PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO SOBRE GENERATIVE PRE-TRAINED TRANSFORMER (GPT)

MARCOS VINICIUS DE PAULA ALVES

GOIÂNIA
2024

MARCOS VINICIUS DE PAULA ALVES

UM ESTUDO SOBRE GENERATIVE PRE-TRAINED TRANSFORMER (GPT)

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. Fábio Barbosa Rodrigues

Banca Examinadora:
Prof. Me. Fernando Gonçalves Abadia
Prof. Me. Max Gontijo De Oliveira

GOIÂNIA
2024

MARCOS VINICIUS DE PAULA ALVES

UM ESTUDO SOBRE
GENERATIVE PRE-TRAINED TRANSFORMER (GPT)

Este Trabalho de Conclusão de Curso julgado adequado para obtenção o título de Bacharel em Ciências da Computação, e aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, em __/__/__.

_____

Profa. Me. Ludmilla Reis Pinheiro dos Santos

Coordenador(a) de Trabalho de Conclusão de Cursos

Banca Examinadora:

_____

Orientador: Prof. Dr. Fábio Barbosa Rodrigues

_____

Prof. Me. Fernando Gonçalves Abadia

_____

Prof. Me. Max Gontijo De Oliveira

GOIÂNIA
2024

AGRADECIMENTOS

Agradeço a Deus pelo privilégio e benefício de estudar, de completar este curso e pelo trabalho que prestarei a sociedade. Agradeço à minha família pelo apoio aos meus estudos, em particular aos meus pais, e ao meu irmão Gustavo, o qual na minha juventude deu a oportunidade de fazer um curso de informática e contribuiu para o meu primeiro emprego, e consequentemente, a minha formação. Agradeço aos meus amigos pelo encorajamento ao longo dessa jornada, em principal o Hezrai que me ofereceu incentivo e mentoria, aos amigos que fiz e os desafios que passamos juntos. Por fim, agradeço aos professores que contribuíram para minha formação.

RESUMO

Este trabalho visa apresentar um estudo sobre *Generative Pre-trained Transformer* (GPT), que é capaz de criar textos, áudios e imagens por meio de autorregressão usando parte da arquitetura *Transformers* para gerar a probabilidade da próxima palavra considerando as palavras anteriores, e é aplicado em ferramentas como o *ChatGPT* que tem milhões de usuários e está em constante aprimoramento. No entanto, os *Transformers* e o GPT têm limitações, como complexidade computacional, número elevado de parâmetros, questões éticas, dificuldades com línguas não inglesas, alto custos e limites no pré-treinamento. Ainda assim, o progresso dos *Transformers* não parece ter chegado ao seu limite, novas propostas continuam a surgir, enquanto outras antigas são reinventadas.

Palavras-chave: inteligência artificial, aprendizado de máquina, processamento de linguagem natural, *generative pre-trained transformer*.

ABSTRACT

This work aims to present a study on Generative Pre-trained Transformer (GPT), which is capable of creating texts, audios, and images through autoregression using part of the Transformers architecture to generate the probability of the next word considering the previous words, and is applied in tools like ChatGPT that has millions of users and is constantly improving. However, Transformers and GPT have limitations, such as computational complexity, high number of parameters, ethical issues, difficulties with non-English languages, high costs, and limits in pre-training. Even so, the progress of Transformers does not seem to have reached its limit, new proposals continue to emerge, while other old ones are reinvented.

Keywords: artificial intelligence, machine learning, natural language processing, generative pre-trained transformer.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

BERT *Bidirectional Encoder Representations from Transformers*
EESA Estimativa de Esforço de Software por Analogia
EUA Estados Unidos da América
GPT *Generative Pre-trained Transformer*
GPU *Graphics Processing Unit*
IA Inteligência Artificial
LLM *Large Language Models*
PLN Processamento de Linguagem Natural
RLHF *Reinforcement Learning from Human Feedback*
RNA Redes Neurais Artificiais

1 INTRODUÇÃO

Desde o início da computação “a tradução automática entre línguas foi um dos primeiros problemas submetidos aos primeiros computadores” (CASELI; NUNES, 2024), onde surgi questões como “as máquinas podem pensar?” (ALAN TURING, 1950). A Inteligência Artificial (IA) abrange diversos campos de conhecimentos como: resoluções de problemas, reconhecimento de padrões, aprendizado e compreensão de linguagem natural. Sendo assim, ela é capaz de realizar tarefas relevantes para qualquer campo de conhecimento da atividade intelectual humana (RUSSELL; NORVIG, 2020).

