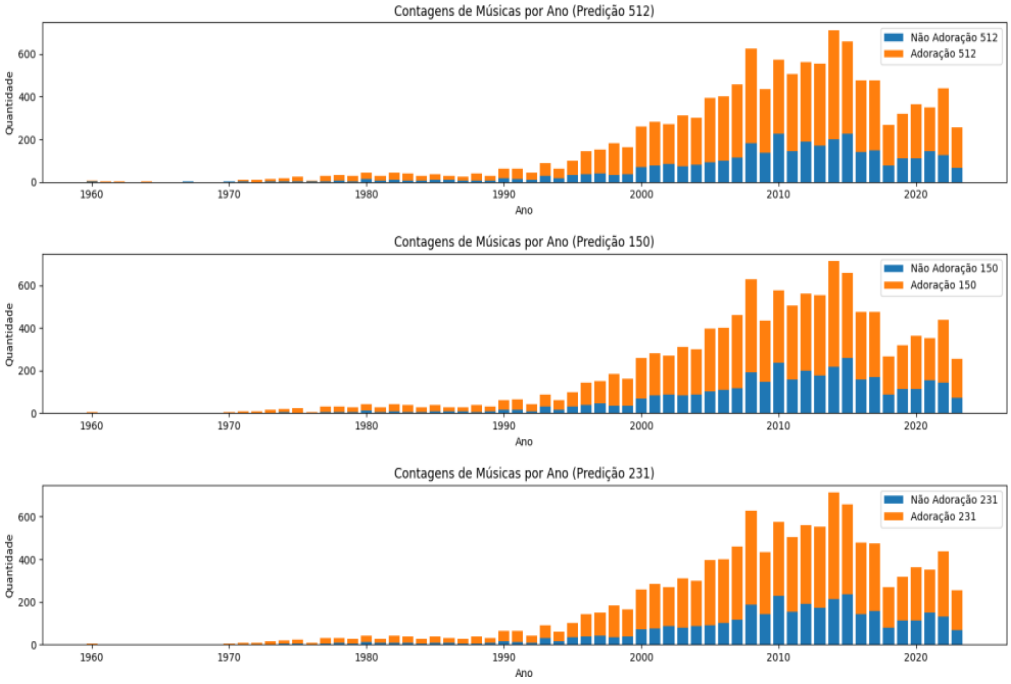
Fonte: O Autor (2024)

Os dados utilizados abrangem o período de 1960 a 2023 e incluem informações sobre a quantidade de músicas de "adoração" e "não adoração" para três categorias de predição que variam de acordo com o tamanho do texto analisado pelo BERT (Predição_512, Predição_150, e Predição_231). A porcentagem de músicas de "não adoração" foi calculada para cada ano, baseando-se na fórmula 5.1.

$$Q = \frac{N_{deIncidencia}}{N_{Total}} \tag{5.1}$$

A Figura 5.1 ilustra a relação entre músicas de "adoração" e "não adoração" ao longo dos anos, considerando diferentes tamanhos de predição de texto analisado pelo BERT (512, 150 e 231 *tokens*).

Figura 5.1 - Resultado Análise Bert



Fonte: O Autor (2024)

Para facilitar a análise e interpretação dos resultados, os dados apresentados no gráfico foram normalizados, transformando as contagens absolutas em porcentagens, conforme mostrado na Tabela 5.3, a normalização permite comparar de forma intuitiva a proporção de músicas classificadas como "Não Adoração" ao longo das décadas, independentemente do número total de músicas em cada ano. Isso é especialmente útil para identificar tendências ao longo do tempo e avaliar se as mudanças observadas são proporcionais ao crescimento ou diminuição geral do número de composições.

A Tabela 5.3 indica a porcentagem das músicas classificadas como "Não Adoração". Observa-se que a variação nas avaliações é mínima entre os diferentes tamanhos de entrada, embora as entradas com menor número de *tokens* apresentem variações ligeiramente maiores.

Para responder à segunda pergunta norteadora da pesquisa: ’Investigar se as composições de músicas Cristãs contemporâneas, em comparação com as de décadas anteriores, apresentam tendências evolutivas na forma de adoração’, geramos a Tabela 5.4 que mostra uma tendência evolutiva ao longo das décadas, com variações nas porcentagens de músicas classificadas como ’Não Adoração’, sugerindo mudanças na temática das composições.

Tabela 5.3 – Porcentagens de Música com classificação "Não adoração" por Década

Década	% Não_Adoração_512	% Não_Adoração_150	% Não_Adoração_231
1960	22.14	22.14	22.14
1970	18.89	19.64	18.89
1980	23.91	24.41	23.91
1990	33.04	33.72	33.04
2000	30.79	31.41	30.79
2010	29.47	28.89	29.47
2020	33.02	32.97	33.02

Fonte: O Autor (2024)

Tabela 5.4 – Médias das Porcentagens de Música por Década, Incluindo Estimativa para 2030

Década	Média % N_Ador_512	Média % N_Ador_150	Média % N_Ador_231
1960	27.78	27.78	27.78
1970	18.24	18.14	18.24
1980	23.15	23.20	23.15
1990	30.71	30.62	30.71
2000	30.35	29.90	30.35
2010	28.90	28.94	28.90
2020	31.97	32.07	31.97
2030 (Est)	33.24	33.44	33.24

Fonte: O Autor (2024)

Para projetar as tendências futuras, utilizamos um modelo de regressão linear, treinado com dados históricos de porcentagens de músicas de não adoração, calculadas com base nos dados fornecidos pelo BERT. Os resultados obtidos foram:

$$Predio_{512} : Coeficiente = 0.2698, \tag{5.2}$$

$$Predio_{150} : Coeficiente = 0.3124, \tag{5.3}$$

$$Predio_{231} : Coeficiente = 0.2899, \tag{5.4}$$

Analisando os dados oferecidos temos:

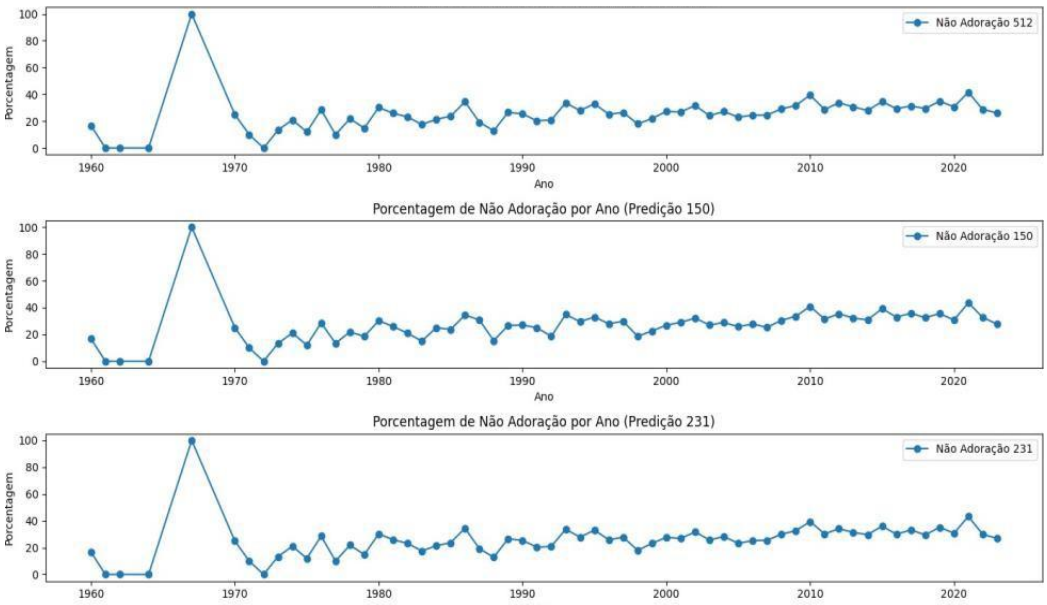
a) predição_512 : O coeficiente de 0.2698 indica que para cada ano adicional, a porcentagem de músicas de "não adoração" aumenta em 0.2698;
b) predição_150:O coeficiente de 0.3124 indica que para cada ano adicional, a porcentagem de músicas de "não adoração" aumenta em 0.3124;
c) predição_231: O coeficiente de 0.2899 indica que para cada ano adicional, a porcentagem de músicas de "não adoração" aumenta em 0.2899.

O gráfico da Figura 5.2 mostra a predição para os próximos anos, o que indica uma tendência a aumentar a incidência de músicas com a classificação de "Não Adoração". Isto pode ser uma explicação pelas mudanças sociais vivenciadas na nossa sociedade, como exemplificado no capítulo 1 (WAINER, 2017). Também vale salientar que a base de dados trabalhada é específica já que o foco era uma classificação de músicas de composição cristã.

Ao analisar o gráfico, é importante notar um ponto fora da curva significativo entre os anos 1960 e 1970. Esse pico se deve ao fato de que o número de músicas registradas foi excepcionalmente baixo, e todas as músicas analisadas foram classificadas como "Não Adoração". Isso gerou uma distorção na média anual, resultando em uma porcentagem de 100% para "Não Adoração" naquele ano, o que não reflete uma tendência geral, mas sim a falta de diversidade nas classificações devido ao pequeno número de amostras.

Para compreender melhor a tendência na música de adoração, calculamos a média por década com o objetivo de estimar qual a relação de "Adoração" e "Não Adoração". A Tabela 5.4 sugere que a porcentagem de músicas que não enaltecem a Deus tende a aumentar nos próximos anos.

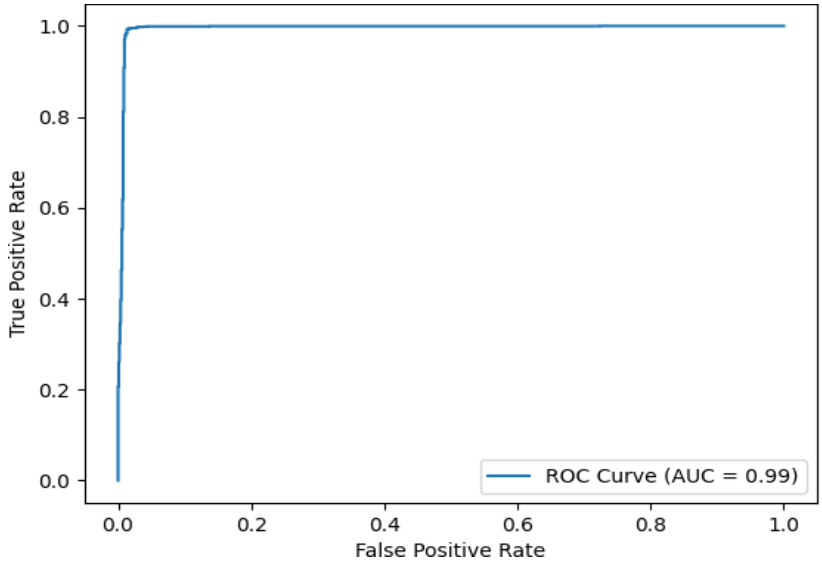
Figura 5.2 - Previsão até 2030



Fonte: O Autor (2024)

Para validarmos o algoritmo e o resultado apresentado anteriormente, analisamos também a perda de treinamento através da épocas e geramos a curva ROC e a área AUC (Figura 5.3) e seu valor como 0.9944769745967582, como ilustrado na imagem 5.3 e pode-se notar que a taxa de perda durante o tempo vai diminuindo o que indica que o modelo está aprendendo 5.4.

