JOSÉ CARLOS FERREIRA NETO

DESENVOLVIMENTO DE MODELOS DE LINGUAGEM PARA EXTRAÇÃO DE ASPECTOS EM LÍNGUA PORTUGUESA

LAVRAS – MG

2023

JOSÉ CARLOS FERREIRA NETO

DESENVOLVIMENTO DE MODELOS DE LINGUAGEM PARA EXTRAÇÃO DE ASPECTOS EM LÍNGUA PORTUGUESA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas, para obtenção do título de Mestre.

Prof. Dr. Danton Diego Ferreira
Orientador

Prof. Dr. Bruno Henrique Groenner Barbosa
Coorientador

Prof. Dr. Denilson Alves Pereira
Coorientador

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).

Ferreira Neto, José Carlos.
Desenvolvimento de Modelos de Linguagem para Extração de Aspectos em Língua Portuguesa / José Carlos Ferreira Neto. - 2023.
93 p. : il.

Orientador(a): Danton Diego Ferreira.
Coorientador(a): Bruno Henrique Groenner Barbosa, Denilson Alves Pereira.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2023.
Bibliografia.

1. Extração de Aspectos. 2. Processamento de Linguagem Natural. 3. Língua Portuguesa. I. Ferreira, Danton Diego. II. Barbosa, Bruno Henrique Groenner. III. Pereira, Denilson Alves.

JOSÉ CARLOS FERREIRA NETO

DESENVOLVIMENTO DE MODELOS DE LINGUAGEM PARA EXTRAÇÃO DE ASPECTOS EM LÍNGUA PORTUGUESA

DEVELOPMENT OF LANGUAGE MODELS FOR ASPECT EXTRACTION IN PORTUGUESE

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas, para obtenção do título de Mestre.

APROVADA em 08 de Dezembro de 2023.

Prof. Dr. Danton Diego Ferreira	UFLA
Prof. Dr. Bruno Henrique Groenner Barbosa	UFLA
Prof. Dr. Denilson Alves Pereira	UFLA
Profª. Dra. Paula Christina Figueira Cardoso	UFLA
Prof. Dr. Giovani Bernardes Vitor	UNIFEI

gov.br
Documento assinado digitalmente
DANTON DIEGO FERREIRA
Data: 22/01/2024 16:37:00-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Danton Diego Ferreira
Orientador

LAVRAS – MG
2023

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, aos meus familiares, minha mãe Alessandra Carlos, meu pai Luiz Henrique e minha avó Maria Célia por sempre acreditarem em mim. A minha companheira Lorrana Flores por me incentivar em mais essa etapa da minha vida.

Gostaria de agradecer também ao professor Dr. Danton Ferreira e ao professor Dr. Denilson Pereira, por terem aceitado esse desafio de me orientar e co-orientar, e pela paciência que demonstraram para comigo durante todo esse ano. Ao professor Dr. Bruno Barbosa pelas contribuições valiosas no decorrer da pesquisa. A UFLA é melhor com vocês aqui.

O primeiro pecado da humanidade foi a fé; a primeira virtude foi a dúvida.
(Autor Desconhecido)

RESUMO

A identificação e extração de aspectos é essencial na análise de textos para discernir opiniões e emoções. Contudo, há uma lacuna na aplicação dessas técnicas ao português. Este trabalho visa adaptar abordagens originalmente desenvolvidas para o inglês a este idioma no conjuntos de dados TV e ReLi. O objetivo deste trabalho consiste em avaliar a aplicação de modelos de linguagem para extração de aspectos na língua portuguesa no contexto de revisões de aparelhos de TV e resenhas literárias nos conjuntos de dado TV e ReLi. Para alcançar este objetivo, modelos baseados na arquitetura BERT foram empregados, tanto na forma pré-treinada para domínios gerais (BERTimbau) quanto para domínios específicos (BERTtv e BERTreli). Além disso, uma técnica de duplo *embedding* foi implementada, combinando modelos de domínio geral e específico. Também foram avaliados Modelos de Linguagem de Larga Escala (*Large Language Models* - LLM), incluindo variantes do GPT-3 via API da OpenAI e uma variante do LLaMa, Cabrita, que é treinada para a língua portuguesa. Para otimizar a demanda por recursos de *hardware*, técnicas de ajuste fino eficiente, como LoRA (*Low-Rank Adaptation*) para o BERTimbau e QLoRa (*Quantized Low-Rank Adaptation*) para o Cabrita, foram aplicadas. Os resultados demonstraram que o modelo BERTimbau ajustado com LoRa se mostrou superior nos dois conjuntos de dados, alcançando F1 *scores* de 0.846 para o conjunto TV e 0.615 para o ReLi. Em contraste, o modelo Cabrita apresentou desempenho inferior, com resultados menos favoráveis para ambos os conjuntos de dados, 0.68 para o TV e 0.46 para o ReLi. Este estudo, portanto, oferece uma contribuição valiosa para a pesquisa em extração de aspectos em língua portuguesa, demonstrando a viabilidade e eficácia de adaptar e otimizar técnicas e modelos desenvolvidos originalmente para outros idiomas.

Palavras-chave: Processamento de Linguagem Natural. Extração de Aspectos. BERT. Modelos de Linguagem.

ABSTRACT

The identification and extraction of aspects are essential in text analysis for discerning opinions and emotions. However, there is a gap in applying these techniques to Portuguese. This work aims to adapt approaches originally developed for English to this language in the TV and ReLi datasets. The goal of this work is to evaluate the application of language models for aspect extraction in Portuguese in the context of TV device reviews and literary reviews in the TV and ReLi datasets. To achieve this goal, models based on the BERT architecture were employed, both in the pre-trained form for general domains (BERTimbau) and for specific domains (BERTtv and BERTreli). Additionally, a double embedding technique was implemented, combining general and specific domain models. Large Language Models (LLMs) were also evaluated, including variants of GPT-3 via the OpenAI API and a variant of LLaMa, Cabrita, which is trained for the Portuguese language. To optimize hardware resource demand, efficient fine-tuning techniques such as LoRA (Low-Rank Adaptation) for BERTimbau and QLoRa (Quantized Low-Rank Adaptation) for Cabrita were applied. The results showed that the BERTimbau model adjusted with LoRA was superior in both datasets, achieving F1 scores of 0.846 for the TV dataset and 0.615 for ReLi. In contrast, the Cabrita model showed inferior performance, with less favorable results for both datasets, 0.68 for TV and 0.46 for ReLi. This study, therefore, offers a valuable contribution to research in aspect extraction in Portuguese, demonstrating the feasibility and effectiveness of adapting and optimizing techniques and models originally developed for other languages.

Keywords: Natural Language Processing. Aspect Extraction. BERT. Language Models.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A identificação e extração de aspectos são tarefas fundamentais no campo do Processamento de Linguagem Natural (PLN) e desempenham um papel crucial na Análise de Sentimentos (WANKHADE; RAO; KULKARNI, 2022; PEREIRA, 2021). A identificação e extração de aspectos é um processo fundamental na análise de textos, que consiste em reconhecer e selecionar os elementos mais relevantes associados às opiniões e emoções expressas pelos indivíduos. Tais elementos podem incluir entidades, características ou propriedades que são cruciais para a compreensão dos diversos contextos em que os textos podem ser encontrados, tais como avaliações de produtos, serviços e eventos.

Essencial para compreender opiniões, a extração de aspectos possibilita discernir componentes específicos mencionados nos textos, em vez de uma análise superficial da opinião global. Este procedimento viabiliza uma análise detalhada das opiniões e sentimentos expressos (HU; LIU, 2004; MUKHERJEE; LIU, 2012; PORIA; CAMBRIA; GELBUKH, 2016). Por exemplo, considere a frase: "A câmera do smartphone é incrível, mas a bateria dura pouco". Neste caso, os aspectos são "câmera" e "bateria", enquanto as opiniões/emoções associadas são "incrível" e "dura pouco", respectivamente. No contexto empresarial, a extração de aspectos e suas associações com opiniões e emoções pode oferecer informações valiosas sobre a percepção dos consumidores em relação aos produtos ou serviços, sendo útil para embasar decisões baseadas nas opiniões dos clientes.

Modelos de linguagem baseados em *Deep Learning*, como BERT (*Bidirectional Encoder Representations from Transformers*) e seus derivados, têm revolucionado o campo da extração de aspectos, oferecendo capacidades de compreensão contextual profunda de textos (DEVLIN et al., 2018; SUN; HUANG; QIU, 2019). Estes modelos são treinados em vastos conjuntos de dados e aprendem representações de palavras que capturam nuances semânticas complexas e relações contextuais. A aplicação desses modelos ao problema da extração de aspectos tem demonstrado resultados promissores, permitindo não apenas a identificação de termos relevantes, mas também a inferência de sentimentos e opiniões complexas associadas a eles (SUN; HUANG; QIU, 2019; XU et al., 2019). Além disso, técnicas de aprendizagem profunda especializadas, como redes

neurais convolucionais (CNNs) e redes neurais recorrentes (RNNs), têm sido adaptadas para melhor capturar relações sequenciais e estruturais em dados textuais, potencializando ainda mais a eficácia dos modelos de linguagem na extração de aspectos (ZHANG; WANG; LIU, 2018).

A maioria dos estudos relacionados a análise de sentimentos, incluindo as tarefas como a extração de aspectos, são realizados na língua inglesa, gerando uma lacuna considerável no estudo e aplicação de técnicas voltadas para outros idiomas, incluindo o português. A complexidade linguística, as peculiaridades gramaticais e os recursos disponíveis para o português trazem desafios adicionais para qualquer tarefa de PLN (PEREIRA, 2021).

1.1 Objetivos

O objetivo deste trabalho consiste em avaliar a aplicação de modelos de linguagem para extração de aspectos na língua portuguesa no contexto de revisões de aparelhos de TV e resenhas literárias. Para isso, serão utilizados modelos baseados na arquitetura *Transformers* (VASWANI et al., 2017), como é caso do BERT, que foi aplicado em Karimi, Rossi e Prati (2020) e Karimi, Rossi e Prati (2021) com sucesso e de Modelos de Linguagem de Larga Escala (LLM).

Como contribuição, dois modelos seguindo a arquitetura BERT serão pré-treinados em domínios específicos. Um deles será treinado com base em dados de resenhas literárias e o outro será baseado em avaliações de produtos, especificamente, em análises de modelos de TV. Esses dois modelos poderão ser utilizados em situações nas quais um domínio específico seja vantajoso para a execução da tarefa.

Para avaliar as técnicas abordadas, dois conjuntos de dados serão utilizados. O primeiro é o ReLi (Resenha de Livros) (FREITAS et al., 2012), que conta com resenhas literárias de usuários diversos e o segundo é o conjunto TV (CARDOSO; PEREIRA, 2020). Esses conjuntos serão estratificados e, assim como os dois modelos pré-treinados produzidos neste trabalho, serão disponibilizados para que outros pesquisadores possam comparar outras técnicas.

1.1.1 Objetivos Específicos

- Examinar a aplicação de modelos de linguagem pré-treinados em domínios específicos para a extração de aspectos e avaliar seu desempenho nessa tarefa.
- Avaliar técnicas de Ajuste Fino Eficiente de Parâmetros (*Parameter-Efficient Fine-Tuning*), como Adaptação de Baixo Posto (*Low-Rank Adaptation* - LoRa) e Adaptação de Baixo Posto Quantizada (*Quantized Low-Rank Adaptation* - QLoRA), para diminuir a necessidade de recursos avançados de hardware.
- Comparar o desempenho entre modelos treinados em domínio geral e modelos treinados em domínios específicos.
- Combinar os modelos de domínio geral e de domínio específico e avaliar se existe vantagem nessa abordagem.

1.2 Estrutura do Texto

Este trabalho segue a seguinte estrutura:

- Capítulo 1 (Introdução): Estabelece o cenário do estudo, introduzindo a relevância da identificação e extração de aspectos em PLN e análise de sentimentos, enfatizando a inovação trazida pelos modelos de linguagem e *Deep Learning* no tratamento de idiomas menos representados como o português.
- Capítulo 2 (Fundamentação Teórica): Detalha a teoria, abordando a análise de sentimentos, suas particularidades no contexto da língua portuguesa, e análise de sentimento baseada em aspecto, além de descrever os modelos sequenciais como o *Transformers*, modelos derivados dessa arquitetura, e adaptadores como LoRA e QLoRa.
- Capítulo 3 (Materiais e Métodos): Discorre sobre as ferramentas e os conjuntos de dados utilizados, os métodos de pré-treino e ajuste fino implementados, e as estratégias de avaliação empregadas, como o método *hold-out* e a métrica F1 *score*.
- Capítulo 4 (Resultado): Apresenta os resultados dos experimentos, comparando a eficácia de diferentes configurações de pré-treino e ajuste fino .

- Capítulo 5 (Conclusão): Sumariza as descobertas e propõe direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os conceitos fundamentais e as técnicas predominantes em Análise de Sentimentos na Seção 2.1, estabelecendo uma base sólida para a compreensão dos tópicos subsequentes. A Seção 2.2 aborda especificamente a Análise de Sentimentos em Língua Portuguesa, destacando os desafios associados ao processamento desse idioma. Em seguida, a Seção 2.3, foca na Análise de Sentimento Baseada em Aspecto (ABSA), uma abordagem que busca entender os sentimentos em relação a aspectos específicos. Modelos Sequenciais são explorados na Seção 2.4, seguido por uma explicação detalhada da arquitetura dos *Transformers* na Seção 2.5, um avanço significativo na modelagem de sequências de texto. A Vetorização Textual é o foco da Seção 2.6, onde são discutidas técnicas cruciais como *Word2Vec* e BERT, cada uma com suas próprias subseções detalhadas. Posteriormente, é apresentado os seguintes modelos de linguagem na Seção 2.7: GPT, LLaMa e o Cabrita. A Seção 2.8 introduz o LoRA e o QLoRa, duas técnicas avançadas para aprimorar a eficiência dos modelos de *Transformer*. Finalmente, a Seção 2.9 proporcionará uma visão compreensiva dos Trabalhos Relacionados, conectando o conteúdo deste capítulo ao estado atual da arte e fornecendo uma ponte para as discussões nos capítulos subsequentes.

2.1 Análise de Sentimentos

O estudo da análise de sentimentos envolve a compreensão computacional das opiniões, sentimentos, emoções e avaliações das pessoas em relação a entidades como produtos, serviços, organizações, indivíduos, questões, eventos e tópicos, bem como seus atributos. Desde o início do século 21, a análise de sentimentos tem se tornado cada vez mais relevante e sendo uma das áreas mais ativas de pesquisa em processamento de linguagem natural (ZHANG; WANG; LIU, 2018).

A análise de sentimentos pode ser útil para muitas entidades, incluindo empresas, governos e indivíduos comuns. Muito do que as pessoas produzem precisa ser avaliado com técnicas de mineração de texto e análise de sentimentos. Contudo, há desafios na avaliação e análise de sentimentos, o que pode dificultar a detecção exata do significado dos sentimentos e da polaridade correta do sentimento (HUSSEIN, 2018).

Como abordado por Medhat, Hassan e Korashy (2014), a análise de sentimentos é o processo de classificar os sentimentos expressos em um texto como positivos, negativos ou de acordo com uma escala, que varia de muito bom até muito ruim. Existem três principais níveis de classificação na análise de sentimentos: o nível de documento, o nível de sentença e o nível de aspecto. O nível de documento classifica um documento como positivo ou negativo considerando o documento como uma unidade que aborda apenas um único tópico. O nível de sentença classifica o sentimento expresso em cada sentença, determinando se é positivo ou negativo. No entanto, esses níveis não fornecem detalhes suficientes das opiniões sobre todos os aspectos de uma entidade. O nível de aspecto classifica o sentimento em relação aos aspectos específicos de uma entidade, identificando primeiro as entidades e seus aspectos.

Um aspecto, no contexto da análise de sentimentos, refere-se a uma característica específica ou atributo de um objeto, produto ou serviço mencionado em um texto. Por exemplo, considere a frase: "O processador deste notebook é rápido, mas a tela possui resolução baixa". Neste caso, "processador" e "tela" são aspectos do objeto em discussão, que é o notebook. As opiniões relacionadas a esses aspectos são "rápido" (uma avaliação positiva) e "resolução baixa" (uma avaliação negativa).

2.2 Análise de Sentimentos em Língua Portuguesa

As técnicas no campo do processamento de linguagem natural, direcionadas à análise de sentimentos, são amplamente investigadas no contexto da língua inglesa. No entanto, para a língua portuguesa, essa investigação é consideravelmente limitada (FREITAS; VIEIRA, 2015; GARCIA; BERTON, 2021; PEREIRA, 2021). A fim de ilustrar essa disparidade, foi realizada uma busca nas bases de dados Scopus e CAPES Periódicos, embora seja importante notar que essa busca por si só não constitui uma revisão sistemática da literatura. É preciso ter em mente que a ausência do termo "português" no título de um trabalho não implica necessariamente que o idioma português não tenha sido abordado nas análises realizadas, mas pode servir de parâmetro para demonstrar a disparidade mencionada. A Tabela 2.1 exibe o número de artigos encontrados que discutem a análise de sentimentos em inglês, em contraste com aqueles que também tratam da análise de

sentimentos em português, destacando a diferença na quantidade de pesquisas disponíveis para ambos os idiomas.

Tabela 2.1 – Número de artigos por busca e buscador

Buscador	Resultados busca 1	Resultados busca 2
Scopus	5.525	18
Periódicos	4.876	7

Fonte: Do autor.

A premissa da busca é capturar o total de artigos voltados principalmente para a língua inglesa, removendo os 9 idiomas mais falados do mundo[1] depois do inglês para a primeira busca e captar apenas os artigos voltados para a língua portuguesa para a segunda busca. Ambas as buscas foram feitas para capturar artigos publicados nos últimos 10 anos.

- Busca 1: (TITLE (sentiment AND analysis) AND NOT ALL (portuguese) AND NOT ALL (chinese) AND NOT ALL (spanish) AND NOT ALL (arabic) AND NOT ALL (indonesian) AND NOT ALL (malaysian) AND NOT ALL (french) AND NOT ALL (japanese) AND NOT ALL (russian) AND NOT ALL (german)) AND PUBYEAR > 2012 AND PUBYEAR > 2012
- Busca 2: (TITLE (sentiment AND analysis) AND TITLE (portuguese) AND NOT ALL (chinese) AND NOT ALL (spanish) AND NOT ALL (arabic) AND NOT ALL (indonesian) AND NOT ALL (malaysian) AND NOT ALL (french) AND NOT ALL (japanese) AND NOT ALL (russian) AND NOT ALL (german)) AND PUBYEAR > 2012 AND PUBYEAR > 2012

Com base na premissa de que as abordagens linguísticas podem não funcionar da mesma forma em diferentes idiomas, a exploração de novas técnicas pode ser um caminho para avanços significativos na compreensão de idiomas distintos e no desenvolvimento de novos métodos. Neste contexto, o trabalho de Araújo, Pereira e Benevenuto (2020) demonstrou relevância ao comparar distintas abordagens de análise de sentimentos, englobando aquelas que recorrem à tradução para o inglês, seguida do emprego de técnicas específicas para esse idioma, e aquelas que se baseiam

[1] https://www.internetworldstats.com/stats7.htm

em técnicas nativas para outras línguas. O resultado indicou que a técnica que utiliza tradução para o inglês apresentou os melhores resultados. Essa abordagem também é interessante do ponto de vista do custo, uma vez que os tradutores comerciais são eficientes e amplamente disponíveis. No entanto, é preciso ressaltar que, embora a tradução para o inglês possa ser uma opção viável para determinados contextos, cada idioma possui particularidades que podem requerer técnicas específicas de análise, e estas estratégias multilíngue não são capazes de aprender estas especificidades (CHEN et al., 2019).

O estudo realizado por Pereira (2021) consiste em uma revisão do estado da arte na área de análise de sentimentos para a língua portuguesa. O autor do estudo identifica as principais particularidades dos idiomas, tais como gírias, abreviações e expressões idiomáticas, que não podem ser plenamente capturadas pelos mecanismos de tradução automática. A fim de ilustrar as limitações desses mecanismos, a Tabela 2.2 é apresentada com alguns exemplos relevantes.

Tabela 2.2 – Exemplos de resenhas com expressões em português, sua tradução para o inglês pelo Google *Translator* e o significado das expressões

Revisões	Tradução	Significado em português
Smartphone **meia-boca**	***Half-mouthed*** *smartphone*	De qualidade média a ruim
Cheguei no hotel, o quarto não estava pronto, **armei o maior barraco**	*I arrived at the hotel, the room was not ready,* ***I set up the biggest shack***	Criar confusão em público, discutindo ou brigando com alguém
A investigação do político **acabou em pizza**	*The politician's investigation* ***ended in pizza***	O caso de corrupção acabou encerrado sem a punição do político
A comida do restaurante **está daqui**!	*The restaurant's food is* ***from here****!*	Comida muito saborosa

Fonte: (PEREIRA, 2021).

2.3 Análise de Sentimento Baseada em Aspecto

O objetivo da Análise de Sentimento Baseada em Aspecto é detectar os sentimentos presentes em textos em relação a determinados aspectos, que são entendidos como características de

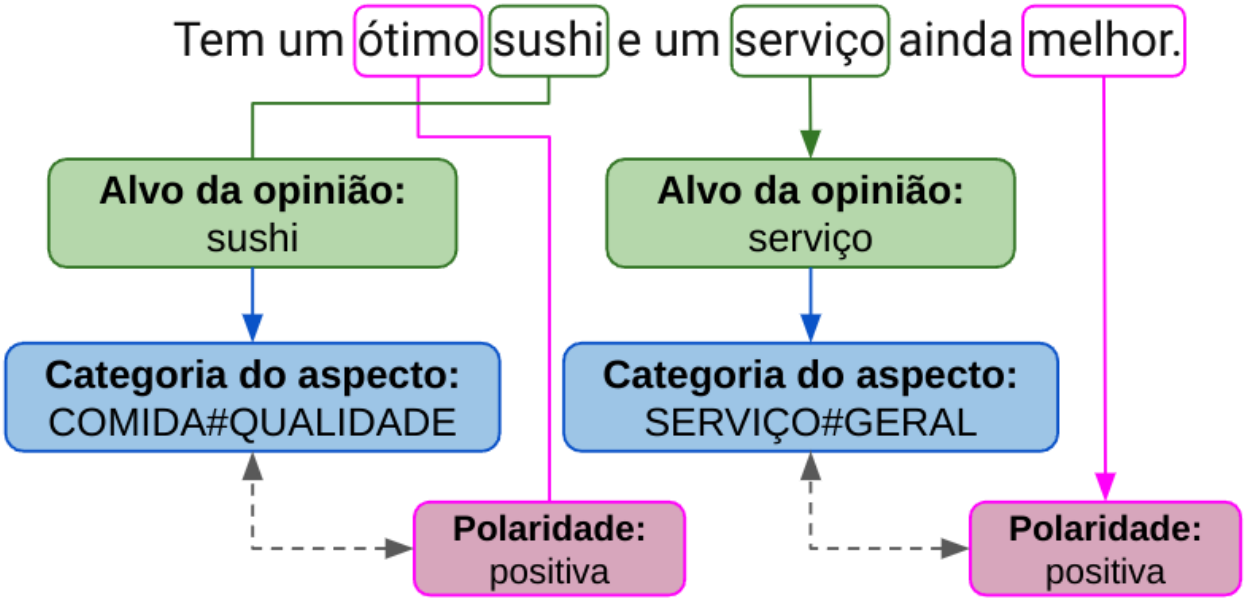
uma entidade. Estes podem ser comunicados de forma explícita ou implícita. O aspecto explícito é prontamente perceptível na sentença, como no exemplo “Adorei a câmera deste smartphone, as fotos ficam incríveis”. Observe que, nesse caso, o aspecto “câmera” é claramente explicitado e o sentimento é positivo. Por outro lado, em se tratando do aspecto implícito, este não é diretamente declarado no texto, tal como no exemplo “Este celular me parece excessivamente caro”. O autor da sentença se refere ao aspecto “preço”, que, neste caso, não é apresentado explicitamente no texto, mas transmite um sentimento negativo em relação ao custo.