Passando por diversas transformações significativas, a IA evoluiu de sistemas simbólicos e métodos baseados em regras para sofisticados modelos, como aprendizado de máquina e aprendizado profundo (GOODFELLOW; BENGIO; COURVILLE, 2016), gerando impacto na evolução da sociedade moderna, remodelando indústrias, negócios, profissões, simplificando processos de tomada de decisões e aumentando as capacidades humanas em vários domínios (KAPLAN; HAENLEIN, 2019). Influenciando e participando cada vez mais em vários setores como a saúde, o transporte, as finanças e a educação (BUGHIN, 2017), na predição médica de câncer (CHEN; KONG; QUIAG, 2023), no trânsito em controle, monitoramento inteligente e carros autônomos (KHAN; SIKANDAR, 2022). Conhecida como a que “revolucionará todos os setores e apresenta uma oportunidade de vários trilhões de dólares” (HUANG, 2023).

Dentre os campos da IA, o Processamento de Linguagem Natural (PLN) foca em modelos para reconhecer, interpretar e gerar linguagem humana (JURAFSKY; MARTIN, 2024), modelos estes que determinam o sentimento, emoção ou opinião expressada em um texto (PANG; LEE, 2008). Estes sistemas podem ter diálogos semelhantes ao humano, entendendo e resolvendo às entradas dos usuários (WEIZENBAUM, 1966; SEBAN, 2016). Outros modelos que geram textos semelhantes aos humanos com base em uma determinada entrada ou contexto, podem gerar histórias, artigos, postagens em redes sociais ou respostas em uma conversa (RAFORD *et al.*, 2019).

Os avanços nessas áreas têm o potencial de revolucionar a forma como interagimos com os computadores e processamos informações, tornando-as mais acessíveis e compreensíveis tanto para humanos quanto para as máquinas

(JURAFSKY; MARTIN, 2024). As “aplicações de PLN saíram dos laboratórios acadêmicos e entraram definitivamente no nosso cotidiano. O desenvolvimento de diversas áreas da computação de forma integrada (hardware, software, tratamento de grandes volumes de dados, aprendizado de máquina e aprendizado profundo) impulsionou o desenvolvimento do processamento da língua para uma nova esfera” (CASELI; NUNES, 2024).

O modelo *Generative Pre-Trained Transformer* foi apresentado por Radford *et al.* (2018), como um modelo de linguagem neural baseado na arquitetura *Transformer* de Vaswani *et al.* (2017). Tendo uma série de novos modelos posteriormente, conhecido como GPT-n, os quais são GPT-2, GPT-3, GPT-3.5 e GPT-4. Sendo altamente adaptável a várias tarefas e domínios (OPENAI, 2022), ao contrário de muitos modelos tradicionais de aprendizado de máquina projetados para tarefas específicas, o modelo GPT pode ser ajustado para executar uma ampla variedade de tarefas sem grandes alterações (DEVLIN *et al.*, 2019), demonstrando alto desempenho em uma ampla variedade de tarefas em PLN, incluindo geração de texto, análise de sentimento, resumo e chats (RAFORD *et al.*, 2019; BROWN *et al.*, 2020), apresentando também, um desempenho competitivo em tarefas do campo de domínio da química, como na previsão de rendimento e seleção de reagentes (GUO *et al.*, 2023).

Estudos que investigaram as implicações potenciais do modelo GPT e de tecnologias relacionadas no mercado de trabalho dos Estados Unidos da América (EUA), indicam que; “aproximadamente 80% da força de trabalho dos EUA poderá ter pelo menos 10% das suas tarefas de trabalho afetadas pela introdução dos modelos GPT, enquanto cerca de 19% dos trabalhadores poderão ver pelo menos 50% das suas tarefas afetadas. A influência abrange todos os níveis salariais, com empregos com rendimentos mais elevados potencialmente enfrentando maior exposição. Notavelmente, o impacto não se limita às indústrias com maior crescimento recente da produtividade” (YENDURI *et al.*, 2023). O GPT têm características de tecnologias de uso geral, sugerindo-se que estes modelos podem ter notáveis implicações econômicas, sociais e políticas (YENDURI *et al.*, 2023).