Figura 5.3 - Curva ROC e AUC

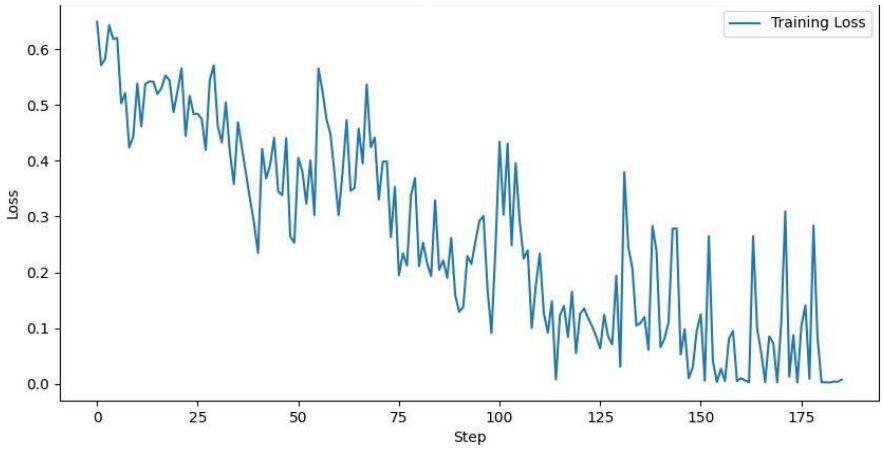


Fonte: O Autor (2024)

A matriz de confusão referenciada na Tabela 5.5 indica que o modelo classificou corretamente um alto número de verdadeiros positivos (1934) e verdadeiros negativos (1011), o que é indicativo de um bom desempenho. No entanto, há um pequeno número de falsos positivos (29) e falsos negativos (7), sugerindo que o modelo cometeu poucos erros de classificação.

Esses resultados são consistentes com as métricas de precisão, revocação e F1-score apresentadas na Tabela 5.7, que também indicam um desempenho robusto do modelo.

Figura 5.4 - Perca através das épocas



Fonte: O Autor (2024)

Tabela 5.5 - Matriz de Confusão

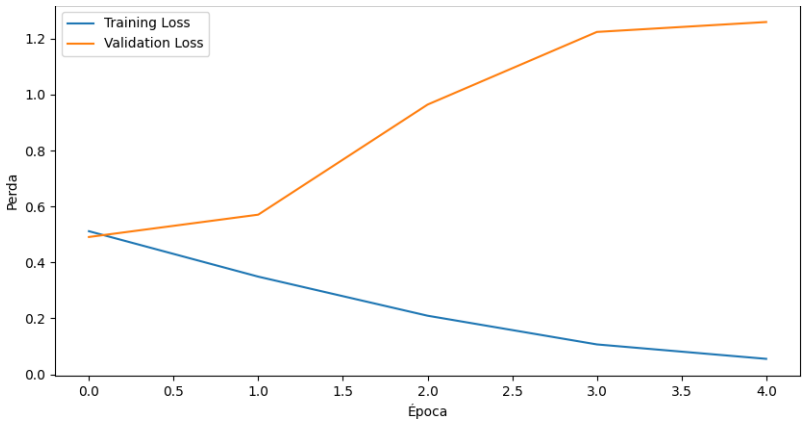
		Predição	
		Não Adoração	**Adoração**
Real	**Não Adoração**	1011	29
	Adoração	7	1934

Fonte: Adaptado de Fawcett (2006)

Os valores de perda de cross-entropy durante o treinamento do modelo para os dados de cada época indicam que o modelo pode ser considerado adequado após três épocas de treinamento. O principal critério para avaliar a adequação foi observar a perda mínima de cross-entropy no subconjunto de Teste 5.5. Consideramos o modelo adequado quando a perda de

cross-entropy no subconjunto de Teste atinge seu valor mais baixo e não apresenta redução significativa nas duas interações subsequentes. A técnica de interrupção antecipada do treinamento (*early stopping*) foi utilizada para evitar que o modelo se tornasse overfit, a Figura 5.6 ilustra essa ação do algoritmo. A prática de ajuste fino é um processo interativo que envolve a modificação de hiperparâmetros (Tabela 5.6 (OLIAEE et al., 2023)) ou da estrutura da rede com base no desempenho do modelo nos conjuntos de dados de treinamento e teste.

Figura 5.5 - Treinamento e Validação nas Épocas



Fonte: O Autor (2024)

Dessa forma, é possível otimizar o modelo para que ele tenha o melhor desempenho possível no subconjunto de dados de teste, mesmo que seja uma amostra aleatória. Os dados de validação foram utilizados apenas após todas as adaptações necessárias terem sido aplicadas. O resultado obtido é semelhante ao encontrado no uso do BERT na análise de severidade em acidentes de carro, um problema com multiclasses (OLIAEE et al., 2023).

Tabela 5.6 - Hiperparâmetros usados nos experimentos

Hiperparâmetro	**Valor**
Taxa de *dropout* da camada de classificação	0.05
Taxa de aprendizado	2e-5
Épocas	5
Tamanho do lote (*Batch size*)	8
Passos de aquecimento (*Warmup steps*)	0

Fonte: O Autor (2024)

Descrição dos Hiperparâmetros:

a) taxa de dropout da camada de classificação: Proporção de neurônios que são desativados aleatoriamente durante o treinamento para evitar o *overfitting*;

b) taxa de aprendizado: O tamanho do passo que o modelo dá durante a otimização. Um valor menor permite um ajuste mais fino dos pesos, mas pode exigir mais épocas de treinamento;