Em Pavlopoulos (2014) são apresentadas três principais sub-tarefas que compõe a ASBA, que são: extração dos aspectos, agregação dos aspectos e estimativa de polaridade dos aspectos. A Figura 2.1 ilustra estas três tarefas. A primeira tarefa busca identificar a palavra, ou as palavras, que são entendidas como os aspectos, na frase da imagem os aspectos são as palavras “sushi” e “serviço”. A segunda tarefa visa agregar os aspectos que representam uma mesma característica em um aspecto único, na imagem o aspecto “sushi” pode ser agregada a categoria comida. E por fim, o objetivo da terceira tarefa é identificar o sentimento relacionado aos aspectos encontrados, para os aspectos da imagem, ambos estão relacionados à sentimentos positivos. Este trabalho foca apenas na primeira sub-tarefa para aspectos explícitos.

De acordo com Liu (2012) existem quatro principais abordagens para realizar tarefas de extração de aspectos, são elas: baseadas em frequência, baseados em relação, baseadas em aprendizado supervisionado e baseadas em tópicos.

A abordagem que se baseia na frequência de termos considera que, dentro de um determinado domínio, como as análises de produtos, os usuários tendem a utilizar palavras semelhantes para expressar suas opiniões sobre a qualidade dos produtos. Com base nessa suposição, a abordagem busca identificar os termos mais frequentes nas avaliações e utilizá-los para identificar os aspectos (LIU, 2012). A extração de aspectos em revisões de produtos foi abordada por Hu e Liu (2004), que utilizou a técnica descrita. Nesse trabalho, os substantivos e frases nominais foram identificados com o auxílio de um POS (*part-of-speech*) *tagger*, e em seguida, os termos mais frequentes foram selecionados como aspectos relevantes para a análise. O limiar de frequência pode ser determinado de forma experimental para ajustar a sensibilidade da técnica.

Figura 2.1 – As três tarefas da ASBA



A imagem exibe um diagrama de análise de sentimentos baseado em aspectos, onde uma sentença é decomposta para avaliar aspectos específicos. O termo "sushi" é associado ao grupo de aspectos "COMIDA#QUALIDADE" com uma polaridade positiva, enquanto "serviço" é relacionado ao grupo de aspectos "SERVIÇO#GERAL", também com polaridade positiva.

Fonte: Adaptado de Do et al. (2019).

Partindo da premissa de que toda opinião possui um alvo, as abordagens baseadas em relação estabelecem uma conexão entre a opinião e seu objeto de referência, o aspecto. Nesse sentido, é possível explorar tais relações quando as palavras de sentimento são conhecidas (LIU, 2012). Hu e Liu (2004) utilizam essa abordagem para identificar os aspectos menos frequentes. Nesse trabalho, os substantivos ou frases nominais mais próximos de palavras de sentimento são extraídos como aspectos.

Os modelos de aprendizado supervisionado dependem de dados para treinamento que sejam rotulados, isto é, identificando quais palavras são ou não aspectos. As principais técnicas nessa abordagem são os modelos sequenciais (LIU, 2012). Algumas das principais técnicas são: *Hidden Markov Models* (HMM) (JIN; HO; SRIHARI, 2009), *Conditional Random Fields* (CRF) (JAKOB; GUREVYCH, 2010) e mais recentemente modelos baseados na arquitetura *Transformers* (SANTOS; MARCACINI; REZENDE, 2021).

Os modelos de tópicos são abordagens de aprendizado não supervisionado usadas para descobrir padrões em grandes conjuntos de documentos. Em termos gerais, os modelos de tópicos usam técnicas de modelagem probabilística para identificar os tópicos presentes em um conjunto de documentos. Eles são baseados na premissa de que cada documento é uma mistura de tópicos e cada tópico é uma mistura de palavras. Em tarefas de extração de aspectos, estes tópicos são vistos como os aspectos a serem extraídos (PEREIRA, 2021). Uma das implementações mais comuns de modelos de tópicos é o *Latent Dirichlet Allocation* (LDA), que é um modelo estatístico que identifica tópicos latentes em um conjunto de documentos (JELODAR et al., 2019).

2.4 Modelos Sequenciais

As palavras em modelos sequenciais são entendidas como atributos para o modelo. Sendo assim, os modelos sequenciais são expostos à sequência original de um texto, aprendendo assim quais são os principais atributos. Esses modelos são especializados em processar sequências de valores $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(T)}$.

As Redes Neurais Recorrentes (RNN) são um exemplo de modelos sequenciais, capazes de processar sequências muito mais longas do que redes que não são baseadas em sequências. Além disso, as RNNs são capazes de processar diferentes comprimentos de sequências (GOODFELLOW; BENGIO; COURVILLE, 2016). Por conta dessa característica, as RNNs são amplamente utilizadas em tarefas de PLN.

A arquitetura mais difundida que faz uso das RNNs foi proposta em (CHO et al., 2014), uma arquitetura do tipo RNN *encoder-decoder* para tarefas de tradução de textos. Nesse caso, o *encoder* realiza o mapeamento de uma sequência de tamanho variável em um vetor de tamanho fixo e o *decoder* decodifica esse vetor de tamanho fixo novamente em uma sequência de tamanho variável.

Uma outra arquitetura que também faz uso do conceito de *encoder-decoder* foi apresentada em Vaswani et al. (2017). Essa arquitetura se baseia totalmente em mecanismos de atenção neural, eliminando a necessidade do uso de recorrência ou convolução. Esse trabalho foi uma virada de chave na área de PLN, gerando uma série de avanços significativos, elevando o estado da arte em

muitas tarefas, tais como: classificação de texto, perguntas e resposta, resumo automático de texto, entre outros. Algumas modelos baseados nessa arquitetura surgiram desde então, como é o caso do BERT e do GPT.

2.5 Transformers

Apesar de ser um modelo sequencial que processa as palavras de entrada uma por vez, o *Transformer* apresenta uma abordagem baseada em atenção que o diferencia de outras arquiteturas sequenciais. Essa abordagem permite que o modelo considere todas as palavras da sequência simultaneamente, em vez de processá-las em uma ordem fixa. Dessa forma, o modelo é capaz de capturar as relações entre as palavras em uma sentença, independentemente de sua ordem original (CHOLLET, 2021).

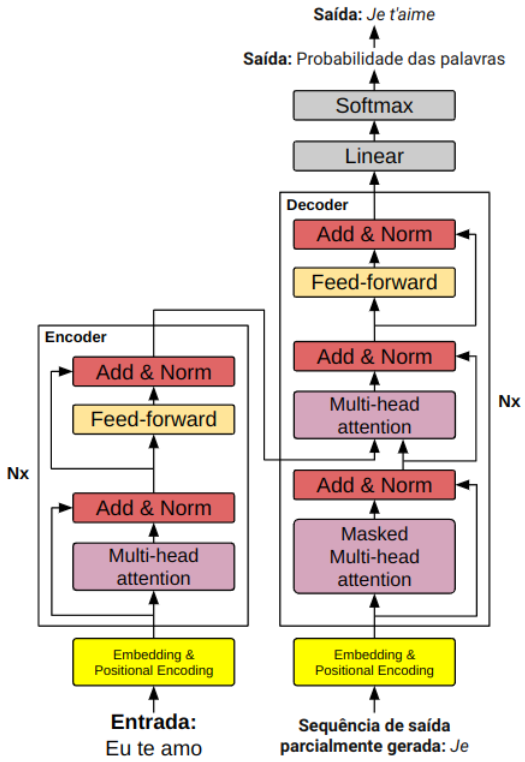
A Figura 2.2 ilustra a arquitetura do *Transformers*. O bloco da esquerda representa o *encoder*, que pode ser empilhado *N* vezes, bem como o *decoder*, representado pelo bloco da direita. As subseções a seguir irão descrever cada elemento presente no *Transformers*.

2.5.1 Atenção

O mecanismo de atenção permite a captura de informações de relacionamento entre todas as palavras em uma sequência, independente da sua posição relativa na sequencia. Isto é feito calculando os pontos de atenção para cada palavra de entrada, as palavras com maior pontuação são mais relevantes e as com menor pontuação são menos relevantes, tornando este mecanismo capaz de criar novos atributos sensíveis ao contexto (CHOLLET, 2021).

A atenção é calculada como uma função que mapeia a consulta e um conjunto de pares chave-valor, onde a saída é uma soma ponderada dos valores obtidos. Isto é, as palavras de entrada são projetadas em três vetores separados: *query* (Q), *key* (K), *value* (V) (consulta, chave e valor) e multiplicadas por três matrizes de projeção diferentes. Em seguida, o produto escalar entre o vetor de consulta e o vetor de chave é calculado para cada palavra de entrada. O resultado dessa operação

Figura 2.2 – *Transformers*

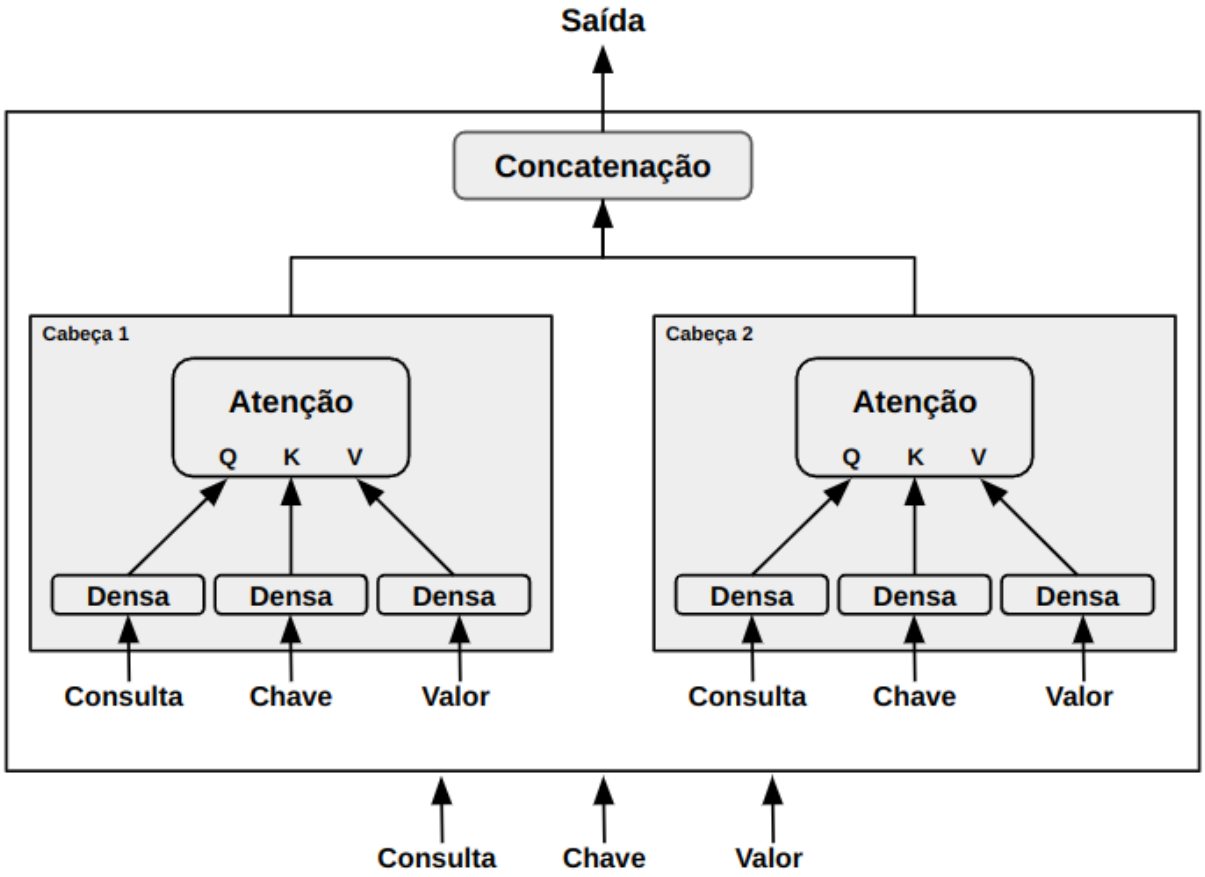


A imagem mostra a arquitetura *Transformer* aplicada à tradução automática. No *encoder*, a entrada de texto é convertida em vetores através de camadas de *embedding* e *positional embedding*, seguida de múltiplas camadas de *multi-head attention* e normalização. O *decoder* utiliza uma estrutura similar, mas inclui *masked multi-head attention* para gerar a sequência de saída. Ambos *encoder* e *decoder* aplicam normalização e camadas *feed-forward*, com o *decoder* finalizando com camadas lineares e *softmax* para determinar as probabilidades das palavras seguintes na sequência de saída.
Fonte: Adaptado de Vaswani et al. (2017).

gera uma matriz de pontuação, que é normalizada através de uma normalização *softmax*, e por fim multiplicada pelo vetor de valor, obtendo assim a saída final da atenção (VASWANI et al., 2017).

O mecanismo de atenção utilizado no *Transformers* é conhecido como "*multi-head attention*" (atenção de múltiplas-cabeças em tradução direta), que consiste em dividir a representação de entrada em várias projeções lineares, denominadas "cabeças", e calcular a atenção de forma paralela e independente em cada uma delas. Após o cálculo da atenção em todas as cabeças, as saídas são concatenadas para gerar a representação final (VASWANI et al., 2017). Essa técnica é ilustrada na Figura 2.3, que representa o mecanismo aplicado em duas cabeças de atenção.

Figura 2.3 – *Multi-head Attention*



A imagem descreve o mecanismo de *multi-head attention* com duas cabeças (*heads*), cada uma com seus próprios fluxos de consulta (Q), chave (K) e valor (V), processam informações através das camadas densas. Essas informações são então combinadas em uma operação de atenção centralizada para cada cabeça. As saídas de ambas cabeças são concatenadas para formar a saída final do mecanismo. Fonte: Adaptado de Chollet (2021).

2.5.2 *Encoder*

O *encoder* utilizado isoladamente, ou seja, separado do *decoder*, pode ser utilizado para tarefas de classificação de texto, por exemplo. Uma vez que se trata de um módulo bastante genérico capaz de ingerir uma sequência de entrada e aprender a tranformá-la em representações mais úteis (CHOLLET, 2021).

Esse bloco (à esquerda da Figura 2.2) é composto de N camadas idênticas empilhadas, cada uma delas contendo duas subcamadas. A primeira subcamada é responsável por calcular a aten-

ção de múltiplas-cabeças e a segunda subcamada é uma camada densa (*feed-foward*). A saída de cada uma destas subcamadas é incrementada por conexões residuais seguida de uma normalização, tornando assim o treinamento mais estável, garantindo que as informações sejam transmitidas de forma adequada entre as camadas (VASWANI et al., 2017).

2.5.3 *Decoder*

De forma similar ao *encoder*, o *decoder* (à direita da Figura 2.2) é constituído por N camadas empilhadas, no entanto, a sua arquitetura apresenta uma subcamada extra. Esta subcamada extra fica entre a atenção de múltiplas-cabeças da sequência alvo e da camada *feed-foward*, recebendo a saída do *encoder* para ser processada. A primeira subcamada é uma máscara de atenção, que permite que o *decoder* atue apenas em informações anteriores da sequência alvo, evitando que ele use informações futuras que ainda não foram geradas (VASWANI et al., 2017).

2.5.4 *Positional Encoding*

A ideia por trás do *positional encoding* é permitir que o modelo tenha acesso às informações referente a ordem das palavras. Para isso, é introduzido informações relativas ou absolutas da posição de cada *token* na sequência. Estas informações são adicionadas ao *embedding* e introduzidas tanto ao *encoder* quanto ao *decoder* (VASWANI et al., 2017).

Esse vetor de posições pode ser fixo ou aprendido. Em Vaswani et al. (2017) é demostrado que não existem diferenças significativas entre estas duas abordagens. Uma forma eficaz de codificar as posições é restringir os valores possíveis no intervalo [-1, 1] usando funções seno e/ou cosseno (CHOLLET, 2021).

2.6 Vetorização Textual

Para que qualquer algoritmo de aprendizado de máquina possa interpretar conteúdos textuais, uma das primeiras etapas em qualquer tarefa de PLN é transformar estes textos em vetores

numéricos, conhecidos como *embeddings*. Este processo é chamado de vetorização de texto ou vetorização textual.

Os *embeddings* podem ser gerados principalmente por meio de modelos baseados em previsão (*prediction-based models*) ou através de modelos baseados em contagem (*count-based models*). O primeiro utiliza informações contextuais, isto é, informações locais (a nível de sentença, por exemplo), e o segundo utiliza informações globais (estatísticas a nível de documento ou coleção de documentos) (ALMEIDA; XEXÉO, 2019). Neste trabalho, faz-se uso apenas dos modelos baseados em previsão.

Um *word embedding* é um vetor numérico onde cada dimensão corresponde a um recurso que representa uma característica da palavra, capturando propriedades sintáticas e semânticas. Esses *embeddings* são tipicamente gerados por meio de modelos neurais (TURIAN; RATINOV; BENGIO, 2010).

Modelos neurais são responsáveis por produzir representações complexas, mas o treinamento desses modelos pode levar dias ou até semanas, devido à grande quantidade de dados a que são expostos. Embora esses modelos tendam a ter desempenho superior a modelos baseados em frequência, essa complexidade pode ser uma desvantagem em relação a esses últimos, devido ao tempo de treinamento necessário (ALMEIDA; XEXÉO, 2019).

Em situações onde há uma quantidade insuficiente de dados de treinamento, uma opção é utilizar representações pré-calculadas, que são altamente estruturadas e possuem propriedades úteis. Nessa situação, reutilizar recursos aprendidos em outro problema é uma escolha sensata (CHOLLET, 2021).

Nesse contexto, destacam-se alguns modelos neurais, que são: *Word2Vec* (MIKOLOV et al., 2013) e o BERT (DEVLIN et al., 2018), apesar de que eles são muito mais do que apenas modelos de vetorização.

2.6.1 *Word2Vec*

Word2Vec é uma técnica de PLN que visa aprender representações vetoriais de palavras a partir de um grande corpus[2] de texto, compostos por bilhões de palavras. A premissa por trás dessa técnica é que palavras que aparecem em contextos semelhantes tendem a ter significados similares, logo, palavras com significados similares devem ter representações vetoriais similares também. Dessa forma, é possível realizar operações algébricas simples nos vetores resultantes, como, por exemplo, "vetor(Rei) - vetor(Homem) + vetor(Mulher)" que resulta em um vetor próximo do vetor da palavra "Rainha" (MIKOLOV et al., 2013).

Essencialmente, existem dois modelos de arquitetura no *Word2Vec*: o *Continuous Bag-of-Words* (CBOW) e o *Continuous Skip-Gram*. Ambos os modelos têm camadas de entrada, oculta e saída e foram projetados principalmente para minimizar a complexidade computacional de trabalhos anteriores similares, a fim de permitir que as representações de palavras sejam aprendidas. Os desenvolvedores perceberam que o que causava a complexidade da rede era a não-linearidade da camada oculta, então utilizaram ativação log-linear para essa camada (MIKOLOV et al., 2013).

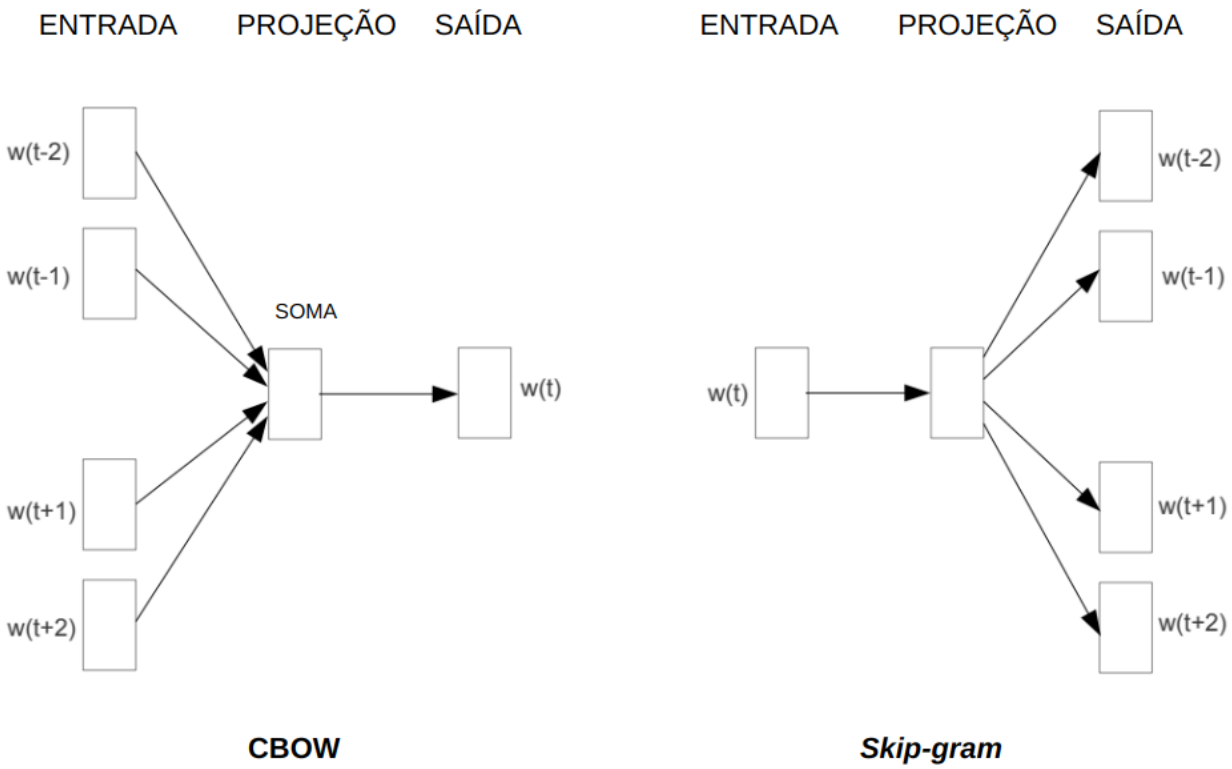
O objetivo do CBOW é prever uma palavra alvo dado as palavras que a precedem e a seguem na sentença. A Figura 2.4 apresenta a arquitetura descrita. Na camada de entrada, as palavras da sentença pré-processada são fornecidas nas posições $t-2$ a $t+2$ para a camada de projeção. Nesta camada, essas palavras são projetadas na mesma posição (calculada como a média dos vetores de entrada). Finalmente, a camada de saída prevê a palavra t, que é a que tem a maior probabilidade de se encaixar no meio da sentença (MIKOLOV et al., 2013).

Conforme ilustrado na Figura 2.4, o *Skip-Gram* trata-se de uma arquitetura oposta ao CBOW, ao invés de prever a palavra alvo t dado as palavras vizinhas ($t-2$ a $t+2$), como no CBOW, o modelo tenta prever as palavras vizinhas dado a palavra alvo t (MIKOLOV et al., 2013).

Em suma, o *Word2Vec* é uma técnica de gerar representações vetoriais densas e de dimensão reduzida para palavras, que captura muitas das propriedades sintáticas e semânticas delas. Além

[2] Em linguística e processamento de linguagem natural, "corpus" refere-se a uma coleção de textos escritos ou transcrições de fala que são usados para análise linguística e treinamento de modelos de inteligência artificial. Um corpus pode variar em tamanho e pode ser especializado em um tópico específico ou abranger uma ampla variedade de textos para fornecer uma visão mais geral da linguagem (BENNETT, 2010).

