UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Avaliando a utilização de Redes de Grafos com Atenção para a tarefa de Análise de Sentimentos em Nível de Aspecto em Línguas de Baixo Recurso

Gabriel Almeida Gomes

Pelotas, 2024

Gabriel Almeida Gomes

Avaliando a utilização de Redes de Grafos com Atenção para a tarefa de Análise de Sentimentos em Nível de Aspecto em Línguas de Baixo Recurso

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ulisses Brisolara Corrêa
Coorientadores: Profa. Dra. Larissa Astrogildo de Freitas
Prof. Dr. Ricardo Matsumura de Araujo

Pelotas, 2024

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

G633a Gomes, Gabriel Almeida

Avaliando a utilização de Redes de Grafos com Atenção para a tarefa de Análise de Sentimentos em Nível de Aspecto em Línguas de Baixo Recurso [recurso eletrônico] / Gabriel Almeida Gomes ; Ulisses Brisolara Corrêa, orientador ; Larissa Astrogildo de Freitas, Ricardo Matsumura de Araujo, coorientadores. — Pelotas, 2024.
81 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2024.

1. Análise de sentimentos baseada em aspectos. 2. Redes neurais em grafos. 3. Redes de grafos com atenção. 4. Processamento de linguagem natural. I. Corrêa, Ulisses Brisolara, orient. II. Freitas, Larissa Astrogildo de, coorient. III. Araujo, Ricardo Matsumura de, coorient. IV. Título.

CDD 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Gabriel Almeida Gomes

Avaliando a utilização de Redes de Grafos com Atenção para a tarefa de Análise de Sentimentos em Nível de Aspecto em Línguas de Baixo Recurso

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 18 de novembro de 2024

Banca Examinadora:

Prof. Dr. Ulisses Brisolara Corrêa (orientador)
Doutor em Computação pela Universidade Federal de Pelotas.

Prof. Dr. Marilton Sanchotene de Aguiar
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Marcelo Rita Pias
Doutor em Computação pela University College London.

Prof. Dr. Calebe Micael de Oliveira Conceição
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Aquilo que se faz por amor está sempre além do bem e do mal.
— FRIEDRICH NIETZSCHE

RESUMO

GOMES, Gabriel Almeida. **Avaliando a utilização de Redes de Grafos com Atenção para a tarefa de Análise de Sentimentos em Nível de Aspecto em Línguas de Baixo Recurso**. Orientador: Ulisses Brisolara Corrêa. 2024. 81 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2024.

Opiniões sobre produtos e serviços possuem uma importância imensa para a indústria, pois auxilia o processo de tomada de decisões. Contudo, devido ao constante crescimento de conteúdo gerado por usuários, principalmente da internet, a análise manual destes dados tornou-se inviável. Técnicas de Análise de Sentimentos são essenciais para compreender e quantificar sentimentos humanos expressos em diferentes tipos de dados, como áudio, vídeo e imagens. No entanto, uma parte substancial dessas análises é conduzida em dados textuais, que constituem o foco deste estudo.

Dentre os diversos níveis de análise, o de menor granularidade é em nível de aspecto, pois a análise é realizada em todas as partes de uma entidade em que se é possível atribuir um sentimento. A utilização de grafos para representar dados textuais permite a utilização das relações estruturais gramaticais do texto para expansão da representação, que pode trazer benefícios para a tarefa de Análise de Sentimentos baseado em Aspectos, já que além de enriquecer a representação com informações extraídas de uma análise sintática do texto, essa abordagem permite a aplicação de novas técnicas, ampliando as possibilidades de análise. Embora estudos tenham demonstrado a eficácia dessa representação para ABSA utilizando Redes Neurais em Grafos no inglês, há evidências limitadas de melhorias com o uso dessas técnicas em línguas de baixo recurso, como o português. Desenvolvemos um modelo de Rede de Atenção em Grafos para a tarefa de ABSA em português brasileiro. A abordagem proposta atinge uma acurácia balanceada de 0,74, apresentando resultados competitivos e alcançando o desempenho equivalente ao terceiro lugar na competição ABSAPT.

Palavras-chave: análise de sentimentos baseada em aspectos; redes neurais em grafos; redes de grafos com atenção; processamento de linguagem natural.

ABSTRACT

GOMES, Gabriel Almeida. **Evaluating the Use of Graph Attention Networks for Aspect-Based Sentiment Analysis in Low-Resource Languages**. Advisor: Ulisses Brisolara Corrêa. 2024. 81 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2024.

Opinions about products and services hold immense importance for the industry, as they aid in the decision-making process. However, due to the continuous growth of user-generated content, particularly on the internet, the manual analysis of such data has become unfeasible. Sentiment Analysis techniques are essential for understanding and quantifying human sentiments expressed through different types of data, including audio, video, and images. Nonetheless, a substantial part of these analyses is conducted on textual data, which is the focus of this study.

Among the various levels of analysis, aspect-level analysis is the most granular, as it examines all parts of an entity where a sentiment can be assigned. The use of graphs to represent textual data allows for the integration of the grammatical structural relationships of the text, enriching the representation and potentially benefiting the task of Aspect-Based Sentiment Analysis (ABSA). This approach not only incorporates information derived from syntactic analysis of the text but also enables the application of new techniques, expanding the possibilities of analysis. Although studies have demonstrated the effectiveness of this representation for ABSA using Graph Neural Networks in English, there is limited evidence of improvements with these techniques in low-resource languages such as Portuguese. We developed a Graph Attention Network model for the ABSA task in Brazilian Portuguese. The proposed approach achieves a balanced accuracy of 0.74, presenting competitive results and achieving a performance equivalent to the third place in the ABSAPT competition.

Keywords: aspect-based sentiment analysis; graph neural networks; graph attention networks; natural language processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABSA	*Aspect-based Sentiment Analysis*
AE	*Aspect Extraction*
AS	Análise de Sentimento
ASC	*Aspect Sentiment Classification*
BPE	*Byte Pair Encoding*
CharCNN	*Character level Convolutional Neural Network*
GAT	*Graph Attention Network*
GCN	*Graph Convolutional Networks*
GNN	*Graph Neural Network*
LSTM	*Long Short-Term Memory*
MPM	*Message-Passing Mechanism*
MLP	*MultiLayer Perceptron*
NER	*Named Entity Recognition*
PLN	Processamento de Linguagem Natural
ReLU	*Rectified Linear Unit*
RNA	Redes Neurais Artificiais
RNN	Redes Neurais Recorrentes

SUMÁRIO

1 INTRODUÇÃO

Opiniões são extremamente importantes para a tomada de decisões, seja para indivíduos ou para entidades de setores que lidem com opinião pública. Graças a elas, é possível ter como base o nível de aceitação que um grupo de pessoas possui em relação a um serviço ou produto. Porém, devido ao aumento crescente de conteúdos gerados por usuários em virtude da popularização da internet, tornou-se impraticável a análise manual de todos estes dados (LIU, 2012), necessitando assim da criação de técnicas automatizadas para transformar este tipo de dado em conhecimento.

A Análise de Sentimento (AS) é o nome dado para o campo de Processamento de Linguagem Natural (PLN) que busca criar algoritmos que focam em automatizar este processo. Algoritmos deste tipo possuem como objetivo processar conjuntos de textos que possuem opiniões dos autores expressas, e identificar a opinião, sentimento ou atitude expressada por um autor, detentor de opinião, em relação a um alvo, entidade, em um texto opinativo (MEDHAT; HASSAN; KORASHY, 2014). Técnicas deste campo podem ser aplicadas em diversos segmentos, como em análises políticas, visualização dinâmica de comentários sobre produtos online e monitoramento em redes sociais.

Para podermos criar uma técnica de AS, é preciso primeiramente decidir o nível de análise a ser realizada. Segundo Kolkur; Dantal; Mahe (2015), os três principais níveis estudados na literatura atual são: nível de documento, nível de sentença, e nível de aspecto.

A AS em nível de documento é realizada estimando o sentimento geral expresso dentro de um texto opinativo, isto é, estima uma polaridade de sentimento que representa a opinião geral dada pelo autor para uma entidade. Esta análise por fim acaba não sendo precisa, pois é possível que o autor tenha expresso mais de uma opinião sobre diferentes entidades.

Já na AS em nível de sentença estimamos a polaridade do sentimento por sentença do documento, tornando assim a análise um pouco mais precisa, porém ainda é possível que o autor tenha expresso diferentes sentimentos sobre diferentes entidades em uma única sentença.

Análise de Sentimento baseado em Aspectos (do inglês, *Aspect-based Sentiment*

Analysis - ABSA) é o nome da AS em nível de aspectos. Neste nível, analisamos cada possível alvo de opinião descrita no texto opinativo, podendo propiciar uma análise mais precisa. Devido a natureza da ABSA, é necessário dividi-la em duas sub-tarefas, a Extração de Aspectos (do inglês, *Aspect-Extraction* - AE), que busca identificar todos os termos de aspectos presentes no texto, ou seja, os alvos de opinião, e a tarefa de Classificação de Sentimento do Aspecto (do inglês, *Aspect Sentiment Classification* - ASC), que realiza a classificação da polaridade de sentimento dos aspectos extraídos.

A Figura 1 ilustra as duas sub-tarefas da ABSA. Para a sub-tarefa de AE, identificamos os aspectos presentes no texto, como “Quarto” e “Café da manhã”. Depois de extrair os aspectos, para a sub-tarefa de ASC atribuímos para cada aspecto um sentimento, como no caso de “Quarto” é atribuída polaridade Negativa, pois é desconfortável, e “Café da manhã” é atribuída polaridade Positiva, pois é ótimo.

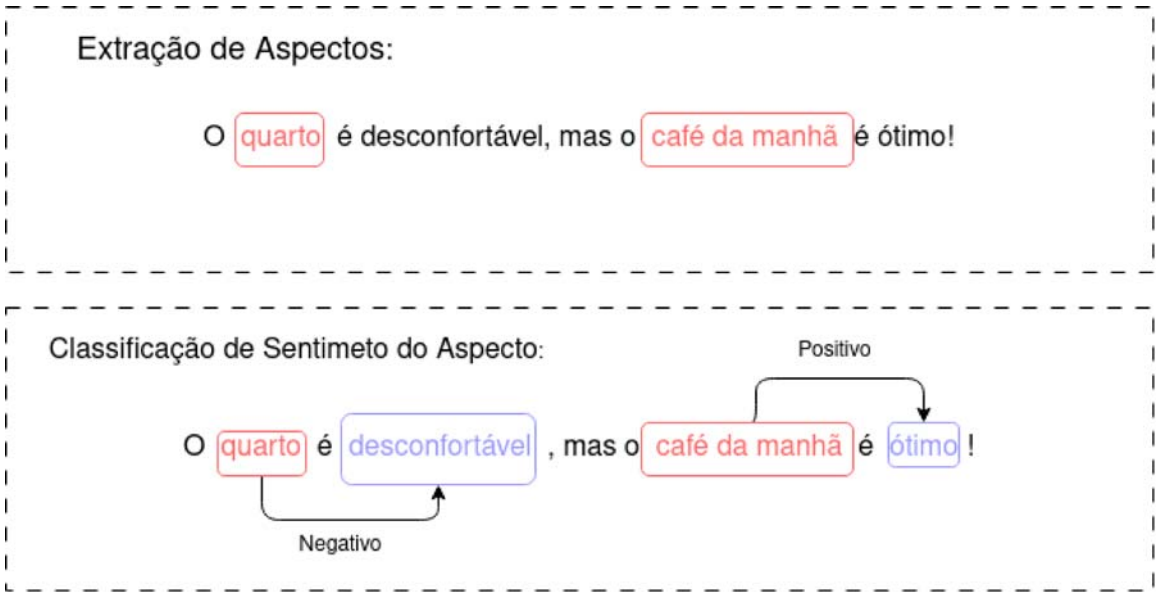

Figura 1 – Sub-tarefas derivadas da Análise de Sentimento Baseado em Aspectos. Adaptado de: Gomes et al. (2024).

.

Abordagens clássicas de ABSA se apoiam no uso da estrutura gramatical das sentenças, assim como heurísticas para definir a polaridade do sentimento. A partir de uma definição de conjunto de regras linguísticas é possível identificar termos de sentimento expressos pelo detentor de uma opinião em um texto opinativo, e analisando as relações gramaticais destas palavras é possível construir a estrutura sintática da sentença e detectar os aspectos. Para definir a polaridade de sentimento a partir dos termos de sentimento são utilizados léxicos de sentimento, que são análogos a dicionários que possuem heurísticas para a estimação de um valor numérico que busca representar a conotação positiva, negativa ou neutra de um determinado termo. O problema deste tipo de abordagem é a dependência da precisão gramatical da sentença devido ao uso de léxicos, além da necessidade da criação de um conjunto de

regras definidas manualmente (DO et al., 2019; PORIA; CAMBRIA; GELBUKH, 2016). Abordagens baseadas em léxicos e regras gramaticais tendem a perder eficácia em domínios onde não possuímos uma estrutura gramatical bem definida, como é o caso do domínio de comentários publicados/postados na internet.

Abordagens baseadas em Redes Neurais Artificiais (RNA) possuem a capacidade de contornar grande parte das limitações das abordagens baseadas em regras e léxicos de sentimento. Como os algoritmos desta área são caracterizados pela sua capacidade de se adaptar aos dados de treinamento, eles podem ser aplicados para qualquer domínio apresentado durante o treinamento, além de serem menos sensíveis a erros gramaticais pelo mesmo motivo. Além disso, este tipo de abordagem pode identificar padrões ocultos presentes no conjunto de dados, retirando a necessidade da criação manual de um conjunto de regras específico para cada uso.

Redes Neurais são uma ferramenta poderosa para ABSA, visto que existem diversos trabalhos que exploram o uso dos algoritmos para as suas sub-tarefas (ZHU et al., 2022). Porém, RNAs necessitam de uma etapa de pré-processamento a mais do que as necessárias por abordagens clássicas, que diz respeito a transformação dos dados em formato de texto para uma representação numérica, conhecido como *Embeddings*. Existem diversos tipos de métodos que realizam essa transformação, e os métodos mais utilizados consistem em criar um espaço dimensional em que cada *token* do texto assume um ponto diferente do espaço. Dessa forma, podemos definir uma certa semântica para as palavras, dependendo do lugar que elas ocupam dentro deste espaço , esse conhecimento é passado para a rede através do treinamento.

Nos últimos anos, arquiteturas do tipo *Transformers* (VASWANI et al., 2017) representam o estado da arte para diversos tipos de tarefas de PLN. Porém, essas arquiteturas normalmente possuem a característica de necessitar vasto poder computacional, visto que, os modelos gerados por essas arquiteturas possuem milhões ou até mesmo bilhões de parâmetros, além de precisarem de uma quantidade massiva de dados para serem treinados. A utilização deste tipo de modelo pode se tornar onerosa, inviabilizando seu uso para pesquisadores que não conseguem arcar com os custos, criando assim a necessidade do desenvolvimento de metodologias mais acessíveis do ponto de vista computacional.

A alta capacidade preditiva de modelos *Transformers*, deve-se em grande parte, ao mecanismo de *Self-Attention*. Este mecanismo busca adicionar uma ideia de "contexto" a cada *token*, onde parte das informações dos *tokens* adjacentes são adicionadas à representação vetorial do *token* alvo, dependendo da relação entre *tokens*. O mecanismo de *Self-Attention* proporciona uma melhoria significativa em termos de capacidade preditiva para os modelos, porém impõe uma penalização computacional, uma vez que, requer multiplicações matriciais para cada *token* com todos demais, inclusive aqueles que possuem conexões fracas, resultando em um alto custo compu-

tacional.

Uma forma de reduzir o custo do mecanismo de *Self-Attention* empregado em modelos *Transformers* é diminuindo o escopo da codificação dos vetores de atenção para cada *token*, e uma forma de abordar isso é por meio de grafos. Grafos são estruturas de representação de dados que representam entidades e as relações entre elas. É possível transformar dados em formato textual para uma representação em grafo, assumindo cada palavra como um nodo e suas relações sintáticas como arestas. Redes Neurais em Grafos (do inglês, *Graph Neural Networks* - GNN) são Redes Neurais capazes de utilizar Grafos como entrada, e são uteis para fazermos uso da conectividade entre nodos, característica dos dados em grafos. GNNs utilizam um mecanismo de Passagem de Mensagem (do inglês, *Message-Passing Mechanism* - MPM) para processar as conexões entre nodos. As Redes em Grafos com Atenção (do inglês, *Graph Attention Networks* - GAT) (VELIČKOVIĆ et al., 2018) são GNNs que aplicam um mecanismo de *Self-Attention* como função de agregação para o MPM, onde a partir das conexões entre nodos, são calculados valores de atenção de um nodo com seus adjacentes.

Acreditamos que, como os grafos gerados para texto são esparsos, a utilização de GATs para a tarefa de ABSA podem trazer os benefícios do mecanismo de *Self-Attention* para Redes Neurais, com um custo de computação inferior a modelos *Transformers*. Isso ocorre pois faz uso das conexões chave baseadas nas relações sintáticas da sentença, tornando o modelo mais rápido, necessitando de menos *hardware* e obtendo resultados comparáveis em termos de acurácia.

Após realizar uma busca na literatura, encontramos alguns trabalhos que exploram o uso de GATs para a tarefa de ABSA para o idioma inglês, porém, não foram encontrados trabalhos que abordassem o uso deste tipo de arquitetura para o idioma português.

Dito isto, nosso objetivo geral é avaliar modelos GATs para a sub-tarefa de ASC no idioma português brasileiro.

O trabalho está estruturado em seis capítulos principais. O Capítulo 1 – Introdução apresenta, de forma sucinta, os conceitos fundamentais do estudo, a problemática abordada e a hipótese formulada para sua resolução. O Capítulo 2 – Referencial Teórico explora detalhadamente as áreas relacionadas ao tema, fornecendo definições formais dos conceitos e tecnologias centrais utilizadas. O Capítulo 3 – Trabalhos Relacionados discute abordagens correlatas ao presente estudo, destacando diferentes perspectivas relevantes para o contexto investigado. O Capítulo 4 – Metodologia descreve os procedimentos experimentais adotados, qual a metodologia definida para a resolução do problema, assim como os critérios utilizados para a avaliação dos resultados. O Capítulo 5 – Discussão dos Resultados oferece uma análise crítica dos achados, comparando o desempenho da abordagem proposta com métodos previamente

estabelecidos. Por fim, o Capítulo 6 – Conclusão sintetiza as principais contribuições do trabalho, destacando as informações mais relevantes obtidas ao longo do estudo.

2 REFERENCIAL TEÓRICO

Neste capítulo, abordamos os principais conceitos teóricos necessários para o desenvolvimento deste trabalho, assim como o histórico das técnicas utilizadas.

2.1 Análise de Sentimento

Análise de Sentimento é o nome denominado à área que engloba algoritmos que buscam estimar o sentimento ou opinião de um autor de uma mensagem opinativa, em relação a uma entidade. Entidades podem ser tópicos, indivíduos ou eventos (MEDHAT; HASSAN; KORASHY, 2014). Os trabalhos desenvolvidos geralmente se concentram na análise realizada no domínio textual, porém a área não se limita apenas a texto. Há diversos estudos aplicando metodologia de identificação de polaridades de sentimento para imagens, vídeos, áudio e até mesmo mais de um tipo de dado (ORTIS; FARINELLA; BATTIATO, 2019) (LUO; XU; CHEN, 2019) (PORIA et al., 2016).

A AS em formato textual geralmente é realizada como um problema de classificação. Onde a entrada do algoritmo consiste em um texto opinativo, que é pré-processado utilizando técnicas para adequar os dados ao modelo preditivo e ao nível da análise, e é então utilizado como entrada para o modelo preditivo que estimará a polaridade do sentimento expressa no texto ou em parte dele.

Normalmente, AS é realizada em termos de polaridade de sentimento, isto é, positivo, neutro ou negativo. Isto se deve ao fato de sentimentos serem subjetivos, algo que provoca tristeza em alguém pode causar raiva em outrem. A polaridade do sentimento tende a ser bem menos subjetiva do que o sentimento em si, além disso, resolver uma classificação com uma menor quantidade de rótulos tende a ser mais simples do que com uma maior quantidade. É preciso também ressaltar que não temos um consenso de sentimentos universais, e isto pode dificultar a definição de um processo de anotação para dados deste tipo.

É possível dividir a AS em diversos níveis (KOLKUR; DANTAL; MAHE, 2015), porém existem três níveis em que se concentram a maior parte das abordagens realizadas na literatura, são eles:

- **Nível de Documento**. A Análise de Sentimento em Nível de Documento classificam a opinião geral expressa pelo autor em um documento opinativo. Desta forma, métodos deste tipo assumem a existência de apenas um sentimento expresso sobre apenas uma entidade por documento de texto (BEHDENNA; BARIGOU; BELALEM, 2018). Devido ao fato de que podemos expressar mais de uma opinião e ainda sobre diferentes entidades em um único documento de texto, a análise neste nível carece de precisão em comparação aos demais níveis.

- **Nível de Sentença**. Já trabalhos de Análise de Sentimento em Nível de Sentença classificam a opinião geral expressa para cada sentença de um documento opinativo. Neste nível possuímos duas sub-tarefas, a tarefa de classificação de subjetividade da sentença e a tarefa de classificação de sentimento da sentença (JAGTAP; PAWAR, 2013). A tarefa de classificação de subjetividade da sentença envolve determinar se uma sentença transmite uma informação factual ou expressa uma opinião do autor (BEHDENNA; BARIGOU; BELALEM, 2018). Este nível realiza uma análise mais precisa quando comparamos com o nível mencionado anteriormente, porém ainda corremos o risco de não extrairmos toda a informação presente no texto, já que ainda é possível que o autor emita opiniões variadas a respeito de diferentes entidades em uma única sentença.

- **Nível de Aspecto**. A Análise de Sentimento em Nível de Aspecto se baseia na ideia de que toda a opinião se baseia em um sentimento e um alvo de opinião. Portanto, realiza-se a análise em cada possível alvo de opinião, que pode ser um aspecto ou uma entidade (LIU, 2012). A natureza deste nível de análise implica na criação de duas sub-tarefas para realizá-la, a tarefa de Extração de Aspectos, que consiste na identificação dos termos de aspectos presentes no documento, e a tarefa de Classificação de Sentimento do Aspecto, que realiza a classificação do sentimento expresso para cada aspecto identificado. A análise neste nível é a mais precisa, devido ao fato de ser realizada diretamente em cada alvo de opinião, porém tende a ser mais custosa computacionalmente pelo mesmo motivo.

Todos os níveis de AS possuem suas características e são melhor adequados a determinados tipos de situações, talvez a AS em nível de documento possa ser mais útil em um cenário de análise de opinião geral de relações públicas de uma personalidade, e ABSA pode ser melhor aplicado para um contexto de análise de *reviews* de produtos ou serviços, pois torna possível identificar quais são as partes daquela entidade que possuem falhas ou são elogiadas pelo público. Fato é, a escolha do nível de análise necessita ter o contexto da aplicação em mente.