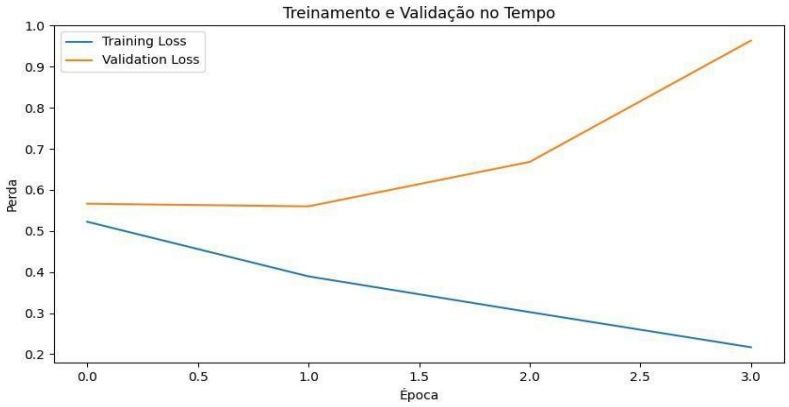
c) épocas: Número de passagens completas pelo conjunto de dados de treinamento.

d) tamanho do lote: Número de amostras processadas antes de atualizar os parâmetros do modelo;

e) passos de aquecimento: Número de passos durante os quais a taxa de aprendizado é gradualmente aumentada a partir de um valor inicial pequeno até a taxa de aprendizado definida.

Os resultados da Tabela 5.7, foram obtidos quando o algoritmo de predição usou a base de teste igual a base de treinamento e indicam que o modelo possui uma alta precisão e revocação para ambas as classes. Especificamente, a precisão para as classes "Não Adoração" e "Adoração" é de 0.99, sugerindo uma pequena taxa de falsos positivos. A classe "Adoração", por sua vez, teve um resultado perfeito no quesito de revocação, indicando que todos os exemplos verdadeiramente positivos foram identificados de forma assertiva, enquanto a classe "Não Adoração" possui uma revocação de 0.97.

Figura 5.6 - Treinamento e Validação nas Épocas com *early stopping*



Fonte: O Autor (2024)

O F1-score é elevado para ambas as classes, com 0.98 para "Não Adoração" e 0.99 para "Adoração", refletindo um bom equilíbrio entre precisão e revocação. A acurácia geral do modelo é de 0.99, demonstrando que 99% das previsões totais estão corretas.

As métricas de macro-média e média ponderada são quase idênticas, ambas em torno de 0.99. Isso sugere que o desempenho do modelo é consistente em ambas as classes e não está enviesado para uma classe específica.

Tabela 5.7 - Relatório de Classificação

Classe	**Precision**	**Recal**	**F1-score**	**Support**
Não Adoração	0.99	0.97	0.98	1040
Adoração	0.99	1.00	0.99	1941
Acurácia		0.99		
Macro média	0.99	0.98	0.99	2981
Média ponderada	0.99	0.99	0.99	2981

Fonte: Adaptado de H. M., M.N (2015)

Com isso, podemos dizer que o modelo de classificação mostra um desempenho com alta precisão, revocação e F1-score para ambas as classes analisadas. A alta acurácia geral e as métricas agregadas reforçam a robustez e a confiabilidade do modelo para a tarefa de distinção entre "Não Adoração" e "Adoração". Esses resultados indicam que o modelo é adequado e eficaz para a aplicação proposta, proporcionando previsões precisas e consistentes.

No intuito de validar se o treinamento do fine-tunning é relevante nesse problema de validação binária (adoração e não adoração), alterou-se o algoritmo em duas abordagens distintas:

a) *zero-shot Learning* - onde o algoritmo só usa a base do BERT treinada em português, mas não o treinamento para a validação de "adoração" e "não adoração".
b) *few-shot Learning* - onde o algoritmo usa a mesma premissa com o *fine-tunning*, mas limitou-se a usar 10 músicas como treinamento.

5.3 *Zero-shot Learning*

Como explicado no Capítulo 2 e 4 o *Zero-shot Learning* não tem base de treinamento, usa-se só o pré-treinamento para o português desenvolvido em outro trabalho (SOUZA; NOGUEIRA; LOTUFO, 2020). Após a execução do BERT sem o fine-tunning obteve-se os resultados abaixo.

As métricas de avaliação podem ser visualizadas nas Tabelas 5.8 e 5.9.

Tabela 5.8 - Matriz de Confusão *Zero-shot Learning*

		Classe Predita	
		Não Adoração	**Adoração**
Classe Real	**Não Adoração**	998	42
	Adoração	1896	45

Fonte: Adaptado de Fawcett (2006)

O modelo tem alto *recall* para a classe "Não Adoração" (96%), porém baixa precisão (34%), indicando que muitas previsões "Não Adoração" estavam incorretas. Para a classe "Adoração", o desempenho é muito fraco, com precisão (52%) e *recall* (2%) baixos, resultando em um F1-Score muito baixo (0.04). Isso indica que o modelo tem dificuldades significativas em identificar corretamente exemplos de "Adoração".

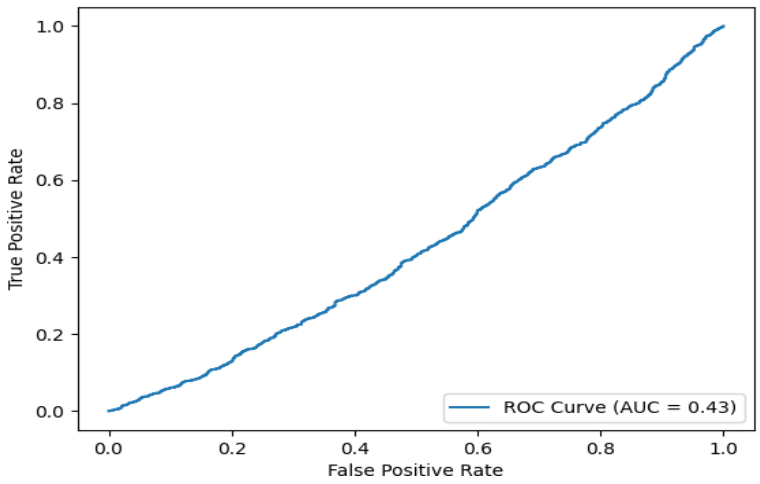
Tabela 5.9 - Relatório de Classificação *Zero-shot Learning*