Figura 2.4 – Arquiteturas CBOW e *Continuous Skip-Gram*



A imagem apresenta as duas arquiteturas do *Word2Vec*: CBOW, que prediz uma palavra com base no contexto fornecido das palavras adjacentes, e *Skip-gram*, que usa uma palavra para prever o contexto circundante.
Fonte: Adaptado de (MIKOLOV et al., 2013).

disso, é computacionalmente eficiente e essas representações podem ser utilizadas como entrada para outros modelos, melhorando seu desempenho em tarefas como classificação de texto, agrupamento e reconhecimento de entidades nomeadas. No entanto, é importante notar que o *Word2Vec* não leva em consideração a ordem das palavras, e também não é sensível ao contexto, o que significa que a mesma palavra pode ter significados diferentes dependendo do contexto e, mesmo assim, gera uma mesma representação vetorial para ela.

2.6.2 BERT

A maioria das técnicas que produzem representações geradas por modelos de linguagem pré-treinados geralmente têm uma limitação comum: esses modelos foram treinados unidirecionalmente, ou seja, eles só conseguem ver e aprender com o contexto anterior a uma palavra específica,

como é o caso do GPT. Isso pode ser prejudicial para tarefas que requerem compreensão do contexto global da sentença (DEVLIN et al., 2018).

O BERT, como o nome sugere, também utiliza arquitetura *Transfomers* e é treinado bidirecionalmente, o que significa que ele aprende a considerar as palavras tanto na ordem da esquerda para direita quanto da direita para esquerda. Ele foi treinado utilizando duas estratégias: modelo de linguagem mascarada (*masked language model* - MLM) e previsão da próxima sentença (*next sentence prediction* - NSP).

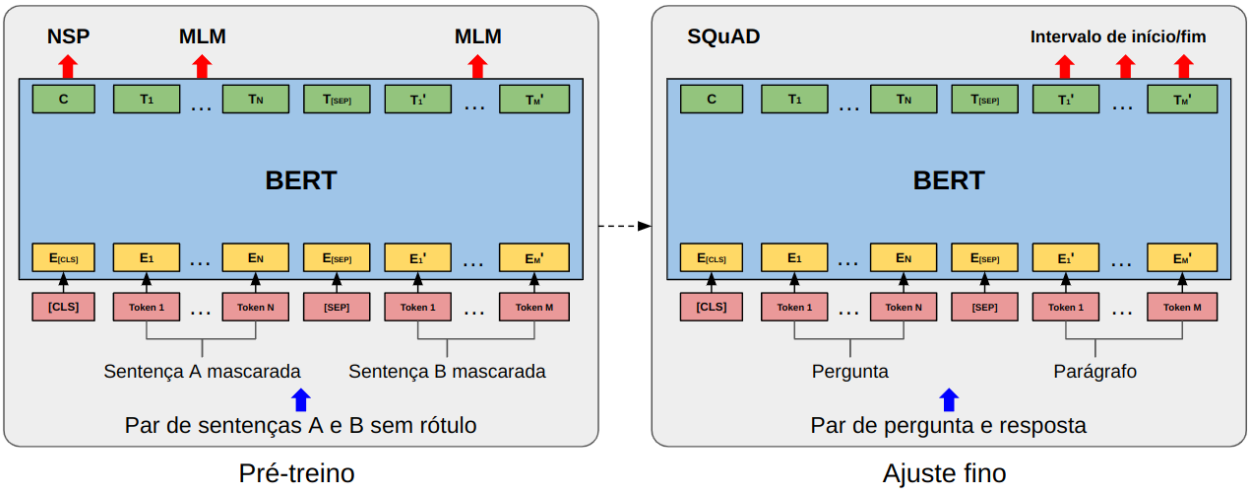
O processo do MLM consiste em mascarar aleatoriamente uma porcentagem dos *tokens* de entrada e prever o *token* que foi mascarado. Para isso, são selecionados aleatoriamente 15% dos *tokens* de entrada, onde 80% desses *tokens* são substituídos por [*MASK*], 10% por um *token* aleatório e os outros 10% não são alterados (DEVLIN et al., 2018).

O NSP tem como objetivo capturar as relações entre sentenças. Este aprendizado é obtido por meio de uma tarefa de classificação. O modelo é treinado para prever se uma segunda sentença é uma continuação lógica da primeira. Para tal, os dados de treinamento são compostos por pares de sentenças, metade das quais são efetivamente continuações da primeira, e são rotuladas como "*IsNext*", enquanto a outra metade é selecionada aleatoriamente do corpus e rotuladas como "*NotNext*" (DEVLIN et al., 2018).

As duas estratégias descritas nos últimos dois parágrafos fazem parte da etapa de pré-treinamento do BERT. A Figura 2.5 ilustra essas estratégias (bloco Pré-treino na Figura). O *token* [CLS] é utilizado no início de todas as sentenças de entrada e o *token* [SEP] é utilizado para separar o par de sentenças de entrada. Os *embeddings* dos *tokens* iniciais são denotados por E. O *token* da camada final C representa o vetor final para o *token* [CLS], sendo esta a representação utilizada para tarefas de classificação, como é o caso do NSP, e os *tokens* T são os vetores finais, isto é, os *embeddings* de cada *token*. A etapa seguinte é o ajuste fino (bloco Ajuste fino da Figura), onde os parâmetros aprendidos no pré-treinamento são inicializados para o treinamento da tarefa desejada.

Inicialmente, duas arquiteturas foram desenvolvidas, BERT_{BASE} ($L = 12$, $H = 768$, $A = 12$, 110M de de parâmetros) e o BERT_{LARGE} ($L = 24$, $H = 1024$, $A = 15$, 340M de parâmetros), sendo que L é o número de camadas de *encoders*, H é o tamanho das camadas ocultas e A o número de cabeças de *self-attention* (DEVLIN et al., 2018).

Figura 2.5 – Arquitetura e processos de pré-treinamento e ajuste fino do BERT



A imagem apresenta a etapa de pré-treino do BERT usando as estratégias NSP e MLM e a etapa de ajuste fino para tarefa de *question-answering* (pergunta e resposta) no conjunto de dados *Stanford Question Answering Dataset* (SQuAD), onde o BERT é treinado para identificar a localização exata da resposta em um texto, dadas perguntas específicas. Fonte: Adaptado de Devlin et al. (2018).

2.7 Modelos de Linguagem

Do ponto de vista técnico, a modelagem de linguagem representa um dos pilares fundamentais no progresso da capacidade linguística em sistemas computacionais. Tipicamente, essa abordagem se concentra em estabelecer um modelo para a probabilidade generativa de sequências textuais, com o objetivo de calcular a probabilidade de ocorrência de *tokens* subsequentes ou não presentes (ZHAO et al., 2023).

2.7.1 GPT

O *Generative Pre-trained Transformer* (GPT) é um modelo de Processamento de Linguagem Natural (PLN) que adota a arquitetura de *Transformer*, treinada de maneira autoregressiva para a geração de texto (RADFORD et al., 2018; RADFORD et al., 2019; BROWN et al., 2020). A metodologia central desse modelo é aprender a prever o *token* subsequente em uma sequência textual, com base nos *tokens* precedentes. Esta abordagem autoregressiva capacita o modelo a gerar repre-

sentações contextuais de *tokens*, variando de palavras individuais a textos completos, resultando em desempenho aprimorado em diversas tarefas de PLN.

A versão inaugural do GPT (RADFORD et al., 2018) representou um marco inicial substancial, sobretudo em relação à compreensão textual. Contudo, foi com o advento do GPT-2 que se observou uma melhoria notável na capacidade de geração textual, despertando atenção tanto acadêmica quanto industrial. Este modelo foi treinado em um corpus de dados mais extenso e exibiu avanços consideráveis em múltiplas tarefas de PLN, incluindo tradução automática, sumarização e geração autônoma de texto (RADFORD et al., 2019). Posteriormente, o GPT-3 expandiu tanto a arquitetura do modelo quanto o corpus de treinamento. Este modelo é notório por sua aptidão em executar diversas tarefas de PLN "*out-of-the-box*", requerendo apenas ajustes mínimos nos parâmetros de entrada e eliminando a necessidade de treinamento adicional especializado (BROWN et al., 2020).

A versão mais recente, GPT-4, difere de suas antecessoras por ser treinada através de Treinamento por Reforço com Feedback Humano (RLHF). Conforme delineado em (OPENAI, 2023), em virtude do ambiente competitivo em que a organização desenvolvedora opera, detalhes específicos relacionados à arquitetura, à escala do modelo, ao hardware empregado e ao conjunto de dados são intencionalmente omitidos. O relatório concentra-se em elucidar as competências e limitações inerentes ao modelo.

A arquitetura do GPT é relativamente simples em termos de componentes: é composta por camadas de *decoders* do *Transformers* que trabalham de forma autoregressiva, olhando apenas para as palavras anteriores na sequência para prever a próxima palavra. Essa simplicidade, no entanto, é compensada pelo enorme número de parâmetros (110M na primeira versão, 1.5B na segunda versão e 175B na terceira) e pela sofisticação dos algoritmos de otimização e treinamento.

Em síntese, o GPT e suas variantes constituem uma abordagem robusta para uma ampla gama de tarefas em PLN, abrangendo desde a geração de texto até análise de sentimento e tradução automática. O modelo distingue-se não apenas pela eficácia em tarefas focalizadas, mas também pela sua versatilidade em compreender e gerar linguagem de maneira abrangente, o que o torna aplicável em uma multiplicidade de domínios.

2.7.2 LLaMa

Em Touvron et al. (2023) é apresentado o *Large Language Model Meta AI* (LLaMa), um modelo de linguagem de grande escala, que busca alcançar o melhor desempenho possível em diversos orçamentos para a etapa de inferência, utilizando mais *tokens* em seu treinamento do que o comumente empregado. Este modelo se baseia na premissa apresentada por Hoffmann et al. (2022) de que, para um determinado orçamento computacional, os melhores desempenhos não são alcançados pelos maiores modelos, mas sim por modelos menores treinados em mais dados.

O LLaMA foi treinado em arquiteturas que variam de acordo com seu tamanho, entre 7 bilhões e 65 bilhões de parâmetros. O modelo com 13 bilhões de parâmetros se mostrou superior ao GPT-3 (BROWN et al., 2020), que possui 175 bilhões de parâmetros, na maioria dos *benchmarks*, mesmo possuindo uma arquitetura bem menor (TOUVRON et al., 2023). Basicamente, o LLaMA é um *Transformers* com algumas modificações, que são:

- Pré-normalização: Em vez de normalizar a saída de cada sub-camada do *Transformers*, conforme ilustra a Figura 2.2, a entrada de cada sub-camada é normalizada, e para isso, utiliza-se a função RMSNorm (ZHANG; SENNRICH, 2019).
- Função de ativação SwiGLU: Substituição da ReLU pela SwiGLU (SHAZEER, 2020) como função de ativação da camada *feed-foward*.
- *Rotary Embeddings*: O *Positional Embedding* é removido e, em vez disso, foram adicionados em cada camada de atenção, os *Rotary Positional Embeddings* (RoPE) (SU et al., 2021).

2.7.3 Cabrita

O trabalho de Larcher et al. (2023) delineia uma estratégia para melhorar a eficiência e o desempenho dos modelos de linguagem em idiomas não-ingleses, particularmente o português, através de uma metodologia nomeada Cabrita. Esta abordagem é concebida para abordar as limitações percebidas em modelos pré-treinados predominantemente com dados em inglês, quando aplicados ao contexto linguístico em português. O foco central do trabalho é a introdução e avaliação

dos modelos *openCabrita*, que são desenvolvidos com o objetivo de proporcionar uma tokenização mais eficiente e um desempenho aprimorado em tarefas de processamento de linguagem natural.

Um novo tokenizador é treinado exclusivamente em dados em português e posteriormente fundido com o tokenizador original do *OpenLLaMA* (GENG; LIU, 2023), mantendo a informação do idioma original enquanto adapta o modelo para lidar eficazmente com o português. Este processo de adaptação visa garantir que o texto em português seja tokenizado de maneira eficaz, minimizando a verbosidade que é típica quando se utiliza um tokenizador padrão para idiomas sub-representados (LARCHER et al., 2023).

Os modelos *openCabrita* são então submetidos a um processo de pré-treino contínuo utilizando um corpus em português. Esse processo é realizado sobre a infraestrutura de TPUs, empregando o framework *EasyLM* para facilitar o treinamento. O modelo *openCabrita3B*, que possui 3 bilhões de parâmetros, é uma evolução do modelo *OpenLLaMA*, otimizado para o idioma português, tanto em termos de tokenização quanto de desempenho em várias tarefas de processamento de linguagem natural (LARCHER et al., 2023).

2.8 Adaptadores

Com a expansão contínua do tamanho e complexidade dos modelos de linguagem, confrontamo-nos com a escalada correspondente nos recursos computacionais requeridos, uma barreira significativa para a pesquisa e aplicação prática. Diante dessa realidade, torna-se necessário investigar e aprimorar métodos inovadores de ajuste fino que não apenas economizem recursos valiosos, mas que também democratizem o acesso a tais tecnologias de ponta. Nesse sentido, destacamos duas abordagens nos tópicos a seguir.

2.8.1 LoRA

À medida que a dimensionalidade dos modelos pré-treinados de aprendizado de máquina se expande, a reconfiguração completa desses modelos através do re-treinamento de todos os parâmetros torna-se cada vez menos eficiente do ponto de vista computacional. Utilizar um modelo como

o GPT-3, que apresenta 175 bilhões de parâmetros, para ajuste fino exige um consumo de recursos computacionais extremamente caro. Para abordar essa limitação, Hu et al. (2021) propõem uma técnica de ajuste fino com eficiência de parâmetros denominada *Low-Rank Adaptation* (LoRA), a qual mantém os pesos da arquitetura pré-treinada estáticos e introduz matrizes de decomposição de baixo *rank* treináveis em cada camada da arquitetura do *Transformers*.

Considere uma matriz de peso pré-treinada $W_0 \in \mathbb{R}^{d \times k}$. A técnica de LoRA propõe uma atualização para essa matriz no formato $W_0 + BA$, onde $B \in \mathbb{R}^{d \times r}$ e $A \in \mathbb{R}^{r \times k}$ e o *rank* $r \leq \min(d,k)$. O passo de propagação para a frente modificado torna-se $h = W_0 x + BAx$, onde x é o vetor de entrada, e durante o treinamento, W_0 é mantido constante enquanto A e B contêm parâmetros treináveis (HU et al., 2021).

Mesmo que modelos como o GPT-3 possuam um número excessivo de parâmetros, a informação efetivamente útil que eles capturam pode ser representada em um espaço dimensional muito menor. Com base nisso, Hu et al. (2021) mostram que as mudanças nos pesos do modelo durante o ajuste fino também possuem um "*rank* intrínseco" baixo (HU et al., 2021). Isso significa que essas mudanças podem ser eficientemente capturadas e representadas usando uma matriz de baixo *rank*.

Este método representa uma redução significativa na quantidade de parâmetros treináveis necessários para futuras aplicações, otimizando assim a viabilidade do ajuste fino em modelos de grande escala. Por exemplo, a implementação de LoRA conseguiu reduzir o número de parâmetros treináveis no GPT-3 em uma magnitude de 10.000 e a exigência de memória da GPU por um fator de três (HU et al., 2021). Notavelmente, apesar da drástica redução na complexidade do modelo, o LoRA demonstrou um desempenho igual ou superior ao alcançado por métodos de ajuste fino tradicionais, sem acréscimo na latência de inferência (HU et al., 2021).

2.8.2 QLoRa

Em Dettmers et al. (2023) é introduzida uma abordagem inovadora para o ajuste fino de Grandes Modelos de Linguagem (LLMs) que permite fazer o ajuste fino de modelos de até 65 bilhões de parâmetros em uma única GPU de 48GB sem sacrificar a performance, sem esta técnica seria necessário uma GPU, ou várias GPUs, totalizando mais de 780GB de memória de vídeo.

Esta conquista é possibilitada através da quantização de 4-bits, especificamente utilizando o tipo de dado 4-bit *NormalFloat* (NF4), que é teoricamente ótimo para pesos de redes neurais distribuídos normalmente. Além disso, a técnica de *Double Quantization* é proposta, que visa quantizar as próprias constantes de quantização, resultando em economias substanciais de memória.

A metodologia se destaca não apenas pela quantização, mas também pela integração dos *Low-rank Adapters* (matrizes de baixo *rank*) do LoRA. O QLoRA desquantiza os pesos do modelo para realizar operações, mas apenas computa gradientes para os adaptadores LoRA, que utilizam o tipo de dado 16-*bit BrainFloat* (DETTMERS et al., 2023).

Outra inovação introduzida é o *Paged Optimizers*, que emprega a memória unificada da NVIDIA para gerenciar transferências de memória entre CPU e GPU, evitando erros de falta de memória durante o treinamento. Isso é essencial para garantir que grandes modelos possam ser treinados em máquinas convencionais (DETTMERS et al., 2023).

2.9 Trabalhos Relacionados

Em Machado et al. (2017), os pesquisadores analisaram um conjunto de dados significativo de aproximadamente 3,4 milhões de *tweets* para extrair aspectos relacionados a problemas de saúde de alta relevância no Brasil. A metodologia empregada envolveu várias etapas de pré-processamento, incluindo a remoção de ruídos, como *stopwords*, termos com menos de quatro caracteres e termos formados por símbolos ou números. A identificação dos termos substantivos relevantes foi realizada através do método *FREQ Baseline*, gerando uma lista de termos baseada na frequência de ocorrência. Os autores também aplicaram esse método, *FREQ Baseline*, para a criação de listas frequentes de bigramas e trigramas nos *tweets*. Para aprimorar a relevância contextual dos termos extraídos, foi incorporada uma etapa de "poda" utilizando o *Word2Vec*. Esse método permitiu a eliminação de termos não relacionados ao contexto das doenças em questão, resultando em listas de aspectos estritamente pertinentes aos temas estudados. A aplicação do método *FREQ Baseline*, combinado com o uso do *Word2Vec*, demonstrou uma melhoria significativa na qualidade dos resultados.

No trabalho de Machado, Pardo e Ruiz (2017) é realizada uma investigação de três abordagens para extração de aspectos: um método baseado em frequência (FREQ), o método híbrido proposto por Hu e Liu (2004), e uma combinação desses com a aplicação do modelo *Word2Vec*. O método FREQ foi aprimorado com o uso do modelo *Word2Vec*, que permitiu a extensão do passo de poda de limiar e a redução de termos detectados não relacionados. Similarmente, o método Hu e Liu (2004) foi adaptado para incorporar a poda *Word2Vec* após a detecção de termos frequentes, visando remover termos não-contextuais. Além disso, foi proposto um processo de análise *Word2Vec* para termos infrequentes localizados perto de palavras de opinião, buscando adicionar termos contextualizados à lista de termos infrequentes.

O estudo de Cardoso e Pereira (2020) investiga a eficácia de um método supervisionado para a extração de aspectos em textos em português e inglês. Baseado em um modelo proposto por Jakob e Gurevych (2010), que utiliza um modelo sequencial de *Conditional Random Fields* (CRF), o estudo analisou três bases de dados, duas em português e uma em inglês, com conclusão de que a tradução dos textos para a extração de aspectos em português não é significativamente benéfica. O método implementado analisa cada sentença com um classificador que leva em consideração a sequência de *tokens* e seus atributos. Ferramentas de PLN são utilizadas para extração de atributos tais como lema, classificação gramatical das palavras e a árvore de dependência das sentenças. O estudo também contribuiu com a criação de uma nova base de dados em português e ajustes na tradução.

A pesquisa de Lopes, Correa e Freitas (2021) propôs uma metodologia para extração de aspectos utilizando os modelos pré-treinados BERT Multilingual (DEVLIN et al., 2018), pré-treinado em diversos idiomas, incluindo o português, e o BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), pré-treinado apenas em português. Essa abordagem utilizou o *Sentence Pair Classifier* do BERT para prever a relação de aspectos com o texto, com entradas e rótulos "relacionados" e "não relacionados", e recorreu a rótulos ponderados para equilibrar o conjunto de dados. Os modelos foram treinados com o pré-processamento limitado à correção de erros ortográficos. Testes variados foram realizados, alterando hiperparâmetros e comparando dados pré-processados e não pré-processados. Para avaliar o desempenho, foi aplicada validação cruzada de 10 vezes, medindo precisão, F1, acurácia e acurácia balanceada, com configurações de hiperparâmetros baseadas nas

recomendações de Devlin et al. (2018), modificando principalmente o número de épocas para otimizar resultados. Os resultados indicaram que os modelos pré-treinados especificamente para a língua portuguesa alcançaram uma alta precisão balanceada mesmo com uma sequência máxima menor, superando os modelos multilíngues.

Em Li et al. (2023) é proposto um método para recomendações baseadas em aspectos, que maximiza a eficiência do ajuste de *prompts* suaves (*soft prompt tuning*) para extrair aspectos significativos das avaliações dos usuários. Utilizando modelos de linguagem pré-treinados como o BERT e o GPT-2, o método concentra em primeiro na personalização da extração de aspetos através da incorporação de *embeddings* de usuários e item, otimizando essas representações latentes para capturar preferências individuais. Após o ajuste fino do modelo para se adequar ao domínio específico dos dados, foi empregado um mecanismo de atenção em conjunto com uma rede de recomendação profunda para gerar predições de avaliações baseadas nos aspectos extraídos. O modelo é então refinado através de um procedimentos de treinamento alternado, que não apenas melhora a convergência mas também facilita a sincronização entre a representatividade dos termos de aspectos e sua utiliade nas recomendações. O modelo resultante superou significativamente os modelos de base em extração de termos de aspectos e recomendação baseada em aspectos, com melhorias consistentes nos conjuntos de dados utilizados (Amazon, Yelp e TripAdvisor).

Este estudo avança a tarefa de extração de aspectos ao realizar uma avaliação sistemática da aplicação de Modelos de Linguagem em domínios específicos, destacando-se pela inovação na aplicação de modelos pré-treinados adaptados ao domínio do problema. A pesquisa se distingue ao explorar métodos de Ajuste Fino Eficiente de Parâmetros, particularmente as técnicas LoRa e QLoRa, demonstrando como a restrição de recursos de *hardware* pode ser mitigada sem comprometer a eficácia da extração de aspectos. Além disso, este trabalho propõe um comparativo entre modelos treinados especificamente para certos domínios e aqueles de aplicação mais genérica, assim como a fusão dessas abordagens, fornecendo uma metodologia potencialmente superior para o aprimoramento da tarefa de extração de aspectos.

3 MATERIAIS E MÉTODOS

Este capítulo descreve a metodologia adotada neste estudo, ressaltando os procedimentos e técnicas empregados para atingir os objetivos estabelecidos. Inicialmente, as ferramentas e os *softwares* essenciais, os quais facilitaram a execução dos experimentos e análises, são apresentados na Seção 3.1. Em seguida, a Seção 3.2 oferece uma descrição pormenorizada do conjunto de dados utilizado.

Na sequência, a Seção 3.3 define o *baseline*, criando um ponto de comparação para avaliação do desempenho dos demais modelos. A técnica de *embedding* duplo é explorada na Seção 3.4. Posteriormente, a Seção 3.5 aborda de forma concisa o pré-treino do BERT de domínio específico, cobrindo desde a seleção do corpus até as particularidades das arquiteturas e estratégias do modelo.

A jornada que percorre desde a escolha das melhores configurações até o ajuste fino dos modelos para cada abordagem é detalhada na Seção 4.2. Adicionalmente, o método de avaliação adotado neste trabalho, o *hold-out*, é descrito na Seção 3.7. O capítulo se encerra com a Seção 3.8, que estabelece a métrica de avaliação utilizada para avaliar as abordagens.

3.1 Ferramentas e softwares

Para a captura e processamento dos dados, foi utilizado um computador com as seguintes especificações: processador (CPU) Ryzen 5 3600 com 6 núcleos, 12 *threads*, velocidade de 3.6GHz, GPU GTX 1660 TI com 6Gb de VRAM e duas memórias (RAM) DDR4 de 16Gb e velocidade 2666MHz cada, totalizando 32Gb de RAM.