Neste trabalho, realizaremos a análise em nível de aspecto. A seguir, apresentaremos a tarefa de forma mais detalhada.

ABSA é o nome dado à área que compreende os trabalhos realizados de AS em nível de aspectos. Para podermos abordar mais sobre ABSA, precisamos primeiro definir formalmente o que é opinião.

Segundo Liu (2012), é possível definir formalmente o conceito de opinião como uma quádrupla (*g*, *s*, *h*, *t*), onde cada item é:

- *g* - É o alvo da opinião.
- *s* - O sentimento expresso em relação ao alvo da opinião.
- *h* - É o detentor da opinião.
- *t* - É o tempo em que a opinião foi expressa.

Por mais que esta seja a definição formal de opinião, é difícil aplicá-la na prática dependendo do domínio a ser abordado. Muitas vezes nos referimos a um alvo implícito, o que acaba dificultando a utilização da definição formal. Por exemplo, assumindo um contexto de *reviews* de hotéis, se considerarmos a sentença “Os quartos são maravilhosos”, o autor quer pontuar que os quartos do hotel X são maravilhosos, mesmo não sendo explícito na mensagem, o que torna o hotel X o alvo em questão, também denominado por Liu (2012) como entidade.

Segundo o autor, podemos definir entidade como um produto, serviço, tópico, pessoa, organização ou evento, e pode ser descrito como um conjunto hierárquico de partes e atributos. Aspectos nada mais são do que as partes e atributos de uma entidade, então para o exemplo “Os quartos são maravilhosos”, o aspecto presente no exemplo é “quartos”, justamente por ser uma parte da entidade hotel X, que é passível de receber uma opinião.

Considerando que o alvo da opinião poder ser dividido entre uma entidade e seus aspectos, podemos transformar a quadrupla da definição formal de opinião para uma quíntupla (e_i, a_{ij}, s_{ijkl}, h_k, t_l):

- e_i - É o nome da entidade.
- a_{ij} - É um aspecto de e_i.
- s_{ijkl} - O sentimento expresso em relação ao aspecto a_{ij} da entidade e_i.
- h_k - É o detentor da opinião.
- t_l - É o tempo em que a opinião foi expressa pelo detentor da opinião h_k.

Indo além da definição postulada por Liu (2012), podemos destrinchar ainda mais os elementos de aspecto e sentimento. De acordo com (ZHANG et al., 2023), na prática, podemos dividir o alvo da opinião como dois elementos: a categoria do aspecto

e o termo do aspecto. A lógica desta divisão deriva do fato de que aspectos possuem uma hierarquia, então, desta forma, podemos assumir como categoria de um aspecto um outro aspecto de hierarquia mais alta. Como exemplo, utilizando a frase “Os quartos são maravilhosos”, podemos assumir que o termo do aspecto é “quartos” enquanto a categoria do aspecto poderia ser algo como “instalações” ou o próprio “hotel”. As categorias dos aspectos devem ser definidas durante o processo de anotação do conjunto de dados, pois elas podem não ser muito intuitivas e claras para os anotadores, podendo assim ser suscetíveis à interpretação de cada anotador.

Também é possível dividir a opinião em **termo da opinião** e **polaridade de sentimento**, na mesma frase utilizada anteriormente, podemos assumir que o termo da opinião é “maravilhosos” e a polaridade da opinião é “Positiva”.

A extração de algum conjunto destes quatro elementos é onde se concentra a maior parte das pesquisas de ABSA modernas. Cada trabalho foca em identificar pelo menos um destes elementos mencionados, ou pares, trios ou até mesmo os quatro elementos ao mesmo tempo, mas vale comentar que a complexidade do algoritmo utilizado é proporcional à quantidade de elementos extraídos.

Neste trabalho, focamos apenas na tarefa de Classificação de Polaridade de Sentimento do Aspecto, o motivo principal se dá pelos desafios de avaliar métodos de extração dos outros elementos, assim como a dificuldade de encontrar conjuntos de dados que vão além da extração do termo do aspecto e a polaridade de sentimento.

Grande parte das abordagens mais clássicas para a solução das tarefas de ABSA consistem na utilização da estrutura gramatical das sentenças para a criação de um conjunto de regras e heurísticas para definir a polaridade de sentimento ou extração de aspecto.

O *pipeline* de uma aplicação que segue essa ideia funciona da seguinte forma: Normalmente, é utilizada uma ontologia que possui os aspectos mais comuns de determinado domínio onde a metodologia será aplicada, assim como uma possível hierarquia entre os aspectos anotados. Com os aspectos alvos já extraídos, é possível identificar a presença deles na sentença a ser analisada, também é possível assumir que os substantivos possam ser candidatos a aspectos, embora esse tipo de abordagem possa causar a extração de muitos aspectos falsos.

Utilizando métodos de análise sintática como *Part-of-Speech taggers* (LI et al., 2024) ou *Dependency Parsers* (AZIZ; BAKAR; YAAKUB, 2024)(RANA; CHEAH, 2016), é possível identificar a estrutura gramatical da sentença e os termos de opinião ligados ao termo do aspecto. A partir disto, o uso de léxicos de sentimento (SEBASTIANI; ESULI, 2006) pode auxiliar para a quantificação do sentimento que o termo da opinião carrega, possibilitando assim a estimativa de polaridade de sentimento para aquele aspecto.

O ponto mais complexo de se lidar quando utilizamos metodologias baseadas na

estrutura gramatical e léxicos é a inconsistência gramatical, além da generalização de domínio. Dependendo do domínio onde a análise é realizada, pode ser comum a escrita informal de certas palavras, como abreviações e contrações, além de estar sujeita também a falhas de escrita. Se utilizarmos como exemplo novamente um contexto de *reviews* de hotel, como a maior parte das opiniões são dadas na internet, onde é comum a informalidade e falta de preocupação com a escrita correta, diversos comentários são realizados com erros gramaticais. Além disso, algumas palavras podem ter diferentes significados dependendo do contexto, ou pelo menos valores de sentimento diferentes. Em outras palavras, o maior problema neste tipo de metodologia diz respeito a falta de flexibilidade da língua, necessitando a utilização de outros algoritmos para contornar essas características dependendo do contexto.

Em inglês, diversos trabalhos abordam a utilização de metodologias baseadas na estrutura gramatical e léxicos para ABSA (KHARDE; SONAWANE et al., 2016) (KIRITCHENKO; ZHU; MOHAMMAD, 2014). Para o português, também possuímos referências sobre a utilização dessas ferramentas e desse tipo de metodologia adaptadas para a língua (MACHADO; PARDO; RUIZ, 2018) (FREITAS, 2015) .

2.2 Redes Neurais Artificiais

Nesta seção abordaremos as RNAs, desde os elementos fundamentais de cada rede até as suas variações que utilizamos para a criação deste trabalho.

2.2.1 Perceptron

RNAs são compostas por neurônios artificiais, também conhecidos como perceptrons (ROSENBLATT, 1958). Perceptrons são funções matemáticas simples que buscam aprender como interpretar sinais de entrada. O funcionamento se dá pelo seguinte algoritmo: Dado um conjunto de entradas de tamanho fixo, possuímos um conjunto de parâmetros denominados de pesos com o mesmo tamanho, cada parâmetro então é multiplicado pela respectiva entrada e os resultados são todos somados para um valor escalar, podendo ainda acrescentar outro valor escalar como viés. Após isso, o resultado da soma final é dado como entrada para uma função de ativação, que determinará se aquele neurônio será ativado ou não, produzindo o valor de saída do perceptron. Existem diversos tipos de funções de ativação que são melhores utilizadas em determinados casos. Porém, um dos motivos de utilizarmos funções de ativação é que podemos trazer um elemento de não linearidade ao modelo, podemos assim aumentar o escopo de possíveis representações usando um neurônio.

Na Figura 2 é possível identificar uma representação visual de um neurônio perceptron único, onde dada um vetor de entradas X, multiplicado por um vetor de pesos W e somando todos os valores resultantes, alimentamos uma função de ativação ϕ

para obter uma saída *y*. Nesta representação, temos a presença de um *threshold* θ, que possui a mesma utilidade de um valor de viés, permitindo alterar o valor inicial de ativação do neurônio, sendo apenas o valor negativo do valor de viés.

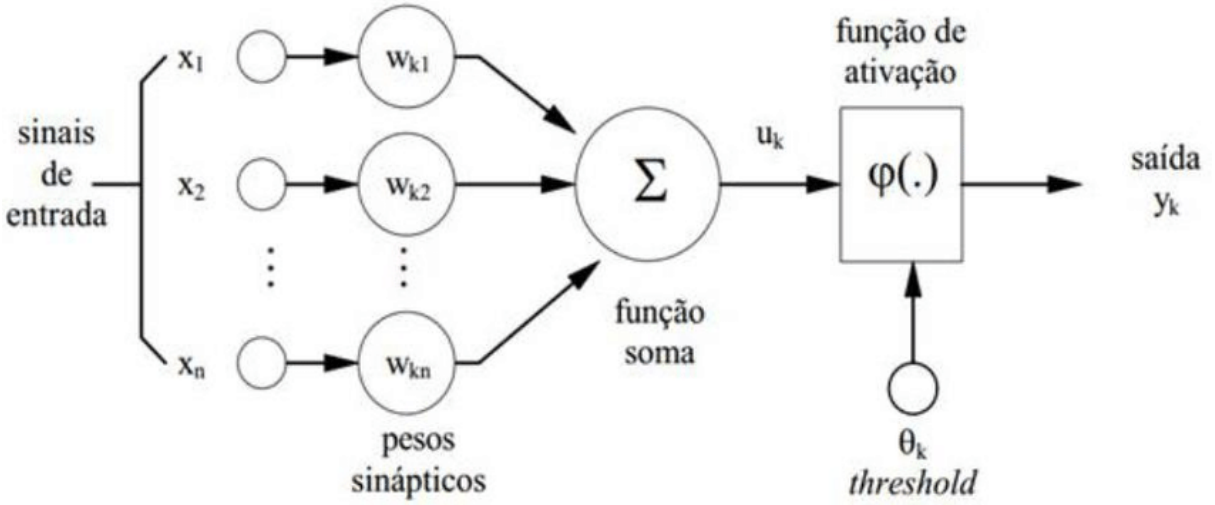

Figura 2 – Representação visual do Neurônio Artificial. Fonte: Assis et al. (2016).

Para podermos adaptar um neurônio artificial para solucionar um problema de classificação binária ou de regressão, precisamos encontrar o valor ideal dos pesos que processam os sinais de entrada, o nome dado a este processo é treinamento. No treinamento de um neurônio, nós selecionamos um conjunto de dados destinado apenas para isto, e outro para avaliar o desempenho do neurônio treinado. Com o conjunto de treinamento, aplicamos os dados e comparamos o valor real com o valor de saída gerado pelo modelo por meio de uma função de perda. A partir do resultado da função, podemos ajustar os parâmetros caso o valor de saída seja insatisfatório, por meio de um hiper-parâmetro chamado de *learning rate*, ou podemos não ajustar o modelo caso o modelo tenha predito corretamente.

Sendo baseado em como os neurônios funcionam, este algoritmo trouxe como novidade a alta adaptabilidade da arquitetura para novos dados, além de poderem ser utilizados em larga escala para criar representações ainda mais complexas, por meio das RNAs.

2.2.2 Redes Neurais Artificiais

Por mais útil que o algoritmo perceptron seja por si só, um neurônio possui uma capacidade muito limitada para predição. Assumindo uma tarefa de classificação, um neurônio pode apenas trabalhar com predições binárias, além de não ter uma capacidade preditiva em termos de processamento dos sinais de entrada forte o suficiente para problemas complexos.

Redes de Perceptron Multicamadas (MLP - Do Inglês *Multilayer Perceptron*) são redes de perceptrons conectadas em diversas camadas, com o objetivo de aumentar

a representatividade de conhecimento que um único perceptron possui. Podemos utilizar vários perceptrons conectados em que cada neurônio é responsável por processar parte do sinal e enviar a informação para a próxima camada (RUMELHART; HINTON; WILLIAMS, 1986). Este tipo de algoritmo revolucionou a área de Inteligência Artificial, devido a melhora significativa para problemas mais complexos que a técnica proporcionou.

MLPs podem possuir diversas camadas, porém normalmente seguem uma ideia principal de três tipos de camadas principais: a camada de entrada, camadas ocultas e a camada de saída (TAUD; MAS, 2018), e o nome do processo realizado da MLP receber a entrada e predizer a saída é *feedforward*. A camada de entrada é responsável por receber os dados de entrada, a saída de cada neurônio desta camada é então dada como entrada para cada neurônio da primeira camada oculta do modelo. Uma arquitetura MLP pode ter diversas camadas ocultas, porém apenas uma camada de entrada e uma de saída. As camadas ocultas servem para modelar o conhecimento oculto da rede, e também segue o mesmo processo de todos os neurônios da camada anterior servirem como entrada para todos os neurônios da próxima camada. Após o processamento de todas as camadas ocultas, os resultados dos neurônios da ultima camada oculta é ligado a camada de saída, onde a quantidade de neurônios da camada de saída normalmente é o mesmo que a quantidade de classes presentes nos dados, já que cada neurônio da camada de saída é responsável pela estimativa de uma classe. Para um problema de regressão, apenas utilizamos um neurônio na camada de saída.

Na Figura 3 podemos observar uma imagem ilustrativa de uma MLP de uma camada oculta para uma classificação binária, com representações do processo de *feedforward*.

A utilização de uma arquitetura de perceptron multicamadas até pode parecer intuitiva de certa forma, porém não é tão intuitivo a forma em que os neurônios corrigem seus pesos. É importante para um modelo MLP que cada perceptron atualize seus pesos individualmente e com valores diferentes, com base na importância que aquele neurônio teve para a estimação final da rede. Para isso, precisamos de uma forma de identificar qual o papel que cada neurônio desempenhou para a estimativa final e corrigirmos com base nisto, e a forma utilizada para isto é por meio do mecanismo de *back-propagation* (RUMELHART; HINTON; WILLIAMS, 1986) que é capaz de atualizar cada peso com base na sua influência com o resultado final.

A utilização de RNAs nos permitiu utilizar a Inteligência Artificial para um escopo muito maior de problemas, que necessitavam de superfícies de decisão mais complexas, tornando assim possível o uso deste tipo de algoritmo em tarefas como visão computacional e processamento de linguagem natural.

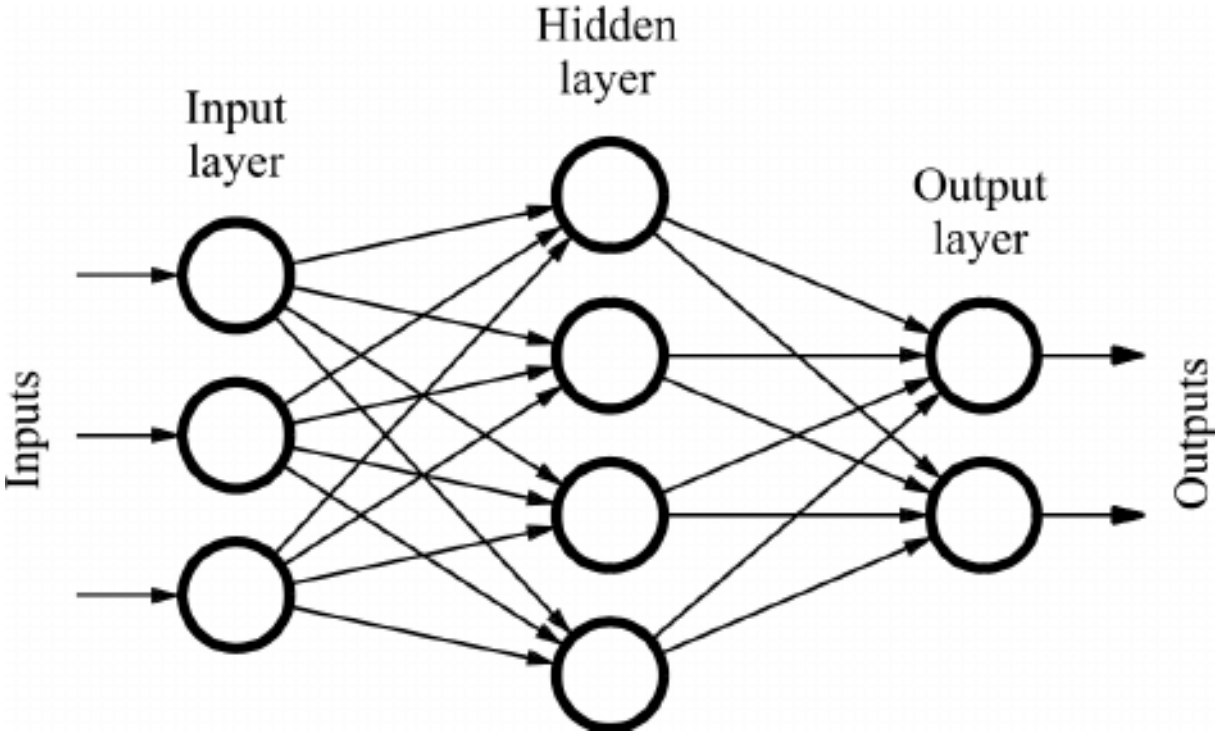

Figura 3 – Representação visual do processo de *FeedForward* em uma Rede Neural Artificial. Fonte: Quiza; Davim (2011)

2.2.3 Transformers

Nos últimos anos, arquiteturas Transformers (VASWANI et al., 2017) têm demonstrado resultados estado-da-arte para diversas tarefas de PLN. Transformers foram criados com o objetivo de substituir a recorrência empregada em arquiteturas de redes neurais recorrentes (RNN) utilizando módulos de *self-attention* para a modelagem da linguagem. Além disso, a forma em que o algoritmo foi criado também permite a paralelização de muitos componentes internos, o que acelera o treinamento.

A família de arquiteturas Transformers possui algumas variantes, porém o tipo de modelo original foi empregado para tarefas de sequência para sequência (do inglês, Sequence-to-Sequence - seq2seq), isto é, dado uma sequência de entrada, o modelo tem como saída outra sequência (SUTSKEVER, 2014). Normalmente, modelos para tarefas *seq2seq* são compostos por arquiteturas *encoder* e *decoder*, como é o caso do modelo original Transformer. Devido a sua eficácia, variantes de modelos *Encoder-only* e *Decoder-only* surgiram baseados na arquitetura Transformer original, dando origem a três famílias principais de modelos Transformer: *Encoder-only*, *Decoder-only* e *seq2seq*.

Na Figura 4, podemos observar uma arquitetura de modelos *seq2seq*, onde no bloco a esquerda é apresentado uma ilustração visual da arquitetura *Encoder* do Transformer, e a direita a arquitetura *Decoder*.

Nas próximas subseções, apresentaremos os blocos de *Encoder* e *Decoder* das arquiteturas Transformer, assim como as utilidades de uso de arquiteturas que pos-

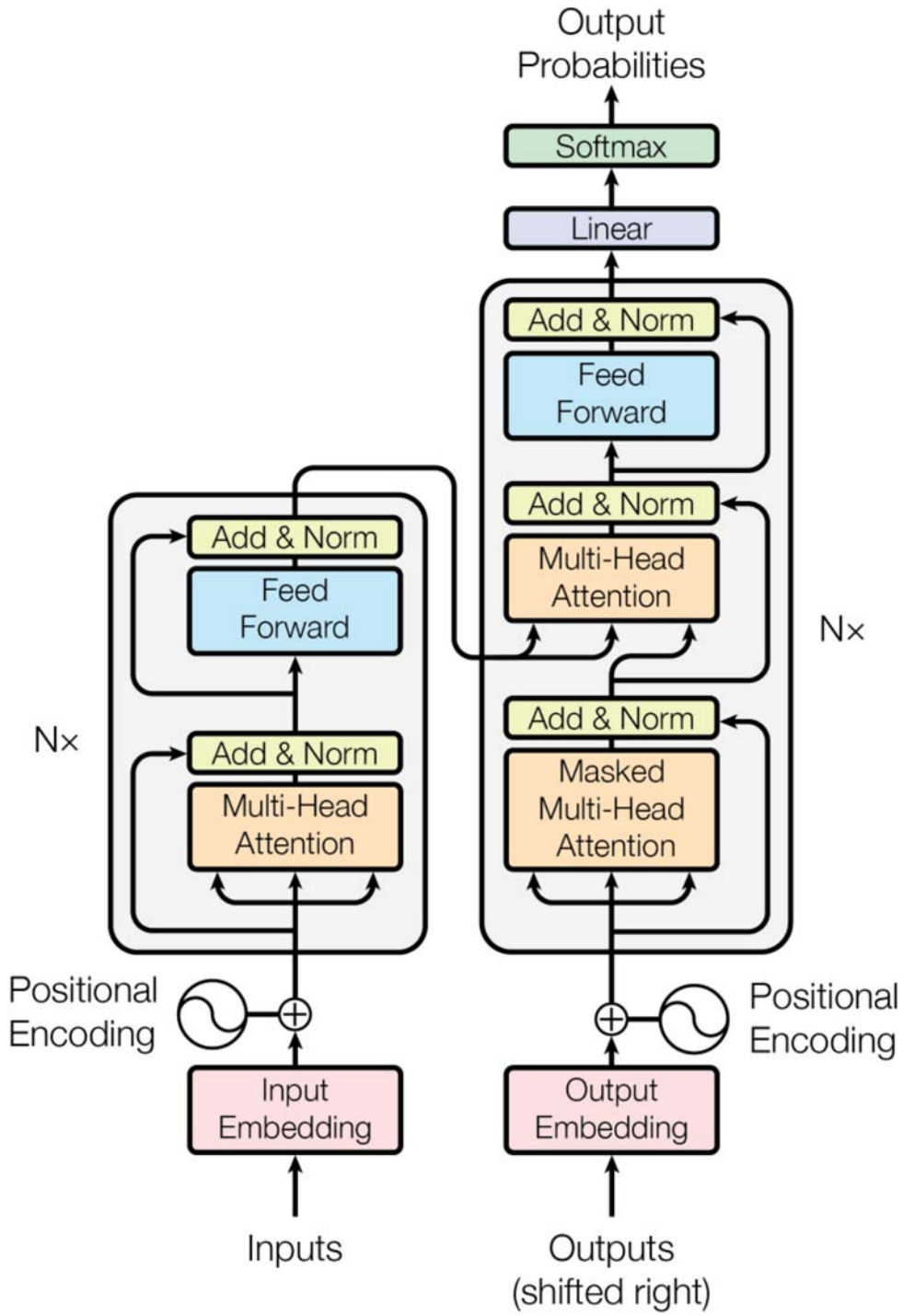

Figura 4 – Representação visual de um modelo Transformer Encoder-Decoder. Fonte: Vaswani et al. (2017)

suem somente um dos blocos.

2.2.3.1 Encoder

O bloco *Encoder* possui a responsabilidade de "interpretar" a sequência de entrada. Este bloco busca gerar *Embeddings* de representação que levem o contexto da entrada para cada *token*. A inferência se dá da seguinte forma: Uma sentença

de entrada é alimentada para um tokenizador, que é um método especifico para cada modelo que é responsável por quebrar os documentos ou sentenças em *tokens* (ALI et al., 2023), que são unidades minímas do texto que geram um *embedding*. Os *tokens* podem assumir formatos de palavras (BENGIO; DUCHARME; VINCENT, 2000), subpalavras(WANG; CHO; GU, 2020) ou caracteres (GAO et al., 2020).