Classe	**Precisão**	**Recall**	**F1-Score**	**Suporte**
Não Adoração	0.34	0.96	0.51	1040
Adoração	0.52	0.02	0.04	1941
Acurácia	0.35 (2981 amostras)			
Média Macro	0.43	0.49	0.28	2981
Média Ponderada	0.46	0.35	0.21	2981
AUC	0.4348			

Fonte: Adaptado de H. M., M.N (2015)

A Figura 5.7 representa a curva AUC de 0.4348 que está abaixo de 0.5, sugerindo que o modelo está performando pior do que uma classificação aleatória. Isso pode ser devido ao grande número de falsos negativos e falsos positivos, particularmente na classe "Adoração". Esse resultado indica que o modelo, na forma atual, não está funcionando bem para a tarefa de classificação pretendida.

Figura 5.7 - Curva ROC e AUC - *Zero-Shot Learning*



Fonte: O Autor (2024)

5.4 *Few-shot Learning*

Como explicado no Capítulo 2 e 4 o *Few-shot Learning* utiliza uma pequena base de treinamento adicional ao pré-treinamento para o português desenvolvido por SOUZA; NOGUEIRA; LOTUFO 2020. Após a execução do BERT com um pequeno ajuste (*fine-tunning*) no contexto das composições de música cristã, obtiveram-se os resultados apresentados abaixo, Tabela 5.10 indicando a Matriz de Confusão e a Tabela 5.11 indicando o relatório de classificação.

Tabela 5.10 - Matriz de Confusão *Few-Shot Learning*

		Classe Predita	
		Não Adoração	**Adoração**
Classe Real	**Não Adoração**	77	963
	Adoração	19	1922

Fonte: Adaptado de Fawcett (2006)

Tabela 5.11 - Relatório de Classificação *Few-Shot Learning*

Classe	**Precisão**	**Recall**	**F1-Score**	**Suporte**
Não Adoração	0.80	0.07	0.14	1040
Adoração	0.67	0.99	0.80	1941
Acurácia	0.67 (2981 amostras)			
Média Macro	0.73	0.53	0.47	2981
Média Ponderada	0.71	0.67	0.57	2981
AUC	0.7187			

Fonte: Adaptado de H. M., M.N (2015)

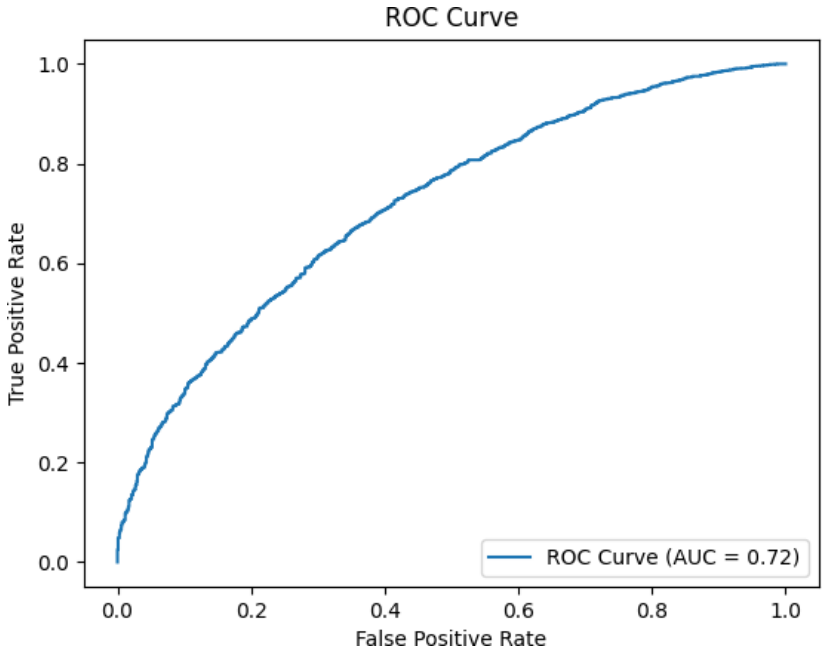
O modelo apresenta um *recall* extremamente alto para a classe "Adoração" (99%), o que significa que quase todos os exemplos reais de "Adoração" foram corretamente identificados. No entanto, o modelo tem um *recall* muito baixo para a classe "Não Adoração" (7%), indicando que a maioria dos exemplos dessa classe foi erroneamente classificada como "Adoração".

Em termos de precisão, a classe "Não Adoração" tem uma precisão alta (80%), mas como o *recall* é baixo, o F1-Score final é também baixo (0.14). A classe "Adoração" tem uma precisão mais moderada (67%), mas devido ao *recall* muito alto, o F1-Score resultante é bom (0.80).

A Figura 5.8 o valor de AUC de 0.7187 sugere que o modelo tem um bom poder discriminativo. Um AUC acima de 0.7 indica que o modelo é capaz de diferenciar bem entre as duas classes ("Adoração" e "Não Adoração"). No entanto, o desempenho pode ser melhorado, principalmente no que diz respeito à identificação correta dos exemplos de "Não Adoração", onde o *recall* é muito baixo.

O valor de AUC acima de 0.7, em combinação com uma acurácia de 67%, mostra que o modelo consegue capturar a maioria dos exemplos de "Adoração" corretamente, mas a identificação da classe "Não Adoração" precisa de melhorias significativas para tornar o modelo mais balanceado e confiável.