Para a realização do treinamento dos modelos, optou-se pelo emprego de duas configurações computacionais distintas. A primeira delas consiste em uma máquina pertencente ao Laboratório de Recuperação da Informação, vinculado ao Departamento de Ciência da Computação da Universidade Federal de Lavras. Este computador está equipado com um processador Intel i7-10700F, que opera a uma frequência base de 2.9 GHz e pode alcançar até 4.8 GHz em modo turbo, além de possuir 8 núcleos e 16 *threads*. A máquina ainda dispõe de 128 GB de memória RAM DDR4, operando a 2667 MHz, e uma GPU NVIDIA GeForce RTX 3090, que conta com 24 GB de VRAM. A

segunda configuração utilizada encontra-se disponível na versão paga do Google Colab, ofertando 83,5 GB de memória RAM e uma GPU A100, equipada com 40 GB de VRAM.

Todos os códigos foram desenvolvidos utilizando a linguagem de programação *Python*. *Python* é uma linguagem de alto nível, interpretada e de propósito geral, caracterizada por sua legibilidade e simplicidade. A comunidade Python é notavelmente ativa e engajada, contribuindo para o seu amplo uso em diversas áreas, incluindo aprendizado de máquina, processamento de dados e inteligência artificial. As bibliotecas que foram utilizadas durante este trabalho são:

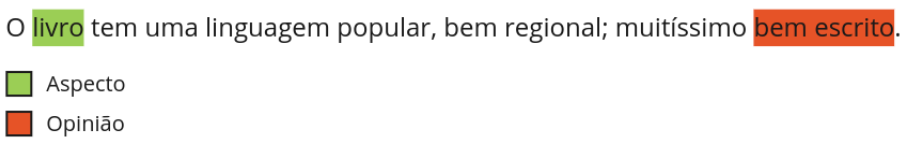
- *Numpy*: Uma biblioteca para computação numérica, fornecendo suporte para *arrays* multidimensionais e funções de alto nível.
- *Pandas*: Uma biblioteca para manipulação e análise de dados.
- *TensorFlow*: Uma biblioteca de aprendizado. Utilizada para construir, treinar e implementar redes neurais.
- *Torch*: Uma biblioteca de aprendizado. Utilizada para construir, treinar e implementar redes neurais.
- *Keras*: Uma interface de alto nível para *TensorFlow* que simplifica o processo de criação, treinamento e avaliação de modelos de aprendizado profundo.
- *Optuna*: Uma biblioteca para ajuste de hiperparâmetros de modelos de aprendizado de máquina.
- *Datasets*: Uma biblioteca para carregar e pré-processar conjuntos de dados para treinamento e avaliação de aprendizado de máquina.
- *Transformers*: Uma biblioteca para trabalhar com modelos de PLN baseados em *Transformers*, como BERT, GPT e entre outros. Oferece ferramentas para treinar, ajustar e implementar esses modelos em várias tarefas.
- *Selenium*: Uma biblioteca que permite automatizar a navegação em páginas na web, podendo extrair informações destas páginas.
- *BeautifulSoup*: Uma biblioteca para extrair informações de documentos HTML e XML.
- *NLTK*: Uma biblioteca para trabalhar com linguagem humana. Fornece ferramentas para processar e analisar texto, incluindo tokenização, remoção de *stopwords*, extração de entidades nomeadas e classificação de texto.

- PEFT: Uma bibliotecapara ajuste fino eficiente de parâmetros em modelos de linguagem pré-treinados, facilitando a adaptação a diferentes aplicações sem alterar todos os parâmetros do modelo. Integra técnicas como o LoRa.
- TRL: Oferece ferramentas para treinar modelos de linguagem com *Transformers* com Aprendizado por Reforço, cobrindo etapas como ajuste fino supervisionado, modelagem de recompensas e outras.

3.2 Conjunto de Dados

Neste trabalho, foram utilizados dois conjuntos de dados anotados em língua portuguesa. A coleção de dados conhecida como ReLi (Resenha de Livros) (FREITAS et al., 2012), que contém 12.429 sentenças distribuídas em 1.598 revisões de livros. Essas resenhas foram coletadas de leitores que as publicaram na web e, posteriormente, foram manualmente anotadas quanto à expressão de opinião, sendo que 18% das sentenças possuem aspectos. Um exemplo de sentença dessa coleção com aspecto marcado pode ser observado na Figura 3.1.

Figura 3.1 – Exemplo de sentença com aspecto

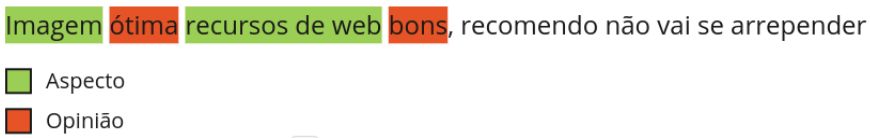


Fonte: Do autor (2023).

A segunda coleção é denominada TV (CARDOSO; PEREIRA, 2020) e possui 1.091 revisões de usuários que avaliaram um modelo de televisão em um site de comércio eletrônico. Os autores coletaram essas revisões e as anotaram manualmente, tendo constatado que 90,65% delas apresentam aspectos. A Figura 3.2 traz um exemplo de sentença com aspecto marcado para esta coleção.

Os conjuntos de dados foram cuidadosamente estratificados para possibilitar a utilização de validação cruzada com 10 partições (*folds*), mantendo o balanceamento original em cada uma das partições (particionamento estratificado). Este processo envolveu inicialmente a identificação da distribuição de classes no conjunto de dados completo, contando o número de instâncias de cada

Figura 3.2 – Exemplo de sentença com aspecto



Fonte: Do autor (2023).

classe. Em seguida, para cada *fold*, foi selecionada, aleatoriamente, uma amostra de tal forma que a proporção de aspectos nesse *fold* correspondesse, ou estivesse muito próxima, da proporção no conjunto de dados completo. Dessa forma, cada instância aparece em apenas um *fold*, garantindo que cada *fold* mantenha a distribuição do conjunto de dados original. Essa estratégia é fundamental para garantir que os resultados obtidos sejam confiáveis e generalizáveis. A estratificação preserva a distribuição de aspectos originais dos conjuntos durante o processo de validação, mitigando o risco de viés devido ao desequilíbrio de classes em algum *fold* específico. Os conjuntos de dados particionados estão disponíveis em https://github.com/jcfneto/aspect-extraction.

Tabela 3.1 – Repartição da base de dados ReLi

Fold	Total de Resenhas	Total de Aspectos
1	158	280
2	168	281
3	154	261
4	164	267
5	171	281
6	159	281
7	155	270
8	152	286
9	155	280
10	162	281
Total	1598	2768

Fonte: Do autor (2023).

As Tabelas 3.1 e 3.2 apresentam o resultado da estratificação para os conjuntos ReLi e TV.

Tabela 3.2 – Repartição da base de dados TV

Fold	Total de Revisões	Total de Aspectos
1	109	234
2	110	230
3	109	236
4	109	234
5	109	235
6	109	235
7	109	226
8	109	234
9	109	229
10	109	234
Total	1091	2327

Fonte: Do autor (2023).

3.3 *Baseline*

Uma estratégia relevante para avaliar a eficácia das técnicas empregadas neste trabalho é a criação de um modelo de base (*baseline*), com o objetivo de comparar o desempenho dos modelos construídos com o desse modelo de referência.

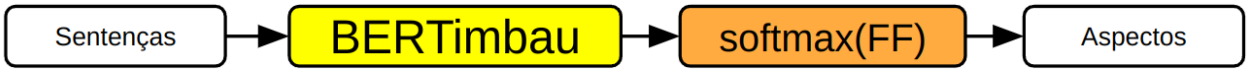
O modelo de base, ilustrado na Figura 3.3, foi construído por meio do ajuste fino de um modelo BERT pré-treinado em domínio geral. "Domínio geral" refere-se ao treinamento do modelo em um amplo espectro de textos, abrangendo diversos temas e estilos, sem se concentrar em um tópico ou gênero específico. A esse modelo foi adicionada uma camada *feed-forward* (FF) com ativação *softmax* para classificação. O modelo pré-treinado utilizado foi o $BERTimbau_{BASE}$ (SOUZA; NOGUEIRA; LOTUFO, 2020), treinado com um corpus de 2.68 bilhões de *tokens* em português. Este modelo contém 110M de parâmetros e pode processar até 512 *tokens* de entrada, gerando *embeddings* com dimensão igual a 768. Antes do ajuste fino, realizou-se uma otimização de hiperparâmetros bayesiana para aprimorar a configuração inicial do modelo, que consiste em selecionar os valores de hiperparâmetros mais eficazes com base em resultados anteriores (CHOLLET, 2021). Os hiperparâmetros otimizados para o *baseline* incluíram a taxa de *dropout* e a taxa de aprendizagem, conforme detalhado na Tabela 3.3.

Tabela 3.3 – Espaço de Busca dos Hiperparâmetros para o *Baseline*

Hiperparâmetro	Espaço de Busca
Taxa de *dropout*	0.1:0.1:0.5
η	$[2 \times 10^{-5}, 1 \times 10^{-3}]$

Fonte: Do autor (2023).

Figura 3.3 – Arquitetura do *Baseline*



A Figura ilustra o modelo de referência (*baseline*) onde as sentenças são analisadas pelo BERTimbau, seguidas por uma camada *softmax* que calcula probabilidades de cada *token* ser ou não um aspecto.
Fonte: Do autor (2023).

3.4 *Embedding* Duplo

Identificar aspectos específicos é uma atividade difícil, que pode demandar representações vetoriais detalhadas do domínio em questão. Por exemplo, ao analisar determinadas características, pode ser necessário empregar diferentes representações para termos gerais e específicos, devido à variedade de contextos que um mesmo termo pode assumir. Termos gerais podem ser adequadamente representados por representações gerais, enquanto termos específicos de um domínio possuem um sentido mais exato, o que torna as representações dentro do domínio essenciais (XU et al., 2018).

Sendo assim, em Xu et al. (2018) é proposto um modelo simples utilizando CNN para extração de aspectos. O modelo utiliza dois tipos de *embeddings* pré-treinados, um de propósito geral, que foi o GloVe (PENNINGTON; SOCHER; MANNING, 2014), e outro específico do domínio do problema, usando *fastText*. Isto é, tanto representações gerais quanto as de domínio são aplicadas, permitindo que a rede decida quais representações trazem informações de maior valor para a a extração do aspecto.

Neste trabalho, inspirado em Xu et al. (2018), foi proposta uma abordagem similar: utilizar incorporação dupla (*double embedding*), um de domínio geral e outro de domínio específico. Em

contraste com a utilização do GloVe e do *fastText*, optou-se por empregar as incorporações pré-treinadas com base na arquitetura BERT.

O *embedding* de domínio geral escolhido foi o *BERTimbau*, treinado com dados extraídos da Wikipédia em português, que proporciona uma ampla gama de informações sobre diversos temas e contextos. Para os *embeddings* de domínio específico, optou-se por treinar um modelo BERT utilizando dados coletados de sites especializados em resenhas literárias, para corresponder ao domínio do conjunto de dados ReLi, e dados de revisões de TV em sites de *e-commerce*, para corresponder o domínio do conjunto de dados TV. Espera-se que, através da combinação dessas duas incorporações baseadas na arquitetura BERT, tanto para o domínio geral quanto para o domínio específico, seja possível uma análise mais profunda e precisa dos aspectos relevantes em cada contexto.

3.5 Pré-treino do BERT de Domínio Específico

Para o pré-treino do BERT de domínio específico, foi necessário extrair os dados para cada um dos domínios, definir a arquitetura a ser utilizada e fazer a otimização dos hiperparâmetros. As seções a seguir descrevem cada uma destas etapas.

3.5.1 Corpus para Pré-treino

O corpus utilizado para o pré-treino no domínio de resenhas literárias foi cuidadosamente compilado a partir dos websites Minha Vida Literária[1] e Leitor Compulsivo[2]. Esse corpus possui ao todo aproximadamente 1.27 milhões de tokens. Uma característica distintiva deste corpus é a sua natureza mais profissional, em contraste com o conjunto ReLi, que compreende resenhas escritas por um público geral.

[1] https://www.minhavidaliteraria.com.br/
[2] https://www.leitorcompulsivo.com.br/

Para o domínio das revisões do aparelho TV, as revisões foram coletadas a partir dos websites do Mercado Livre[3] e da Amazon[4]. Ao todo, esse corpus possui 1.46 milhões de *tokens*.

3.5.2 Arquitetura do Modelo

A arquitetura utilizada segue o modelo padrão do BERT, com hiperparâmetros ajustados por meio de otimização bayesiana, coforme descrito na seção subsequente. Portanto, a arquitetura utilizada consiste em múltiplas camadas de codificadores (*encoders*) empilhados, e cada camada é formada por duas subcamadas: a subcamada de atenção multi-cabeça e a subcamada de *feed-forward*. Essas camadas de codificadores são responsáveis por extrair e processar informações contextuais de sequências de entrada.

Além disso, a saída das camadas codificadoras alimentam dois módulos de previsão, que são baseados em duas estratégias de treinamento: *Masked Language Modeling* (MLM) e *Next Sentence Prediction* (NSP). Esses módulos são adicionados como cabeças de previsão na arquitetura do BERT, atuando sobre a saída das camadas de codificadores. O módulo MLM tem como objetivo prever *tokens* mascarados com base no contexto bidirecional, enquanto o módulo NSP visa prever a relação entre duas sentenças. A arquitetura segue a mesma apresentada na Figura 2.5 em Pré-treino.

3.5.3 Otimização dos Hiperparâmetros

Neste estudo, a busca pelos hiperparâmetros para o BERT foi automatizada utilizando a técnica de otimização bayesiana. Para isso, separou-es 10% do corpus de pré-treinamento para realizar essa busca (esta porção não será utilizada para o pré-treinamento) e definiu-se o espaço de busca.

Em Devlin et al. (2018), os autores empregaram as duas variantes do modelo BERT que foram apresentadas no trabalho, denominadas BERT_{BASE} ($L = 12$) e BERT_{LARGE} ($L = 24$), para realizar, entre outras, a tarefa de Reconhecimento de Entidades Nomeadas (NER). Duas abordagens distintas foram adotadas: a primeira envolveu o ajuste fino do modelo, enquanto a segunda

[3] https://www.mercadolivre.com.br/
[4] https://www.amazon.com.br/

consistiu em extrair as representações aprendidas durante o pré-treinamento e utilizá-las como atributos para uma arquitetura adicional responsável pela classificação. Ambas as variantes do BERT (*BASE* e *LARGE*) foram empregadas na primeira abordagem, enquanto apenas a variante BASE foi utilizada na segunda. Os resultados obtidos demonstraram que a segunda abordagem apresentou um desempenho altamente competitivo em comparação à primeira, mesmo ao empregar uma arquitetura menor. Já em Vaswani et al. (2017), a maior arquitetura testada foi com $L = 6$. Tendo isto em vista, explorou-se no espaço de busca para este hiperparâmetro valores entre 6 e 12, com incremento de 2.

Para testar qual o efeito ao se variar o tamanho da arquitetura BERT, Devlin et al. (2018) avaliaram, entre outros hiperparâmetros, o número de cabeças do *multi-head attention* entre 3, 12 e 16. Inspirado nisso, utilizou-se essa mesma faixa para enriquecer o espaço de busca dos hiperparâmetros.

Na arquitetura original do BERT, a subcamada *feed-forward* contém 4 camadas de dimensões idênticas. No caso do BERT_{BASE}, cada camada possui uma dimensão oculta de 768, enquanto no BERT_{LARGE}, a dimensão é de 1.024. Como resultado, a dimensão do *embedding* é igual à saída dessa subcamada. Neste estudo, foi investigado o valor ótimo para este hiperparâmetro, considerando uma faixa de busca que inclui 256, 512 e 768. Além disso, explorou-se a possibilidade de que a última camada, ou seja, a dimensão do *embedding*, possa ser diferente das três camadas anteriores.

A taxa de aprendizado empregada na implementação original do BERT foi de 1×10^{-4}. Neste estudo, realizou-se uma busca dentro de um intervalo que engloba esse valor, variando entre 2×10^{-5} e 1×10^{-3}. A estratégia de amostragem empregada foi logarítmica, assegurando uma distribuição uniforme dos valores no espaço logarítmico entre o limite inferior e o limite superior do intervalo.

Finalmente, o último hiperparâmetro a ser otimizado é a taxa de *dropout*. O espaço de busca para este hiperparâmetro abrange valores no intervalo de 0 a 0.5, com um incremento de 0.1.

Todos os hiperparâmetros mencionados foram otimizados utilizando tamanhos de lote B de 8, 16 e 32. Embora o ideal fosse também avaliar tamanhos de lote de 64 e 128, isso não foi realizado devido às limitações de recursos de *hardware* disponíveis. A Tabela 3.4 apresenta

os hiperparâmetros e os respectivos intervalos de busca empregados, em que L é o número de camadas, A é o número de cabeças do *multi-head attention*, H_{FF} é a dimensão de cada uma das três primeiras camadas da sub-camada *feed-foward*, H_{EMBED} é a dimensão da última camada da sub-camada *feed-foward*, isto é, do *embedding*, B é o tamanho do lote e η é a taxa de aprendizado.

Tabela 3.4 – Espaço de Busca dos Hiperparâmetros

Hiperparâmetro	Espaço de Busca
L	6:2:12
A	4, 12 e 16
H_{FF}	1024:1024:3072
H_{EMBED}	256:256:718
Taxa de *dropout*	0.0:0.1:0.5
B	8:8:32
η	$[2 \times 10^{-5}, 1 \times 10^{-3}]$

Fonte: Do autor (2023).

3.5.4 Estratégia do Pré-treino

O treinamento do modelo foi realizado utilizando o método MLM e NSP. No MLM será empregado uma probabilidade de 15% de mascaramento dos *tokens* no conjunto de dados. Durante o processo de mascaramento:

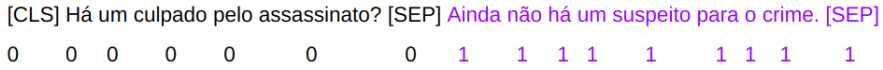
- 80% dos *tokens* selecionados foram substituídos pelo *tokens* especial [MASK], permitindo que o modelo aprenda a prever o *tokens* oculto. Exemplo: Há um culpado pelo assassinato? $\rightarrow$ Há um [MASK] pelo assassinato?
- 10% dos *tokens* foram substituídos por um *tokens* aleatório, introduzindo ruído. Exemplo: Há um culpado pelo assassinato? $\rightarrow$ Há um maçã pelo assassinato?
- Os 10% restantes dos *tokens* selecionados permanecerão inalterados. Exemplo: Há um culpado pelo assassinato? $\rightarrow$ Há um culpado pelo assassinato?

Na tarefa de NSP, pares de sentenças são extraídos dos dados de treinamento. Cada par consiste em duas sentenças: sentença A e sentença B. Em metade dos casos (50%), a sentença B é de fato a sentença subsequente à sentença A no texto original. Na outra metade (50%), a

sentença B é escolhida aleatoriamente de uma parte distinta do texto. As sentenças A e B são então concatenadas para criar uma sequência de entrada única. *Tokens* especiais são adicionados no início ([CLS]) e entre as sentenças ([SEP]), resultando na estrutura: [CLS] sentença A [SEP] sentença B [SEP].

Para permitir que o modelo distingua as duas sentenças, uma representação de segmento é incorporada na entrada. Essa representação associa a sentença A ao segmento 0 e a sentença B ao segmento 1, conforme ilustrado na Figura 3.4. Dessa forma, o modelo pode diferenciar e processar adequadamente o contexto de cada sentença durante o treinamento.

Figura 3.4 – Exemplo de Representação de Segmento



Fonte: Do autor (2023).

No pré-treino será utilizado o otimizador AdamW, uma variante do otimizador Adam, que incorpora decaimento de peso como técnica de regularização. Com taxa de aprendizado proveniente da otimização dos hiperparâmetros. Com decaimento de 0,004, $\beta_1 = 0.9$ e $\beta_2 = 0.999$.

O pré-treinamento do modelo será executado por um máximo de 10 épocas, com a adoção de uma estratégia de parada antecipada (*early stopping*). A definição de um máximo número de épocas tem o objetivo de limitar o tempo de execução por questões de limitação do uso dos recursos computacionais disponíveis. Caso o critério de parada não interrompa o treinamento por falta de melhoria, nesse caso, o processo será finalizado ao alcançar o limite de épocas estabelecido. Neste trabalho, a parada antecipada será configurada com um delta de 10%, indicando que o treinamento será interrompido se a métrica de validação não melhorar em pelo menos 10% em relação ao melhor valor observado anteriormente. Além disso, será estabelecida uma paciência de 3 épocas, o que significa que o treinamento continuará por até 3 épocas adicionais após a detecção de estagnação no desempenho, a fim de verificar se ocorre alguma melhoria adicional antes de interromper o processo definitivamente.

3.6 Ajuste Fino

Nesta seção, são discutidas as estratégias implementadas para refinar os modelos de linguagem utilizados neste trabalho. Dentre as abordagens de ajuste-fino empregadas, explorou-se o uso do *embedding* duplo com BERT de domínio geral e de domínio específico. Adicionalmente, investigou-se a utilização do GPT por meio da API da OpenAI.

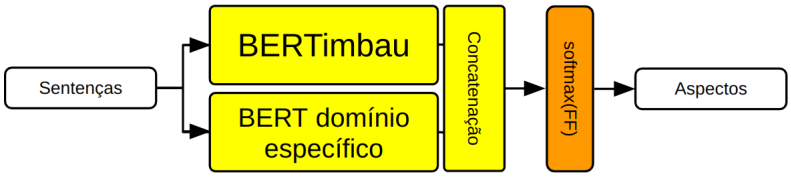
Ainda nesta seção é descrita a aplicação do LoRa ao BERT, uma estratégia que procura reduzir a complexidade do modelo enquanto mantém, ou até mesmo melhora, seu desempenho. Finalmente, introduz-se o QLoRa aplicado ao *openCabrita*, uma inovação que incorpora quantização no processo de ajuste-fino, promovendo uma otimização mais eficaz dos parâmetros do modelo.

Cada uma dessas abordagens apresenta características únicas que podem ser exploradas para melhorar a eficácia dos modelos de linguagem em tarefas específicas. A análise detalhada das metodologias de ajuste-fino, bem como a discussão sobre sua implementação e resultados alcançados (apresentado no capítulo seguinte), fornecerá *insights* valiosos sobre a capacidade de adaptação e otimização dos modelos selecionados para o problema de extração de aspectos.

3.6.1 *Embedding* Duplo

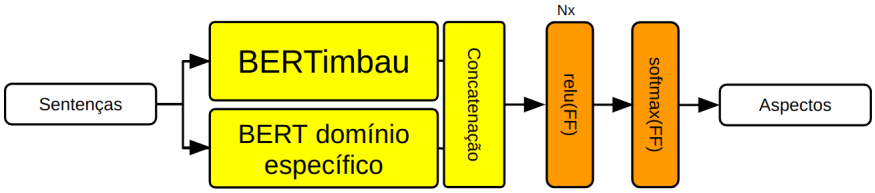
Para entender a melhor forma de aproveitar a representação combinada dos modelos pré-treinados de BERT de domínio geral e específico, optou-se por concatenar a saída de cada um deles e alimentar camadas subsequentes que serão responsáveis por definir a melhor maneira de combinar as representações dos dois domínios. A abordagem fundamenta-se na premissa de que a combinação de representações semânticas fornecidas por ambos os modelos pode fornecer uma compreensão mais completa dos dados e, portanto, aprimorar a extração de aspectos. A fim de explorar essa hipótese, foram testadas duas arquiteturas diferentes, uma apenas concatenando a saída dos dois modelos alimentando uma camada para classificação (Figura 3.5), esse modelo chamaremos de *concat*, e outra adicionando algumas camadas *feed-forward* antes da camada de classificação (Figura 3.6) que chamaremos de *concatFFN*.