Após obtido os *tokens*, eles são dados como entrada para uma camada de criação de *Embeddings* iniciais. Para fazer uso da ordem de palavras em uma sentença, modelos Transformers aplicam uma codificação posicional para cada *token*, isto é, aplicam uma função que adicionará um valor depedente da posição daquele *token* na sentença. Existem diversos tipos de codificadores posicionais, mas o utilizado no trabalho original de Vaswani et al. (2017) utiliza as equações 1 e 2.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{1}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2}$$

Onde *pos* é a posição do *token*, *i* é a dimensão e d_{model} é as dimensões do modelo de *embeddings*. Dessa forma, é possível alterar os valores do vetor de *embeddings* para adicionar uma informação referente a posição daquele *token* em uma sentença.

Após adicionado as informações referentes a posição do *token*, os *embeddings* passam pela primeira camada de *Multi-Head Attention*. Uma camada *Multi-Head Attention* é composta por diversas subcamadas de algum mecanismo de *Self-Attention* para os dados de entrada, no caso do trabalho de Vaswani et al. (2017), o mecanismo escolhido foi o produto escalar normalizado, uma adaptação do mecanismo de *Self-Attention* de produto escalar porém adicionando um fator de escala para lidar com problemas de *vanish gradients*. *Vanish gradients* é um comportamento que algumas arquiteturas possuem em que os gradientes se tornam tão pequenos a medida que o erro é propagado que a velocidade de treinamento pode ser reduzida ou até mesmo parada (BASODI et al., 2020).

O objetivo do mecanismo de *Self-Attention* é inserir dados do contexto da sentença para cada representação individual de *token*, assim como identificar quais destes *tokens* de contexto devem ser adicionados a representação.

Abordando o funcionamento do mecanismo, inicialmente precisamos obter um conjunto de vetores de *Queries*, *Keys* e *Values* para cada *token* da sentença. Isso se dá por meio de camadas lineares que se alimentam dos *embeddings* gerados pelas camadas anteriores e formam os três vetores mencionados anteriormente. Cada vetor possui um papel fundamental para o funcionamento do mecanismo, o vetor de *Query* do *token* a ser codificado é usado para obter o produto vetorial com os vetores de *Keys*

de todos os outros *tokens* dentro de uma janela de contexto. Os resultados passam por uma divisão pelo tamanho da dimensão dos *embeddings* do vetor de *Keys*, com o objetivo de minimizar o problema de *vanishing gradients*. É aplicado a função de *Softmax* nos dados gerados, criando assim um vetor de pesos normalizado de importância de quais *tokens* devem ser codificados junto ao *token* alvo. Multiplicando esse vetor pelos vetores de *Value*, temos os valores de atenção finais da *Self-Attention Head*.

Os autores do artigo argumentam que é interessante utilizar múltiplos vetores de *Queries*, *Keys* e *Values* que são formados de diferentes projeções lineares para melhorar o desempenho do modelo final. Cada conjunto de *Queries*, *Keys* e *Values* podem ser utilizados em paralelo, os resultados são concatenados e projetados por outra camada linear, resultando nos valores finais.

A definição da função de *Self-Attention* se dá pela equação 3, onde Q são os valores de *Query*, K os valores de *Keys*, V os valores de *Values* e d_k a dimensão de *Keys*.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

A Figura 5 representa o processo documentado anteriormente, porém com uma camada adicional e opcional de máscara, que é utilizada no mecanismo de modelos *Decoder*. A Figura 6 representa visualmente o processo de *Multi-Head Self-Attention*, onde *h* representa a quantidade de “cabeças” de atenção.

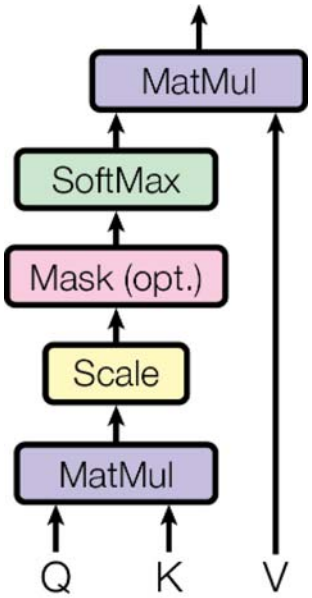

Figura 5 – Representação visual do mecanismo de atenção de produto escalar escalado. Fonte: Vaswani et al. (2017).

Após obtidos os valores de atenção finais, é aplicado uma normalização e adição do sinal anterior à camada de *Self-Attention*, com o objetivo de minimizar problemas de *vanishing gradients*. As representações são então alimentadas para uma rede neural

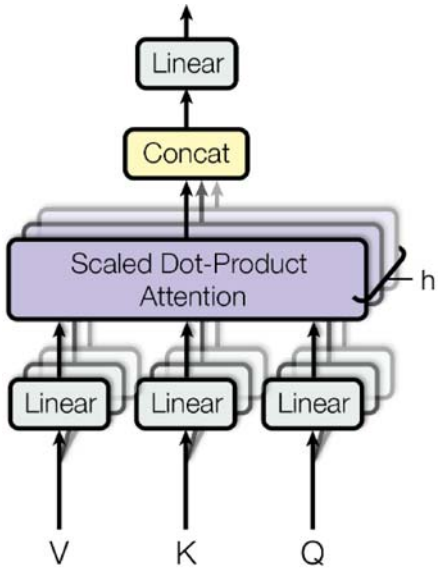

Figura 6 – Representação visual do processo de *Multi-Head Self-Attention*. Fonte: Vaswani et al. (2017).

que aprenderá mais padrões ocultos, e depois novamente passando por uma camada de normalização e adição. É possível assumir estes últimos passos como blocos, e empilhá-los N vezes, no artigo original, os autores empilharam 6 destes blocos. Aqui é onde se difere os modelos *Encoder-only* e *Encoder-Decoder*. Modelos *Encoder-only* são utilizados para a criação de *embeddings*, e possuem foco para tarefas de classificação ou interpretação do texto de entrada, logo, a saída esperadas para estes modelos são os valores obtidos neste ponto. Exemplos de modelos *Encoder-only* são os modelos BERT (DEVLIN et al., 2018) e RoBERTa (LIU et al., 2019).

2.2.3.2 Decoder

O bloco de modelo *Decoder* possui a finalidade de gerar o texto de saída. Em um modelo *Encoder-Decoder*, o modelo *Decoder* utiliza os dados de entrada codificados pelo bloco de *Encoder* para a geração de texto analisá-los como contexto geral.

Um bloco *Decoder* realiza a tarefa de geração de próximo *token*. Isto é, na primeira iteração do algoritmo, o *token* de entrada é um *token* especial que marca um início de sentença, e a saída do modelo deverá ser as probabilidades de qual será o próximo *token*. Após geradas, é escolhido o *token* sucessor e agregado ao *token* de entrada, formando uma janela de contexto que será utilizada para prever o próximo *token* sucessivamente, terminando as iterações quando for gerado um *token* especial simbolizando o final de sentença, ou por meio de uma interrupção manual.

A arquitetura de um modelo *Decoder* possui os níveis de tokenização e codificação posicional da mesma forma do que o modelo *Encoder*. Porém, as próximas camadas

são compostas por dois tipos diferentes de *Self-Attention*.

O primeiro mecanismo de *Multi-Head Self-Attention* possui semelhanças com o mecanismo empregado por modelos *Encoder*, porém a diferença fundamental é que a janela de contexto leva como informação para a análise de atenção apenas os *tokens* à esquerda do *token* predito. Como o intuito de um modelo *Decoder* é a geração de texto, é preciso que ele não tenha acesso aos *tokens* posteriores, pois teria o mesmo efeito de um vazamento de dados. Após a aplicação deste mecanismo, passamos por outra camada de adição e normalização.

Já o segundo mecanismo de *Multi-Head Self-Attention* liga as saídas do modelo Encoder com os *Embeddings* gerados pelo modelo *Decoder*, aplicando o mesmo processo para obter os valores de atenção com os dados de entrada como um contexto para a geração do novo *token*. Após o segundo mecanismo, outra adição e normalização é efetuada.

Posteriormente a aplicação de cada camada, passamos por outro modelo *Feed Forward* e outra camada de normalização e adição. Estas 6 camadas compõem um bloco *Decoder* e podem ser empilhadas N vezes, no artigo original, os autores empilharam 6 camadas.

A saída é então obtida com o uso de uma camada linear e a aplicação de uma função *Softmax* para gerar as probabilidades finais.

Modelos *Decoder-only* possuem bastante visibilidade nos dias atuais. A diferença destes modelos se dá pela não aplicação da camada de *Self-Attention* que une os dados de saída do modelo *Encoder*, já que não utilizam um modelo deste tipo. A tarefa principal empregada por modelos desta família é geração de texto, enquanto modelos *seq2seq* são mais indicados para tarefas como geração condicional, tradução de texto, sumarização e afins. Exemplos de modelos *Decoder-only* são: GPT 4, empregado na tecnologia ChatGPT (OPENAI et al., 2024), Llama (TOUVRON et al., 2023) e Falcon (ALMAZROUEI et al., 2023). Já exemplos de modelos *seq2seq* são os modelos T5 (RAFFEL et al., 2023) e BART (LEWIS et al., 2019).

2.2.3.3 Pré Treinamento

Como mencionado anteriormente, modelos Transformers são utilizados por diversas metodologias estado da arte para tarefas de PLN. Grande parte destes modelos não são eficazes em termos de acurácia apenas devido ao uso da arquitetura, mas sim devido ao conjunto arquitetura e pré treinamento.

O pré treinamento de um modelo baseado em Transformers consiste na tarefa de modelagem de língua. Existem diferentes tarefas destinadas a modelagem línguistica, como no caso da tarefa de *Masked Language Modeling* empregada na metodologia de pré treino do modelo BERT (DEVLIN et al., 2018), que é focada para a modelagem de língua para modelos *encoder*, pois a tarefa consiste na mascaragem de tokens

aleatórios e o modelo prevê qual será o valor deste *token* baseado nos *tokens* anteriores e posteriores. E para modelos *decoder*, é utilizado a tarefa de *Causal Language Modeling*, que consiste na predição do próximo *token*, apenas com base nos *tokens* anteriores.

Geralmente, essas etapas envolvem o treinamento do modelo utilizando vastos corpora na linguagem-alvo para a qual o modelo será aplicado. Alguns exemplos de modelos baseados em Transformers pré treinados em português são os modelos BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), Sabiá (PIRES et al., 2023) um modelo *decoder* baseado na arquitetura Llama e PTT5 (CARMO et al., 2020) baseado nos modelos T5.

2.3 Grafos

Para podermos abordar os conceitos de Redes Neurais de Grafos, primeiro precisamos definir formalmente a estrutura.

Grafos são estruturas matemáticas em conjuntos de elementos com conexões binárias entre si (XU, 2013). Dessa forma, podemos definir grafos como uma tripla (V, E, ψ) onde:

- V - É um conjunto de vértices (ou nodos) não vazio.
- E - É um conjunto de arestas.
- ψ - É uma função que dita quais arestas ligam pares de vértices.

A Figura 7 ilustra como um grafo é representado visualmente, com um exemplo de uma rede social com quatro pessoas, onde os vértices representam as pessoas na rede social e as arestas as amizades entre si.

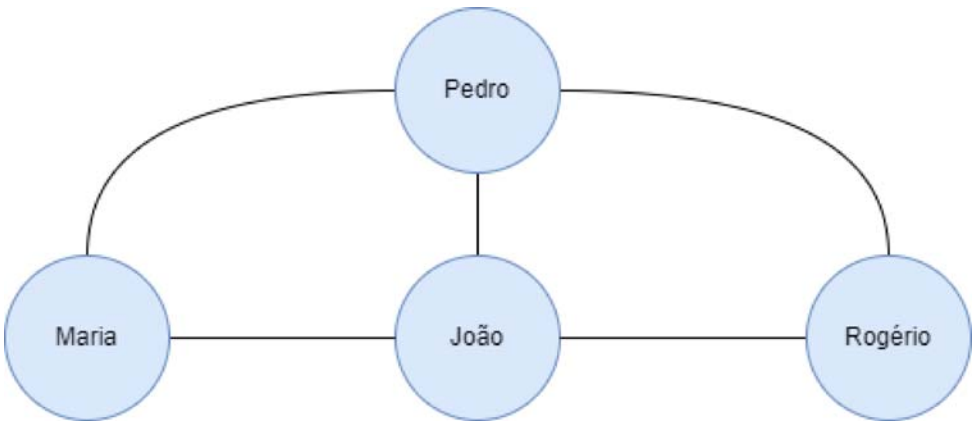

Figura 7 – Representação visual de um grafo de rede social. Fonte: Autoria própria.

As arestas dos grafos podem apresentar direcionalidade, onde introduzimos uma informação de conexões não bidirecionais entre vértices que podem ser úteis dependendo da aplicação da estrutura. Além disso, também podemos atribuir pesos para as

	Pedro	Maria	João	Rogério
Pedro	0	1	1	1
Maria	1	0	1	0
João	1	1	0	1
Rogério	1	0	1	0

Tabela 1 – Representação de uma matriz de adjacência para a Figura 7. Fonte: Autoria Própria.

arestas e vértices, podendo assim definir categorias de vértices e arestas assim como graus de conexão.

Geralmente, grafos são a melhor escolha para representar dados não euclidianos que possuem conexões entre si. Diversas são as áreas que se beneficiam da estrutura de conexões, como é o caso da representação visual de redes sociais e representação de moléculas.

Além disso, é possível representar dados euclidianos utilizando grafos também, podendo trazer conexões que agregam informação a representação padrão. Em um domínio como o textual, é possível utilizar a estrutura gramatical para adicionar informação a *embeddings* textuais (TRAN; MIWA; ANANIADOU, 2020). Por meio de técnicas como *Part-of-Speech tagging* e *Dependency Parsing*, podemos criar conexões sintáticas que podem ser assumidas como arestas e palavras como nodos, adicionando informação a representação textual e potencialmente trazendo ganhos de acurácia para uma rede neural.

Porém, é preciso um pré processamento a mais para utilização de grafos como entrada para uma rede neural.

2.3.1 Representação

Para podermos representar grafos de forma em que possam ser utilizados como entrada para uma rede neural, é preciso transformá-los para um formato numérico. As formas mais comuns de representar um grafo são por meios de matriz de adjacência e lista de adjacência (SINGH; SHARMA, 2012).

Uma matriz de adjacência pode ser definida da seguinte forma, dado um grafo finito G com n vértices, é criado uma matriz nXn onde a posição ij representa a aresta do vértice i com o vértice j. Por meio desta matriz é possível representar tanto grafos direcionados como não direcionados, além de a posição ij poder possuir um valor que serve como o peso da aresta entre os dois vértices. A Tabela 1 demonstra visualmente uma matriz de adjacência para o grafo de rede social representado pela Figura 7.

A principal limitação dessa representação torna-se evidente ao lidarmos com grafos esparsos. Ao gerar uma matriz que considera todas as possíveis conexões, mesmo que apenas algumas posições correspondam a conexões reais entre os no-

Nodo	Conexões
Pedro	Maria, João, Rogério
Maria	Pedro, João
João	Pedro, Maria, Rogério
Rogério	Pedro, João

Tabela 2 – Representação de uma lista de adjacência para a figura 7. Fonte: Autoria Própria.

dos, muitas posições permanecem sem valor significativo. Isso resulta em um desperdício de espaço de memória computacional e aumenta o tempo de processamento necessário. Em grafos com muitos nodos e poucas conexões, essa abordagem é particularmente ineficiente, pois amplia a complexidade sem agregar informações relevantes, impactando negativamente a eficiência computacional.

Para minimizar este problema, é possível utilizar as listas de adjacência. Essas listas possuem o mesmo objetivo da matriz de adjacência, porém buscam representar apenas as conexões válidas entre nodos, não representando conexões não existentes. A lista é definida da seguinte forma, toda entrada na lista é dada por uma tupla de dois nodos, sendo o primeiro nodo o nodo raiz e o segundo nodo aquele a receber a conexão. Para definimos uma bidirecionalidade, é possível apenas replicar o inverso de cada entrada. A Tabela 2 representa uma lista de adjacência da Figura 7.

Esta representação faz melhor uso da memória computacional, apenas armazenando as conexões existentes e sem desperdício de memória. Porém, pode tornar mais complexo o processo de inserção de novas entradas na lista.

2.3.2 Redes Neurais em Grafos

GNNs são redes neurais capazes de fazer uso da conectividade dos grafos. Este tipo de RNA necessita do pré-processamento anterior de converter um grafo para uma matriz de adjacência ou lista de adjacência para que possa representar numericamente seus dados.

Existem diversos tipos de arquiteturas GNNs diferentes, porém as duas mais abordadas na literatura são as Redes Convolucionais em Grafos (do inglês *Graph Convolutional Networks* - GCN) (KIPF; WELLING, 2017) e as GATs (VELIČKOVIĆ et al., 2018). Cada uma dessas arquiteturas explora a natureza conexionista dos grafos de maneira distinta, oferecendo abordagens complementares para o processamento de dados estruturados em grafos.

2.3.2.1 Redes Convolucionais em Grafos

As GCNs são fortemente inspiradas em redes neurais convolucionais para sua utilização em grafos. O funcionamento de uma GCN é baseado na codificação de cada nó sendo uma média ponderada de seus nós vizinhos. A equação que define a GCN original em (KIPF; WELLING, 2017) é dada por:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad (4)$$

Onde $\tilde{A} = A + I_N$ é a matriz de adjacência de um grafo não direcionado com auto conexões. I_N é uma matriz identidade, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ é uma matriz diagonal de graus, ou seja, a soma das conexões para cada nó. A utilização dessa normalização se dá pelo fato que sem ela, nodos com mais conexões estariam em uma escala maior do que nodos com menos conexões, necessitando assim aplicar uma normalização desta forma. W^l é uma matriz de pesos treináveis, σ é uma função de ativação e H^l é uma matriz de ativação da camada l sendo que $H^{(0)} = X$.

Mais tarde, este tipo de mecanismo começou a ser conhecido como *Message Passing Mechanism* e serviu de base para a criação de outras metodologias. Diversos outros trabalhos começaram a surgir propondo diferentes métodos de agregação que não a média para unir os vetores das características dos nodos vizinhos, como por exemplo a função de agregação por somatório, agregação por memória de longo e curto prazo (do inglês *Long Short-Term Memory* - LSTM) (HAMILTON; YING; LESKOVEC, 2018), por ordem (ZHANG et al., 2018), entre muitos outros. Sendo que algumas funções de agregação podem funcionar melhor dependendo da problemática em que está sendo utilizada. A Figura 8 demonstra a utilização de uma função de agregação de somatório em um MPM.

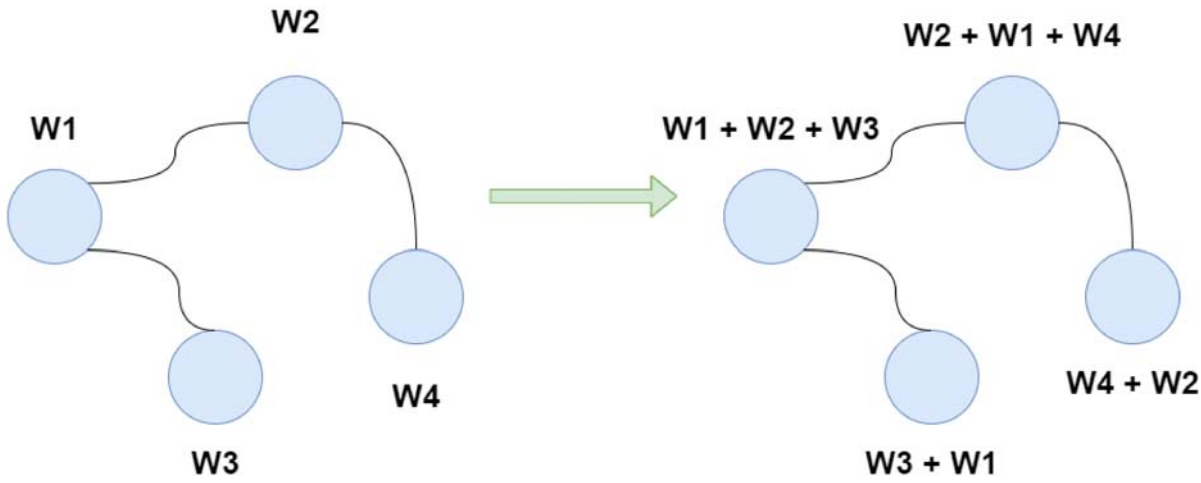

Figura 8 – Representação visual de um *Message Passing Mechanism* com uma função de agregação de somátorio. Fonte: Autoria Própria.

GCNs são bem exploradas na literatura e funcionam bem para diversos tipos de tarefas, principalmente para classificação de nodos. Porém uma grande parte das funções de ativação possuem um problema latente. Dado uma vizinhança parecida, funções de agregação baseadas em média ou somatório possuem o problema de que diferentes nodos podem acabar com o mesmo valor, que pode ser desfavorável dependendo do contexto do problema. Como por exemplo em uma análise textual, já

que esse tipo de agregação não leva muito bem em consideração diferentes contextos. Além disso, pode ser que algumas conexões ainda não sejam tão úteis de serem codificadas para determinados nodos em certos cenários. Dito isto, o próximo tipo de arquitetura surgiu para minimizar estes problemas.

2.3.2.2 Redes de Atenção em Grafos

Com base no conceito de que os mecanismos de atenção são amplamente utilizados em tarefas baseadas em sequências, os autores de Veličković et al. (2018) propuseram um mecanismo de atenção adaptado para grafos, visando tornar a codificação dos nós e suas conexões mais dinâmica. Esse mecanismo permite a atribuição de pesos diferenciados às conexões, de forma a destacar quais atributos dos nós vizinhos são mais relevantes para a codificação de um determinado nó. Isso proporciona uma forma mais eficiente de capturar informações contextuais no grafo, tornando a tarefa de aprendizado mais precisa e flexível. Além disso, a arquitetura criada por eles possui alguns benefícios, como a operação poder ser paralelizada por pares de nodo/vizinho e pode ser aplicado para grafos com nodos de diferentes graus de vizinhança devido aos pesos gerados.

Seguindo os passos de Veličković et al. (2018), para definirmos a arquitetura, precisamos primeiro definir uma camada de atenção em grafos. A entrada da camada pode ser dada como h_n que representa um conjunto de características de n nodos, e cada saída de camada produz um novo conjunto de características de nodos, que podem ser de diferentes tamanhos.