Figura 5.6 - Curva ROC e AUC - *Few-Shot Learning*



Fonte: O Autor (2024)

5.5 Discussão

Ao analisar os resultados das matrizes de confusão das Tabelas 5.5, 5.8 e 5.10, as métricas de avaliação 5.7, 5.9, 5.11 e as curvas Roc/Auc 5.3, 5.7 e 5.8, observa que o BERT, quando treinado, tem resultado melhores do que sem treinamento (*Zero-shot Learning*) e com pouco treinamento (*Few-shot Learning*), um dos problemas que para tal divergência de resultado pode ser causado pela pouca generalização que esse modelo desenvolvido possui, os trabalhos relacionados para o uso dessas métricas de avaliação indicam que a generalização é um dos fatores que pode impactar o seu uso (LUO et al., 2023; BLODGETT et al., 2020). O problema de generalização é encontrado em estudos com o uso do BERT específico para avaliação de sentimento (REVATHY; PILLAI; DANESHFAR, 2023) e evidenciado também na Figura 5.5 onde a perda de validação aumenta com as épocas.

Em relação à anotação dos dados, os resultados apresentados na primeira abordagem mostram que o CKF negativo (−0.1243) indica que a concordância entre os anotadores era pior do que a esperada ao acaso (LANDIS; KOCH, 1977).

Isso sugere que, sem diretrizes claras ou treinamento, os anotadores enfrentaram dificuldades significativas para classificar consistentemente as músicas como "Adoração" ou "Não Adoração". Após o treinamento, o CKF aumentou para 0.425, o que, segundo LANDIS;

KOCH 1977, indica uma concordância moderada entre os anotadores após o treinamento e a explicação das diretrizes. Embora este seja um avanço em relação ao cenário sem treinamento, ainda há espaço para melhoria, o que sugere que a subjetividade continua a influenciar as classificações.

5.6 Conclusão e Trabalhos Futuros

Com base nos resultados e conclusões desta dissertação, evidenciou-se que a diretriz criada para a anotação de dados no contexto de música cristã é eficaz para melhorar a consistência das anotações em relação a classificações feitas ao acaso (LANDIS; KOCH, 1977). No entanto, os critérios de avaliação pelos anotadores podem ser aprimorados com o uso de ferramentas de PLN. Por exemplo, a implementação de sistemas que auxiliem os anotadores na detecção de palavras-chave relevantes pode aumentar a objetividade e a precisão das anotações. Além disso, o desenvolvimento de mecanismos que comparem as anotações dos avaliadores com um conjunto de dados anotado por especialistas pode fornecer resposta em tempo real, aprimorando ainda mais a acurácia das anotações.

Com relação ao BERT, uma das possíveis melhorias para trabalhos futuros seria a realização de uma busca por hiperparâmetros mais adequada. Isso pode ser feito por meio da geração de um grid search para explorar diferentes variações de hiperparâmetros, como a taxa de aprendizado, o número de épocas e o tamanho do lote (*batch size*).

Além das melhorias propostas, estudos comparativos com outros modelos de aprendizado profundo ou técnicas de PLN também poderiam ser executados. A avaliação de modelos alternativos ou complementares pode revelar abordagens que superem os resultados obtidos com o BERT neste contexto específico de classificação de músicas cristãs. Tais comparações podem fornecer alternativas para a adequação do BERT em relação a outros métodos disponíveis, identificando oportunidades para aprimoramentos futuros.

Entre as limitações deste estudo, destaca-se o fato de que as anotações foram realizadas por apenas um avaliador, o que pode introduzir viés subjetivo no processo. Uma abordagem futura poderia envolver múltiplos anotadores, além de aplicar técnicas de validação cruzada para mitigar essa limitação. Além disso, tanto o treinamento quanto a base de teste foram realizados com o mesmo conjunto de dados, o que pode ter introduzido um grau de sobreajuste no modelo. Para estudos futuros, é recomendada a divisão mais robusta entre os conjuntos de treinamento e teste, ou o uso de conjuntos de dados externos para validação, a fim de garantir

que os resultados sejam generalizáveis e que o modelo possa ser aplicado a novos dados com maior confiabilidade.

Em conclusão, o trabalho apresentado nesta dissertação oferece uma base sólida para a classificação automática de músicas cristãs, mas melhorias nos processos de anotação e treinamento do modelo, bem como a inclusão de estudos comparativos com outras abordagens, podem contribuir para o avanço significativo do estado da arte nesse campo.

REFERÊNCIAS

A, P. Y. C.; PULABAIGARI, V.; B, E. **Semi-supervised learning: a brief review**. International Journal of Engineering & Technology, v. 7, p. 81, 02 2018.

AGRAWAL, Y.; SHANKER, R. G. R.; ALLURI, V. **Transformer-based approach towards music emotion recognition from lyrics**. In: HIEMSTRA, D. et al. (Ed.). Advances in Information Retrieval. Cham: Springer International Publishing, 2021. p. 167–175. ISBN 978-3-030-72240-1.

AIJMER, K.; RüHLEMANN, C. **Corpus Pragmatics: A Handbook**. [S.l.]: Cambridge University Press, 2014.

ARTSTEIN, R.; POESIO, M. **Inter-Coder Agreement for Computational Linguistics**. Computational Linguistics, v. 34, n. 4, p. 555–596, 12 2008. ISSN 0891-2017. Disponível em: <https://doi.org/10.1162/coli.07-034-R2>.

BA, J. L.; KIROS, J. R.; HINTON, G. E. **Layer Normalization**. 2016.

BAHDANAU, D.; CHO, K.; BENGIO, Y. **Neural Machine Translation by Jointly Learning to Align and Translate**. 2016.

BARTHET, M.; FAZEKAS, G.; SANDLER, M. **Music emotion recognition: From content- to context-based models**. In: . [S.l.: s.n.], 2013. p. 228–252. ISBN 978-3-642-41247-9.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural Language Processing with Python**. [S.l.: s.n.], 2009. ISBN 978-0-596-51649-9.