Em novembro de 2022, o mundo conheceu o *ChatGPT*, um *chatbot* capaz de responder a qualquer pergunta ou solicitação, em língua natural, incluindo o português. Além de surpreender pelo seu desempenho linguístico, ele acendeu um sinal de alerta para a comunidade de IA e PLN, bem como para vários setores da

sociedade. Várias indagações surgiram: Será que a IA está prestes a atingir a meta de criar sistemas que possam vir a substituir o homem em todas as tarefas? Quais os riscos que as diferentes sociedades correm se o mercado de trabalho for dominado por sistemas inteligentes? E se apenas poucas empresas dominarem o mercado de sistemas inteligentes? E o que pode acontecer com países não desenvolvidos? A conscientização desse cenário fez crescer o interesse pelas atividades da comunidade que desenvolve sistemas de IA, incluindo a de PLN, já que o *ChatGPT* é uma aplicação típica dessa área (CASELI; NUNES, 2024).

É relevante estudar esse tema, pois o GPT é um modelo de IA com uma abordagem de aprendizado profundo que aproveita de mais dados e mais computação para criar modelos de linguagem cada vez mais sofisticados, capazes e com bons resultados (OPENAI, 2023), impactando na forma como as pessoas e empresas utilizam IA e compartilhando conceitos com outros modelos de IA. Além disso, é um assunto novo e que muitas pessoas da computação ainda não possuem conhecimento do modelo GPT ou até mesmo da sua existência.

Diante deste contexto, esse trabalho visa responder a seguinte questão de pesquisa: Como funciona o GPT e onde pode ser aplicado?

O objetivo final deste trabalho é apresentar um estudo sobre GPT, descrevendo os principais conceitos do seu modelo, funcionamento e aplicações.

Os objetivos específicos são:

- Descrever o conceito de *Transformers;*
- Descrever os principais conceitos do GPT;
- Apresentar aplicações do GPT;
- Apresentar limitações do GPT.

Espera-se que os resultados deste trabalho possam contribuir no entendimento dos *Transformers* e GPT na geração de texto, com uma revisão de literatura recente e conhecimento organizado e facilitado.

Quanto aos procedimentos metodológicos é um resumo de assunto conforme sua natureza, quanto aos objetivos é uma pesquisa exploratória e quanto aos procedimentos técnicos é uma pesquisa bibliográfica.

Esta monografia está estruturada da seguinte forma: No capítulo 1 informa o contexto do trabalho, o problema a ser resolvido, objetivo da pesquisa e resultados esperados. O capítulo 2 traz os conceitos, definições e trabalhos relacionados com o tema. No capítulo 3 descreve o método utilizado para a pesquisa, mostrando como o

trabalho foi desenvolvido e o que foi feito para atingir o objetivo final. No Capítulo 4 descreve os principais conceitos, aplicações e limitações do GPT. Capítulo 5 apresenta as considerações finais do TCC e as sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo fornece uma visão do campo da IA com foco em PLN, nas abordagens essenciais para geração de texto e trabalhos relacionados a pesquisa.

2.1 Inteligência artificial

Usada para construir sistemas que simulam os comportamentos, a IA, abrange uma ampla gama de abordagens, incluindo aquelas baseadas na lógica, pesquisa e raciocínio probabilístico. O aprendizado de máquina é um subconjunto da IA, que aprende a tomar decisões ajustando modelos matemáticos aos dados observados. (PRINCE, 2023)

As Redes Neurais Artificiais (RNA) são uma classe de modelos de aprendizado de máquina inspirados na estrutura e função do cérebro humano (GOODFELLOW; BENGIO, 2016). Em seu núcleo, as RNA consistem em nós interconectados ou "neurônios" que podem processar e transmitir informações. As RNA tornaram-se uma ferramenta popular em muitos campos diferentes devido à sua capacidade de aprender com os dados e fazer previsões ou decisões com base nesse aprendizado (BISHOP, 2006). Esses neurônios são normalmente organizados em camadas, com cada camada consistindo em um conjunto de neurônios que processam entradas e produzem saídas que são passadas para a próxima camada (BISHOP, 2006).