Para podermos alterar as características para outras novas, utilizamos então uma matriz de pesos W que é aplicada para cada nodo. Com estas variáveis, é possível aplicarmos um mecanismo de atenção a, determinando a equação para obter o coeficiente de atenção e_{ij} com:

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \tag{5}$$

Para injetar a estrutura do grafo com um mecanismo, os autores argumentam que é possível utilizar um método de *masked attention*, computando e_{ij} apenas para os nodos j que fazem parte da vizinhança N do nodo i. Após a geração dos valores, eles são normalizados por meio da função *softmax* garantindo que a soma das atenções seja unitária. Dessa forma, obtemos o valor normalizado de e_{ij} para cada vizinho j como α_{ij} com:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \tag{6}$$

Esse processo permite a que o modelo atribua diferentes pesos para cada nodo da vizinhança de i, destacando a importância relativa de cada vizinho j.

Na arquitetura desenvolvida pelos autores, eles utilizaram uma rede neural simples de uma camada como mecanismo de atenção, com os parâmetros definidos como um vetor de pesos $\vec{a}$, e aplicaram uma função de ativação *LeakyReLU* para trazer a não linearidade, desta forma, a equação final para computar os coeficientes de atenção é definido por:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\text{LeakyReLU}\left(\vec{a}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k\right]\right)\right)} \tag{7}$$

Onde ^{T} é a transposição do vetor e $\|$ é a operação de concatenação. Após obtidos os valores, é possível só aplicar uma combinação linear para obter parte das características dos nodos vizinhos.

Para estabilizar o processo de aprendizado do mecanismo, os autores ainda implementaram o conceito de *multi-head attention* que foi empregado nas arquiteturas Transformers. A Figura 9 demonstra visualmente como é aplicado o mecanismo de atenção para cada nodo e a aplicação de *multi-head attention* para cada cálculo.

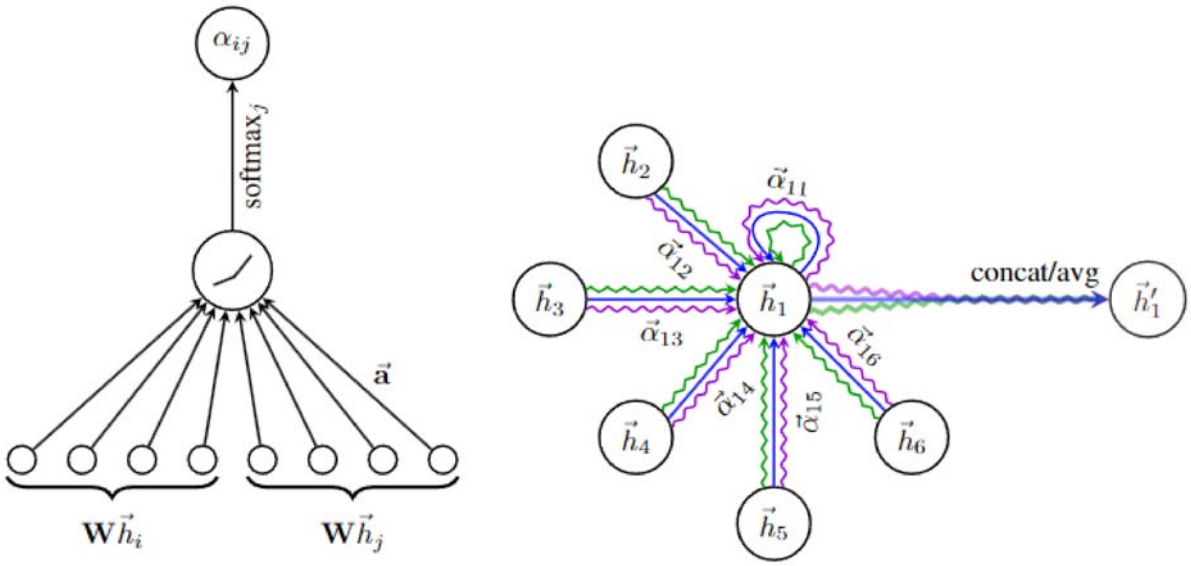

Figura 9 – Representação visual da aplicação do mecanismo de atenção e *multi-head attention* empregado pela arquitetura GAT. Fonte: (VELIČKOVIĆ et al., 2018).

Essa arquitetura permite a implementação de um mecanismo de atenção, incluindo a técnica de *multi-head attention*, de forma eficiente, sem a necessidade de aplicá-lo a todos os *tokens* de uma sentença. Isso resulta em uma significativa economia de recursos computacionais, o que pode ser extremamente valioso dependendo do contexto e das limitações de quem utiliza a tecnologia. A abordagem concentra-se apenas nas conexões previamente extraídas, que podem ser as mais relevantes, tornando o processo mais leve e rápido, sem comprometer a qualidade da representação das

informações.

3 TRABALHOS RELACIONADOS

Neste capítulo, abordaremos os principais trabalhos relacionados ao tema da metodologia. Os dividiremos nas seguintes seções: Trabalhos de ABSA em Português, Trabalhos em Grafos para ABSA em inglês e Trabalhos em Redes de Grafos com Atenção para ABSA em inglês.

3.1 Análise de Sentimento em Nível de Aspecto em Português

A área de ABSA está amplamente consolidada como uma tarefa essencial de PLN na comunidade acadêmica. Devido a sua utilidade em uma variedade de aplicações e contextos, novos estudos são continuamente publicados com o objetivo de propor soluções inovadoras para esse problema. Porém, o grande foco das metodologias é na língua inglesa, dado o seu status de língua mais falada globalmente, além do fato de os países falantes da língua possuírem mais recursos que auxiliam o desenvolvimento desses métodos.

Diante a isso, é evidente a importância da criação de metodologias voltadas para a língua portuguesa brasileira, dado a crescente demanda de aplicação dessas técnicas em diversos cenários. Além disso, é importante destacar a escassez de recursos e corpora em português quando comparado ao inglês, que acaba criando um obstáculo significativo para o avanço de novas tecnologias.

Apesar desta limitação, ainda surgem diversos trabalhos baseados em diferentes metodologias que visam abordar e solucionar o problema de ABSA para português brasileiro. Esses trabalhos exploram abordagens inovadoras e adaptativas, superando desafios relacionados à escassez de recursos e contribuindo significativamente para o avanço do PLN no contexto da língua portuguesa.

No trabalho de Freitas (2015), a autora destaca a necessidade de criação de um novo corpus, devido à escassez de recursos disponíveis para a condução dos experimentos. Assim, foi proposta a criação de um conjunto de dados a partir da extração de *reviews* de hotéis obtidos no site TripAdvisor, seguido da anotação manual dos aspectos e suas respectivas polaridades. Com o corpus definido, a autora desenvolveu uma

metodologia para ABSA, organizada em quatro etapas: pré-processamento, identificação dos aspectos, identificação da polaridade e sumarização. Na primeira etapa, são obtidas as categorias gramaticais e os lemmas das palavras por meio da técnica de *Part-of-Speech Tagging*. A extração dos aspectos ocorre na segunda etapa, utilizando ontologias de domínio. Na terceira fase, a metodologia emprega léxicos de sentimentos e regras linguísticas para estimar a polaridade dos aspectos identificados. Como resultado final, a metodologia gera um sumário contendo os aspectos extraídos e suas respectivas polaridades. Esta metodologia possui as mesmas limitações mencionadas anteriormente, pelo fato de ser baseada em abordagens clássicas. Dessa forma, a abordagem depende da precisão gramatical, não sendo o ideal para determinados domínios.

No trabalho de Corrêa (2021), o autor também aborda a criação de um novo conjunto de dados do setor hoteleiro extraído do site TripAdvisor, utilizando a mesma ontologia empregada por Freitas (2015) para extrair os aspectos do domínio, auxiliando o processo de anotação. Após a anotação do conjunto de dados, o autor emprega uma arquitetura de Rede Neural Convolucional em nível de caractere (do inglês, *Character level Convolutional Neural Networks* - CharCNN) para as tarefas de estimação de polaridade de sentimento do aspecto, além da criação de um modelo único para resolver a tarefa *end-to-end*, adicionando uma camada para extração dos aspectos. A utilização de um modelo convolucional a nível de caractere traz uma característica interessante, pois o modelo se torna menos sensível a erros de ortografia, que costumam ser muito presentes em domínios baseados em dados da internet. O modelo desenvolvido demonstrou resultados superiores aos estado da arte da época. Porém, a metodologia se baseia na aplicação de convolução entre caracteres, tornando a codificação menos dinâmica em comparação com mecanismos de atenção.

No trabalho de Silva et al. (2022), os autores organizaram uma competição focada nas duas tarefas principais de ABSA: extração de aspectos e classificação de polaridade de sentimento. O corpus utilizado foi composto pelos dados desenvolvidos por Freitas (2015) e Corrêa (2021). A competição contou com a participação de cinco equipes de universidades brasileiras para a tarefa de AE e quatro equipes para a tarefa de ASC. A equipe Deep Learning Brasil, vencedora da tarefa de classificação de polaridade de sentimento, utilizou uma técnica baseada em um *ensemble* de modelos PTT5 (CARMO et al., 2020), os quais foram ajustados com diferentes hiperparâmetros para uma tarefa de geração de texto condicional, conforme descrito em Gomes et al. (2023).

A equipe PiLN obteve o segundo lugar da tarefa de classificação de polaridade de sentimento, empregando uma metodologia baseada no modelo BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), para a tarefa de classificação de sentença com sentença auxiliar, usando o *review* como uma sentença e o aspecto como uma sentença

auxiliar, além do fato de adotarem pesos diferentes entre as *labels*, como forma de lidar com o desbalanceamento do conjunto de dados (NETO et al., 2022).

Já a equipe UFSCar, que alcançou terceiro lugar, emprega uma abordagem diferente das demais, ao identificar palavras significativas em relação ao aspecto utilizando a ferramenta SpaCy (HONNIBAL et al., 2020), os autores utilizaram um modelo BERTimbau ajustado para estimar valores de sentimento baseados no *dataset* GoEmotions (HAMMES; FREITAS, 2021), e a partir dos valores gerados foi possível estimar a polaridade (ASSI et al., 2022).

A equipe UFPR, que obteve o quarto lugar, também empregou um modelo BERTimbau em sua metodologia, porém não deixou claro como foi implementada sua estratégia, apenas afirmaram que pelo fato do modelo demorar um tempo considerável para ser treinado, aliando-se com a falta de tempo que o time possuía, os autores acabaram limitando o número de épocas para realizar o treinamento, e argumentam que se possuíssem mais tempo poderiam ter resultados melhores (HEINRICH; MARCHI, 2022).

Podemos perceber que todos os trabalhos competidores utilizaram modelos Transformers em suas abordagens, creditamos isso ao fato desse tipo de algoritmo normalmente obter bons resultados para tarefas de PLN, além de estar sendo muito bem explorado nos últimos anos. Porém, estas metodologias são custosas em termos computacionais, como mencionado anteriormente.

3.2 Trabalhos em Grafos para Análise de Sentimento em Nível de Aspecto em Inglês

Diversos trabalhos exploram a aplicação de GNNs como parte de suas metodologias para a tarefa de ABSA na língua inglesa. Em alguns casos, há aqueles que abordam o uso de mais de uma arquitetura de GNN para criar o modelo final, onde cada uma complementa a outra, como é o caso de (LI et al., 2021). Neste trabalho, os autores propõem um modelo que utiliza duas GCNs para solucionar a tarefa de classificação de polaridade de sentimento, onde cada modelo fica responsável por analisar uma perspectiva diferente do texto. O primeiro modelo é chamado de *SynGCN*, seu foco é na formação de uma codificação sintática da entrada, este modelo alcança seu objetivo pelo meio da criação de uma matriz de adjacência gerada por um modelo estado da arte da tarefa de *dependency parsing* (MRINI et al., 2019). O segundo modelo, *SemGCN* é responsável por fazer uma análise semântica da entrada, a partir de um mecanismo de *Self-Attention* para realizar a criação da matriz de adjacência. Para integrar informações entre modelos, é empregado um módulo *BiAffine* que aprenda as características mais importantes entre cada arquitetura para formar a representação final. Os autores realizaram seus experimentos em três corporas diferentes, sendo eles

os *datasets* de restaurantes e *laptops* da competição SemEval (PONTIKI et al., 2016) e um de *tweets* (DONG et al., 2014). Seus resultados superaram todos os modelos escolhidos para a comparação.

Em Sun et al. (2019) os autores tomam um rumo diferente, com apenas um modelo. Neste trabalho, é explorado o uso de uma árvore sintática como entrada para o modelo. Os autores propõe um modelo BiLSTM para a geração de *embeddings* iniciais, e assumindo que podemos interpretar uma árvore sintática como um grafo, onde nodos representam palavras e as arestas como as dependências sintáticas entre as palavras, os autores adotam uma GCN para melhorar ainda mais a codificação da entrada. Para a classificação final, eles exploraram o uso de uma função de *average pooling* como agregador nas representações dos *tokens* dos aspectos, e a saída é então obtida a partir de uma função de *softmax*. Utilizando os mesmos conjuntos de dados para a avaliação do que o trabalho anterior, os conjuntos de *reviews* de restaurantes, *laptops* e *tweets*, o modelo final obteve melhores resultados do que os demais escolhidos para a comparação.

3.3 Trabalhos em Redes de Grafos com Atenção para Análise de Sentimento em Nível de Aspecto em Inglês

Na literatura, conseguimos observar alguns trabalhos que já apresentam evidências sobre a eficiência da arquitetura GAT para a tarefa de ABSA na língua inglesa. Como é o caso do trabalho de Wang et al. (2020), onde os autores utilizam uma metodologia baseada em GATs relacionais para solucionar o problema de classificação de polaridade de sentimento. Os autores argumentam que as até então utilizações de mecanismos de *Self-Attention* falhavam ocasionalmente em determinadas situações devido a complexidade da morfologia da linguagem e sintaxe, onde determinados termos de opinião influenciavam na predição de aspectos diferentes dos quais estes termos se referiam. Para lidar com este problema, a abordagem proposta dos autores se baseia em três passos. O primeiro passo consiste na extração da árvore de dependência sintática por meio de um *dependency parser*. O segundo passo se refere na transformação do termo do aspecto como raiz da árvore, e por último, é realizada uma poda na árvore para permanecer somente as relações de dependência diretamente ligadas ao aspecto, tornando assim mais simples de realizar a paralelização e focar apenas nas conexões entre os aspectos e os termos de opinião. Após este processamento dos dados, essa codificação é alimentada para uma GAT relacional, que codifica as novas árvores de dependência. Os conjuntos de dados utilizados para avaliar a metodologia foram os mesmos dos dois trabalhos descritos anteriormente, e o modelo desenvolvido superou os demais modelos escolhidos pelos autores como *baseline*.

Buscando solucionar o mesmo problema do trabalho de Wang et al. (2020), os autores de Liang et al. (2022) propõem uma nova abordagem. Eles argumentam que a utilização de uma árvore de dependência introduz ruído com conexões não muito significativas para o aspecto, além de apenas trazer informações sintáticas entre as palavras, não sendo capazes de modelar relações complexas de uma sentença. Como abordagem para solucionar o problema, eles exploram o uso de uma árvore constituinte como substituto para a árvore de dependência sintática. Eles argumentam que essa representação contém segmentações mais precisas e discriminativas das frases, além de uma estrutura hierárquica de composição, que é útil para alinhar os aspectos com seus termos de sentimento.

Para sumarizar a abordagem proposta, os autores afirmam que a arquitetura é composta de três componentes, um módulo intra-contexto que se responsabiliza por codificar os dados de entrada para obter representações dos aspectos, que contém dois *encoders*, um responsável por codificar as informações inerentes ao contexto, baseado no modelo BERT, e outro responsável por analisar a sintaxe, baseado em uma arquitetura de GAT hierárquica multi-nível. O segundo módulo realiza uma análise extra-contexto, onde um *encoder* de relação é aplicado no grafo gerado para extrair representações das relações entre aspecto e contexto. Este processo é definido por um conjunto de regras que são responsáveis por criar um grafo de relações entre aspectos e contexto. E por último um classificador de polaridade de sentimento é utilizado para obter a saída final. Os autores avaliaram a abordagem por meio de quatro conjuntos de dados, sendo três os mesmos citados anteriormente, com adição de um conjunto de dados multi aspecto e multi sentimento (MAMS) (JIANG et al., 2019). Os resultados demonstraram que o modelo proposto superou os outros modelos selecionados para comparação.

3.4 Comparação dos Trabalhos

A tabela 3 apresenta uma comparação entre os trabalhos relacionados, indicando se é realizado para a língua Portuguesa ou Inglesa, se são aplicadas técnicas de aprendizado de máquina, se são utilizadas arquiteturas de GNNs e se são utilizadas arquiteturas de GATs. Nosso trabalho consiste em todas as categorias, dado ao fato de que nossa proposta consiste na avaliação do uso da arquitetura GAT para ABSA na língua portuguesa.

Trabalho	Idioma	ML	GNN	GAT
Freitas (2015)	PT			
Corrêa (2021)	PT	X		
Gomes et al. (2023)	PT	X		
Neto et al. (2022)	PT	X		
Assi et al. (2022)	PT	X		
Heinrich; Marchi (2022)	PT	X		
(LI et al., 2021)	EN	X	X	
Sun et al. (2019)	EN	X	X	
Wang et al. (2020)	EN	X	X	X
Liang et al. (2022)	EN	X	X	X

Tabela 3 – Tabela comparativa entre abordagens relacionadas. A primeira coluna identifica cada estudo. A segunda indica a língua em que a metodologia foi realizada. A terceira indica o uso de técnicas de aprendizado de máquina. A quarta coluna aborda a utilização de GNNs, enquanto a última coluna identifica se foram empregadas arquiteturas de GATs

4 METODOLOGIA

Neste capítulo, abordaremos a metodologia utilizada para a realização do trabalho, assim como nossa abordagem proposta.

4.1 Conjunto de dados

O conjunto de dados escolhido para avaliar a metodologia é o mesmo usado na competição ABSAPT 2022 (SILVA et al., 2022). A competição foi organizada pelo Hub de Inovação em Inteligência Artificial[1] da Universidade Federal de Pelotas e contou com a presença de cinco times de diferentes universidades nacionais. A competição abrangeu as duas principais sub-tarefas de ABSA, Extração de Aspectos e Classificação de Polaridade de Sentimento do Aspecto. A escolha da utilização deste conjunto de dados se dá pelo fato de ser um dos poucos conjuntos anotados que temos disponíveis na língua Portuguesa.

Além disso, a partir dos resultados dos modelos gerados pelos pesquisadores que participaram da competição, é possível utilizá-los como modelos *baseline* para a comparação de resultados, agregando assim um pouco mais de contexto geral para entendermos os resultados.

O corpus é formado por um conjunto de duas extrações diferentes de um mesmo domínio e mesma fonte de dados, os trabalhos de Freitas (2015) e Corrêa (2021) abordam com detalhes todo o processo de anotação.

O corpus é composto apenas por *reviews* de viajantes sobre acomodações de hotéis, coletados do site TripAdvisor [2]. Todas as anotações são na língua portuguesa, os *reviews* são de hotéis das cidades de Nova Iorque, Paris, Las Vegas e Porto Alegre, e todos possuem pelo menos 300 caracteres.

Cada anotação é um conjunto de *review* e aspecto, ou seja, múltiplos exemplos no dataset podem conter a mesma *review*. Isso se dá pelo fato de que um mesmo *review* pode conter mais de um aspecto. As anotações podem ser vistas como tuplas de *review*-aspecto únicos.

[1] https://ia.ufpel.edu.br
[2] https://www.tripadvisor.com.br

Informação	Quantidade
Número de Anotações	3.797
Número de *Reviews*	1.031
Aspectos únicos	77

Tabela 4 – Informações sobre o Corpus utilizado

Para identificar os aspectos a serem anotados, os autores utilizaram os conceitos da ontologia de domínio HOntology (CHAVES; FREITAS; VIEIRA, 2012). Nem todos os conceitos da HOntology são utilizados no processo de anotação, apenas os três primeiros níveis da ontologia. Nas anotações foram consideradas os níveis de hierarquia presente na ferramenta, sendo assim para cada exemplo é anotado seu nível hierárquico, se é um aspecto, um subaspecto ou subsubaspecto. Na prática, apenas usamos o aspecto que é explicitamente mencionado no texto e de nível mais profundo na taxonomia. O corpus total também possui referências a aspectos implícitos no texto, mas para este trabalho analisaremos apenas os aspectos explícitos.

Cada anotação é realizada considerando o nível de polaridade do sentimento, classificado como positivo, negativo ou neutro, sendo representado no corpus pelos valores **1** para polaridade positiva, **0** para neutra e **-1** para polaridade negativa. Além disso, as anotações incluem a posição do aspecto no texto em nível de caractere, indicando o caractere inicial e final do aspecto. Os aspectos anotados podem consistir em conjuntos de palavras, não se limitando a apenas uma única palavra. Como os dados são extraídos de *reviews* da Internet, uma característica desse domínio é uma alta quantidade de erros gramaticais presentes nos textos.

Quanto ao tamanho do corpus, a Tabela 4 apresenta a quantidade de anotações, *reviews* e aspectos únicos presentes no conjunto inteiro. Ressaltando que anotações são conjuntos de tuplas únicas de *reviews*-aspectos, sendo assim uma mesma *review* pode aparecer em diferentes anotações com diferentes aspectos.

Na Tabela 5 é apresentado alguns exemplos de anotações e como estão no formato que empregamos para a realização deste trabalho.

Como utilizamos o mesmo conjunto empregado na competição ABSAPT, os dados de treino e de teste são os mesmos usados pelos competidores, onde cerca de 20% do conjunto total é destinado a teste e o restante é destinado apenas para o treinamento. Este não é o número exato pois nestas tabelas estamos apenas nos referindo aos dados de teste da tarefa 2 como teste, sem levar em consideração os dados da tarefa 1. A divisão oficial foi realizada da seguinte forma, os autores selecionaram dos dados totais as *reviews* únicas para que não acontecesse nenhum tipo de vazamento de dados entre anotações, e a partir dessas *reviews* únicas foram separadas 20% das anotações com aquelas *reviews* para teste. Destes 20%, 50% foram destinadas para a tarefa 1 e os demais para a tarefa 2. Desses dados, 682 anotações foram usadas para teste neste trabalho, os mesmos selecionados para o teste da tarefa 2.

Review	Aspecto	Início	Final	Polaridade
Viajamos eu e minha irmã. O hotel tem uma extrutura excelente! Elevadores modernos e rápidos. O quarto bem climatizado e roupas de cama novas e limpas. O café da manhã farto e variado. O fato de ser ao lado da gare Montparnasse facilitou a ida a Versailles, pois pudemos pegar um trem lá. O ponto negativo é que o metrô não é muito perto, (6 quadras grandes) e a caminhada no frio muito angustiante! Mas retornaria com certeza!	Elevador	63	71	1
Muito simples, apartamentos pequenos, e parecem ter sido decorados por São Francisco pessoalmente, 1 cama, 1 cadeira, 1 mesa, 1 frigo bar, e 1 armário (sem portas, que funciona mesmo como cabideiro). Para piorar as coisas, os 2 únicos elevadores do hotel estavam quebrados e não havia outra escolha, se não usar as escadas. O prédio tem 14 andares. No café da manhã, o bufê é também franciscano (pobre de opções), mas dá pro gasto se você vai ficar uma ou duas noites no hotel.	hotel	249	254	0
Gostei muito deste hotel pela localização, no coração de Nova York, na Times Square, próximo de vários restaurantes, lojas,teatros, metrô e etc... Excelente para se hospedar com a família porque nos quartos há uma cozinha com geladeira, microondas e cafeteira. A internet no quarto nos atendeu muito bem.	cozinha	215	222	1

Tabela 5 – Exemplo de Anotações

Na Figura 10, podemos observar um gráfico representando a distribuição de polaridade dos 31 aspectos mais frequentes no conjunto de treino, podemos observar que não há um padrão exato e também não é balanceado, com grande parte dos aspectos contendo muito mais exemplos de classe positiva do que as demais.