BISWAS, J. et al. **Sentiment analysis on user reaction for online food delivery services using bert model**. In: . [S.l.: s.n.], 2021.

BLODGETT, S. L. et al. **Language (technology) is power: A critical survey of"bias"in nlp**. arXiv preprint arXiv:2005.14050, 2020.

BONTCHEVA, K.; ROUT, D. **Making sense of social media streams through semantics: a survey**. Semantic Web, IOS Press, v. 5, n. 5, p. 373–403, 2014.

CHEN, W.-Y. et al. **A closer look at few-shot classification**. arXiv preprint arXiv:1904.04232, 2019.

CHO, K. et al. **On the Properties of Neural Machine Translation: Encoder-Decoder Approaches**. 2014.

CHUNG, J. et al. **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling**. 2014.

CUNHA, M. do N. **A explosão gospel: um olhar das ciências humanas sobre o cenário evangélico no Brasil**. [S.l.]: Mauad Editora Ltda, 2007. v. 2.

DEGROOT, M.; SCHERVISH, M. **Probability and Statistics**. Addison-Wesley, 2002. (Addison-Wesley series in statistics). ISBN 9780321204738. Disponível em: <https://books.google.ne/books?id=qi6KmgEACAAJ>.

DEVLIN, J. et al. BERT: **Pre-training of deep bidirectional transformers for language understanding**. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>.

DOLGHIE, J. Z. **A igreja renascer em cristo e a consolidação do mercado de música gospel no brasil: uma análise das estratégias de marketing**. Ciencias Sociales y Religión, v. 6, n. 6, p. 201–220, 2004.

EDMONDS, D.; SEDOC, J. **Multi-emotion classification for song lyrics**. In: CLERCQ, O. D. et al. (Ed.). Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. Online: Association for Computational Linguistics, 2021. p. 221–235. Disponível em: <https://aclanthology.org/2021.wassa-1.24>.

FAWCETT, T. **An introduction to roc analysis. Pattern Recognition Letters**, v. 27, n. 8, p. 861–874, 2006. ISSN 0167-8655. ROC Analysis in Pattern Recognition. Disponível em: <https://www.sciencedirect.com/science/article/pii/ S016786550500303X>.

GHAROUN, H. et al. **Meta-learning approaches for few-shot learning: A survey of recent advances.** ACM Computing Surveys, ACM New York, NY, v. 56, n. 12, p. 1–41, 2024.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GUDIVADA, V. N.; ARBABIFARD, K. **Chapter 3 - open-source libraries, application frameworks, and workflow systems for nlp**. In: GUDIVADA,

V. N.; RAO, C. (Ed.). **Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications**. Elsevier, 2018, (Handbook of Statistics, v. 38). p. 31–50. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169716118300221>.

HALEVY, A.; NORVIG, P.; PEREIRA, F. **The unreasonable effectiveness of data**. IEEE Intelligent Systems, v. 24, n. 2, p. 8–12, 2009.

HE, K. et al. **Deep residual learning for image recognition**. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. p. 770–778.

HELGEN, E.; HOEK, S. van der. **Religious Conflict in Brazil. Protestants, Catholics, and the Rise of Religious Pluralism in the Early Twentieth Century.** [S.l.]: International Journal of Latin American Religions, 2022.

HENDRYCKS, D.; GIMPEL, K. **Gaussian Error Linear Units** (GELUs). 2016.

HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. Neural computation, v. 9, p. 1735–80, 12 1997.

HOVY, D.; SPRUIT, S. L. **The social impact of natural language processing**. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). [S.l.: s.n.], 2016. p. 591–598.

INDURKHYA, N.; DAMERAU, F. J. **Handbook of natural language processing**. [S.l.]: Chapman and Hall/CRC, 2010.

JAIN, S. M. Bert. In: . **Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems**. Berkeley, CA: Apress, 2022. p. 37–49. ISBN 978-1-4842-8844-3. Disponível em: <https://doi.org/10.1007/978-1-4842-8844-3_3>.

JOACHIMS, T. **Text categorization with support vector machines: Learning with many relevant features**. In: NÉDELLEC, C.; ROUVEIROL, C. (Ed.). Machine Learning: ECML-98. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 137–142. ISBN 978-3-540-69781-7.

JORDAN, M. I.; MITCHELL, T. M. **Machine learning: Trends, perspectives, and prospects. Science**, v. 349, n. 6245, p. 255–260, 2015. Disponível em: <https://www.science.org/doi/abs/10.1126/science.aaa8415>.

JURAFSKY, D.; MARTIN, J. **Speech and Language Processing: An Introduction to Natural Language Processing**, Computational Linguistics, and Speech Recognition. [S.l.: s.n.], 2008. v. 2.

KIM, Y. **Convolutional Neural Networks for Sentence Classification**. 2014.

KIMURA, M.; HINO, H. **A Short Survey on Importance Weighting for Machine Learning**. 2024. Disponível em: <https://arxiv.org/abs/2403.10175>.

LANDIS, J. R.; KOCH, G. G. **The measurement of observer agreement for categorical data**. biometrics, JSTOR, p. 159–174, 1977.

LANE, H.; HOWARD, C.; HAPKE, H. M. **Natural Language Processing in Action**. Manning Publications, 2019. 544 p. ISBN 9781617294631. Disponível em: <https://www.manning.com/books/natural-language-processing-in-action>.

LAUSCHER, A. et al. **From zero to hero: On the limitations of zero-shot cross-lingual transfer with multilingual transformers**. arXiv preprint arXiv:2005.00633, 2020.

LEVY, O. et al. **Zero-shot relation extraction via reading comprehension**. arXiv preprint arXiv:1706.04115, 2017.

LIU, Y. et al. RoBERTa: **A Robustly Optimized BERT Pretraining Approach**. 2019. Disponível em: <https://arxiv.org/abs/1907.11692>.