Figura 3.5 – Arquitetura do *concat*



A Figura retrata o modelo onde os *embeddings* se fundem, mantendo constante o tamanho da sequência, mas expandindo a dimensão de *features* para a próxima camada de processamento, que trata-se de uma camada *feed-forward* com ativação *softmax*.
Fonte: Do autor (2023).

Figura 3.6 – Arquitetura do *concatFFN*



A Figura retrata o modelo onde os *embeddings* se fundem, mantendo constante o tamanho da sequência, mas expandindo a dimensão de *features* para as próximas *Nx* camadas de processamento e por fim é passado para a camada *feed-forward* com ativação *softmax*.
Fonte: Do autor (2023).

As camadas *feed-forward* são uma das arquiteturas mais simples para treinamento de redes neurais. Elas são compostas por múltiplas camadas de neurônios que se conectam diretamente aos neurônios da camada seguinte, formando uma estrutura sem ciclos ou conexões de retorno. Isso significa que os dados fluem da entrada para a saída sem passar por nenhuma camada mais de uma vez. A principal vantagem das camadas *feed-forward* é a sua simplicidade e eficiência computacional, sendo que, no contexto de extração de aspectos, elas podem ser úteis para capturar relacionamentos lineares entre os diferentes aspectos e o contexto no qual eles aparecem. Por conta disso, avaliaremos o uso dessas camadas após a concatenação dos *embeddings*.

Foi realizada uma busca exaustiva no espaço de hiperparâmetros usando a otimização Bayesiana (os espaços de busca são apresentado nas Tabelas 3.5 e 3.6). Isso permitiu encontrar a combinação de hiperparâmetros que maximizava F1 no conjunto de validação. Uma vez que os melhores hiperparâmetros foram encontrados, prosseguiu-se com o treinamento completo dos modelos utilizando esses hiperparâmetros.

Tabela 3.5 – Espaço de Busca dos Hiperparâmetros para o *concat*

Hiperparâmetro	Espaço de Busca
Taxa de *dropout*	0.0:0.1:0.5
η	$[2 \times 10^{-5}, 1 \times 10^{-3}]$

Fonte: Do autor (2023).

Tabela 3.6 – Espaço de Busca dos Hiperparâmetros para o *concatFFN*

Hiperparâmetro	Espaço de Busca
Taxa de *dropout*	0.0:0.1:0.5
η	$[2 \times 10^{-5}, 1 \times 10^{-3}]$
Unidades na primeira camada N_1	256, 512 e 768
Unidades na segunda camada N_2	0, 256, 512 e 768
Unidades na terceira camada N_3	0, 256, 512 e 768

Fonte: Do autor (2023).

Para a camada de classificação, optou-se pela função de ativação *softmax*, que é amplamente empregada em problemas de classificação multiclasse devido à sua capacidade de fornecer uma distribuição de probabilidade normalizada sobre as classes alvo. Esta função facilita a interpretação dos resultados, proporcionando não apenas a classificação, mas também a confiança associada às predições.

Por outro lado, nas camadas *feed-foward*, a função de ativação *relu* foi a escolhida. A função *relu* é conhecida por sua eficiência computacional e capacidade de acelerar a convergência durante o treinamento, ao mitigar o problema do desaparecimento do gradiente. Além disso, a *relu* introduz a não-linearidade necessária que permite à rede aprender a partir dos dados, mantendo a complexidade computacional sob controle.

Em relação ao tamanho do lote, foi estabelecido um valor fixo de 8, uma escolha guiada por limitações computacionais. Esta configuração representa um compromisso entre a eficiência do treinamento e a utilização eficaz dos recursos computacionais disponíveis. Lotes maiores poderiam, potencialmente, proporcionar estimativas de gradiente mais estáveis e uma convergência mais suave durante o treinamento. No entanto, no contexto deste trabalho, não foi possível adotar

lotes maiores devido às restrições de recursos computacionais, mais especificamente limitações de memória. Tentativas para aumentar o tamanho do lote para valores superiores a 8 mediante aos recursos disponíveis gerariam quebras de memória que impediriam o progresso do treinamento. Portanto, o tamanho do lote foi meticulosamente definido em 8 para garantir um treinamento contínuo e eficiente dentro das capacidades computacionais disponíveis.

A quantidade de unidades nas camadas *feed-forward* também foi meticulosamente calibrada, com o propósito principal de adequar-se às restrições computacionais enfrentadas. O número de unidades nessas camadas foi limitado devido às limitações de memória, similares às enfrentadas com o tamanho do lote. Expandir a quantidade de unidades resultaria em quebras de memória, o que impossibilitaria a exploração de configurações com maior número de unidades. Portanto, o espaço de busca adotada foi cuidadosamente escolhido para assegurar que o modelo pudesse ser treinado e avaliado de maneira eficaz dentro das capacidades computacionais disponíveis, evitando assim interrupções no treinamento devido a quebras de memória.

3.6.2 GPT

A OpenAI[5], uma organização no campo da inteligência artificial que é a desenvolvedora do GPT, oferece uma API que possibilita o ajuste fino de vários modelos, entre eles o próprio GPT. Entre as variantes oferecidas para o ajuste fino do GPT-3, foram utilizadas três variantes específicas: Ada, Babbage e Curie. Estes modelos diferem em vários aspectos, porém, a única informação disponibilizada pela empresa é a capacidade máxima de tokens, que é de 2.049 para todas as variantes.

Quanto à estrutura econômica associada ao uso dessas variantes, o custo para o ajuste fino é estabelecido em aproximadamente $0,0060 por mil *tokens* processados. Adicionalmente, para operações de inferência — que compreendem tanto o *input* quanto o *output* de dados — o custo está fixado em $0,0016 para cada mil *tokens*.

Dado que o modelo é comercializado exclusivamente através de uma API, não há possibilidade de modificar sua arquitetura intrínseca. Consequentemente, os dados devem ser submetidos

[5] https://openai.com/

em um formato prescrito pela documentação oficial da OpenAI, o que constitui uma distinção significativa em relação aos métodos adotados em outras abordagens deste estudo. É apresentado um exemplo do formato requerido para o processo de ajuste fino nesta metodologia na Figura 3.7.

Figura 3.7 – Exemplo do Formato dos Dados para GPT

```
{
  "prompt": "Excelente smart tv. E foi entregue vem antes do prazo informado
  ↪  . ->",
  "completion": "smart tv."
}
{
  "prompt": "Gosto muito dos produtos de vcs sempre entrega com antecedência
  ↪  e ótimo ->",
  "completion": "produtos\entrega."
}
{
  "prompt": "PRODUTO DE ÓTIMA QUALIDADE, MUITO SATISFEITO PELA COMPRA DO
  ↪  PRODUTO. ->",
  "completion": "produto."
}
```

Fonte: Do autor (2023).

Os modelos conversacionais, como é o caso das variantes do GPT utilizados, são projetados e otimizados para interagir de forma fluente e natural com os usuários respondendo a perguntas, fornecendo informações, ou até mesmo auxiliando em tarefas mais complexas. O ajuste fino destes modelos é realizado por meio de uma técnica que utiliza "*prompts*", que são essencialmente instruções dadas ao modelo na forma de exemplos de entrada e saída (como mostra a Figura 3.7). Este método permite que os modelos aprendam a produzir respostas específicas ou a realizar tarefas especializadas com base em exemplos fornecidos. O processo capitaliza a capacidade inerente dos modelos de generalizar a partir de exemplos, permitindo que eles se adaptem a domínios específicos ou tarefas particulares com eficácia.

É relevante destacar que os hiperparâmetros empregados no processo de ajuste fino foram os valores padrão fornecidos pela própria OpenAI. Esta decisão foi estratégica e fundamentada em considerações econômicas, uma vez que a identificação de um conjunto ótimo de hiperparâmetros

requereria uma série de experimentações iterativas, acarretando, assim, um custo computacional e financeiro considerável. Dado o modelo de precificação da API, que cobra tanto pelo ajuste fino quanto pelas operações de inferência, a opção por utilizar os parâmetros padrão demonstrou ser uma abordagem eficiente para a contenção de despesas.

3.6.3 LoRA com BERT

Foi empregado também o ajuste fino usando o LoRA com o BERT de domínio geral (*BERTimbau*), porém, semelhante aos demais, foi realizado uma otimização de hiperparâmetros usando otimização Bayesiana. A Tabela 3.7 apresenta o espaço de busca para a otimização, onde *r* especifica a dimensão da atenção em LoRa, esse hiperparâmetro refere-se à dimensão do espaço de baixa classificação que LoRa opera, reduzindo o número total de parâmetros que precisam ser treinados durante o ajuste fino. O α é o parâmetro de escala para o LoRa, este pode influenciar o impacto dos adaptadores de baixo *rank* nas camadas alvo do modelo. O viés determina quais vieses são atualizados durante o treinamento e podem impactar o comportamento do modelo mesmo quando os adaptadores são desativados. O decaimento *W* é um hiperparâmetro de regularização usado para prevenir *overfitting*, aplicando uma penalidade aos pesos do modelo durante o treinamento, incentivando-os a terem valores menores.

Tabela 3.7 – Espaço de Busca dos Hiperparâmetros para LoRa com *BERTimbau*

Hiperparâmetro	Espaço de Busca
r	8:8:32
α	8:8:32
Viés	*none*, *all*, *lora_only*
LoRa *dropout*	0.0:0.1:0.5
W_{DECAY}	[0, 0.1]
η	$[2 \times 10^{-5}, 1 \times 10^{-3}]$

Fonte: Do autor (2023).

Após a fase de otimização, determinou-se as configurações ideais de cada hiperparâmetro que otimizam o F1 no conjunto de validação. Essas configurações foram subsequentemente empregadas para o ajuste fino do modelo com LoRA.

3.6.4 QLoRa com Cabrita

Em contraste com outras metodologias empregadas, a etapa de otimização de hiperparâmetros para definir as configurações ideais do *openCabrita* em conjunto com QLoRa foi preterida, dada a intensiva demanda por recursos computacionais associada ao carregamento deste modelo específico. O procedimento em questão necessita de uma GPU de elevado desempenho e, consequentemente, de maior custo. Diante desta limitação, optou-se por empregar as seguintes configurações predefinidas: $B = 8$ para equilibrar eficiência computacional e generalização, 3 épocas para assegurar treinamento adequado sem risco de sobreajuste, e taxa de aprendizado $\eta = 1.5e-4$. Para o LoRa, r e α foram definidos como 13, com uma taxa de *dropout* de 0.2, visando captar padrões mais sutis nos dados, ao mesmo tempo que se previne o sobreajuste com uma regularização eficaz.

No intuito de capitalizar a facilidade de utilização proporcionada pelas implementações das bibliotecas do Huggingface, ou seja, evitando a necessidade de trabalho adicional além da declaração do modelo e das configurações do LoRa, foi essencial processar os dados de uma maneira distinta das abordagens anteriores, analogamente a fornecer um *prompt* para um modelo generativo conversacional, conforme Figura 3.8.

3.7 Método de Avaliação: *Hold-out*

Para avaliar o desempenho das arquiteturas descritas na seção anterior, foi utilizado o método de validação cruzada *hold-out*, que inclui uma divisão tríplice dos dados: treinamento, validação e teste. Nesta configuração, o conjunto de treinamento é utilizado para o aprendizado dos modelos, o conjunto de validação é usado para a otimização dos hiperparâmetros e a avaliação do desempenho do modelo durante o treinamento, e o conjunto de teste é usado para fornecer uma avaliação final e imparcial do desempenho do modelo (RASCHKA, 2018).

Neste caso, a divisão ficou da seguinte forma: as oito primeiras partições foram separadas para treino, a nona partição foi utilizada para teste e a última para validação. Dessa forma, é possível garantir que a avaliação final do modelo, no conjunto de teste, seja uma representação justa de como o modelo irá se comportar em dados nunca vistos.

Figura 3.8 – Exemplo do Formato dos Dados para Treino com Cabrita

```
    {
        "instruction": "Qual o(s) aspecto(S) explícito(s) da frase a
        ↪ seguir?",
        "input": "Excelente smart tv. E foi entregue vem antes do prazo
        ↪ informado.",
        "output": "smart tv."
    }
    {
        "instruction": "Qual o(s) aspecto(S) explícito(s) da frase a
        ↪ seguir?",
        "input": "Gosto muito dos produtos de vcs sempre entrega com
        ↪ antecedência e ótimo",
        "output": "produtos\nentrega."
    }
    {
        "instruction": "Qual o(s) aspecto(S) explícito(s) da frase a
        ↪ seguir?",
        "input": "PRODUTO DE ÓTIMA QUALIDADE , MUITO SATISFEITO PELA COMPRA
        ↪ DO PRODUTO.",
        "output": "PRODUTO."
    }
```

Fonte: Do autor (2023).

Embora a validação cruzada possa fornecer uma estimativa mais robusta do desempenho do modelo, ela também é computacionalmente mais intensiva, pois requer o treinamento e a avaliação do modelo várias vezes. Dada a natureza computacionalmente intensiva do treinamento das arquiteturas propostas e da otimização de hiperparâmetros, optou-se pela abordagem de *hold-out* com conjunto de validação para manter o processo gerenciável do ponto de vista de tempo e recursos computacionais (RASCHKA, 2018).

No entanto, é importante reconhecer que esta abordagem tem suas limitações. A maior delas é que a avaliação do desempenho do modelo pode ser sensível à maneira como os dados são divididos entre treinamento, validação e teste. Para mitigar esse risco, tomou-se cuidado para garantir que todas as três divisões sejam representativas do conjunto de dados como um todo, assim como apresentado na Seção 3.2.

3.8 Métrica de Avaliação: F1 *score*

Neste trabalho, optou-se por utilizar a medida F1 *score* como principal métrica de avaliação.

O F1 *score* é uma medida harmônica da precisão e da revocação, que fornece um equilíbrio entre essas duas métricas fundamentais. A precisão é a proporção de identificações positivas feitas corretamente, isto é, verdadeiros positivos (TP) divididos pela soma de verdadeiros positivos e falsos positivos (FP), enquanto a revocação é a proporção de positivos reais que foram identificados corretamente, ou seja, verdadeiros positivos divididos pela soma de verdadeiros positivos e falsos negativos (FN) (DERCZYNSKI, 2016). Conforme as equações a seguir.

$$precision = \frac{TP}{TP+FP} \tag{3.1}$$

$$recall = \frac{TP}{TP+FN} \tag{3.2}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.3}$$

Na extração de aspectos é importante não apenas identificar corretamente os aspectos (precisão), mas também garantir que todos os aspectos relevantes sejam identificados (revocação). O F1 *score* combina essas duas métricas em uma única medida, permitindo avaliar simultaneamente a capacidade do modelo de evitar falsos positivos e falsos negativos.

4 RESULTADO

Neste capítulo são apresentados os resultados obtidos a partir da metodologia aplicada conforme descrito no Capítulo 3.

4.1 Pré-Treino com BERT

Os resultados nas Tabelas 4.1 e 4.2 evidenciam as cinco configurações ótimas dos hiperparâmetros para os BERTs de domínio específico resultantes da otimizaçao bayesiana para o espaço de busca da Tabela 3.4. Ao todo, foram 120 configurações diferentes testadas, 60 para cada conjunto de dados. Para ambos os conjuntos, a primeira configuração de cada tabela foi selecionada para realizar o pré-treino, baseando-se nos resultados favoráveis obtidos.

Tabela 4.1 – BERT Domínio Específico - Reviews de TV

H_{EMBED}	L	A	H_{FF}	*dropout*	η	epochs	loss	B
768	**6**	**12**	**1024**	**0.2**	**1.02e-4**	**9**	**3.493**	**32**
768	6	16	1024	0.4	9.3e-5	8	5.047	32
512	10	4	2048	0.1	4.0e-5	7	5.071	8
512	10	16	3072	0.1	4.7e-5	7	5.132	8
768	10	12	2048	0.3	3.8e-5	7	5.412	16

Fonte: Do autor (2023).

Tabela 4.2 – BERT Domínio Específico - Resenhas Literárias

H_{EMBED}	L	A	H_{FF}	*dropout*	η	epochs	loss	B
768	**6**	**4**	**1024**	**0.1**	**6.7e-5**	**5**	**6.373**	**8**
512	8	16	2048	0.2	1.28e-4	5	6.497	8
512	10	16	2048	0.1	7.5e-5	5	6.512	8
768	6	4	3072	0.1	1.24e-4	5	6.532	16
768	8	12	2048	0.2	4.8e-5	5	6.532	8

Fonte: Do autor (2023).

Os modelos resultantes foram hospedados na plataforma do Hugginface, tanto o BERT de domínio resenhas literárias[1] quanto o de reviews de TV[2].

Os tópicos a seguir detalham os achados relevantes identificados durante a exploração do espaço de busca para otimizar os hiperparâmetros do pré-treino:

- **Tamanho de Embedding (H_{EMBED})**: Observa-se que o tamanho de 768 frequentemente apresentou resultados melhores. Isso sugere que um tamanho de embedding mais amplo pode capturar representações mais ricas para tarefas específicas do domínio.
- **Camadas (L)**: Uma profundidade de 6 camadas foi comum para o conjunto TV, enquanto uma profundidade de 10 camadas foi frequentemente observada para o conjunto ReLi. Essa diferença pode ser atribuída às características inerentes de cada conjunto.
- **Atenções (A)**: O número de cabeças de atenção varia entre as configurações. No entanto, não é evidente se um número maior ou menor é preferível. Isso pode indicar que a eficácia do número de cabeças de atenção pode depender de outros hiperparâmetros.
- **Taxa de *Dropout***: Valores mais altos de dropout (0.2 e 0.4) foram observados para o conjunto TV, possivelmente indicando que o modelo requer regularização mais forte para evitar o *overfitting* neste conjunto.
- **Taxa de Aprendizado (η)**: As taxas de aprendizado mostram uma variação significativa entre os conjuntos, destacando a sensibilidade deste hiperparâmetro. Mais investigações podem ser necessárias para determinar a taxa ótima para conjuntos diferentes.

Os resultados apresentados nas Tabelas 4.3 e 4.4 fornecem informações sobre o desempenho do modelo em termos de *loss*, considerando diferentes tamanhos de *batch*.

Tabela 4.3 – Estatísticas pelo Tamanho do Lote - Reviews de TV

B	*Loss* Média	Desvio Padrão	Mediana
8	6.225	0.649	6.594
16	6.326	0.612	6.774
32	**5.852**	**0.902**	**5.849**

Fonte: Do autor (2023).

[1] https://huggingface.co/jcfneto/bert-br-portuguese
[2] https://huggingface.co/jcfneto/bert-tv-portuguese

Tabela 4.4 – Estatísticas pelo Tamanho do Lote - Resenhas Literárias

B	*Loss* Média	Desvio Padrão	Mediana
8	**6.864**	**0.253**	**6.936**
16	6.985	0.205	7.098
32	7.015	0.121	7.053

Fonte: Do autor (2023).

A relação entre o tamanho do *batch* e o *loss* não é linear ou consistente entre as duas tabelas. Enquanto um padrão é observado na primeira tabela, a segunda mostra uma tendência distinta.

A variabilidade (desvio padrão) nos resultados também varia de acordo com o tamanho do *batch*. Isso sugere que diferentes tamanhos de *batch* podem influenciar não apenas a performance média do modelo, mas também sua consistência.

Dada a inconsistência observada entre as duas tabelas, seria valioso investigar mais profundamente as diferenças entre os conjuntos de dados ou as condições experimentais que podem ter causado tais discrepâncias.

Para aplicações práticas, pode ser aconselhável considerar o equilíbrio entre a eficiência computacional (maiores *batches* tendem a ser mais eficientes em termos de tempo por época) e a qualidade do modelo (indicada pelo *loss*) ao escolher um tamanho de *batch*.

4.2 Ajuste Fino

Nesta seção, discorre-se sobre os resultados alcançados mediante a calibração refinada do modelo *baseline*, o qual incorpora representações de domínio geral (treinamento em uma ampla gama de tópicos e estilos textuais). Analisa-se, ademais, a eficácia de representações específicas de domínio, a implementação da estratégia LoRa em modelos de domínio geral, bem como a combinação de domínios gerais e específicos. Explora-se também o emprego de técnicas avançadas, tais como a utilização do modelo GPT e a aplicação do *openCabrita* acoplado ao QLoRa, visando viabilizar o treinamento em unidades de processamento gráfico (GPUs) de menor exigência computacional.

4.2.1 *Baseline*

As Tabelas 4.5 e 4.6 detalham os resultados da otimização de hiperparâmetros para o ajuste fino do *baseline* baseado no BERT de domínio geral (BERTimbau) nas tarefas de extração de aspectos para os conjuntos de validação. Ao todo foram 20 diferentes configurações testadas, 10 para cada conjunto de dados.

A taxa de *dropout* entre 0.2 e 0.4, juntamente com taxas de aprendizado mais baixas (aproximadamente na faixa de 10^{-5}), demonstrou melhores resultados de F1 para o conjunto de dados de TV. Os melhores resultados para este modelo foram obtidos com uma taxa de *dropout* de 0.3 e uma taxa de aprendizado de 4.4×10^{-5}, alcançando um F1 de 0.824. Taxas de *dropout* mais baixas (0.1), quando combinadas com taxas de aprendizado específicas, resultaram em um F1 de zero. Isso pode indicar que, nesses cenários, o modelo não consegue aprender efetivamente ou ocorreu algum tipo de divergência durante o treinamento.

Tabela 4.5 – Otimização de Hiperparâmetros do *Baseline* - TV

dropout	η	F1
0.3	**4.4e-5**	**0.824**
0.4	5.5e-5	0.823
0.2	7.1e-5	0.809
0.4	2.6e-5	0.804
0.2	2.6e-5	0.799
0.5	5.0e-5	0.797
0.2	1.12e-4	0.792
0.4	2.15e-4	0.756
0.1	5.66e-4	0.000
0.1	6.5e-5	0.000

Fonte: Do autor (2023).

Os resultados do modelo para o conjunto de dados ReLi são marcantemente diferentes dos observados para o conjunto de dados TV. Apenas uma combinação de hiperparâmetro (taxa de *dropout* de 0.1 e taxa de aprendizado de 2.0×10^{-5}) apresentou um desempenho notável com F1 de 0.602. Todas as outras combinações de hiperparâmetros resultaram em um F1 de zero, sugerindo

que o modelo não foi capaz de aprender ou adaptar-se adequadamente para estas configurações específicas no conjunto de dados ReLi.

Tabela 4.6 – Otimização de Hiperparâmetros do *Baseline* - ReLi

dropout	η	F1
0.1	**2.0e-5**	**0.602**
0.2	9.5e-5	0.000
0.2	7.5e-5	0.000
0.2	3.15e-4	0.000
0.5	5.84e-4	0.000
0.5	5.30e-4	0.000
0.5	1.06e-4	0.000
0.3	5.29e-4	0.000
0.2	8.88e-4	0.000
0.1	1.48e-4	0.000

Fonte: Do autor (2023).