Já na Figura 11, podemos observar o mesmo gráfico porém aplicado sobre o conjunto de teste. Todos os aspectos possuem pelo menos 10 ocorrências, e também são

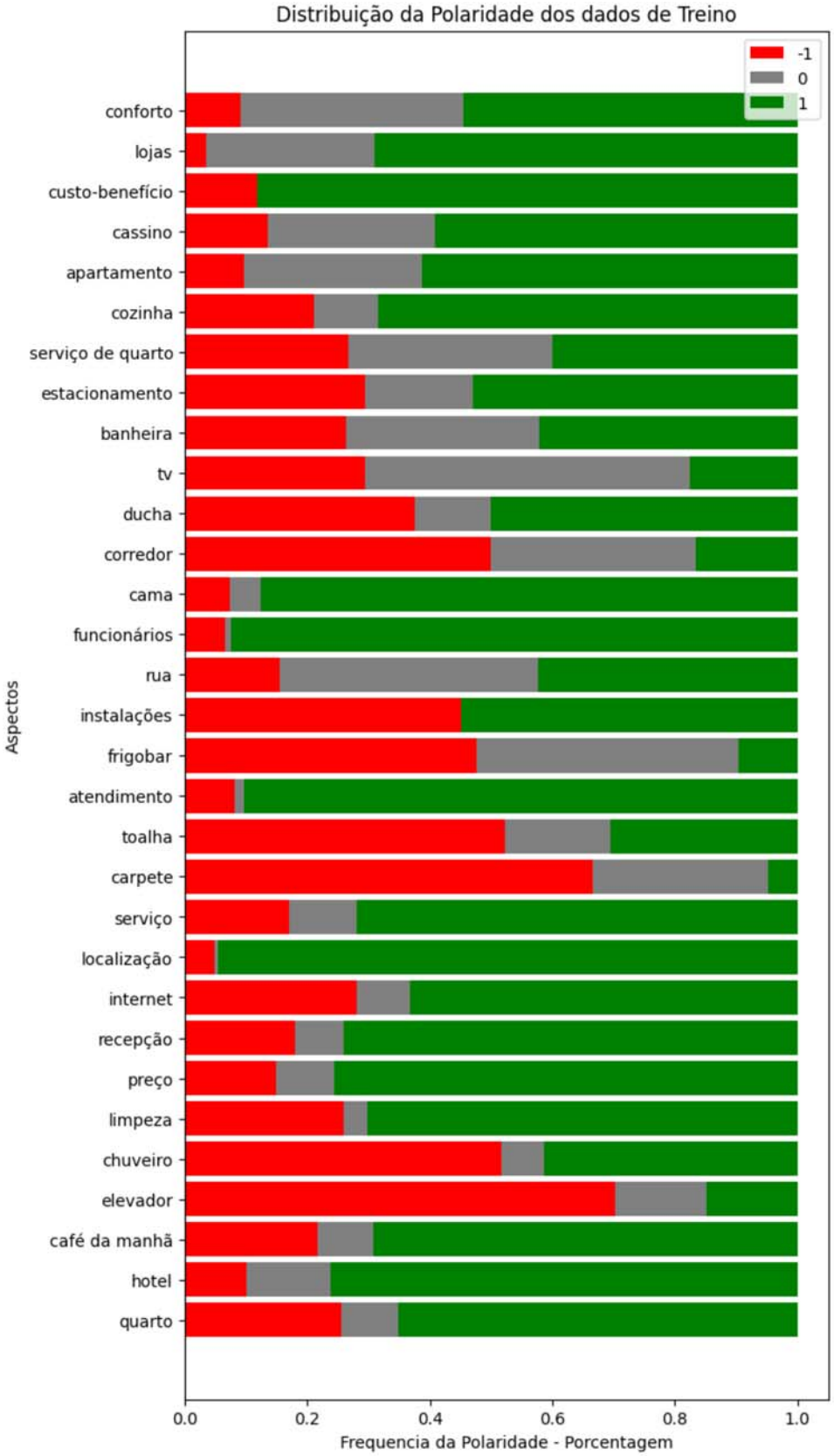

Figura 10 – Gráfico de distribuição de polaridade para os 31 aspectos mais frequentes do conjunto de treino. Fonte: Autoria própria, inspirado em Silva et al. (2022).

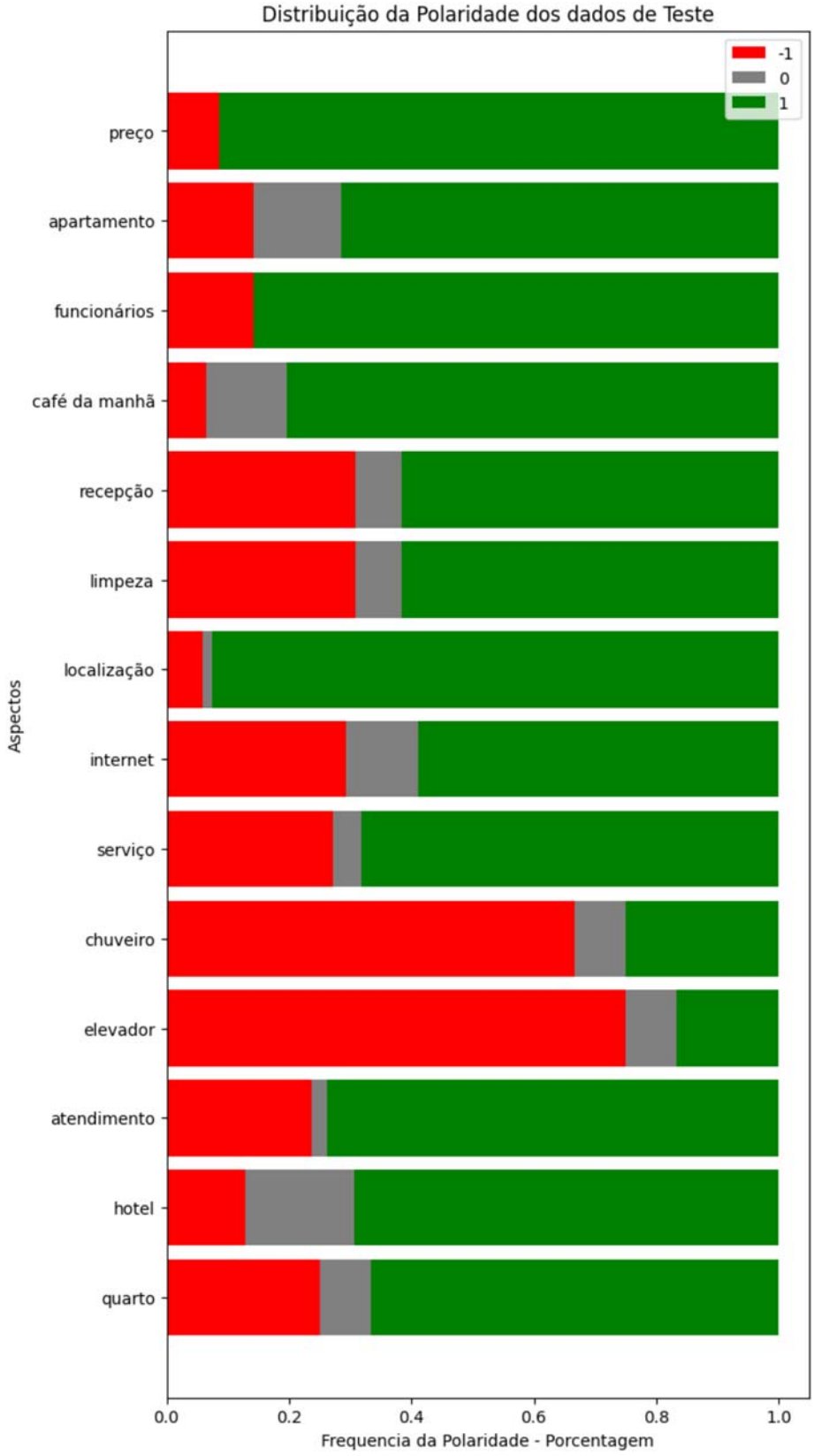

Figura 11 – Gráfico de distribuição de polaridade para os aspectos com pelo menos 10 ocorrências no conjunto de teste. Fonte: Autoria própria, inspirado em Silva et al. (2022).

desbalanceados com a maior parte dos aspectos tendendo a classe positiva.

Em relação à distribuição total das polaridades dos aspectos, na Figura 12 podemos observar um gráfico de barras representando a distribuição da polaridade no conjunto de treino. A partir dele é possível identificar que o conjunto de dados é desbalanceado, isso cria a necessidade de um cuidado maior quanto à avaliação das métricas resultantes da metodologia, já que o desbalanceamento implica que o uso de métricas como acurácia podem ser comprometidas durante a avaliação, dado que se o modelo prever apenas uma classe, e essa classe representar cerca de 70% dos dados, como é o nosso caso, podemos ser levados a acreditar que o modelo conseguiu generalizar bem o problema, sendo que não é o caso.

Devido a isto, escolhemos nossas métricas de avaliação tendo esta característica dos dados em mente. Este assunto será melhor abordado posteriormente neste capítulo.

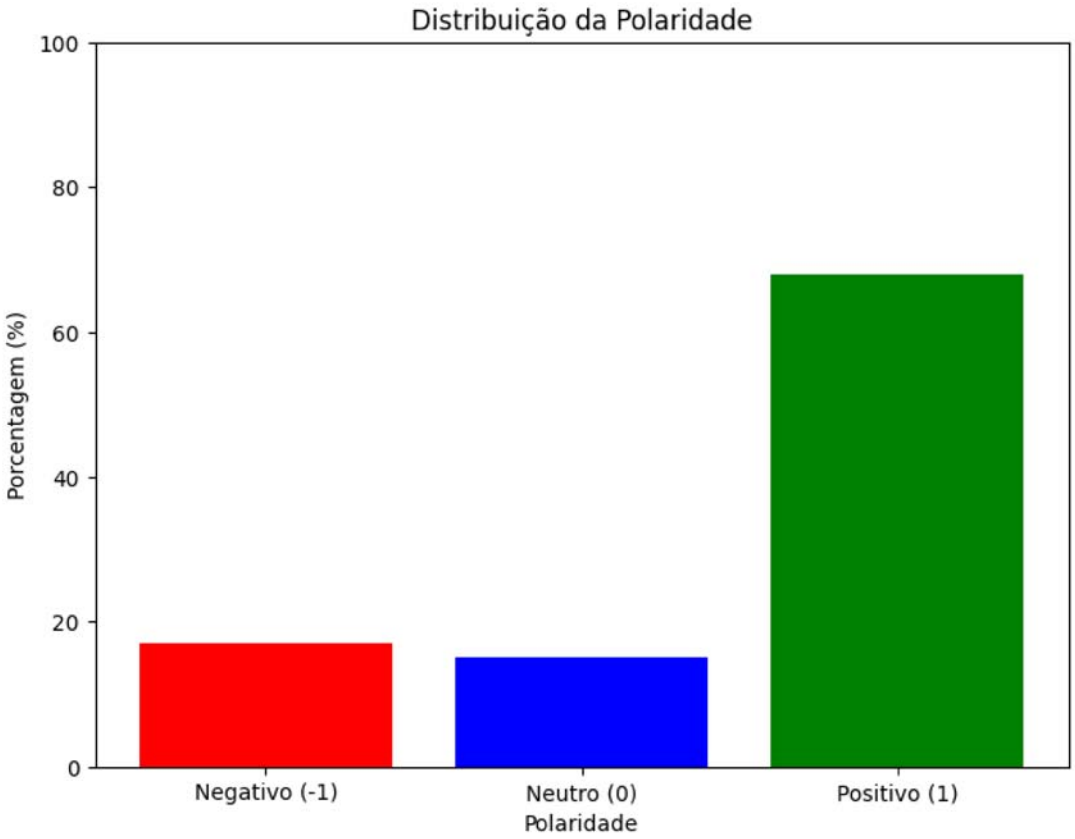

Figura 12 – Gráfico de distribuição de polaridade do conjunto de dados destinado a treino. Fonte: Autoria Própria.

4.2 Abordagem Proposta

A abordagem desenvolvida para avaliar a utilização de GATs para a tarefa de classificação de polaridade de sentimento em português se baseia nos seguintes passos:

1. Geração das árvores de dependência.

2. Codificação dos textos para *embeddings*.
3. Geração dos grafos a partir das árvores de dependência e *embeddings*.
4. Aplicação da GAT.
5. Aplicação do Classificador.

Discutiremos com mais detalhes os motivos da utilização de cada passo, assim como foram aplicados na prática. Na Figura 13, podemos observar uma ilustração da abordagem proposta.

4.2.1 Extração de Relações

Como primeiro passo para nossa abordagem, precisamos realizar os pré-processamentos necessários para a transformação das anotações para um formato de grafos. Como mencionado anteriormente, a estrutura de um grafo é descrita a partir de entidades e relações entre elas. Dito isto, para podermos representar nossos dados em forma de grafos, precisamos definir quais serão as entidades e que tipo de relação elas vão possuir.

Nossa escolha para abordar este problema é baseada nas relações sintáticas das palavras. Assumindo cada nodo como uma palavra, é possível utilizar um *dependency parser* para obter as relações sintáticas entre palavras, que podem ser utilizadas como arestas. Geralmente, modelos de *dependency parser* estimam além da relação, qual o tipo de relação sintática tal palavra tem com outra, porém, para nossa aplicação só há a necessidade de obtermos as relações entre palavras.

A escolha de um modelo para a construção da árvore de dependência possui um papel fundamental para nossa abordagem. Como ele dita como as arestas serão montadas, a acurácia do modelo para a tarefa afeta diretamente a eficácia do nosso modelo final, afinal, conexões falsas ou falta de conexões importantes pode acabar não agregando a informação correta para o modelo.

O *dependency parser* utilizado para o nosso trabalho é o modelo “pt_core_news_lg” da biblioteca SpaCy (HONNIBAL et al., 2020). A biblioteca SpaCy implementa diversas ferramentas muito úteis para facilitar o desenvolvimento de aplicações para tarefas de NLP em diversas línguas, incluindo o português. O modelo escolhido é o maior disponível para a língua portuguesa ofertado pelo SpaCy, assim como o que possui melhores resultados em termos de acurácia. Outro motivo da escolha se dá pelo fato de que não existem muitos modelos de *dependency parser* em português abertos para uso, o que limita nossas possíveis opções.

A Figura 14 ilustra a árvore de dependência gerada pela ferramenta SpaCy, podemos observar as conexões expressas pelas setas que ligam palavras a outras palavras, formando assim uma representação de grafo.

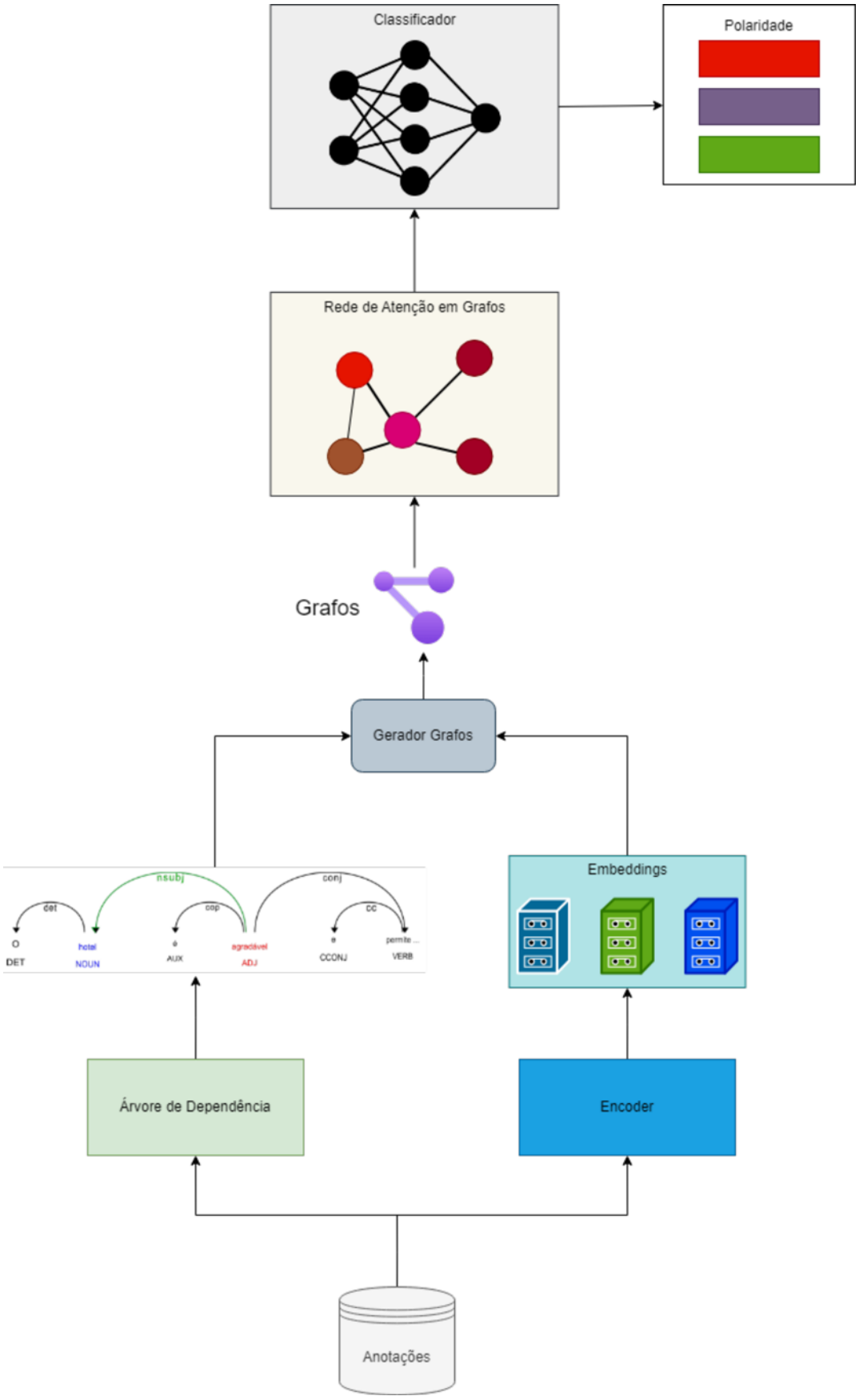

Figura 13 – Ilustração da Abordagem Proposta. Fonte: Autoria Própria.

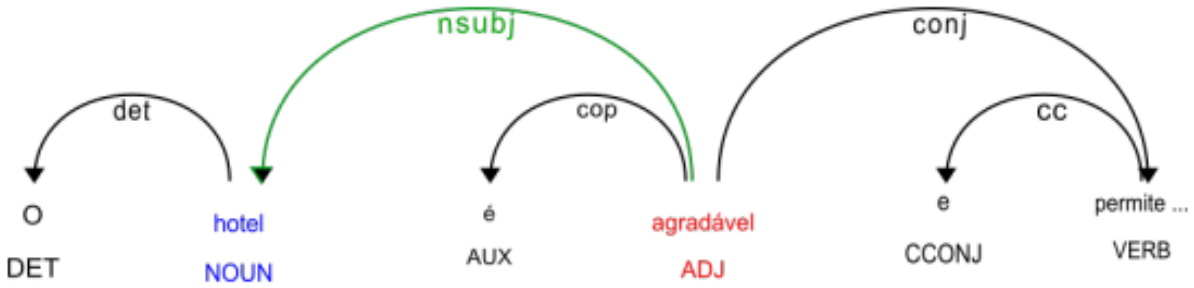

Figura 14 – Árvore de Dependência gerada pelo SpaCy. Fonte: (CORRÊA, 2021).

É importante ressaltar que, como empregamos um modelo para gerar as conexões, nós precisamos quebrar o texto em *tokens*. No modelo escolhido, os *tokens* são em nível de palavra, o que faz sentido devido ao objetivo que o modelo possui, o de solucionar a tarefa de análise sintática .

As únicas informações que retiramos desta etapa são as relações sintáticas entre palavras. Como nosso objetivo é a formatação dos dados para grafos, com o intuito de servirem como entrada para um modelo GAT, precisamos criar as representações matemáticas do texto.

4.2.2 Geração dos *Embeddings*

Atualmente, diversas tarefas de PLN em português possuem como estado da arte soluções que utilizam *embeddings* extraídos com o BERTimbau Large (SOUZA; NOGUEIRA; LOTUFO, 2020). Por isso, escolhemos este modelo para a geração dos *embeddings*. Como o modelo é um *encoder-only* baseado em transformers, acreditamos que o motivo da sua boa eficiência como gerador de representações é devido ao uso de *multi-head self-attention* como um agregador de informações semânticas para cada representação de *token*. A utilização de *embeddings* que codificam a informação semântica da sentença em cada *token* pode trazer diversos benefícios para a nossa representação do texto, como a capacidade de distinguir certos termos homônimos, ter menos sensibilidade a erros de grafia e carregar informações de locuções conjuntivas.

O modelo BERTimbau Large possui um vetor oculto de 1.024 dimensões, o que gera um maior espaço vetorial para capturar informações mais detalhadas. No artigo original da metodologia BERT (DEVLIN et al., 2018), os autores abordam qual a quantidade ideal de saídas de camada de *encoder* devem ser utilizadas para criar a representação do *token* para a tarefa de reconhecimento de entidade nomeada (NER - do Inglês *Named Entity Recognition*). Em seus experimentos, o uso da concatenação das saídas das últimas quatro camadas possuíram o melhor resultado para tarefa. Levando isso em consideração, decidimos também usar as quatro últimas saídas concatenadas, tornando assim nossa representação de cada *token* um vetor de 4.096 dimensões.

É importante mencionar que apenas adotamos o modelo BERTimbau para a inferência, com o objetivo de conseguir apenas os *embeddings* iniciais de cada *token*, em nenhum momento realizamos nenhum tipo de ajuste fino ou treinamento. O custo computacional de inferência é muito menor do que o necessário para realizar o treinamento, e é executado apenas uma vez.