LUO, X. et al. **A closer look at few-shot classification again**. In: PMLR. International Conference on Machine Learning. [S.l.], 2023. p. 23103–23123.

M., H.; M.N, S. **A review on evaluation metrics for data classification evaluations**. International Journal of Data Mining & Knowledge Management Process, v. 5, p. 01–11, 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:61877559>.

MA, Y. et al. **Active learning for name entity recognition with external knowledge**. ACM Trans. Asian Low-Resour. Lang. Inf. Process., Association for Computing Machinery, New York, NY, USA, apr 2023. ISSN 2375-4699. Just Accepted. Disponível em: <https://doi.org/10.1145/3593023>.

MALHEIRO, R. et al. **Music emotion recognition from lyrics: A comparative study**. In: . [S.l.: s.n.], 2013.

MAO, C. et al. **Ad-bert: Using pre-trained language model to predict the progression from mild cognitive impairment to alzheimer's disease**. Journal of Biomedical Informatics, v. 144, p. 104442, 2023. ISSN 1532-0464. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046423001636>.

MARTINOFF, E. H. d. S. **A música evangélica na atualidade: algumas refl exões sobre a relação entre religião, mídia e sociedade**. v. 18, abr. 2014. Disponível em: <https://revistaabem.abem.mus.br/revistaabem/article/view/217>.

MENDONÇA, J. d. S. **O gospel é pop: música e religião na cultura pós-moderna**. Universidade Estadual Paulista (Unesp), 2009.

MENG, Y. et al. **Text classification using label names only: A language model self-training approach**. arXiv preprint arXiv:2010.07245, 2020.

MONTOYO, A.; MARTINEZ-BARCO, P.; BALAHUR, A. **Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments**. Decision Support Systems, v. 53, p. 675–679, 11 2012.

OLIAEE, A. H. et al. **Using bidirectional encoder representations from transformers (bert) to classify traffic crash severity types**. Natural Language Processing Journal, v. 3, p. 100007,

2023. ISSN 2949-7191. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2949719123000043>.

PURI, R.; CATANZARO, B. **Zero-shot text classification with generative language models**. arXiv preprint arXiv:1912.10165, 2019.

PUSTEJOVSKY, J.; STUBBS, A. **Natural Language Annotation for Machine Learning: A guide to corpus-building for applications**. [S.l.]: "O'Reilly Media, Inc.", 2012.

RADFORD, A.; NARASIMHAN, K. **Improving language understanding by generative pre-training**. In: . [s.n.], 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:49313245>.

RADFORD, A. et al. **Improving language understanding by generative pre-training**. San Francisco, CA, USA, 2018.

RESTREPO, S. B. **La reforma protestante y la música. Ideas y valores**, v. 7, n. 25-26, p. 17–26, 1965.

REVATHY, V. R.; PILLAI, A. S.; DANESHFAR, F. **Lyemobert: Classification of lyrics' emotion and recommendation using a pre-trained model**. Procedia Computer Science, v. 218, p. 1196–1208, 2023. ISSN 1877-0509. International Conference on Machine Learning and Data Engineering. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050923000984>.

SANH, V. et al. **DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter**. 2020. Disponível em: <https://arxiv.org/abs/1910.01108>.

SANT'ANA, R. **A música gospel e os usos da "arma da cultura". reflexões sobre as implicações de uma emenda**. Revista Intratextos, v. 5, n. 1, p. 23–41, 2013.

SCHMIDT, R. M. **Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. 2019**.

SHAW, P.; USZKOREIT, J.; VASWANI, A. **Self-Attention with Relative Position Representations**. 2018.

SIMÕES, J. d. R. **Ser músico e viver da música no Brasil: um estudo da trajetória do centro musical Porto-Alegrense (1920-1933).** Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2011.

SNELL, J.; SWERSKY, K.; ZEMEL, R. **Prototypical networks for few-shot learning**. Advances in neural information processing systems, v. 30, 2017.

SNOW, R. et al. **Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks**. In: Proceedings of the 2008 conference on empirical methods in natural language processing. [S.l.: s.n.], 2008. p. 254–263.

SONG, Y. et al. **Utilizing BERT Intermediate Layers for Aspect Based Sentiment Analysis and Natural Language Inference**. 2020.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. **Bertimbau: Pretrained bert models for brazilian portuguese**. In: CERRI, R.; PRATI, R. C. (Ed.). Intelligent Systems. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8.

STEININGER, M. et al. **Density-based weighting for imbalanced regression**. Machine Learning, Springer, v. 110, p. 2187–2211, 2021.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Cambridge, MA, USA: A Bradford Book, 2015. ISBN 0262039249.

T, A.; DELLAPIETRA, V.; PIETRA, S. D. **A maximum entropy approach to natural language processing**. Computational Linguistics, v. 22, 07 2002.

VASWANI, A. et al. **Attention is all you need**. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6000–6010. ISBN 9781510860964.

VOS, I. M. A. de et al. **Comparing in context: Improving cosine similarity measures with a metric tensor**. 2022. Disponível em: <https://arxiv.org/abs/2203.14996>.

WAINER, D. F. **Entre música e tecnologia: condições de existência e funcionamento da indústria fonográfica brasileira no século xxi**. Comunicação e Sociedade, v. 31, p. 311–326, 2017.

WANG, W. et al. **A survey of zero-shot learning: Settings, methods, and applications**. ACM Transactions on Intelligent Systems and Technology (TIST), ACM New York, NY, USA, v. 10, n. 2, p. 1–37, 2019.

XIE, S. et al. **ResiDual: Transformer with Dual Residual Connections**. 2023.

ZEINALI, N. et al. **Symptom-bert: Enhancing cancer symptom detection in ehr clinical notes**. Journal of Pain and Symptom Management, 2024. ISSN 0885-3924. Disponível em: <https://www.sciencedirect.com/science/article/pii/ S088539242400784X>.