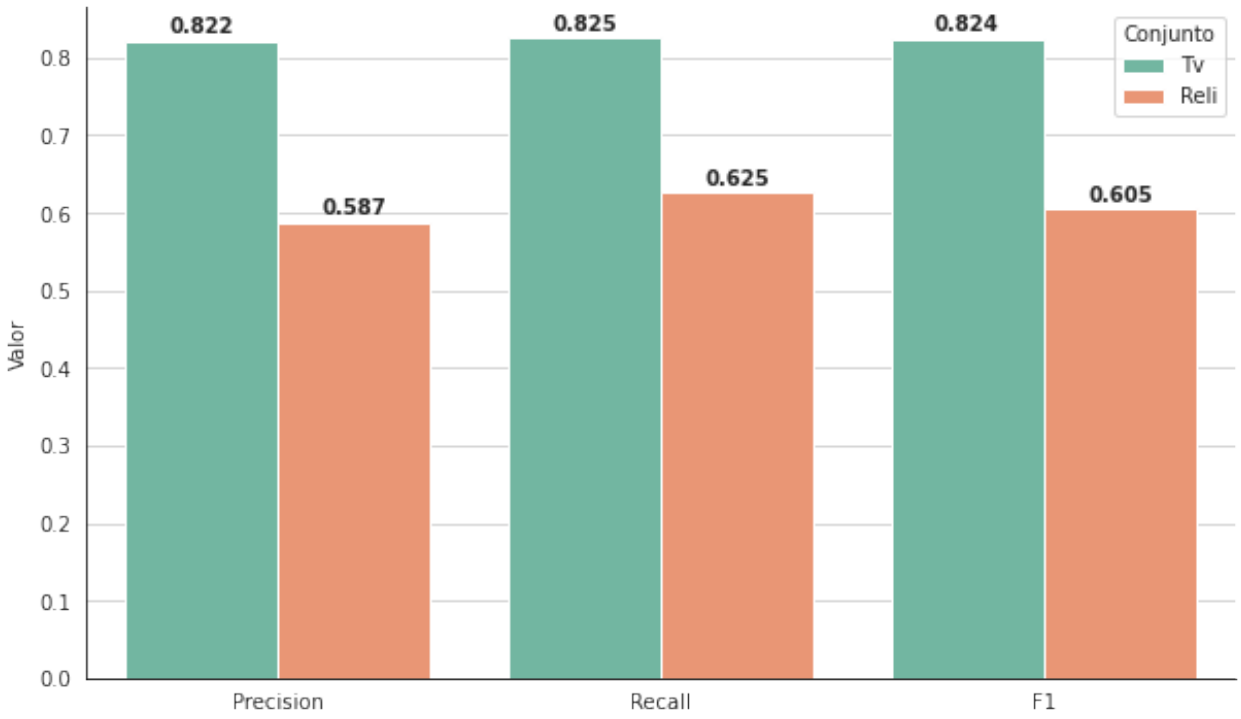
A grande discrepância nos resultados do *baseline* entre os dois conjuntos de dados sugere que as características intrínsecas de cada conjunto têm um impacto significativo na otimização dos hiperparâmetros. O que funcionou bem para um conjunto não foi adequado para o outro.

A taxa de *dropout* mostrou ter um papel crucial na eficácia do ajuste fino do modelo *baseline*. Enquanto valores mais intermediários (0.2-0.4) mostraram-se promissores para o conjunto de dados TV, um valor mais baixo (0.1) foi o mais eficaz para o conjunto de dados ReLi.

Os resultados apresentados na Figura 4.1 correspondem ao desempenho do modelo baseline no conjunto de teste, ajustado com as configurações ótimas, isto é, que obtiveram valores máximos de F1 no conjunto de validação..

A diferença marcante na distribuição de sentenças com aspectos entre os dois conjuntos de dados teve um impacto evidente no desempenho do *baseline*. Enquanto para o conjunto de dados TV, com uma proporção dominante de sentenças contendo aspectos, permitiu um desempenho robusto, o conjunto ReLi apresentou um desafio substancial devido ao seu desequilíbrio extremo.

Figura 4.1 – Resultados do *Baseline*



Fonte: Do autor (2023).

4.2.2 BERT de Domínio Específico

As tabelas 4.7 e 4.8 exibem os resultados provenientes da otimização de hiperparâmetros para os modelos que utilizam variantes do BERT de domínio específico. Os modelos serão referenciados da seguinte forma: BERTtv para o modelo baseado no BERT especializado em revisões de TV e BERTreli para o modelo centrado no BERT de domínio de resenhas literárias. Foram testadas 20 configurações diferentes, 10 em cada conjunto de dados.

Observa-se para o BERTtv que configurações com taxas de *dropout* mais elevadas tendem a obter resultados superiores. A taxa de aprendizado variou amplamente entre as configurações de topo, indicando uma sensibilidade nesta hiperparâmetro.

Nota-se que, diversas configurações resultaram em um F1 igual a zero para o BERTreli, semelhante ao que foi observado no *baseline* para o ReLi, o que pode indicar, além da discrepância

Tabela 4.7 – Otimização de Hiperparâmetros do BERTtv

dropout	η	F1
0.5	**1.5e-4**	**0.744**
0.5	9.0e-5	0.742
0.3	9.9e-5	0.734
0.1	7.3e-5	0.733
0.3	2.35e-4	0.732
0.5	4.2e-5	0.727
0.1	5.0e-5	0.724
0.5	2.79e-4	0.708
0.1	2.19e-4	0.707
0.1	2.4e-5	0.699

Fonte: Do autor (2023).

de representatividade de aspectos no conjunto, que certas combinações de hiperparâmetros não são apropriadas para este conjunto de dados.

Tabela 4.8 – Otimização de Hiperparâmetros do BERTreli

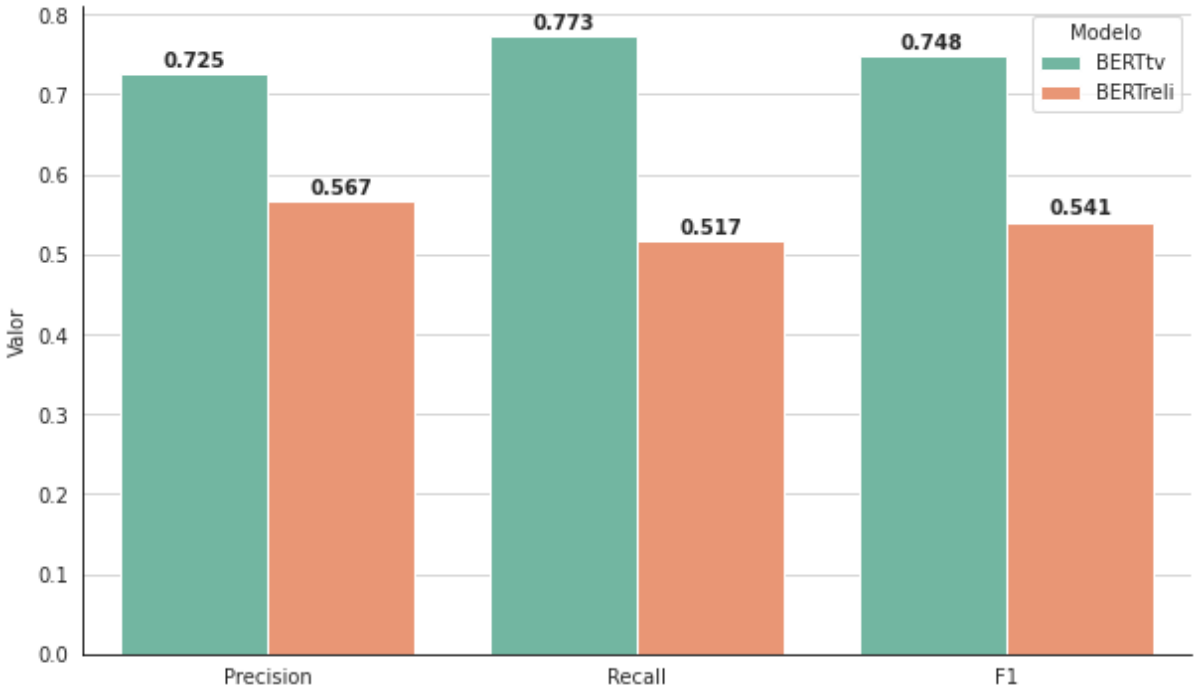
dropout	η	F1
0.5	**3.3e-5**	**0.463**
0.1	2.6e-5	0.439
0.1	9.0e-5	0.436
0.4	8.96e-4	0.000
0.5	2.74e-4	0.000
0.4	7.20e-4	0.000
0.2	6.93e-4	0.000
0.2	4.65e-4	0.000
0.1	4.56e-4	0.000
0.2	9.74e-4	0.000

Fonte: Do autor (2023).

A Figura 4.2 apresenta os resultados referentes às métricas de avaliação para os modelos BERT de domínio específico, BERTtv e BERTreli, após otimização, isto é, no conjunto de teste.

Ambos os modelos BERT de domínio específico, BERTtv e BERTreli, mostraram desempenho consideráveis em suas respectivas tarefas, especialmente considerando o fato de terem sido

Figura 4.2 – Resultados do BERT de Domínio Específico



Fonte: Do autor (2023).

pré-treinados em um conjunto de dados muito menor em comparação aos modelos BERT de domínio geral. Contudo, em comparação com o modelo *baseline*, os dois modelos específicos de domínio apresentaram desempenho inferior. Esses resultados enfatizam a eficiência do modelo *baseline*, ao mesmo tempo que realçam as possibilidades oferecidas por modelos específicos de domínio. Isso indica que, com a inclusão de mais dados ou com otimizações extras, tais modelos têm potencial para obter desempenhos superiores.

4.2.3 BERT de Domínio Geral com LoRa

As Tabelas 4.9 e 4.10 correspondem aos resultados da otimização de hiperparâmetros utilizando o modelo BERT geral em combinação com LoRa. Ao todo foram 40 configurações testadas, 20 para cada conjunto de dados.

Considerando o conjunto de dados TV, observa-se que a melhor configuração alcançou uma pontuação F1 de 0.845. Neste caso, uma porção pequena dos parâmetros do modelo foi

ajustável durante o treinamento, 889.350 de um total de 109.221.894 ($P\% = 0.814\%$), com uma configuração que incluiu um valor relativamente alto para o parâmetro r (22), indicando uma alta dimensionalidade para a adaptação de baixo *rank*, e um valor moderado para α (29), sugerindo uma influência substancial da adaptação de baixo *rank* nos parâmetros do modelo.

Tabela 4.9 – Otimização de Hiperparâmetros do BERTimbau com LoRa para o TV

$P\%$	r	α	*dropout*	Viés	η	B	W_{DECAY}	F1
0.814	**22**	**29**	**0.335**	**all**	**0.00099**	**8**	**0.063**	**0.845**
0.807	17	22	0.217	none	0.00088	8	0.059	0.826
0.975	16	26	0.223	none	0.00089	8	0.056	0.822
0.841	9	24	0.107	none	0.00085	8	0.002	0.821
0.579	22	16	0.226	none	0.00017	8	0.058	0.820
0.539	11	26	0.273	none	0.00061	16	0.032	0.818
0.546	8	26	0.125	none	0.00094	8	0.000	0.814
0.814	23	29	0.334	all	0.00028	8	0.062	0.811
0.874	19	25	0.106	none	0.00019	8	0.001	0.802
1.175	26	15	0.336	all	0.00011	8	0.068	0.800

Fonte: Do autor (2023).

Ao analisar o desempenho no ReLi, percebe-se que o desempenho foi significativamente mais baixo, com uma pontuação F1 máxima de 0.639. A configuração ótima para esse conjunto de dados apresentou uma proporção menor de parâmetros ajustáveis, 446.982 de um total de 108.779.526 ($P\% = 0.411\%$) e valores menores para r (12) e α (27).

A Figura 4.3 apresenta os resultados obtidos no conjunto de teste utilizando as melhores configurações de hiperparâmetros identificadas na otimização. O modelo ajustado no conjunto TV apresenta resultados que refletem um equilíbrio entre precisão e revocação, indicando um desempenho consistente, superando em aproximadamente 2.3% o resultado do *baseline*. Os resultados para o modelo ajustado no ReLi são equiparáveis aos resultados do *baseline*.

Assim como observado nos resultados apresentados até o momento, nota-se que as características específicas de cada conjunto de dados, como a complexidade da linguagem e a variedade de aspectos mencionados, representam desafios significativos para a modelagem, demostrados pela discrepância dos resultados obtidos em cada conjunto.

Tabela 4.10 – Otimização de Hiperparâmetros do BERTimbau com LoRa para o ReLi

P%	*r*	α	*dropout*	Viés	η	*B*	W_{DECAY}	F1
0.411	**12**	**27**	**0.456**	**none**	**0.00031**	**32**	**0.086**	**0.639**
0.310	9	23	0.474	none	0.00033	32	0.099	0.633
0.411	12	20	0.423	none	0.00042	32	0.073	0.631
0.411	12	16	0.383	none	0.00068	32	0.093	0.625
0.276	8	15	0.408	none	0.00062	32	0.081	0.622
0.377	11	21	0.466	none	0.00013	32	0.099	0.620
0.579	17	15	0.497	none	0.00045	32	0.081	0.618
0.370	8	27	0.429	all	0.00027	8	0.077	0.618
0.276	8	27	0.438	none	0.00021	32	0.098	0.616
0.478	14	25	0.454	none	0.00033	32	0.092	0.610

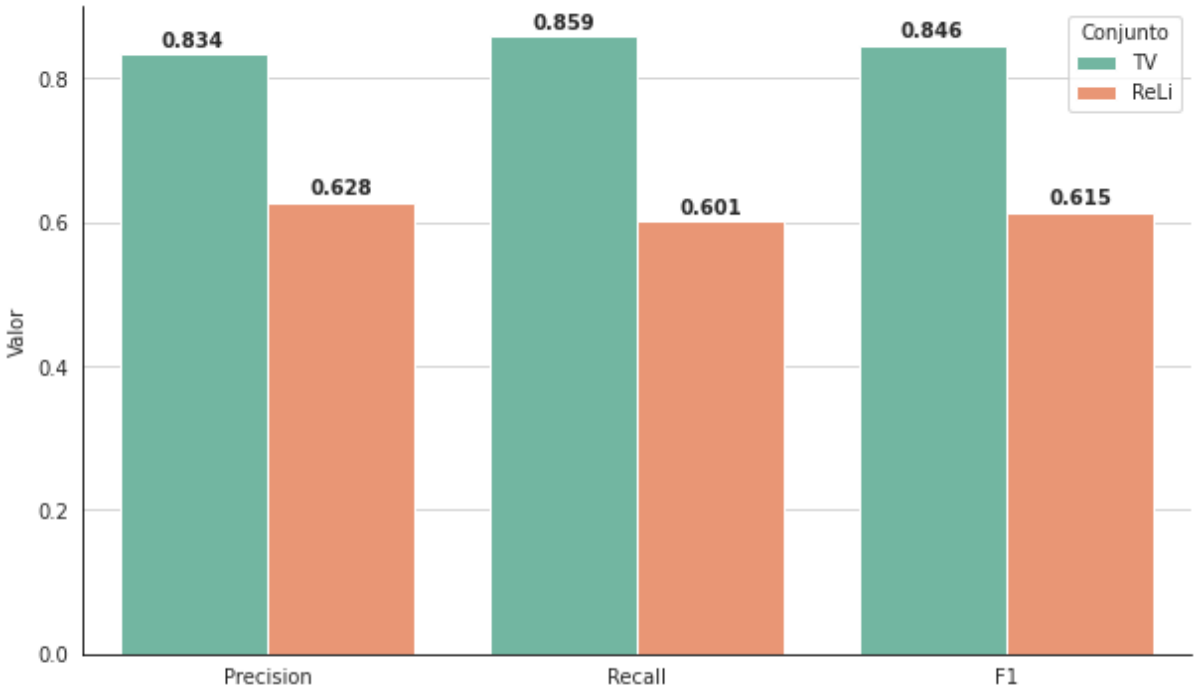
Fonte: Do autor (2023).

Ao comparar os resultados obtidos com o modelo BERT de domínio geral, tanto o *baseline* quanto na versão adaptada com LoRa, observa-se diferenças significativas que realçam a eficácia da técnica LoRa em otimizar a performance do modelo. Enquanto o *baseline* (BERT de domínio geral sem LoRa) alcançou um F1 máximo de 0.824 para o conjunto de dados TV e 0.605 para o conjunto ReLi, a implementação com LoRa superou esses resultados, atingindo um F1 de 0.846 para TV e 0.615 para ReLi. Esta melhoria na pontuação F1 ilustra como a estratégia LoRa, ao permitir ajustes finos em uma porção reduzida dos parâmetros do modelo, pode efetivamente aprimorar a capacidade de adaptação do modelo às especificidades de cada conjunto de dados.

4.2.4 *Embedding* Duplo

As Tabelas 4.11 e 4.12 apresentam os resultados da otimização de hiperparâmetros para o modelo de embedding duplo. Este modelo combina o BERTimbau, de domínio geral, com os BERTs específicos BERTtv e BERTreli, e serão referenciados da seguinte forma: d-BERTtv para ajuste sobre o conjunto TV e d-BERTreli para o ajuste sobre o ReLi. Foram avaliadas 20 configurações para o domínio de TV e 10 para o ReLi. O conjunto ReLi teve menos configurações testadas devido à sua maior demanda por recursos computacionais comparado ao conjunto TV.

Figura 4.3 – Resultados do BERT de Domínio Geral com LoRa



Fonte: Do autor (2023).

Como cada um dos BERTs possui uma saída com dimensão igual a 768, a concatenação desses dois BERTs produziu *embeddings* com dimensão igual a 1.536.

Para d-BERTtv observa-se que a taxa de *dropout* de 0.3 é consistentemente favorável, com diversas configurações apresentando bons resultados. A taxa de aprendizado, no entanto, varia, mas permanece relativamente baixa. Isso indica que pequenas alterações na taxa de aprendizado podem afetar significativamente o desempenho, mas uma taxa de *dropout* moderada é benéfica.

O modelo de *embedding* duplo manifesta uma amplitude maior em desempenhos robustos quando comparado ao BERT específico e ao *baseline*, ambos apresentando diversas combinações com F1 nulo. Notadamente, no domínio das resenhas literárias, certas configurações do embedding duplo levam a um F1 de zero, evidenciando a sensibilidade deste conjunto à otimização de hiperparâmetros. A busca pelo conjunto ideal de hiperparâmetros demonstra uma variação mais expressiva neste contexto.

Os resultados apresentados na Figura 4.4 são referentes às métricas avaliadas no conjunto de teste para as configurações ótimas dos modelos de embedding duplo (d-BERTtv e d-BERTreli).

Tabela 4.11 – Otimização de Hiperparâmetros do d-BERTtv

dropout	η	F1
0.3	**2.1e-05**	**0.850**
0.3	2.6e-05	0.846
0.3	3.4e-05	0.840
0.3	2.1e-05	0.838
0.4	2.1e-05	0.836
0.3	2.9e-05	0.831
0.4	3.5e-05	0.830
0.3	2.1e-05	0.827
0.2	4.4e-05	0.819
0.1	5.4e-05	0.778

Fonte: Do autor (2023).

A abordagem de *embedding* duplo exibiu um desempenho destacado no domínio de revisões de TV, refletindo resultados sólidos em todas as métricas avaliadas. Contudo, no contexto das resenhas literárias, essa metodologia revelou um desempenho mais contido, sinalizando espaço para refinamentos. Esta contraposição ressalta que para o conjunto TV, detentor de uma predominância de sentenças que contêm aspectos, gera um ajuste melhor do modelo. Em contraste, o conjunto ReLi, marcado por sua notável disparidade na distribuição de sentenças contendo aspectos, permanece como um desafio robusto na tarefa de extração, mesmo quando se aplica técnicas avançadas, como o *embedding* duplo.

As tabelas 4.13 e 4.14 apresentadas ilustram os resultados da otimização de hiperparâmetros para o modelo de duplo *embedding* complementado por camadas *feed-forward* subsequentes à concatenação dos *embeddings*. Assim como a otimização de hiperparâmetros para d-BERTtv e d-BERTreli, foram avaliadas 20 configurações para o domínio de TV e 10 para o ReLi.

Na Tabela 4.13, que se refere ao modelo d-BERTtv-FFN, é evidente que a configuração otimizada, com um *dropout* de 0.4, $N_1 = 768$, $N_2 = 1536$, $N_3 = 0$ e uma taxa de aprendizado η de 0.00002, alcançou a maior pontuação F1 de 0.830. As variações na profundidade e tamanho das camadas *feed-forward*, combinadas com diferentes taxas de *dropout* e aprendizado, sugerem uma sensibilidade do modelo às mudanças na arquitetura e hiperparâmetros.

Tabela 4.12 – Otimização de Hiperparâmetros do d-BERTreli

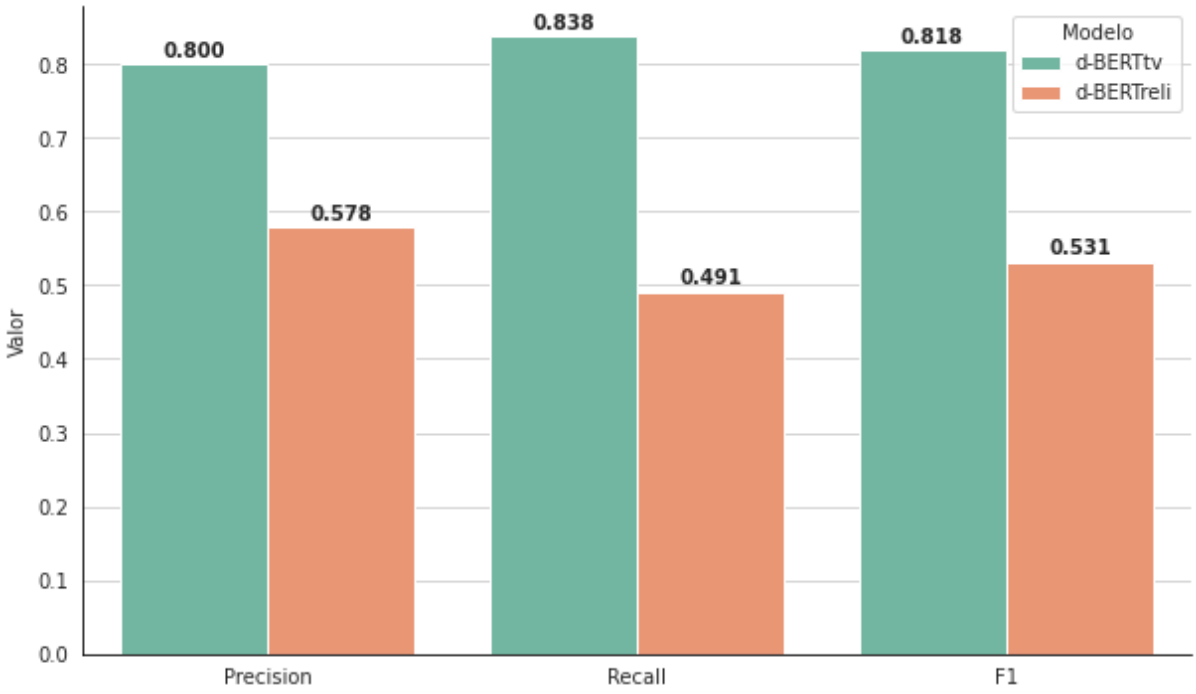
dropout	η	F1
0.5	**2.6e-05**	**0.622**
0.2	8.4e-05	0.547
0.0	8.0e-05	0.537
0.4	3.9e-05	0.478
0.0	3.0e-05	0.456
0.4	1.4e-04	0.364
0.5	3.1e-04	0.000
0.2	7.8e-04	0.000
0.4	5.0e-04	0.000
0.4	3.4e-04	0.000

Fonte: Do autor (2023).

Em contraste, a Tabela 4.14 apresenta os resultados para o domínio das resenhas literárias (d-BERTreli-FFN). O desempenho ótimo foi atingido com um *dropout* de 0, $N_1 = 256$, $N_2 = 512$, $N_3 = 768$ e uma taxa de aprendizado η de 0.000039, resultando em um F1 de 0.469. Nota-se que, para este domínio, as configurações de hiperparâmetros que levaram a um F1 próximo a zero foram mais frequentes. Este comportamento reforça a percepção de que o conjunto ReLi, com suas características distintas e desequilíbrios, representa um desafio intrínseco para a extração de aspectos.

É crucial destacar que o número limitado de testes (apenas 10) durante a otimização dos hiperparâmetros pode ter contribuído para essa variação nos resultados. Uma amostra mais extensa do espaço de hiperparâmetros poderia ter revelado combinações mais promissoras ou evitado configurações subótimas. Esse número restrito de testes também pode ter levado a uma sensibilidade exagerada a certas configurações que não são ideais para o domínio das resenhas literárias. Adicionalmente, riscos associados a *overfitting* ou *underfitting* podem ter sido exacerbados devido à restrição no número de testes. Portanto, enquanto esses resultados fornecem percepções valiosas, uma exploração mais profunda do espaço de hiperparâmetros poderia fornecer uma visão mais clara e robusta do potencial do d-BERTreli-FFN para este domínio.