4.2.3 Criação dos Grafos

Nesta etapa, possuímos as representações numéricas das palavras, levando em contexto a semântica entre elas, e as relações sintáticas entre elas. Porém, as representações numéricas obtidas através do modelo BERTimbau são em nível de subpalavras. Em ordem para podermos processar um conjunto de texto em um modelo *encoder*, precisamos quebrar o texto em *tokens*, que pode ser visto como uma unidade mínima de dado a ser processado. Cada *token* pode representar um caractere, uma sub-palavra ou até uma palavra inteira, dependendo da forma que a divisão será realizada e qual modelo será responsável pela divisão. Modelos de linguagem são altamente dependentes do método de tokenização utilizado, já que o aprendizado realizado é baseado no modelo aprender a associar os padrões, dependências e representações semânticas a *tokens* específicos utilizados no treinamento. Mudar o processo de tokenização quebra a correspondência dos novos *tokens* com os originais, afetando as representações extraídas pelo modelo.

Geralmente, o processo de tokenização segue regras específicas, como é o caso do método empregado pelo BERTimbau, conhecido como *Byte Pair Encoding* (BPE) (SENNRICH; HADDOW; BIRCH, 2016). Este método, que foi inspirado na técnica de compressão de dados do mesmo nome, cria *tokens* baseados no agrupamento iterativo de sequências mais comuns de caracteres e subpalavras de um conjunto de dados. Isto acaba gerando um empecilho para nossa abordagem, dado que nossas representações das palavras estão em níveis de subpalavras e nossas arestas em nível de palavra.

Para contornar este problema, inspirados pelo trabalho de Li et al. (2021) decidimos assumir nossos nodos em nível de subpalavra e replicar as relações entre palavras para cada subpalavra originária de uma palavra raiz. Dessa forma conseguimos utilizar as relações sintáticas em conjunto com as representações semânticas geradas pelo BERTimbau. A Figura 15 ilustra o processo de correção dos diferentes níveis de *token* para a criação do grafo final.

4.2.4 Rede de Atenção em Grafos

Nossa hipótese para o desenvolvimento do trabalho é baseada no fato de que podemos utilizar apenas conexões relevantes extraídas previamente para substituir a necessidade de computar todos os valores de atenção entre *tokens*, como é no caso

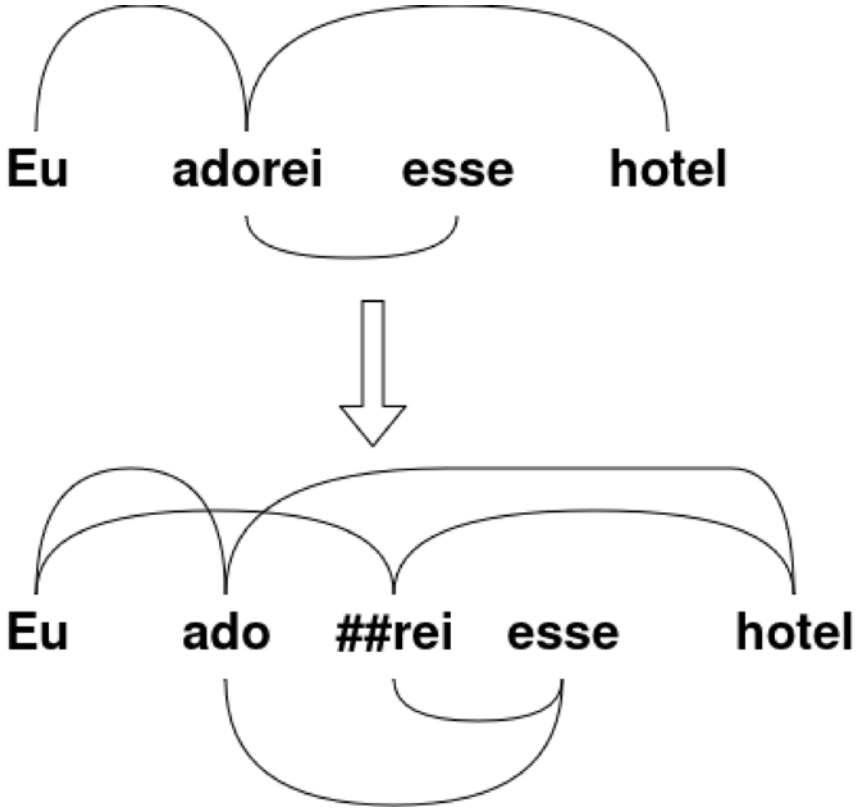

Figura 15 – Ilustração da correção do problema de diferentes níveis de *token*. Fonte: Autoria Própria.

de algoritmos baseados em transformers. E neste processo, economizaríamos no número de operações realizadas, além de diminuir o tamanho do modelo, melhorando a eficiência computacional e energética do algoritmo, sem perder desempenho.

Para realizar isso, propomos a utilização de uma arquitetura GAT, criada por Veličković et al. (2018) com o objetivo de calcular valores de atenção em torno da vizinhança de cada nó, permitindo incorporar à representação do nó central uma fração das informações de seus vizinhos, ponderada pela relevância contextual de cada um. Porém, é importante pontuar que o mecanismo de atenção empregado pela arquitetura GAT original é diferente do mecanismo utilizado em modelos Transformers, embora o objetivo seja o mesmo.

Para a implementação da arquitetura utilizamos o *framework* Pytorch Geometric (FEY; LENSSEN, 2019), uma ferramenta que facilita a criação de diversas arquiteturas de GNNs, assim como a integração com o a biblioteca Pytorch (PASZKE et al., 2019).

A arquitetura escolhida para a realização dos experimentos é composta por duas camadas de GAT, que são aplicadas para cada nodo. A entrada é a representação vetorial de cada palavra, assim como uma lista de adjacência que dita as conexões entre nodos. O tamanho da entrada possui 4.096 dimensões, e a saída de cada camada também possui 4.096 dimensões, sendo que as camadas são ligadas por uma função de unidade linear retificada (ReLU - do inglês *Rectified Linear Unit*). Nossa

escolha diante o tamanho da saída é apenas avaliar o efeito do mecanismo de *self-attention*, sem realizar nenhum tipo de redução dimensional.

Esta arquitetura pode ser vista como um modelo *encoder*, dado que não chegamos a realizar nenhum tipo de classificação e apenas queremos adicionar o contexto das ligações para cada representação.

4.2.5 Classificador

Neste passo, como já realizamos as transformações necessárias e já obtivemos a saída do modelo GAT criado, é preciso definir a arquitetura que criará o modelo que classificará os dados obtidos nas polaridades positiva, negativa ou neutra.

Nós optamos por modelar a tarefa como uma classificação de nodos. Porém apenas utilizamos os nodos referentes aos aspectos presentes no texto. Isto é, nesta etapa, nós extraímos somente as representações dos nodos de aspectos para servirem de entrada para o classificador, já que para nossa análise somente esses nodos devem ser classificados.

Optamos por utilizar uma arquitetura com uma camada linear simples para realizar a classificação final. A entrada dessa camada precisa ter um tamanho fixo, e para definir esse valor, realizamos uma análise para determinar o número máximo de *tokens* necessários para representar o maior aspecto, concluindo que seis *tokens* seriam suficientes. Nós realizamos *padding* para todos os aspectos que foram divididos em menos de seis *tokens* para se adequarem à entrada da rede. Logo, a entrada do modelo é um vetor de 6*4.096 posições.

A saída do classificador é composta por três neurônios, um para cada polaridade de sentimento.

4.3 Configuração dos Experimentos

Nesta seção abordaremos as configurações empregadas para realizar os experimentos, e que tipo de avaliações foram conduzidas.

4.3.1 Busca de Hiper-Parâmetros

Para nossa avaliação da metodologia, é importante que conduzamos uma busca de quais são os melhores hiper-parâmetros para a criação de nossa arquitetura. Hiper-parâmetros podem ser interpretados como valores fixos que afetam o ajuste dos parâmetros do modelo. Existem diversos tipos de hiper-parâmetros para a criação de modelos que devem ser selecionados levando em consideração o conjunto arquitetura, tarefa e dados usados na metodologia.

Para realizar experimentos iniciais, nós dividimos o conjunto de treino em 80% destinados a treino e 20% para validação. Com estes dados, buscamos limitar o conjunto

de hiperparâmetros a ser explorado fazendo alguns treinamentos *a priori*. Nestes experimentos, descobrimos que a utilização de múltiplas cabeças de atenção para cada camada limitava o aprendizado do modelo, causando uma estimativa única da mesma polaridade para todos os exemplos, além de aumentar severamente a quantidade de parâmetros e o tempo de execução necessário pra inferência. Acreditamos que isto seja devido ao aumento de complexidade da arquitetura. Geralmente, arquiteturas mais complexas necessitam de mais dados para o realizar o aprendizado, e nosso conjunto pode ser considerado pequeno para o treinamento de uma rede deste calibre.

Nossos outros experimentos iniciais nos levaram a crer que a utilização do otimizador AdamW (LOSHCHILOV; HUTTER, 2019) e a função de perda *Cross Entropy* seriam as opções que trariam mais benefícios para o treinamento, então os deixamos fixos durante a busca.

Como ferramenta para realizar a busca de hiperparâmetros utilizamos o *framework* Optuna (AKIBA et al., 2019), uma biblioteca Python com diversas técnicas de otimização de ajuste de hiper-parâmetros já implementadas. Em nossos experimentos, utilizamos o algoritmo *Tree-Structured Parzen Estimator* (WATANABE, 2023), um método de otimização bayesiana que busca maximizar ou minimizar uma função, no caso desta abordagem, maximizar a métrica de acurácia balanceada. O algoritmo se baseia na exploração de um espaço de busca estruturado em árvore por meio de estimadores Parzen, separando observações anteriores entre configurações que obtiveram uma performance melhor do que um determinado *threshold* e configurações com performances piores, levando estes conjuntos em consideração para a próxima escolha de parâmetros.

Todos os experimentos realizados com o Optuna foram avaliados utilizando o método de validação cruzada K-fold com cinco *folds*. Essa abordagem, embora aumente o tempo total de treinamento, já que a arquitetura precisa ser treinada cinco vezes para uma avaliação completa, é particularmente adequada para conjuntos de dados pequenos. Isso ocorre porque o método permite o uso de todo o conjunto de dados destinado ao treinamento, além de oferecer uma avaliação mais robusta da performance da arquitetura, minimizando vieses e variações nos resultados.

Os hiperparâmetros a serem explorados são o tamanho da *batch*, *learning rate*, *weight decay*, a utilização ou não da variante AMSGrad do otimizador, o valor ϵ e o número de épocas dadas para o treinamento.

O tamanho da *batch* se refere a quantas entradas em paralelo serão computadas por vez pela rede, assim como de quantos em quantos exemplos o modelo corrigirá seus pesos. O valor de *learning rate* define o quanto os parâmetros serão atualizados. Já *weight decay* é um valor de penalização empregado pelo otimizador AdamW, para incentivar o modelo a manter pesos menores e evitar o sobreajuste. AMSGrad

(REDDI; KALE; KUMAR, 2018) assume um valor binário que define a utilização ou não da variante para o otimizador. Em teoria, essa variante tende a melhorar a estabilidade do treinamento corrigindo um problema de não convergência que o otimizador Adam base possui. O valor ϵ é um termo somado ao denominador para melhorar a estabilidade numérica, servindo quase como um viés. O número de épocas é o número de vezes em que o modelo repetirá o treinamento com os dados originais.

Os valores explorados para cada hiperparâmetro são: Valores de tamanho de *batch* entre 1, 2, 4, 8, 16 e 32. Valores de *learning rate* entre $1e-7$ até $1e-5$, *weight decay* pode assumir valores entre $1e-7$ até $1e-5$, AMSGrad pode ser ativado ou desativado, valores ϵ entre $1e-9$ até $1e-5$ e quantidade de épocas foram exploradas de 1 até 100.

Ao término da busca por hiper-parâmetros, é selecionado o conjunto que possuiu os melhores resultados da média de acurácia balanceada entre *folds* para o treinamento com todos os dados selecionados para treino, incluindo os de validação. O modelo gerado a partir deste ponto pode ser avaliado pelo conjunto de teste.

Todos os experimentos e avaliações deste trabalho foram realizados na mesma máquina, com um processador Intel Core i7-7700K, uma placa de vídeo Nvidia Titan Xp 12GB e 64GB de memória RAM.

4.3.2 Avaliação 1 - Desempenho de Classificação

A primeira avaliação a ser realizada para nossa metodologia é quanto ao desempenho que o modelo possui para a tarefa de classificação, em termos de métricas de acurácia.

As métricas escolhidas são: acurácia balanceada, precisão, revocação e *f1-score*. A escolha foi realizada principalmente pelo fato do conjunto de dados ser desbalanceado, tornando assim necessária a escolha de métricas que fossem robustas para isso, justificando assim nossa escolha. Segue as fórmulas de cada métrica.

$$\text{Acurácia Balanceada} = \frac{1}{2}\left(\frac{\text{VP}}{\text{VP}+\text{FN}} + \frac{\text{VN}}{\text{VN}+\text{FP}}\right) \tag{8}$$

Onde VP são os verdadeiros positivos, VN os verdadeiros negativos, FP os falsos positivos e FN falsos negativos. Buscamos maximizar a métrica de acurácia balanceada por ser uma métrica eficaz que equilibra o desempenho de cada uma das classes igualmente, além de ser de fácil interpretação.

Em seguida, é apresentada a fórmula de precisão, que avalia a proporção de previsões positivas corretas em relação ao total de previsões feitas para a classe positiva.

$$\text{Precisão} = \frac{\text{VP}}{\text{VP}+\text{FP}} \tag{9}$$

A próxima equação é a que define a métrica de revocação, que mede a proporção das previsões positivas corretas em relação a todos os exemplos positivos.

$$\text{Revocação} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{10}$$

E por último, segue a equação da métrica de *f1-score*, que apresenta a média harmônica da precisão e revocação.

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{11}$$

4.3.3 Avaliação 2 - Consumo de Memória

Nossa segunda avaliação diz respeito à quantidade de memória necessária para o treinamento do modelo, assim como o necessário para o uso apenas para inferência.

A avaliação do consumo de memória durante a criação e utilização do modelo é um aspecto fundamental em nosso trabalho, pois é este fator que determinará a viabilidade de implementação da ferramenta para outros pesquisadores ou usuários da rede. A eficiência em termos de memória é crucial para garantir que o modelo possa ser escalado e aplicado em diferentes contextos com recursos computacionais variados, sem comprometer o desempenho ou acessibilidade.

Iremos separar esta etapa em três comparações, a comparação da quantidade de parâmetros do modelo gerado a partir da nossa abordagem com outros modelos *baseline*, o quanto de memória é utilizado durante o treinamento, já que o otimizador também consome memória para o cálculo dos gradientes, e a quantidade de memória necessária para realizar a inferência. Os resultados demonstrarão qual o mínimo de memória necessária para cada etapa da criação ou uso do modelo.

4.3.4 Avaliação 3 - Tempo de Execução

A terceira avaliação é baseada no tempo de execução necessário para o treinamento do algoritmo e para realizar a inferência de cada exemplo.

Essa avaliação é essencial para compreender a eficiência do modelo em ambientes onde a rapidez nas operações pode impactar diretamente a produtividade dos pesquisadores e usuários. A análise do tempo de execução permitirá identificar possíveis otimizações e garantirá que a ferramenta seja viável para aplicações em larga escala.

Assim como a avaliação anterior, separaremos esta análise em duas comparações, a comparação do tempo de execução do treinamento da rede com o de outros modelos *baseline*, e a comparação do tempo de execução para a inferência em diferentes exemplos.

4.3.5 Avaliação 4 - Gasto Energético

Nossa quarta avaliação será realizada em termos de gasto energético durante o treinamento do modelo e em um cenário hipotético de utilização da abordagem em um ambiente de produção.

Termos ciência do gasto energético necessário para o treinamento e uso do modelo é importante em um cenário como o atual, onde a sustentabilidade e a eficiência energética estão se tornando cada vez mais relevantes. Não só em relação a sustentabilidade, mas este gasto também diz respeito ao custo de energia que os implementadores da abordagem deverão arcar. Dessa forma, essa avaliação se torna essencial para a garantia de que a implementação do modelo não seja apenas eficaz, mas também economicamente sustentável e que minimize o impacto ambiental.

Também dividiremos esta avaliação entre duas etapas, uma etapa abordando o gasto energético durante o treinamento, e outra seguindo um cenário hipotético de uso.

5 DISCUSSÃO DOS RESULTADOS

Neste capítulo, são abordados os resultados obtidos através da metodologia apresentada neste trabalho em comparação com outros métodos frequentemente utilizados para a resolução da tarefa de ASC. Cada seção é referente a uma análise realizada, sendo elas descritas no capítulo de metodologia.

O melhor modelo gerado seguindo a metodologia descrita anteriormente alcançou o valor de 0.705 de acurácia balanceada média entre *folds*. Seus parâmetros são:

- *Learning Rate* - 1.458829872029105e-06.
- Tamanho de *Batch* - 2.
- *Weight Decay* - 4.977158795334005e-07.
- AMSGrad - *True*.
- ϵ - 2.1143277753859043e-06.
- Quantidade de Épocas - 83.

As avaliações seguintes serão baseadas no uso de um modelo seguindo a configuração mencionada acima, treinado contendo todos os dados de treino e validação.

5.1 Avaliação 1 - Desempenho de Classificação

Como mencionado no início do capítulo da metodologia, o conjunto de dados selecionado para nossa abordagem é o mesmo da competição ABSAPT 2022. O motivo da escolha se deu por existirem poucos conjuntos de dados para a tarefa de ABSA curados para a língua portuguesa, somando-se a possibilidade do uso dos modelos competidores para a avaliação da nossa metodologia. O *dataset* de teste que utilizamos nesta abordagem é o mesmo usado para a avaliação dos competidores do ABSAPT 2022 na tarefa de ASC, tornando assim possível o uso das abordagens como modelos *baseline*.

A tabela 6 apresenta as métricas de resultado sobre o conjunto de testes de cada um dos times participantes, retirados do artigo que apresenta os resultados, em comparação com as métricas obtidas através da nossa abordagem que escolhemos nomear como GAT-PT. As métricas são em ordem: Acurácia Balanceada, Precisão, Revocação e *F1-Score macro*. Os trabalhos desenvolvidos pelos times foram apresentados no capítulo de trabalhos relacionados, onde foram descritos os modelos utilizados e quais as tarefas empregadas para solucionar o problema de ASC.

Tabela 6 – Resultados no conjunto de teste em comparação com os competidores do ABSAPT 2022.

Abordagem	BAcc	Precisão	Revocação	F1-Score
Time Deep Learning Brasil	**0.82**	**0.81**	**0.82**	**0.81**
Time PiLN	0.78	0.76	0.78	0.77
GAT-PT (*nossa abordagem*)	0.74	0.75	0.74	0.74
Time UFSCAR	0.62	0.65	0.62	0.61
Time UFPR	0.62	0.65	0.62	0.61

Como podemos observar pela tabela, nossa abordagem superou as métricas de classificação do terceiro e quarto lugar, porém obteve métricas inferiores aos do segundo e primeiro. Porém é possível perceber que as métricas estão relativamente próximas do segundo lugar, com cerca de quatro pontos decimais de distância e mais distantes do terceiro.

Relembrando a discussão iniciada no capítulos dos trabalhos relacionados, todos os participantes da competição empregaram modelos baseados em Transformers em suas metodologias, com menção especial ao time Deep Learning Brasil, que criou um *ensemble* de quatro modelos PTT5 *large* em seu trabalho. Para fins de comparação, um modelo PTT5 possui cerca de 740 milhões de parâmetros, logo o *ensemble* de quatro modelos PTT5 *large* possui 2 bilhões e 960 milhões de parâmetros. O modelo gerado pela nossa abordagem necessita de menos de 2% da quantidade de parâmetros para obter um resultado com menos de dez pontos decimais de diferença em qualquer métrica.

Já para as demais abordagens, todas empregaram o treinamento de modelos BERTimbau para realizar a classificação. Principalmente a versão *Large*, que possui 335 milhões de parâmetros, cerca de dez vezes mais do que o necessário para nossa abordagem. Alguns autores não deixaram claro qual a versão do modelo BERTimbau foi utilizada em seus trabalhos, porém, mesmo o modelo base ainda possui aproximadamente 110 milhões de parâmetros, um número muito superior ao nosso. Modelos com muitos parâmetros tendem a ser mais lentos do que com menos parâmetros, porém não temos como demonstrar estas comparações com os trabalhos submetidos à competição.

Devido ao motivo de não termos acesso aos modelos gerados pelos competidores, optamos por criar modelos *baseline* baseados nos modelos BERTimbau *Large*, BERTimbau base e DeBERTinha (CAMPIOTTI et al., 2023). Todos os modelos foram treinados no mesmo computador que utilizamos para os experimentos da nossa abordagem. Os modelos foram treinados por meio de uma classificação de sentença com sentença auxiliar, onde a sentença é o *review* e a sentença auxiliar é o termo do aspecto. Dessa forma, podemos comparar nossa metodologia com outros modelos mais comuns nas outras abordagens, em termos de consumo de memória, tempo de execução e consumo energético. O modelo DeBERTinha foi escolhido por ser baseado em uma arquitetura *encoder-only* com menos parâmetros do que os demais, porém seus resultados são competitivos com as duas outras arquiteturas, alcançando até resultados superiores em algumas tarefas (CAMPIOTTI et al., 2023).

Na tabela 7, apresentamos os resultados obtidos seguindo a metodologia definida neste trabalho em comparação com os modelos gerados para *baseline*.

Tabela 7 – Resultados no conjunto de teste em comparação com os modelos criados para *baseline*.

Abordagem	BAcc	Precisão	Revocação	F1-Score
BERTimbau *Large*	**0.78**	**0.79**	**0.78**	**0.78**
BERTimbau Base	0.77	0.77	0.77	0.77
DeBERTinha	0.76	0.78	0.76	0.77
GAT-PT (*nossa abordagem*)	0.74	0.75	0.74	0.74

Como podemos observar, a abordagem baseada no BERTimbau *Large* apresentou os melhores resultados em todas as métricas, superando inclusive o modelo gerado pelo time PiLN durante a competição ABSAPT. Em comparação, o modelo BERTimbau Base mostrou desempenho ligeiramente inferior em todas as métricas, perdendo consistentemente para sua versão *Large* e apresentando uma precisão inferior ao modelo DeBERTinha. O DeBERTinha por sua vez, obteve uma acurácia balanceada e revocação apenas um ponto percentual inferior ao BERTimbau Base, com resultados próximos também na métrica F1. Nossa abordagem, embora tenha apresentado os menores valores nas diferentes métricas, apresentou uma diferença relativamente pequena. Em termos de acurácia balanceada, por exemplo, a diferença para o modelo que alcançou o primeiro lugar foi de apenas quatro pontos percentuais, o que demonstra uma competitividade razoável, mesmo diante de modelos significativamente maiores.

Dito isto, as próximas análises demonstram como nosso modelo é competitivo, mesmo diante a métricas inferiores.

5.2 Avaliação 2 - Consumo de Memória

Em nossa segunda avaliação, iremos analisar o consumo de memória de cada uma das arquiteturas, tanto no processo de treinamento quanto durante a inferência. Essa análise é fundamental para compreender a viabilidade do uso dos modelos em cenários com recursos computacionais limitados.

A avaliação inclui a quantidade de parâmetros presente em cada arquitetura, o que nos permite entender o potencial impacto no uso de memória. Além disso, é monitorado o espaço ocupado na GPU durante as etapas de treinamento e inferência, para identificar o comportamento de consumo de memória em diferentes fases do experimento.

A figura 16 apresenta a quantidade de parâmetros de cada arquitetura, acompanhada por um gráfico de barras que facilita a comparação visual entre os modelos.

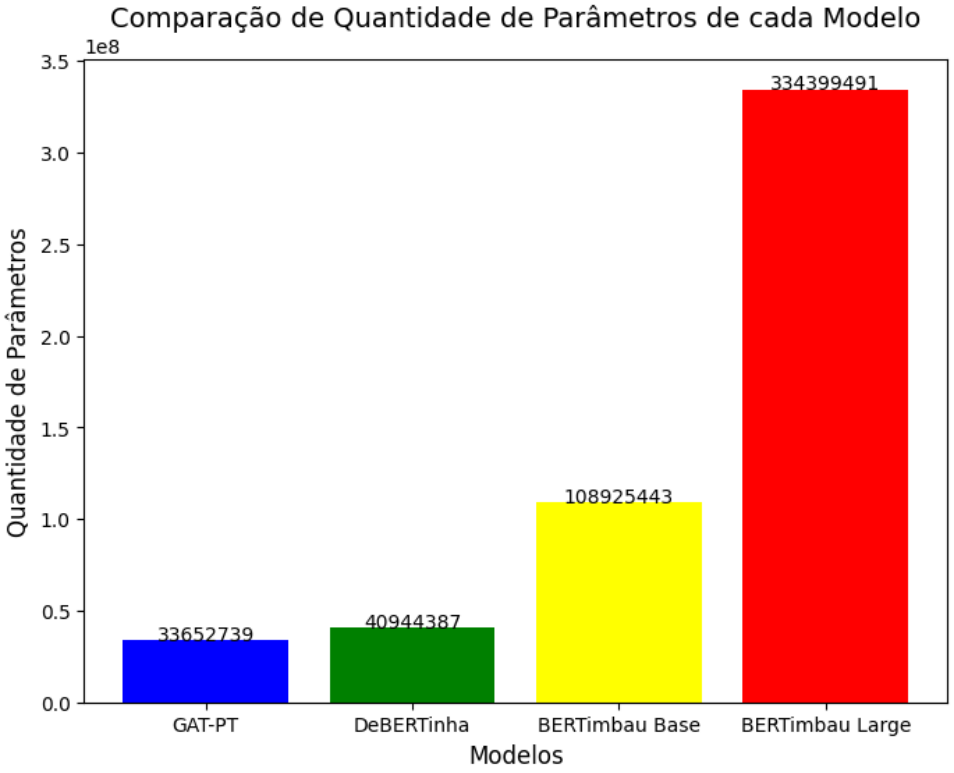

Figura 16 – Comparação da quantidade de parâmetros de cada arquitetura. Fonte: Autoria Própria.

A ilustração permite visualizar com maior clareza a diferença na quantidade de parâmetros entre os modelos. O modelo GAT-PT, com aproximadamente 34 milhões de parâmetros, é o menor entre eles, seguido pelo DeBERTinha, que possui 41 milhões. O BERTimbau Base, com 109 milhões de parâmetros, ocupa a terceira posição, enquanto a versão *Large* do BERTimbau é a mais complexa, com 334 milhões de parâmetros.

A quantidade de parâmetros nos ajuda a estimar qual o valor mínimo de memória

que precisamos ter disponível para a utilização do modelo durante a inferência. Se assumirmos cada valor de parâmetro como uma representação de 4 bytes, fica evidente de que precisaremos de cerca de 129 megabytes de memória para o modelo GAT-PT, 157 megabytes para o DeBERTinha, 416 megabytes para a versão Base do BERTimbau e 1.276 megabytes para a versão *Large*. Por si só, já podemos ver uma diferença discrepante entre tamanhos, porém isso se torna mais acentuado quando analisamos estes números durante o treinamento.

Durante o algoritmo de *backpropagation*, são calculados os gradientes para a atualização de pesos, e a depender do otimizador este processo pode aumentar em algumas vezes a quantidade de memória necessária. Um exemplo é o otimizador AdamW, que cria a necessidade de pelo menos duas vezes o tamanho do modelo para armazenar os estados do otimizador (ZHANG et al., 2024). Além disso, é importante que os dados sejam carregados na GPU para acelerar o processo de treinamento, o que promove ainda mais consumo de memória.

A figura 17 apresenta uma comparação da memória utilizada por cada modelo durante o seu processo de treinamento. Vale destacar que os modelos *baseline* foram todos treinados utilizando tamanho de *batch* 8, enquanto o melhor modelo encontrado para a arquitetura GAT-PT foi com tamanho 2. Além disso, os valores foram obtidos por meio da ferramenta Nvidia-SMI.

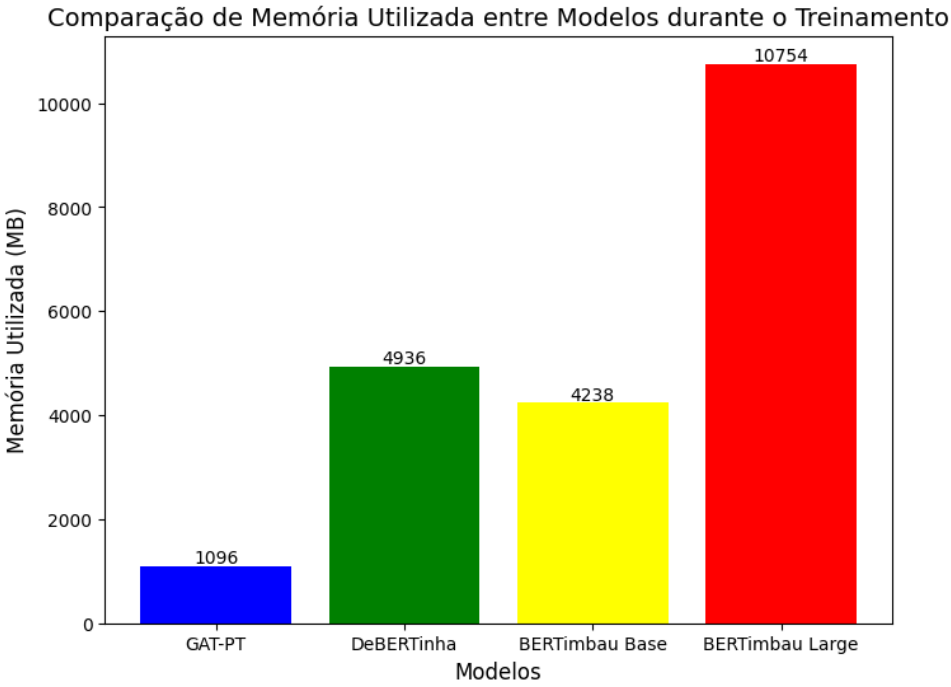

Figura 17 – Comparação da memória utilizada por cada modelo durante o processo de treinamento. Fonte: Autoria Própria.

De fato o tamanho de *batch* escolhido como hiperparâmetro afeta tanto a quantidade de memória necessária quanto a velocidade de execução e acurácia do modelo.

Porém, durante nossos experimentos esse conjunto obteve os melhores resultados, e optamos por não realizar uma procura mais extensa para não correr risco de causar *overfitting* no modelo. Como referência, realizamos experimentos para descobrir qual seria a quantidade de memória necessária para realizar o treinamento do modelo GAT-PT com um tamanho de *batch* 8, e o valor encontrado foi de 1.263 megabytes, representando um aumento modesto em comparação com as demais abordagens.

Podemos notar uma diferença mais acentuada na quantidade de memória necessária para o treinamento de cada arquitetura, com a maioria dos modelos seguindo um comportamento parecido com a quantidade de parâmetros, exceto o modelo DeBERTinha.

Em nossos testes, verificamos que o aumento do tamanho de *batch* durante o treinamento do modelo DeBERTinha causou um aumento da quantidade de memória necessária em uma escala muito maior do que os outros modelos, onde o treinamento com o tamanho de *batch* 1 utilizava 1.434 megabytes de memória, com tamanho 2 necessitava de 1.920 megabytes, com 4 aumentava para 2.818 megabytes e com 8 para 4.936 megabytes.

Acreditamos que isso tenha relação com o fato da arquitetura do modelo ser essencialmente diferente dos demais, onde faz uso de uma matriz de viés de posição relativa como parte do mecanismo de atenção do DeBERTinha. Esta matriz possui as dimensões LxL, onde L é o comprimento máximo da sentença, e cada valor representa um par de tokens. No entanto, no artigo original não são disponibilizadas informações o suficiente para validarmos essa hipótese, categorizando uma limitação da nossa análise no presente estudo.

Porém, é possível notar que em nossos experimentos, o GAT-PT foi o que promoveu melhor uso da memória durante o treinamento, podendo ser realizada em uma GPU com menos de 2 gigabytes de memória.

Para a análise de uso de memória durante o processo de inferência, optamos por anotar o tempo necessário que o modelo levou para estimar as polaridades de sentimento de todo o conjunto de teste. Em nossos experimentos, utilizamos o tamanho de *batch* 8. A figura 18 demonstra os resultados desse experimento.

Pelo gráfico, podemos observar que as diferenças acentuadas de memória utilizada por cada modelo foi alterada. Isso se deve ao motivo de que nossa abordagem utiliza vetores de características de tamanho 4.096, o quadruplo dos modelos da família BERTimbau, o que penaliza em consumo de memória tamanhos de *batch* muito altos. O modelo DeBERTinha possui um comportamento parecido, mesmo com o fato de que o vetor de características possui um tamanho menor do que os demais, acreditamos que seja pelo mesmo motivo do experimento anterior, em relação ao treinamento.

Mesmo assim, assumindo um *batch* de tamanho 8, podemos observar que o GAT-

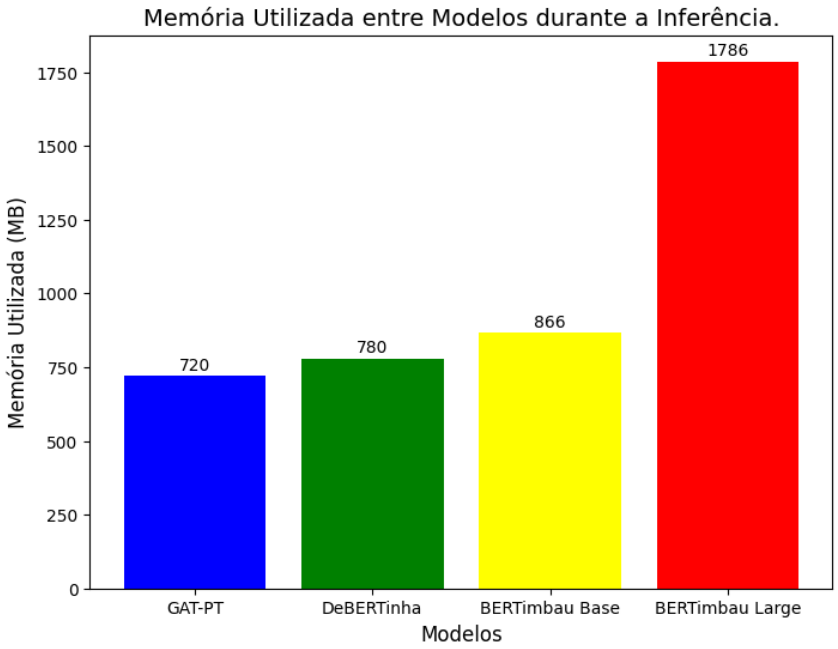

Figura 18 – Comparação da memória utilizada por cada modelo durante a inferência. Fonte: Autoria Própria.

PT é o que utiliza menos espaço em comparação aos demais modelos *baseline*.

5.3 Avaliação 3 - Tempo de Execução

Nossa terceira avaliação foca no tempo de execução da abordagem. Para medir o tempo de treinamento, utilizamos os dados fornecidos pela biblioteca *tqdm* da linguagem Python, que monitora a duração de cada etapa e o tempo total gasto. Já para a inferência, recorremos à biblioteca *time*, que permite uma medição precisa do tempo gasto na execução de previsões.

Nossa avaliação abrange o tempo de execução necessário para o treinamento de cada arquitetura, o tempo total gasto para a inferência de todo o conjunto de teste e o tempo médio de execução por *batch*. Essas métricas são essenciais para entender a eficiência de cada modelo, tanto em cenários de treinamento intensivo quanto em aplicações de inferência, oferecendo uma visão abrangente sobre o desempenho temporal de cada abordagem.

A figura 19 apresenta uma comparação do tempo de execução gasto por cada modelo durante o processo de treinamento.

A figura deixa evidente que o GAT-PT acaba sendo mais lento do que as arquiteturas DeBERTinha e BERTimbau Base durante o processo de treinamento. Isso se dá por dois fatores principais, o tamanho de *batch* e a quantidade de épocas. O tamanho de *batch* tem um papel fundamental para atribuirmos a paralelização durante a inferência ou treinamento, e possuir um valor pequeno para esse hiperparâmetro acaba limitando esse processo, além disso, a quantidade de épocas dita quantas vezes os

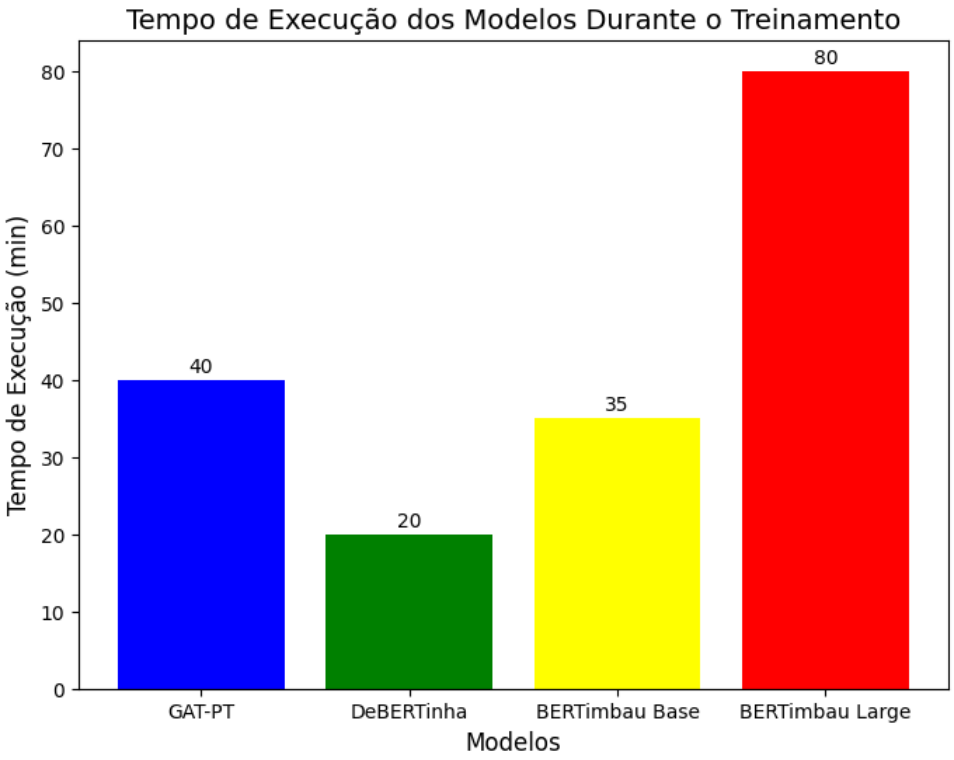

Figura 19 – Comparação do tempo de execução necessário para o treinamento dos modelos. Fonte: Autoria Própria.

exemplos serão apresentados para a rede, então ter um valor alto de épocas também afeta o tempo de treinamento. Como mencionado anteriormente, essa configuração foi a que obteve a melhor acurácia balanceada durante a seleção de hiper-parâmetros, optamos por não explorar mais para não corrermos o risco de *overfitting*.

Para colocarmos em perspectiva a velocidade de execução por época, o GAT-PT foi treinado durante 83 épocas, com uma média de 29 segundos por época. Em contrapartida, o modelo DeBERTinha completou 10 épocas, levando aproximadamente 120 segundos por época. O BERTimbau *Base* também foi treinado por 10 épocas, com um tempo médio de 210 segundos por época. Já a versão *Large* do BERTimbau realizou 8 épocas, com cada uma delas exigindo cerca de 600 segundos. Essa comparação evidencia as diferenças significativas no tempo de treinamento por época entre as arquiteturas.

Considerando a aplicação do modelo em um ambiente de produção, é fundamental conhecer o tempo médio necessário para a rede realizar a inferência. Essa métrica nos permite identificar sua capacidade de escalar o modelo para demandas maiores. Com essa informação, podemos determinar sua adequação para aplicações em larga escala, onde a velocidade de inferência é crucial.

Esses experimentos foram realizados em conjunto com os experimentos de utilização de memória, ou seja, também foram realizados utilizando um tamanho de *batch* 8, e para todos os dados de teste. Para realizar a análise, executamos 10 rodadas

de aquecimento antes de fazer a medição do tempo gasto para a inferência de todo o conjunto de dados destinado a teste.

A figura 20 apresenta o tempo de execução de cada modelo para fazer a inferência em todo o conjunto de teste.

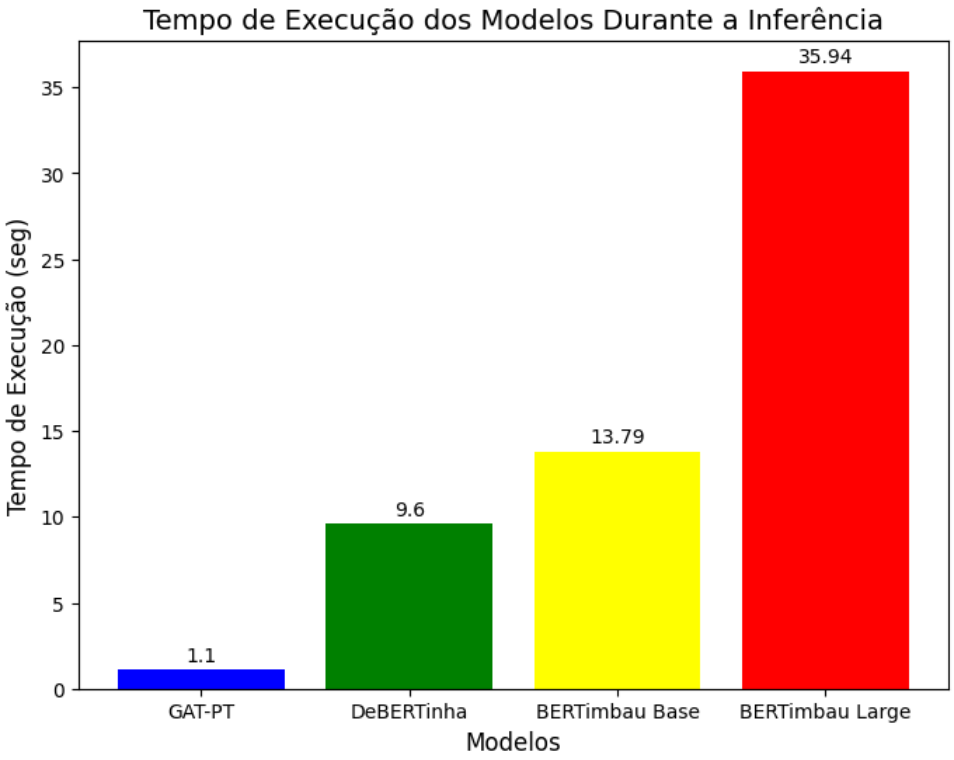

Figura 20 – Comparação do tempo de execução necessário para a inferência dos modelos em todo conjunto de teste. Fonte: Autoria Própria.

É possível observar que o modelo GAT-PT se destaca por ser significativamente mais veloz que os demais. Isso ocorre devido à sua arquitetura mais simples, com conexões mais significativas e menos esparsas, o que resulta em maior eficiência computacional. Por outro lado, os modelos da família BERTimbau, que compartilham uma arquitetura similar, mostram um crescimento no tempo de execução proporcional ao aumento do número de parâmetros, exibindo um comportamento quase linear. O modelo DeBERTinha, embora possua menos da metade dos parâmetros do BERTimbau Base, apresenta uma arquitetura mais complexa. Devido a essa complexidade, o tempo de inferência do DeBERTinha chegou próximo ao do BERTimbau Base, mesmo com sua menor quantidade de parâmetros. Isso reflete a influência do design arquitetônico na eficiência do modelo, além do mero número de parâmetros.

Com base nos valores apresentados, é possível calcular o tempo médio gasto por *batch* de exemplos. Utilizando *batches* de 8 exemplos cada, resultando em um total de 86 *batches*, o modelo GAT-PT levou aproximadamente 0,01 segundos por *batch*, enquanto o modelo DeBERTinha gastou em torno de 0,11 segundos. Já o BERTimbau Base utilizou 0,16 segundos por *batch*, e sua versão *Large* demandou cerca de 0,42

segundos. Dessa forma evidenciamos ainda mais a diferença de tempo de execução necessário para cada modelo.

5.4 Avaliação 4 - Gasto Energético

Nesta seção, abordamos o consumo energético de cada modelo durante a fase de treinamento, além de analisar um cenário hipotético de uso contínuo. Esta avaliação é fundamental para entendermos o impacto do gasto energético das diferentes arquiteturas, tanto em termos econômicos quanto ambientais. Considerar esses fatores permite não apenas otimizar os custos operacionais, mas também reduzir a pegada ambiental do modelo, promovendo uma abordagem mais sustentável e eficiente.

A análise será realizada por meio da ferramenta Nvidia-SMI, uma interface da Nvidia que dá informações gerais do uso da placa de vídeo em tempo real. Realizaremos uma estimativa de consumo energético baseado em uma média de 20 medições tiradas em intervalos de 1 segundo durante o treinamento e inferência.

A fórmula utilizada para o cálculo do consumo energético, em kWh, é apresentada na equação 12. Em nossos experimentos, apenas medimos os tempos em segundos, criando a necessidade da mudança de escala do tempo para segundos.

$$kWh = \frac{\text{Potência (W)} \times \frac{\text{Tempo (s)}}{3600}}{1000} \tag{12}$$

A figura 21 apresenta uma comparação da energia gasta durante o treinamento de cada modelo em kWh.

A figura evidencia que o GAT-PT foi a segunda arquitetura com maior demanda de energia durante o treinamento, ficando atrás apenas do BERTimbau *Large*. Esse resultado era previsível, dado que o treinamento do GAT-PT foi mais demorado em relação ao BERTimbau Base e ao DeBERTinha, o que aumentou o tempo de uso da GPU. Apesar de a rede GAT-PT apresentar um consumo médio de energia por unidade de tempo inferior aos demais, o longo tempo de execução fez com que o consumo total de energia se elevasse.