Figura 4.4 – Resultados do *Embedding* Duplo



Fonte: Do autor (2023).

A Figura 4.5 apresenta os resultados para os *embeddings* obtidos no conjunto de teste. O modelo d-BERTtv-FFN confirma sua eficácia no domínio de revisões de TV, mostrando-se sólido em todas as métricas avaliadas. Em contrapartida, o d-BERTreli-FFN, apesar de apresentar desempenho moderado no domínio das resenhas literárias, ainda sugere espaço para refinamentos.

4.2.5 GPT

Os resultados apresentados na Figura 4.6 referem-se à métricas de avaliação para modelos ajustados para o conjunto TV das variantes do GPT disponíveis via API pela OpenAI: Ada, Babbage e Curie. Estes modelos são caracterizados pela sua natureza conversacional.

Os modelos ajustados através de *prompts* demostram capacidade notáveis em termos de precisão, *recall* e F1. A técnica de ajuste fino por meio de *prompts* permite que estes modelos se especializem em domínios específicos, ampliando sua aplicabilidade e eficácia em diversas tarefas. Entre as variantes, Babbage e Curie mostram-se particularmente promissores, com Curie

Tabela 4.13 – Otimização de Hiperparâmetros do d-BERTtv-FFN

dropout	N_1	N_2	N_3	η	F1
0.4	**768**	**1536**	**0**	**2.0e-05**	**0.830**
0.3	768	1536	512	3.6e-05	0.828
0.3	768	1536	512	2.0e-05	0.827
0.2	768	1536	0	5.9e-05	0.822
0.4	768	256	512	2.9e-05	0.820
0.3	256	1536	0	7.7e-05	0.820
0.4	768	768	768	3.5e-05	0.817
0.4	768	1536	1536	2.0e-05	0.815
0.2	768	1536	0	4.4e-05	0.812
0.2	1536	512	256	5.7e-05	0.807

Fonte: Do autor (2023).

destacando-se em termos de *recall*. Estes resultados evidenciam o potencial e a versatilidade dos modelos conversacionais quando adequadamente ajustados e treinados.

No entanto, é essencial considerar que, devido à sua natureza conversacional, a eficácia destes modelos também dependem da qualidade e representatividade dos *prompts* fornecidos durante o treinamento. Para o conjunto ReLi, os modelos GPT não foram capazes de ajustar-se, não gerando nenhum resultado para avaliar. Uma hipótese plausível para esta limitação é que a quantidade de sentenças com aspectos no conjunto ReLi é significativamente inferior ao conjunto TV. Esta disparidade pode impactar a capacidade do modelo de generalizar e aprender nuances específicas, especialmente em conjuntos de dados mais escassos ou menos representativos.

4.2.6 openCabrita com QLoRa

A Figura 4.7 apresenta os resultados obtidos nos dados de teste para o ajuste fino do openCabrita com QLoRa. Ao analisarmos estes resultados, observamos um desempenho heterogêneo nas métricas de precisão, revocação e F1, tanto para o domínio de TV quanto para o ReLi.

O ajuste fino com QLoRa para os hiperparâmetros utilizados possibilitou um treinamento com apenas 4.3M de parâmetros de um total de 3.5B, isto é, 0.12% dos parâmetros totais.

Tabela 4.14 – Otimização de Hiperparâmetros do d-BERTreli-FFN

dropout	N_1	N_2	N_3	η	F1
0.0	**256**	**512**	**768**	**3.9e-05**	**0.469**
0.5	512	512	512	4.3e-05	0.469
0.5	768	512	0	9.5e-05	0.449
0.0	768	256	0	1.4e-04	0.320
0.4	256	768	256	2.9e-04	0.000
0.2	512	512	768	5.7e-04	0.000
0.0	768	512	0	4.7e-04	0.000
0.2	768	256	512	6.9e-04	0.000
0.2	256	512	512	8.0e-04	0.000
0.2	256	768	256	2.6e-04	0.000

Fonte: Do autor (2023).

Para o conjunto TV, o modelo apresentou uma capacidade inferior as abordagens anteriores de identificar as instâncias relevantes do conjunto de dados, mantendo um balanço entre os falsos positivos e os falsos negativos.

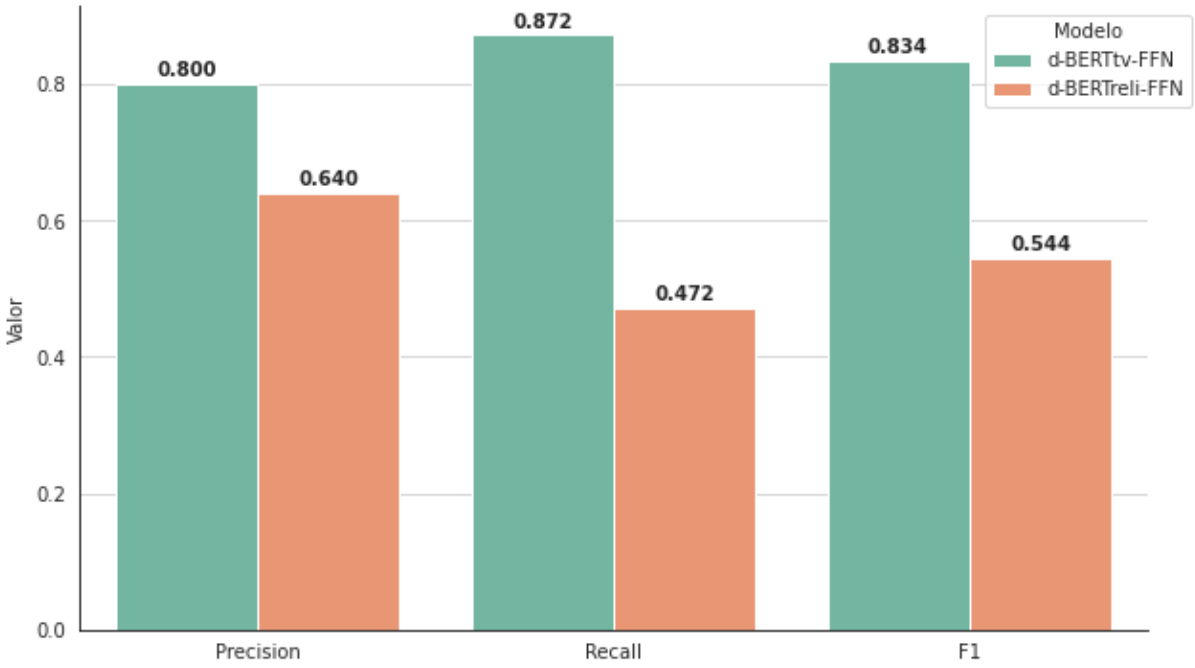
Ao observar os resultados para o ReLi, percebe-se uma discrepância significativa entre precisão e revocação. Isso sugere que, enquanto o modelo é preciso nas instâncias que identifica como relevantes, ele tem uma tendência a não identificar uma quantidade considerável de instâncias positivas reais, resultando em um número elevado de falsos negativos.

4.3 Discussões

Considerando que os treinamentos foram realizados empregando recursos de *hardware* distintos, uma avaliação comparativa direta do tempo de processamento para cada estratégia se revela inadequada. Há uma correlação, embora não linear, entre o tempo de treinamento e a magnitude da arquitetura do modelo. Portanto, as discussões subsequentes sobre a eficiência do treinamento das abordagens aqui avaliadas serão baseadas na quantidade de parâmetros treináveis das redes, isto é, no volume total de pesos aprendidos pela rede ao longo do processo de treinamento.

A relação quantitativa entre o total de parâmetros treináveis e a métrica F1 de cada modelo é apresentada nas Figuras 4.8 e 4.9. A Figura 4.8 detalha os resultados obtidos no domínio das

Figura 4.5 – Resultados do *Embedding* Duplo com Camadas Adicionais

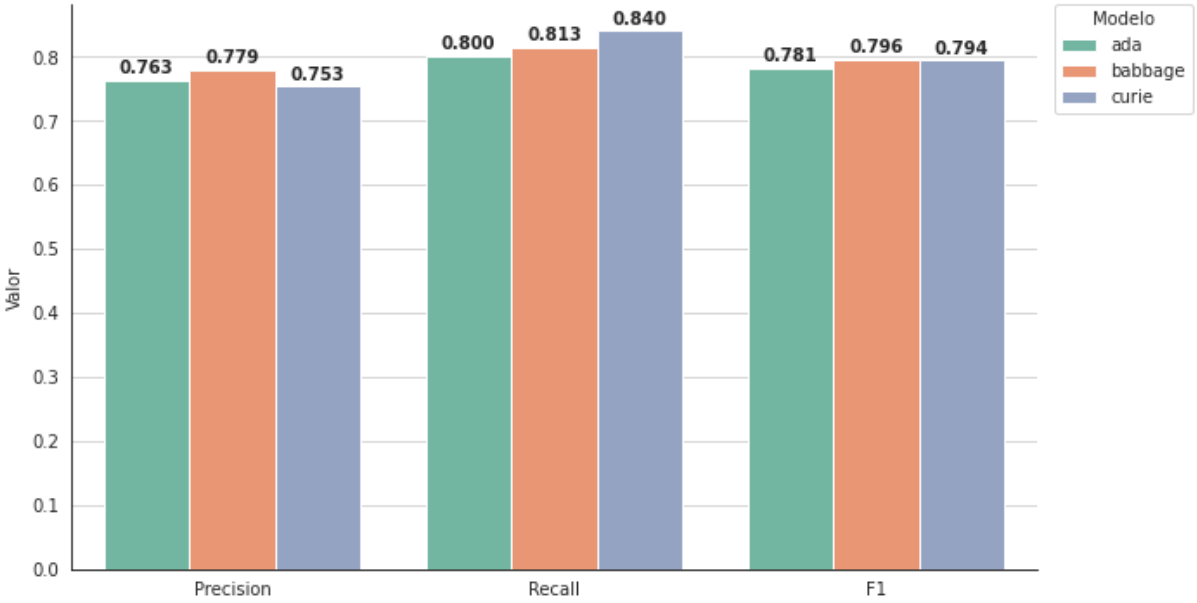


Fonte: Do autor (2023).

revisões de TV, utilizando o conjunto de dados TV para avaliação. Por sua vez, a Figura 4.9 se concentra no domínio das resenhas literárias, com a avaliação realizada sobre o conjunto de dados ReLi.

No conjunto de dados TV, o modelo BERTimbau-LoRa (em Apência A pode ser encontrado um exemplo do fluxo desse modelo com uma previsão), que implementa a estratégia LoRa em um BERT de domínio geral, destaca-se por seu número reduzido de parâmetros treináveis, alcançando o F1 *score* mais alto de 0.846. Esse resultado é notável, pois supera modelos com um número muito maior de parâmetros, como o d-BERTtv-FFN, que, possuindo a maior quantidade de parâmetros, obteve um F1 *score* de 0.834. Além disso, um número menor de parâmetros treináveis geralmente se traduz em um tempo de treinamento mais curto em comparação com modelos mais densos. No entanto, quanto ao uso de memória durante o treinamento, o modelo BERTimbau-LoRa, apesar de ter menos parâmetros treináveis, requer uma quantidade de memória semelhante ao modelo *baseline*, uma vez que ambos utilizam a arquitetura BERTimbau e mantêm uma grande parcela de parâmetros congelados, que não são atualizados, mas ainda ocupam memória.

Figura 4.6 – Resultados das Variantes do GPT para o Conjunto TV

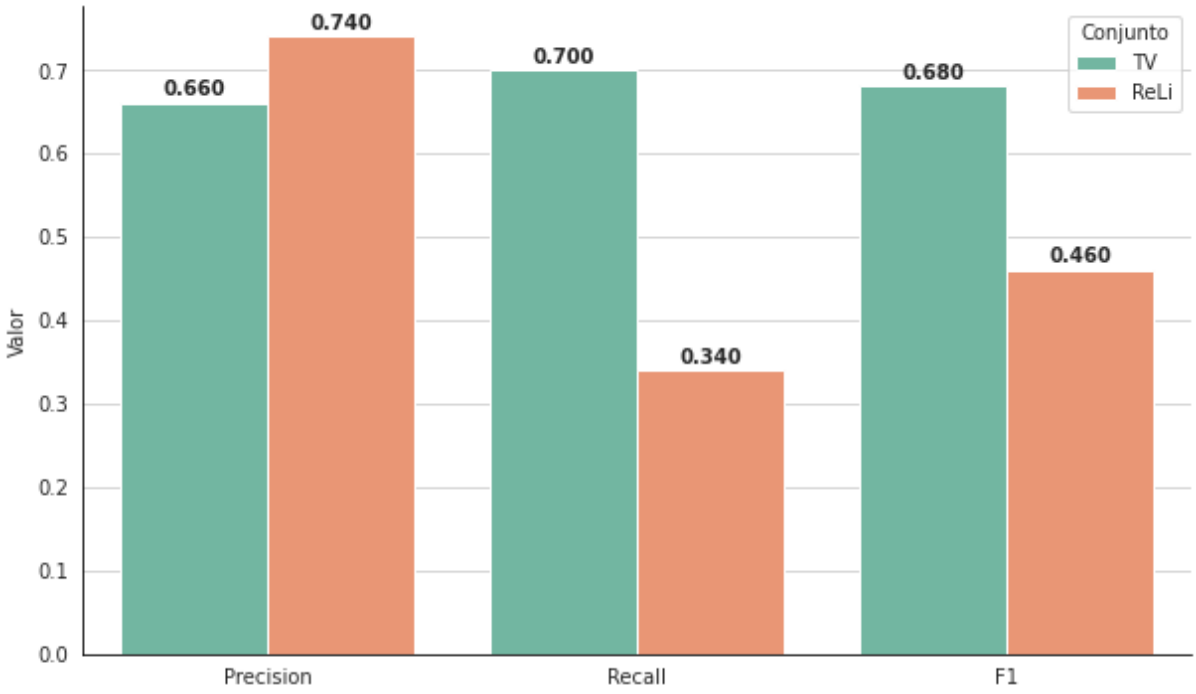


Fonte: Do autor (2023).

Para o conjunto de dados TV, observa-se que a fusão dos modelos de domínio geral (BERTimbau) e específico (BERTtv), representado pelo modelo d-BERTtv-FFN, que integra camadas adicionais após a concatenação desses domínios, resultou em um aumento no F1 *score* em comparação aos desempenhos isolados tanto do *baseline* quanto do BERTtv. Notavelmente, a combinação direta entre os domínios geral e específico levou a uma pequena redução do F1 em relação ao *baseline*. Esta observação sugere que a interação entre os domínios pode ser aprimorada com a introdução de camadas adicionais, as quais têm o potencial de aprender a combinar os domínios de maneira mais eficaz. Tal estratégia indica que a precisão dos modelos pode ser significativamente melhorada pela configuração estratégica das camadas pós concatenação, permitindo assim um aprendizado mais detalhado e refinado das características intrínsecas de cada domínio.

Considerando as limitações inerentes ao ajuste fino das variantes do GPT-3 (resultados apresentado na Figura 4.6), é importante destacar que este processo foi conduzido através da API da OpenAI. Devido a essa metodologia, informações detalhadas sobre o número de parâmetros treinados não estão acessíveis. Além disso, o tempo de treinamento desses modelos é condicionado pela capacidade e disponibilidade dos servidores da OpenAI, fatores esses que estão fora do controle

Figura 4.7 – Resultados do openCabrita com QLoRa

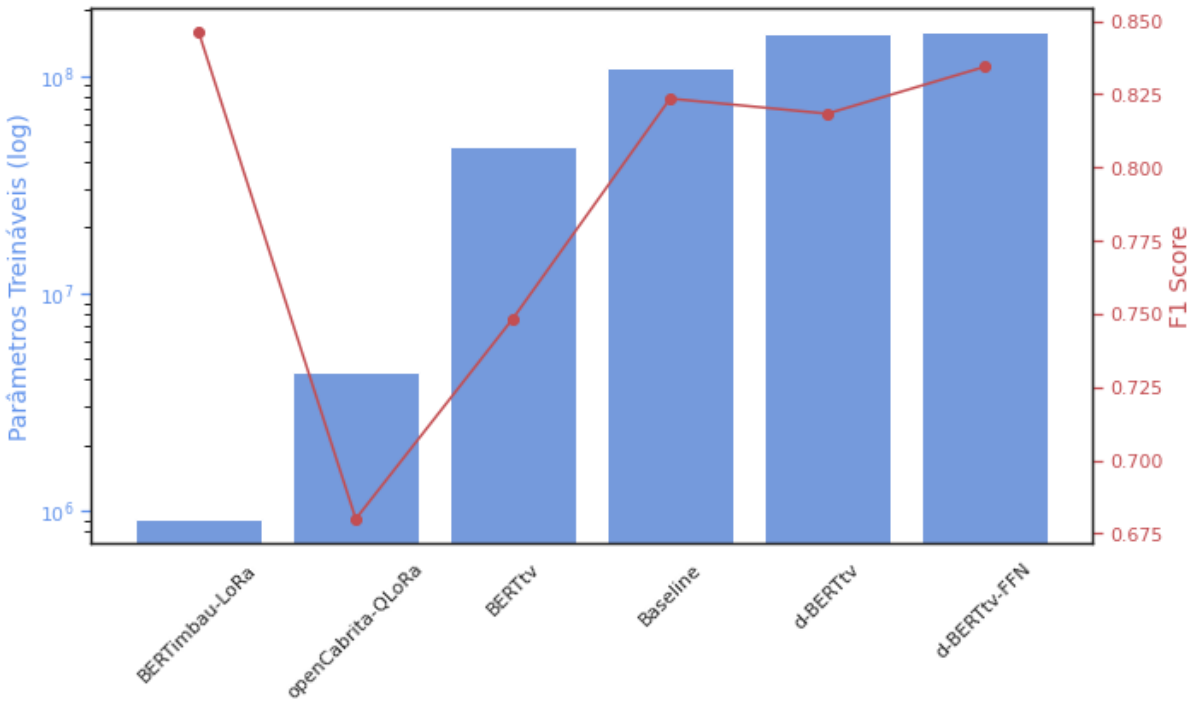


Fonte: Do autor (2023).

deste trabalho. Essas particularidades tornam a avaliação da eficiência computacional destes modelos impraticável, já que uma comparação precisa requer acesso a informações completas sobre os parâmetros e condições de treinamento. Portanto, a análise de eficiência, neste contexto, fica limitada pelos dados disponíveis e pelas condições impostas pelo ambiente de treinamento remoto.

Analogamente ao observado no conjunto de dados TV, no contexto do conjunto ReLi, destaca-se o modelo de domínio geral treinado utilizando a estratégia LoRa, que demonstrou superioridade ao alcançar o F1 *score* mais elevado de 0.615. Este resultado é notório, especialmente considerando que tal modelo possui o menor número de parâmetros treináveis dentre os avaliados, superando os modelos que possuem uma maior quantidade de parâmetros. Em relação à utilização de recursos de memória, o modelo de domínio geral com LoRa demonstra uma necessidade equivalente à do modelo *baseline*, pelas mesmas razões descritas anteriormente. Por sua vez, o resultado do *baseline* também ficou em segundo lugar (F1 *score* de 0.605) no conjunto ReLi. Este resultado evidencia a eficiência da estratégia LoRa não apenas em acurácia de modelagem, mas

Figura 4.8 – Parâmetros Treináveis e F1 (TV)

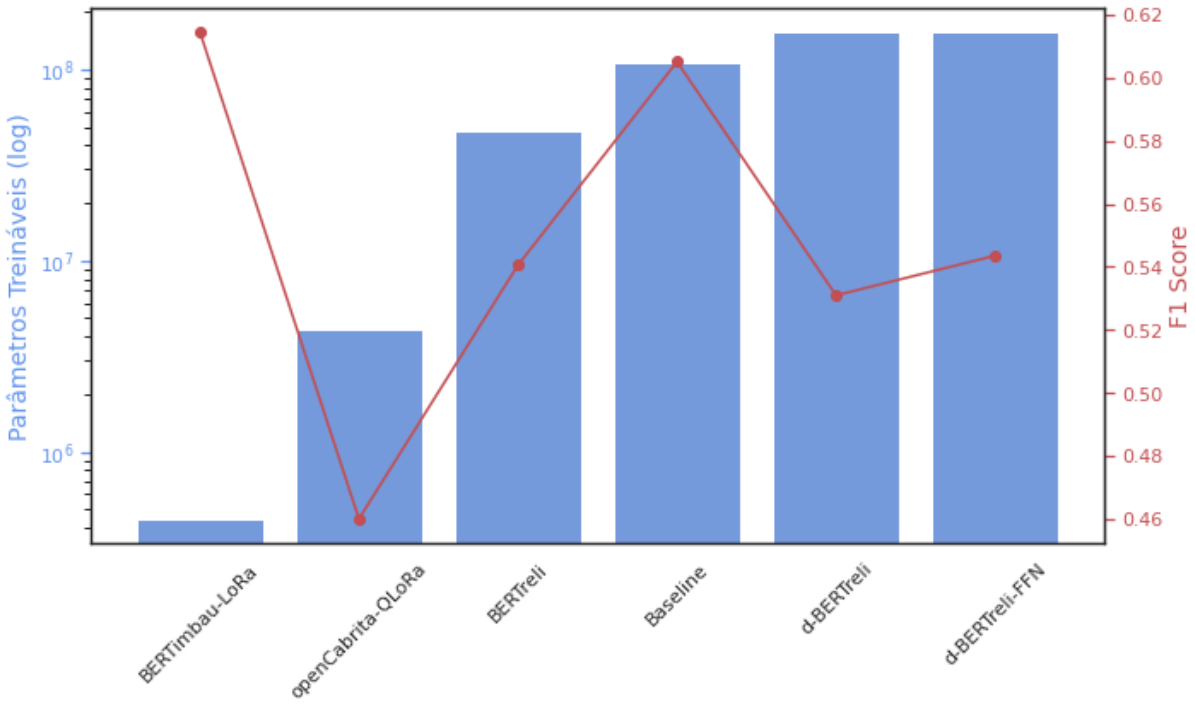


Fonte: Do autor (2023).

também na otimização do número de parâmetros treináveis, representando uma abordagem valiosa em situações onde a harmonia entre desempenho e consumo de recursos é essencial.

Nos modelos que combinam domínios geral e específico (BERTreli) para o conjunto ReLi, observa-se um desempenho distinto do constatado no conjunto TV. Ambos os modelos para o ReLi registraram F1 *scores* significativamente inferiores em comparação às outras estratégias, apresentando resultados até mesmo abaixo ou equivalentes aos do modelo de domínio específico. Para compreender essa discrepância, várias hipóteses podem ser consideradas. Uma delas é a diferença na forma de escrita entre o corpus de pré-treino do BERTreli e os dados utilizados no ajuste fino. O corpus do BERTreli é caracterizado por uma linguagem mais formal, típica de resenhas literárias profissionais, enquanto o conjunto de ajuste fino inclui resenhas de natureza mais informal, escritas por um público mais amplo e diversificado. Adicionalmente, a quantidade de aspectos no ReLi é proporcionalmente menor em comparação ao TV, o que pode ter impacto na capacidade de generalização dos modelos. Esses fatores, em conjunto, podem contribuir para o desempenho atípico dos

Figura 4.9 – Parâmetros Treináveis e F1 (ReLi)



Fonte: Do autor (2023).

modelos no conjunto ReLi, indicando a complexidade e a sensibilidade do processo de modelagem a variações nos dados e nas características do corpus.