Para ilustrar essa análise, criamos um cenário hipotético no qual cada modelo realiza a inferência de 1 milhão de exemplos. Considerando *batches* de tamanho 8, os exemplos são divididos em 125.000 *batches*. Utilizando o tempo médio de inferência por *batch* de cada modelo, calculamos o tempo total gasto para processar todos os exemplos em segundos.

Com o tempo total obtido, aplicamos as estimativas de consumo de energia médio registradas durante a inferência para cada arquitetura, a fim de calcular a quantidade de energia que seria consumida para processar 1.000.000 de exemplos. A figura 22 apresenta uma comparação clara dos valores estimados de consumo de energia para

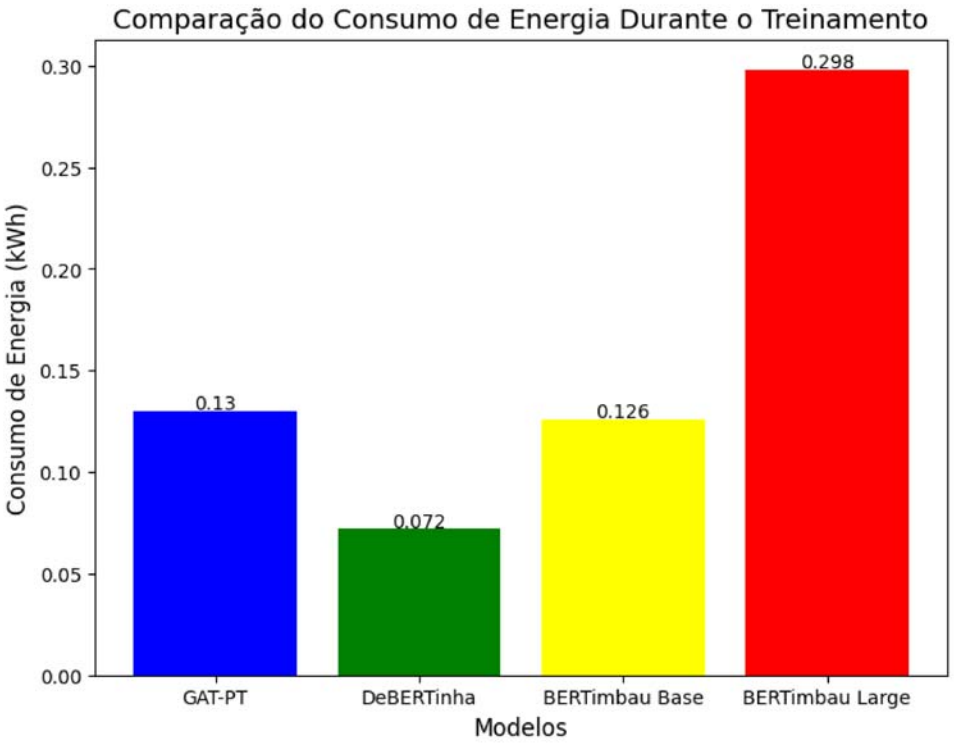

Figura 21 – Comparação da energia gasta durante o processo de treinamento. Fonte: Autoria Própria.

cada modelo, expressos em kWh.

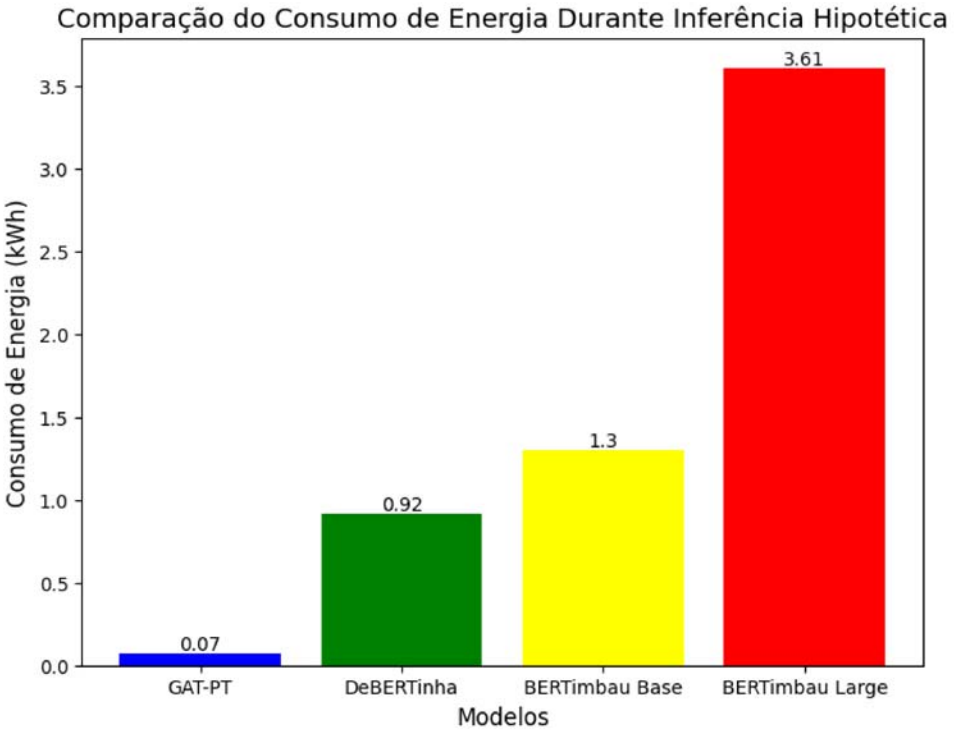

Figura 22 – Comparação da energia gasta durante a inferência hipotética de 1.000.000 de exemplos. Fonte: Autoria Própria.

Seguindo esse exemplo, podemos inferir que nossa abordagem demonstra uma

eficiência energética superior em comparação com os outros modelos testados, ao menos no processo de inferência. Ressaltando um menor uso de energia para a inferência de 1 milhão de exemplos do que para o processo de treinamento. Esse comportamento reforça a viabilidade do modelo em ambientes de produção, onde o uso de energia durante a inferência é um fator crucial, especialmente em aplicações que envolvem grandes volumes de dados.

6 CONCLUSÃO

Este trabalho apresentou uma avaliação em termos de desempenho de classificação, consumo de memória, tempo de execução e gasto energético de uma rede de grafos com atenção para a análise de sentimentos em nível de aspecto para a língua portuguesa brasileira.

Os experimentos indicam que a metodologia proposta alcança resultados competitivos quando comparada a outros modelos amplamente utilizados para a tarefa de ASC, os quais geralmente são baseados em arquiteturas Transformer. Embora essas redes, como os modelos da família BERT, sejam reconhecidas por sua eficácia e frequentemente consideradas estado da arte em diversas tarefas de PLN, elas também tendem a possuir um grande número de parâmetros, o que implica na exigência de *hardware* custoso, com grande quantidade de memória e alta capacidade de processamento. O modelo gerado pela abordagem proposta superou o terceiro colocado na competição ABSAPT em todas as métricas avaliadas. Devido ao fato do modelo usado nas metodologias do segundo e terceiro lugar serem baseados no BERTimbau, podemos inferir que o modelo gerado pela abordagem proposta se destaca por ser mais eficiente energeticamente, além de menor e mais rápido, visto que nossas análises levaram em consideração abordagens que utilizam a mesma arquitetura.

Como mencionado, nossa arquitetura se destaca por utilizar significativamente menos parâmetros do que os modelos comparados, resultando em um consumo de memória reduzido tanto durante o treinamento quanto na inferência. No entanto, o vetor de características do nosso modelo é baseado nos valores de saída das quatro últimas camadas do BERTimbau *Large*, o que o torna consideravelmente maior, com dimensão 4.096. Isso implica que, à medida que o tamanho do *batch* aumenta, o consumo de memória para o treinamento e inferência também tende a ser mais elevado, aproximando-se do consumo dos modelos da família BERTimbau, apesar de nossa rede possuir menos parâmetros.

Apesar da necessidade de utilizar *batches* menores, o modelo proposto se destaca por sua alta velocidade de inferência, exigindo quase um décimo do tempo necessário em comparação com o segundo modelo mais rápido avaliado. Essa rapidez contri-

bui significativamente para a economia de energia durante a inferência, tornando-o altamente eficiente para aplicações em larga escala. Além disso, a combinação de menor consumo de memória e alta velocidade faz com que o modelo seja ideal para cenários que demandam redes rápidas e energeticamente eficientes, reforçando sua viabilidade para ambientes de produção.

Nossos experimentos comprovam que a utilização de GATs para a aplicação da subtarefa ASC de ABSA pode ser uma boa alternativa a modelos baseados em Transformer, sendo uma alternativa mais compacta, rápida e energeticamente eficiente, ao custo de um desempenho de classificação inferior.

Caso conjuntos de dados maiores possam ser obtidos para o treinamento, poderemos ter experimentos futuros com a utilização de mais camadas de GAT, além de mais cabeças de atenção. Isso seria necessário, visto que creditamos a escassez de dados um fenômeno observado em nossos experimentos: o aumento de complexidade da arquitetura piorou o desempenho da abordagem.

A exploração de novas técnicas de *embedding* também pode ser uma alternativa para a melhora no desempenho de classificação além de poder tornar o modelo ainda mais compacto. Nossas análises se basearam apenas nas quatro últimas saídas do modelo BERTimbau *Large*, tornando assim o vetor de características extenso.

A análise de outros modelos de geração de árvores de dependência pode ser uma estratégia valiosa para aprimorar as representações em grafos utilizadas neste trabalho. Embora tenhamos adotado o modelo oferecido pelo *framework* SpaCy, reconhecido por seus bons resultados e disponibilidade, é importante destacar que o desempenho do gerador de árvores de dependência sintática desempenha um papel crucial na construção dos grafos, impactando diretamente as representações finais e, consequentemente, o desempenho da abordagem.

Além disso, a limitação de ferramentas de PLN para a língua portuguesa, uma língua com poucos recursos, representa um obstáculo significativo. Esse cenário restringe a diversidade de opções disponíveis para experimentação e, por sua vez, afeta o desenvolvimento de soluções mais robustas e eficientes. Dessa forma, a disponibilidade de ferramentas adequadas e de alta qualidade é um fator determinante para a realização de trabalhos de PLN com maior precisão e profundidade.

Esperamos que o presente estudo possa enriquecer o campo de PLN no Brasil e acreditamos que esta metodologia possa inspirar futuras pesquisas e soluções voltadas para idiomas com menos recursos, promovendo avanços tanto no desempenho dos modelos quanto na acessibilidade a tecnologias de ponta.

REFERÊNCIAS

AKIBA, T. et al. **Optuna**: A Next-generation Hyperparameter Optimization Framework. Disponível em: <https://arxiv.org/abs/1907.10902>.

ALI, M. et al. Tokenizer Choice For LLM Training: Negligible or Crucial? **arXiv preprint arXiv:2310.08754**, online, 2023.

ALMAZROUEI, E. et al. **The Falcon Series of Open Language Models**. Disponível em: <https://arxiv.org/abs/2311.16867>.

ASSI, F. M. et al. UFSCar's Team at ABSAPT 2022: Using Syntax, Semantics and Context for Solving the Tasks. In: IBERLEF@ SEPLN, 2022. **Anais...** SEPLN, 2022.

ASSIS, R. et al. Uso de Redes Neurais Artificiais para o desenvolvimento de modelos de previsão da condição de pavimentos de aeroportos. In: PLURIS, 2016, Maceió, Brasil. **Anais...** Viva Editora, 2016.

AZIZ, M. M.; BAKAR, A. A.; YAAKUB, M. R. CoreNLP dependency parsing and pattern identification for enhanced opinion mining in aspect-based sentiment analysis. **Journal of King Saud University-Computer and Information Sciences**, Riyadh, Saudi Arabia, v.36, n.4, p.102035, 2024.

BASODI, S. et al. Gradient amplification: An efficient way to train deep neural networks. **Big Data Mining and Analytics**, [S.l.], v.3, n.3, p.196–207, 2020.

BEHDENNA, S.; BARIGOU, F.; BELALEM, G. Document level sentiment analysis: a survey. **EAI endorsed transactions on context-aware systems and applications**, [S.l.], v.4, n.13, p.e2–e2, 2018.

BENGIO, Y.; DUCHARME, R.; VINCENT, P. A neural probabilistic language model. **Advances in neural information processing systems**, Denver, CO, USA, v.13, 2000.

CAMPIOTTI, I. et al. **DeBERTinha**: A Multistep Approach to Adapt DebertaV3 XSmall for Brazilian Portuguese Natural Language Processing Task.

CARMO, D. et al. **PTT5**: Pretraining and validating the T5 model on Brazilian Portuguese data. Disponível em: <https://arxiv.org/abs/2008.09144>.

CHAVES, M.; FREITAS, L.; VIEIRA, R. Hontology: a multilingual ontology for the accommodation sector in the tourism industry. , [S.l.], 2012.

CORRÊA, U. B. **Análise de sentimento baseada em aspectos usando aprendizado profundo**: uma proposta aplicada à língua portuguesa. 2021. Tese (Doutorado em Ciência da Computação) — Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pelotas, Pelotas.

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **CoRR**, online, v.abs/1810.04805, 2018.

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

DONG, L. et al. Adaptive recursive neural network for target-dependent twitter sentiment classification. In: SHORT PAPERS), 52., 2014. **Proceedings...** ACL, 2014. p.49–54.

FEY, M.; LENSSEN, J. E. **Fast Graph Representation Learning with PyTorch Geometric**. Disponível em: <https://arxiv.org/abs/1903.02428>.

FREITAS, L. **Feature-level sentiment analysis applied to brazilian portuguese reviews**. 2015. Tese (Doutorado em Ciência da Computação) — Programa de Pós-Graduação em Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre.

GAO, Y. et al. Character-level translation with self-attention. **arXiv preprint arXiv:2004.14788**, online, 2020.

GOMES, G. et al. Evaluating Graph Attention Networks as an Alternative to Transformers for ABSA Task in Low-Resource Languages. **The International FLAIRS Conference Proceedings**, FL, USA, v.37, n.1, May 2024.

GOMES, J. R. S. et al. Deep learning brasil at absapt 2022: Portuguese transformer ensemble approaches. **arXiv preprint arXiv:2311.05051**, online, 2023.

HAMILTON, W. L.; YING, R.; LESKOVEC, J. **Inductive Representation Learning on Large Graphs**. Disponível em: <https://arxiv.org/abs/1706.02216>.

HAMMES, L.; FREITAS, L. Utilizando BERTimbau para a Classificação de Emoções em Português. In: XIII SIMPóSIO BRASILEIRO DE TECNOLOGIA DA INFORMAçãO

E DA LINGUAGEM HUMANA, 2021, Porto Alegre, RS, Brasil. **Anais...** SBC, 2021. p.56–63.

HEINRICH, T.; MARCHI, F. TeamUFPR at ABSAPT 2022: Aspect Extraction with CRF and BERT. In: IBERLEF@ SEPLN, 2022. **Anais...** SEPLN, 2022.

HONNIBAL, M. et al. spaCy: Industrial-strength Natural Language Processing in Python. , online, 2020.

JAGTAP, V.; PAWAR, K. Analysis of different approaches to sentence-level sentiment classification. **International Journal of Scientific Engineering and Technology**, Bhopal, MP, India, v.2, n.3, p.164–170, 2013.

JIANG, Q. et al. A challenge dataset and effective models for aspect-based sentiment analysis. In: EMNLP-IJCNLP), 2019., 2019. **Proceedings...** ACL, 2019. p.6280–6285.

KHARDE, V.; SONAWANE, P. et al. Sentiment analysis of twitter data: a survey of techniques. **arXiv preprint arXiv:1601.06971**, online, 2016.

KIPF, T. N.; WELLING, M. **Semi-Supervised Classification with Graph Convolutional Networks**. Disponível em: <https://arxiv.org/abs/1609.02907>.

KIRITCHENKO, S.; ZHU, X.; MOHAMMAD, S. Sentiment Analysis of Short Informal Text. **The Journal of Artificial Intelligence Research (JAIR)**, El Segundo, CA, EUA, v.50, 08 2014.

KOLKUR, S.; DANTAL, G.; MAHE, R. Study of different levels for sentiment analysis. **International Journal of Current Engineering and Technology**, [S.l.], v.5, n.2, p.768–770, 2015.

LEWIS, M. et al. **BART**: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Disponível em: <https://arxiv.org/abs/1910.13461>.

LI, R. et al. Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS AND THE 11TH INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING (VOLUME 1: LONG PAPERS), 59., 2021, Online. **Proceedings...** Association for Computational Linguistics, 2021.

LI, X. et al. The Scope of Part of Speech Tagging: A Bibliometric Study. **Lecture Notes on Language and Literature**, [S.l.], v.7, n.4, p.51–58, 2024.

LIANG, S. et al. BiSyn-GAT+: Bi-syntax aware graph attention network for aspect-based sentiment analysis. **arXiv preprint arXiv:2204.03117**, online, 2022.

LIU, B. **Sentiment Analysis and Opinion Mining**. [S.l.]: Morgan & Claypool Publishers, 2012.

LIU, Y. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. **CoRR**, [S.l.], v.abs/1907.11692, 2019.

LOSHCHILOV, I.; HUTTER, F. **Decoupled Weight Decay Regularization**. Disponível em: <https://arxiv.org/abs/1711.05101>.

LUO, Z.; XU, H.; CHEN, F. Audio Sentiment Analysis by Heterogeneous Signal Features Learned from Utterance-Based Parallel Neural Network. In: AFFCON@ AAAI, 2019. **Anais...** [S.l.: s.n.], 2019. p.80–87.

MACHADO, M. T.; PARDO, T. A.; RUIZ, E. E. S. Creating a Portuguese context sensitive lexicon for sentiment analysis. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, 2018. **Anais...** Springer, 2018. p.335–344.

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

MRINI, K. et al. Rethinking self-attention: Towards interpretability in neural parsing. **arXiv preprint arXiv:1911.03875**, online, 2019.

NETO, F. A. R. et al. Team PiLN at ABSAPT 2022: Lexical and BERT Strategies for Aspect-Based Sentiment Analysis in Portuguese. In: IBERLEF@ SEPLN, 2022. **Anais...** SEPLN, 2022.

OPENAI et al. **GPT-4 Technical Report**. Disponível em: <https://arxiv.org/abs/2303.08774>.

ORTIS, A.; FARINELLA, G. M.; BATTIATO, S. An Overview on Image Sentiment Analysis: Methods, Datasets and Current Challenges. **ICETE (1)**, Hyderabad, India, p.296–306, 2019.

PASZKE, A. et al. **PyTorch**: An Imperative Style, High-Performance Deep Learning Library. Disponível em: <https://arxiv.org/abs/1912.01703>.

PIRES, R. et al. **Sabiá**: Portuguese Large Language Models. [S.l.]: Springer Nature Switzerland, 2023. 226–240p.

PONTIKI, M. et al. Semeval-2016 task 5: Aspect based sentiment analysis. In: INTERNATIONAL WORKSHOP ON SEMANTIC EVALUATION, 2016. **Anais...** ACL, 2016. p.19–30.

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

PORIA, S. et al. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. **Neurocomputing**, Amsterdam, Netherlands, v.174, p.50–59, 2016.

QUIZA, R.; DAVIM, J. P. **Computational Methods and Optimization**. [S.l.]: Springer, 2011. p.177–208.

RAFFEL, C. et al. **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**. Disponível em: <https://arxiv.org/abs/1910.10683>.

RANA, T. A.; CHEAH, Y.-N. Aspect extraction in sentiment analysis: comparative analysis and survey. **Artificial Intelligence Review**, [S.l.], v.46, p.459–483, 2016.

REDDI, S. J.; KALE, S.; KUMAR, S. On the Convergence of Adam and Beyond. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2018. **Anais...** ICLR, 2018.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, Washington, DC, USA, v.65, n.6, p.386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, London, England, v.323, n.6088, p.533–536, 1986.

SEBASTIANI, F.; ESULI, A. Sentiwordnet: A publicly available lexical resource for opinion mining. In: OF THE 5TH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION, 2006. **Proceedings...** ACL, 2006. p.417–422.

SENNRICH, R.; HADDOW, B.; BIRCH, A. **Neural Machine Translation of Rare Words with Subword Units**. Disponível em: <https://arxiv.org/abs/1508.07909>.

SILVA, F. L. da et al. ABSAPT 2022 at iberlef: Overview of the task on aspect-based sentiment analysis in portuguese. **Procesamiento del Lenguaje Natural**, [S.l.], v.69, p.199–205, 2022.

SINGH, H.; SHARMA, R. Role of adjacency matrix & adjacency list in graph theory. **International Journal of Computers & Technology**, Amritsar, Punjab, India, v.3, n.1, p.179–183, 2012.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: Pretrained BERT Models for Brazilian Portuguese. In: INTELLIGENT SYSTEMS: 9TH BRAZILIAN CONFERENCE, BRACIS 2020, RIO GRANDE, BRAZIL, OCTOBER 20–23, 2020, PROCEEDINGS, PART I, 2020, Berlin, Heidelberg. **Anais...** Springer-Verlag, 2020. p.403–417.

SUN, K. et al. Aspect-level sentiment analysis via convolution over dependency tree. In: EMNLP-IJCNLP), 2019., 2019. **Proceedings...** ACL, 2019. p.5679–5688.

SUTSKEVER, I. Sequence to Sequence Learning with Neural Networks. **arXiv preprint arXiv:1409.3215**, online, 2014.

TAUD, H.; MAS, J.-F. Multilayer perceptron (MLP). **Geomatic approaches for modeling land change scenarios**, [S.l.], p.451–455, 2018.

TOUVRON, H. et al. **LLaMA**: Open and Efficient Foundation Language Models. Disponível em: <https://arxiv.org/abs/2302.13971>.

TRAN, T. T.; MIWA, M.; ANANIADOU, S. Syntactically-informed word representations from graph neural network. **Neurocomputing**, Amsterdam, Netherlands, v.413, p.431–443, 2020.

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

VELIčKOVIć, P. et al. Graph Attention Networks. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2018. **Anais...** ICLR, 2018.

WANG, C.; CHO, K.; GU, J. Neural machine translation with byte-level subwords. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2020. **Proceedings...** AAAI Press, 2020. v.34, n.05, p.9154–9160.

WANG, K. et al. Relational graph attention network for aspect-based sentiment analysis. **arXiv preprint arXiv:2004.12362**, online, 2020.

WATANABE, S. **Tree-Structured Parzen Estimator**: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. Disponível em: <https://arxiv.org/abs/2304.11127>.

XU, J. **Theory and application of graphs**. [S.l.]: Springer Science & Business Media, 2013. v.10.

ZHANG, M. et al. An end-to-end deep learning architecture for graph classification. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2018. **Proceedings...** AAAI Press, 2018. v.32, n.1.

ZHANG, W. et al. A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.35, n.11, p.11019–11038, 2023.

ZHANG, Y. et al. Adam-mini: Use fewer learning rates to gain more. **arXiv preprint arXiv:2406.16793**, [S.l.], 2024.

ZHU, L. et al. Deep learning for aspect-based sentiment analysis: a review. **PeerJ Computer Science**, San Diego, CA, USA, v.8, p.e1044, 2022.