Em termos de eficácia, os modelos de domínio específico, BERTtv e BERTreli, alcançaram respectivamente 90.9% e 89.3% do desempenho do modelo de domínio geral (*baseline*). Esse desempenho pode ser considerado satisfatório por algumas razões. Primeiramente, ao comparar com o BERTimbau, que foi pré-treinado com um volumoso corpus de 2.68 bilhões de *tokens*, o BERTtv foi pré-treinado com apenas 1.46 milhões de *tokens* e o BERTreli com 1.27 milhões. Isso representa menos de 0.05% do volume de dados utilizado pelo BERTimbau. Tal comparação destaca a eficiência dos modelos de domínio específico em alcançar um desempenho significativo com uma fração ínfima dos dados de treinamento. Além disso, a proporção de 43.6% de parâmetros nos modelos de domínio específico, em comparação ao BERTimbau, é outro ponto crucial. Essa redução substancial no número de parâmetros poderia sugerir uma queda correspondente no desempenho. No entanto, a manutenção de um desempenho comparativamente alto com significativamente menos

parâmetros demonstra a otimização e a adequação desses modelos para capturar as características essenciais dos respectivos domínios. Isso implica que, apesar de operarem com menos parâmetros e dados de treinamento, os modelos de domínio específico conseguem preservar uma grande parte da eficácia do modelo mais complexo e mais amplamente treinado.

O processo de ajuste fino do modelo *openCabrita*, empregando a estratégia QLoRa, resultou nos desempenhos mais modestos dentre as abordagens investigadas neste estudo. Este modelo, que possui 3 bilhões de parâmetros totais, somente 0.14% desse valor são parâmetros treináveis, uma consequência direta da adoção do LoRa. A técnica de quantização desempenha um papel crucial nesta configuração, mantendo os parâmetros originais congelados em NF4 (4-bit *NormalFloat*). Esta abordagem resulta em uma redução drástica dos requisitos de memória de vídeo necessários para o treinamento, tornando viável a execução em uma única GPU (modelo A100). Sem a estratégia de quantização, o treinamento de um modelo dessa magnitude em uma única GPU seria impraticável, mesmo com os parâmetros congelados. Isso se deve ao fato de que, embora congelados, os parâmetros ainda exigem memória para serem carregados e utilizados durante o treinamento. Assim, a quantização não apenas otimiza o uso de memória, mas também possibilita a execução de modelos complexos em *hardware* limitado.

Entre as abordagens analisadas neste estudo, sob as condições estabelecidas, os modelos de domínio geral treinados com a técnica LoRa se sobressaem. Além dos aspectos discutidos anteriormente sobre este modelo, a estratégia LoRa se destaca por possibilitar uma exploração mais eficiente do espaço de hiperparâmetros em comparação com o modelo de domínio geral sem LoRa (*baseline*). Isso é evidenciado pelo número de tentativas realizadas: 40 para o modelo com LoRa, contra apenas 20 para o *baseline*, uma vantagem derivada do menor número de parâmetros treináveis que também resulta em tempos de execução reduzidos.

Além disso, a utilização do LoRa oferece benefícios significativos em termos de armazenamento. Com essa técnica, é possível manter o BERTimbau, bem como adaptadores específicos para cada conjunto de dados ajustados, de forma separada. Esses adaptadores são substancialmente menores do que manter dois modelos de domínio geral ajustados individualmente para cada conjunto de dados. Por exemplo, o BERTimbau utilizado contém 110 milhões de parâmetros, enquanto os adaptadores específicos contêm 915.459 para o conjunto TV e 444.675 para o ReLi, totalizando

um pouco mais de 111 milhões de parâmetros armazenados. Em contraste, manter dois modelos de domínio geral ajustados separadamente para cada conjunto de dados exigiria armazenar 220 milhões de parâmetros, representando, portanto, uma economia significativa de aproximadamente 109 milhões de parâmetros.

Essa economia de armazenamento, aliada à maior eficiência na busca de hiperparâmetros e ao tempo de execução reduzido, reforça a viabilidade e a eficácia do uso da estratégia LoRa em modelos para processamento de linguagem natural, evidenciando uma estratégia eficaz para aprimorar o desempenho dos modelos sem comprometer a capacidade computacional.

5 CONCLUSÃO

A partir do modelo treinado existe uma variedade de aplicações práticas, que são de grande valor em diversos campos, incluindo *marketing*, análise de sentimentos e pesquisa de mercado.

Uma aplicação significativa do modelo de extração de aspectos é na área de análise de *feedback* de clientes. Por exemplo, desenvolvedores de eletrônicos podem utilizar essa técnica para analisar avaliações e comentários de usuários sobre seus produtos. Esta análise permite a identificação de padrões e tendências em relação a aspectos específicos do produto, como desempenho, design, custo-benefício e durabilidade. Ao extrair e analisar esses aspectos de forma detalhada, os desenvolvedores podem obter *insights* valiosos sobre as percepções dos consumidores, possibilitando melhorias direcionadas no design do produto, estratégias de *marketing* e desenvolvimento de novas funcionalidades.

Outro campo de aplicação prática do modelo é na pesquisa de mercado e na inteligência competitiva. Empresas podem utilizar essa técnica para monitorar e analisar as avaliações de produtos de concorrentes, identificando pontos fortes e fracos em produtos similares. Essa análise permite às empresas ajustar suas estratégias de mercado, desenvolver novos produtos que atendam melhor às necessidades dos consumidores, e posicionar seus produtos de forma mais efetiva no mercado. Além disso, a extração de aspectos também pode ser utilizada para monitorar mudanças nas tendências de consumo e preferências do público, o que pode ser crucial para manter a competitividade e relevância em mercados em rápida evolução.

Neste trabalho, foram exploradas algumas abordagens inéditas para o idioma português no contexto da extração de aspectos. Uma das estratégias implementadas envolveu o uso de modelos pré-treinados em domínios específicos baseados na arquitetura BERT. Foram desenvolvidas e disponibilizadas duas variantes do BERT especializadas: uma para resenhas literárias[1] e outra para análises de produtos de TV[2]. Outra abordagem é o conceito de duplo *embedding* usando arquitetura BERT, que consiste na combinação de *embeddings* de domínio geral com *embeddings* específicos ao problema em questão. Além disso, investigou-se o potencial de modelos generativos, através das variantes do GPT-3 (Ada, Babbage e Curie) e o *openCabrita*.

[1] https://huggingface.co/jcfneto/bert-br-portuguese
[2] https://huggingface.co/jcfneto/bert-tv-portuguese

Entre as diversas abordagens desenvolvidas e avaliadas neste estudo, o modelo que utiliza BERTimbau, isto é, pré-treinado em domínio geral ajustada com a técnica LoRa é o mais vantajoso para ambos os conjuntos de dados, alcançando F1 *score* de 0.846 para o conjunto de dados TV e 0.615 para o ReLi. Por outro lado, a estratégia menos vantajosa foi o ajuste fino do modelo de grande escala *openCabrita*, que obteve F1 de 0.68 para o TV e 0.46 para o ReLi.

5.1 Contribuições

Este estudo contribuiu significativamente para a área de extração de aspectos em língua portuguesa. Primeiramente, a implementação e a disponibilização de modelos BERT especializados para domínios específicos, como resenhas literárias e análises de produtos de TV, representam um avanço na personalização de ferramentas de processamento de linguagem natural para contextos específicos. Além disso, a introdução do conceito de duplo *embedding* em arquitetura BERT, combinando *embeddings* de domínio geral e específico, abre novas possibilidades para a personalização e eficácia dos modelos de extração de aspectos. Por fim, a investigação dos modelos generativos, como as variantes do GPT-3 e *openCabrita*, expande o escopo de possíveis abordagens para a tarefa de extração de aspectos.

Para facilitar futuras pesquisas e comparações, os conjuntos de dados utilizados neste estudo estão disponíveis em um repositório no GitHub[3], devidamente estratificados para permitir comparações justas com os resultados obtidos neste trabalho. Esta disponibilização visa promover uma continuidade nos avanços da área, incentivando outros pesquisadores a explorar e melhorar as técnicas de extração de aspectos em língua portuguesa.

5.2 Dificuldades

Este estudo apresentou algumas limitações. Uma delas é o fato dos experimentos terem sidos executados em diferentes tipos de *hardware*, o que impossibilitou de comparar o tempo de processamento de cada uma das estratégias. Outra limitação, já destacada no capítulo de Resulta-

[3] https://github.com/jcfneto/aspect-extraction

dos, é referente à quantidade de dados para o pré-treino dos BERTs de domínio específico, o que pode ter limitado a capacidade do modelo, resultando em desempenho inferior ao *baseline*.

5.3 Propostas de Continuidade

As seções a seguir descrevem algumas propostas de continuidade do trabalho, em busca de alcançar melhores resultados para a tarefa de extração de aspectos em língua portuguesa.

5.3.1 Ajuste Fino com LoRa

A estratégia LoRa demonstrou ser uma ferramenta altamente eficaz no ajuste fino de modelos, mantendo ou até melhorando o desempenho enquanto treina uma fração significativamente menor de parâmetros. A vantagem desta abordagem é evidente na aceleração do processo de treinamento e na simplificação da otimização de hiperparâmetros, possibilitando uma exploração mais ampla e eficiente do espaço de hiperparâmetros.

Nesse contexto, propõe-se a aplicação do LoRa em todas as abordagens que não se aproveitaram dessa estratégia neste estudo, com o intuito de expandir o espaço de busca por hiperparâmetros e aumentar o número de tentativas de otimização. Uma vez que, o ajuste fino é feito com um número de parâmetros treináveis menor, é viável também avaliar modelos mais complexos e de maior dimensão, como o BERTimbau$_{LARGE}$, que possui 335 milhões de parâmetros, e as variantes *BASE* e *LARGE* do XLM-RoBERTa (CONNEAU et al., 2019), com 270 milhões e 550 milhões de parâmetros, respectivamente. Essas abordagens têm o potencial de elevar significativamente a eficácia na tarefa de extração de aspectos para língua portuguesa.

5.3.2 Corpus do Pré-Treino

Observou-se que os modelos ajustados com o BERT de domínio específico obtiveram resultados significativos em comparação com o *baseline*, que foi calibrado sobre um modelo pré-treinado em domínio geral. Este desempenho é notável, especialmente considerando que os modelos de domínio específico foram pré-treinados com um volume significativamente menor de dados.

Para melhorar o desempenho desses modelos, uma estratégia promissora seria a expansão da coleta de dados específicos do domínio para o pré-treino. Entretanto, essa abordagem pode enfrentar desafios consideráveis, como a potencial escassez de dados disponíveis para domínios específicos e questões relativas à permissão para coletar e utilizar esses dados.

5.3.3 *Post-Training*

Em seu trabalho, Xu et al. (2019) aborda os desafios enfrentados no ajuste fino do modelo BERT para tarefas específicas, especialmente em cenários com dados limitados. Esses desafios incluem a discrepância entre os dados de treinamento e os dados reais da tarefa, assim como a adaptação do modelo à natureza específica da tarefa. Para aprimorar o desempenho em tarefas como a Análise de Sentimentos Baseada em Aspectos, é sugerido o uso de uma estratégia de pós-treinamento (*post-training*). Esta abordagem busca mitigar vieses introduzidos por conhecimentos irrelevantes ao domínio e combinar o conhecimento de domínio com o entendimento específico da tarefa.

Inspirando-se nesse estudo, uma abordagem para melhorar a extração de aspectos em língua portuguesa seria empregar o BERT de domínio geral, seguido de uma fase de pós-treinamento, que antecede o ajuste fino, utilizando MLM e NSP. Os dados empregados nesta etapa seriam os mesmos utilizados para o pré-treinamento de domínio específico, promovendo uma afinidade entre o modelo e as características particulares do domínio em questão.

5.3.4 LLMs

Este estudo empregou Modelos de Linguagem de Grande Escala (LLMs), como o GPT-3 e o *openCabrita*, em um contexto de ajuste fino. Contudo, conforme argumentado por Brown et al. (2020), modelos de linguagem de grande porte podem ser classificados como "aprendizes de poucos exemplos"(*few-shot learners*), indicando sua capacidade de aprender e generalizar a partir de um número limitado de exemplos. Para explorar plenamente o potencial desses modelos, a abordagem sugerida para a continuidade dessa pesquisa envolve fornecer-lhes poucos exemplos e

permitir que eles generalizem para outros conjuntos de dados. O desafio central para maximizar a eficácia desses modelos reside na otimização do *prompt*, de forma a extrair o máximo desempenho possível do modelo, aproveitando sua capacidade inerente de aprender e adaptar-se com base em informações limitadas.

REFERÊNCIAS

ALMEIDA, F.; XEXÉO, G. Word embeddings: A survey. **arXiv preprint arXiv:1901.09069**, 2019.

ARAÚJO, M.; PEREIRA, A.; BENEVENUTO, F. A comparative study of machine translation for multilingual sentence-level sentiment analysis. **Information Sciences**, Elsevier, v. 512, p. 1078–1102, 2020.

BENNETT, G. R. Using corpora in the language learning classroom: Corpus linguistics for teachers. **(No Title)**, 2010.

BROWN, T. et al. Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.

CARDOSO, B.; PEREIRA, D. Evaluating an aspect extraction method for opinion mining in the portuguese language. In: **Anais do VIII Symposium on Knowledge Discovery, Mining and Learning**. Porto Alegre, RS, Brasil: SBC, 2020. p. 137–144. ISSN 2763-8944. Disponível em: <https://sol.sbc.org.br/index.php/kdmile/article/view/11969>.

CHEN, Z. et al. Emoji-powered representation learning for cross-lingual sentiment classification. In: **The world wide web conference**. [S.l.: s.n.], 2019. p. 251–262.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

CHOLLET, F. **Deep learning with Python**. [S.l.]: Simon and Schuster, 2021.

CONNEAU, A. et al. Unsupervised cross-lingual representation learning at scale. **arXiv preprint arXiv:1911.02116**, 2019.

DERCZYNSKI, L. Complementarity, f-score, and nlp evaluation. In: **Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)**. [S.l.: s.n.], 2016. p. 261–266.

DETTMERS, T. et al. Qlora: Efficient finetuning of quantized llms. **arXiv preprint arXiv:2305.14314**, 2023.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DO, H. H. et al. Deep learning for aspect-based sentiment analysis: a comparative review. **Expert systems with applications**, Elsevier, v. 118, p. 272–299, 2019.

FREITAS, C. et al. Vampiro que brilha... rá! desafios na anotaçao de opiniao em um corpus de resenhas de livros. **Encontro de Linguística de Corpus**, v. 11, p. 22, 2012.

FREITAS, L. A. D.; VIEIRA, R. Exploring resources for sentiment analysis in portuguese language. In: IEEE. **2015 Brazilian conference on intelligent systems (BRACIS)**. [S.l.], 2015. p. 152–156.

GARCIA, K.; BERTON, L. Topic detection and sentiment analysis in twitter content related to covid-19 from brazil and the usa. **Applied soft computing**, Elsevier, v. 101, p. 107057, 2021.

GENG, X.; LIU, H. **OpenLLaMA: An Open Reproduction of LLaMA**. 2023. Disponível em: <https://github.com/openlm-research/open_llama>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

HOFFMANN, J. et al. Training compute-optimal large language models. **arXiv preprint arXiv:2203.15556**, 2022.

HU, E. J. et al. Lora: Low-rank adaptation of large language models. **arXiv preprint arXiv:2106.09685**, 2021.

HU, M.; LIU, B. Mining and summarizing customer reviews. In: **Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2004. p. 168–177.

HUSSEIN, D. M. E.-D. M. A survey on sentiment analysis challenges. **Journal of King Saud University-Engineering Sciences**, Elsevier, v. 30, n. 4, p. 330–338, 2018.

JAKOB, N.; GUREVYCH, I. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In: **Proceedings of the 2010 conference on empirical methods in natural language processing**. [S.l.: s.n.], 2010. p. 1035–1045.

JELODAR, H. et al. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. **Multimedia Tools and Applications**, Springer, v. 78, p. 15169–15211, 2019.

JIN, W.; HO, H. H.; SRIHARI, R. K. Opinionminer: a novel machine learning system for web opinion mining and extraction. In: **Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2009. p. 1195–1204.

KARIMI, A.; ROSSI, L.; PRATI, A. Improving bert performance for aspect-based sentiment analysis. **arXiv preprint arXiv:2010.11731**, 2020.

KARIMI, A.; ROSSI, L.; PRATI, A. Adversarial training for aspect-based sentiment analysis with bert. In: IEEE. **2020 25th International conference on pattern recognition (ICPR)**. [S.l.], 2021. p. 8797–8803.

LARCHER, C. et al. Cabrita: closing the gap for foreign languages. **arXiv preprint arXiv:2308.11878**, 2023.

LI, P. et al. Prompt tuning large language models on personalized aspect extraction for recommendations. **arXiv preprint arXiv:2306.01475**, 2023.

LIU, B. Sentiment analysis and opinion mining. **Synthesis lectures on human language technologies**, Morgan & Claypool Publishers, v. 5, n. 1, p. 1–167, 2012.

LOPES, É.; CORREA, U.; FREITAS, L. Exploring bert for aspect extraction in portuguese language. In: **The International FLAIRS Conference Proceedings**. [S.l.: s.n.], 2021. v. 34.

MACHADO, M. T.; PARDO, T. A. S.; RUIZ, E. E. S. Analysis of unsupervised aspect term identification methods for portuguese reviews. **Anais do XIV Encontro Nacional de Inteligncia Artificial e Computacional (ENIAC), SBC**, p. 239–249, 2017.

MACHADO, M. T. et al. Mineraçao de tópicos e aspectos em microblogs sobre dengue, chikungunya, zika e microcefalia. In: SBC. **Anais do XVII Workshop de Informática Médica**. [S.l.], 2017.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. **Ain Shams engineering journal**, Elsevier, v. 5, n. 4, p. 1093–1113, 2014.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MUKHERJEE, A.; LIU, B. Aspect extraction through semi-supervised modeling. In: **Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [S.l.: s.n.], 2012. p. 339–348.

OPENAI. **GPT-4 Technical Report**. 2023. ArXiv:2303.08774 [cs.CL]. Disponível em: <https://doi.org/10.48550/arXiv.2303.08774>.

PAVLOPOULOS, I. Aspect based sentiment analysis. **Athens University of Economics and Business**, 2014.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

PEREIRA, D. A. A survey of sentiment analysis in the portuguese language. **Artificial Intelligence Review**, Springer, v. 54, n. 2, p. 1087–1115, 2021.

PORIA, S.; CAMBRIA, E.; GELBUKH, A. Aspect extraction for opinion mining with a deep convolutional neural network. **Knowledge-Based Systems**, Elsevier, v. 108, p. 42–49, 2016.

RADFORD, A. et al. Improving language understanding by generative pre-training. OpenAI, 2018.

RADFORD, A. et al. Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, p. 9, 2019.

RASCHKA, S. Model evaluation, model selection, and algorithm selection in machine learning. **arXiv preprint arXiv:1811.12808**, 2018.

SANTOS, B. N. D.; MARCACINI, R. M.; REZENDE, S. O. Multi-domain aspect extraction using bidirectional encoder representations from transformers. **IEEE Access**, IEEE, v. 9, p. 91604–91613, 2021.

SHAZEER, N. Glu variants improve transformer. **arXiv preprint arXiv:2002.05202**, 2020.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: pretrained bert models for brazilian portuguese. In: SPRINGER. **Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I 9**. [S.l.], 2020. p. 403–417.

SU, J. et al. Roformer: Enhanced transformer with rotary position embedding. **arXiv preprint arXiv:2104.09864**, 2021.

SUN, C.; HUANG, L.; QIU, X. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. **arXiv preprint arXiv:1903.09588**, 2019.

TOUVRON, H. et al. Llama: Open and efficient foundation language models. **arXiv preprint arXiv:2302.13971**, 2023.

TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In: **Proceedings of the 48th annual meeting of the association for computational linguistics**. [S.l.: s.n.], 2010. p. 384–394.

VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANKHADE, M.; RAO, A. C. S.; KULKARNI, C. A survey on sentiment analysis methods, applications, and challenges. **Artificial Intelligence Review**, Springer, v. 55, n. 7, p. 5731–5780, 2022.

XU, H. et al. Double embeddings and cnn-based sequence labeling for aspect extraction. **arXiv preprint arXiv:1805.04601**, 2018.

XU, H. et al. Bert post-training for review reading comprehension and aspect-based sentiment analysis. **arXiv preprint arXiv:1904.02232**, 2019.

ZHANG, B.; SENNRICH, R. Root mean square layer normalization. **Advances in Neural Information Processing Systems**, v. 32, 2019.

ZHANG, L.; WANG, S.; LIU, B. Deep learning for sentiment analysis: A survey. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 8, n. 4, p. e1253, 2018.

ZHAO, W. X. et al. A survey of large language models. **arXiv preprint arXiv:2303.18223**, 2023.

APÊNDICE A – Detalhamento da Extração de Aspectos

A seguir um exemplo do fluxo, de ponta a ponta, para a realização da extração de aspecto utilizando o modelo que combina o BERTimbau e LoRA. Porém, esse fluxo é equivalente para todos os outros modelos avaliados neste trabalho.

O processo inicia-se com a tokenização, que é o processo de dividir um texto em unidades menores, conhecidas como *tokens*. No contexto deste trabalho, temos a seguir um exemplo retirado do conjunto de dados TV:

```
['Recomendo', ',', 'uma', 'otima', 'smart', 'é', 'sansung',
'nao', 'precisa', 'falar', 'mais', 'nada', '.']
```

Essa sequência de *tokens* é convertida em *tokens* numéricos:

```
[101, 2325, 3309, 214, 117, 230, 12512, 22278, 139, 14254, 253,
2903, 2515, 833, 229, 22280, 3804, 5961, 325, 3874, 119, 102],
```

Essa etapa é crucial para preparar os dados para processamento por modelos de aprendizado de máquina, que operam com números em vez de texto bruto. Uma característica importante a ser observada é que algumas palavras podem ser divididas em mais de um *token* durante o processo de tokenização. Isso ocorre frequentemente com palavras que não estão diretamente presentes no vocabulário do modelo ou são compostas por subunidades menores. Por exemplo, a palavra "sansung"pode ser dividida em "sans"e "ung"se estas subunidades estiverem presentes no vocabulário do modelo, enquanto "sansung"como um todo não. Esse método de tokenização em subunidades permite que o modelo lide com uma ampla variedade de palavras, incluindo neologismos e termos especializados, sem a necessidade de um vocabulário excessivamente grande.

Após a tokenização, o modelo de extração de aspectos processa os *tokens* e gera uma matriz de probabilidades. Cada linha desta matriz corresponde a um *token*, e cada coluna a uma categoria possível para classificação, que no caso deste trabalho são três: o *token* não é um aspecto (O), o *token* é o início ou meio de um aspecto (B-ASP) e o *token* é o fim de um aspecto (I-ASP). No exemplo, a matriz gerada foi:

```
[[0.66167724, 0.21418422, -2.0021877], [...], [3.1701941,
-1.3115548, -2.8593087]]
```

A matriz acima representa as probabilidades logarítmicas de cada *token* pertencer a uma das categorias de aspectos. Valores maiores indicam uma maior probabilidade. Por exemplo, um valor alto na primeira coluna indica uma alta probabilidade do *token* estar relacionado à primeira categoria de aspecto, isto é, não ser um aspecto.

A identificação de aspectos relevantes é feita aplicando a função *argmax* a cada linha da matriz de probabilidades. Essa função seleciona o índice da coluna com o valor mais alto, indicando a categoria de aspecto mais provável para cada *token*. No exemplo, o vetor resultante foi:

```
[-100, 0, -100, -100, 0, 0, 0, -100, 1, -100, 0, 0, -100, ...]
```

O valor -100 indica *tokens* que não são relevantes para a extração de aspectos (por exemplo, pontuação ou palavras de ligação), enquanto outros valores representam diferentes categorias de aspectos identificadas pelo modelo.

Convertendo novamente para uma sequência em linguagem natural, temos o seguinte aspecto extraído para o exemplo: “Recomendo, uma otima **smart** (aspecto extraído corretamente) é sansung nao precisa falar mais nada”.