As conexões entre os neurônios são ponderadas, ou seja, cada conexão tem um valor que determina a força do sinal transmitido de um neurônio para outro. Para treinar uma rede neural, normalmente são necessárias grandes quantidades de dados para permitir que a rede aprenda os padrões e relacionamentos dentro dos dados (GOODFELLOW; BENGIO, 2016).

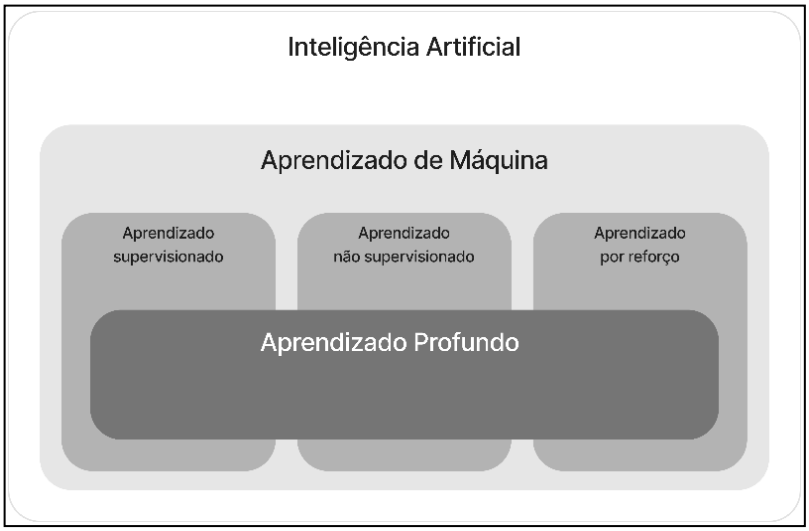
Um dos pontos fortes das redes neurais é sua capacidade de aprender e generalizar a partir de dados, o que significa que muitas vezes elas podem ter um bom desempenho em tarefas, mesmo quando os exemplos específicos que recebem durante o treinamento diferem um pouco dos exemplos que encontrarão durante o teste ou implantação (NG, 2018).

Uma rede neural profunda é um tipo de modelo de aprendizado de máquina, e quando ela é ajustada aos dados é chamado de aprendizado profundo. Redes

neurais profundas são modelos de aprendizado de máquina poderosos, práticos e frequentemente encontrados no dia a dia, como na tradução de um texto ou detecção de objetos em imagens usando visão computacional. Todas essas aplicações são impulsionadas pelo aprendizado profundo.

Os métodos de aprendizado de máquina podem ser divididos em três áreas principais, conforme ilustrado na Figura 1; aprendizado supervisionado, não supervisionado e por reforço. Nesses métodos podemos utilizar o aprendizado profundo para obter melhores resultados (PRINCE, 2023).

Figura 1 – Representação de subáreas da Inteligência Artificial.



Fonte: Modificado a partir de (PRINCE, 2023).

No entanto, as redes neurais também podem ser propensas ao *overfitting*, que ocorre quando uma rede se torna muito especializada para os dados de treinamento e tem um desempenho ruim em dados novos e não vistos. Técnicas de regularização, como *dropout* ou decaimento de peso, podem ajudar a prevenir o *overfitting* (SRIVASTAVA *et al.*, 2014; GOODFELLOW; BENGIO, 2016).

No aprendizado supervisionado são usadas técnicas de classificação ou regressão, no aprendizado não supervisionado a redução de dimensionalidade (BENGIO; COURVILLE; VINCENT, 2013) e no aprendizado por reforço a rede aprende a tomar decisões com base em recompensas ou punições recebidas de seu ambiente (SUTTON; BARTO, 2018).

As redes neurais têm muitas arquiteturas e variantes diferentes, cada uma com seus próprios pontos fortes e fracos. Alguns exemplos incluem Redes Neurais Convolucionais para análise de imagens (LECUN *et al.*, 1998), Redes Neurais

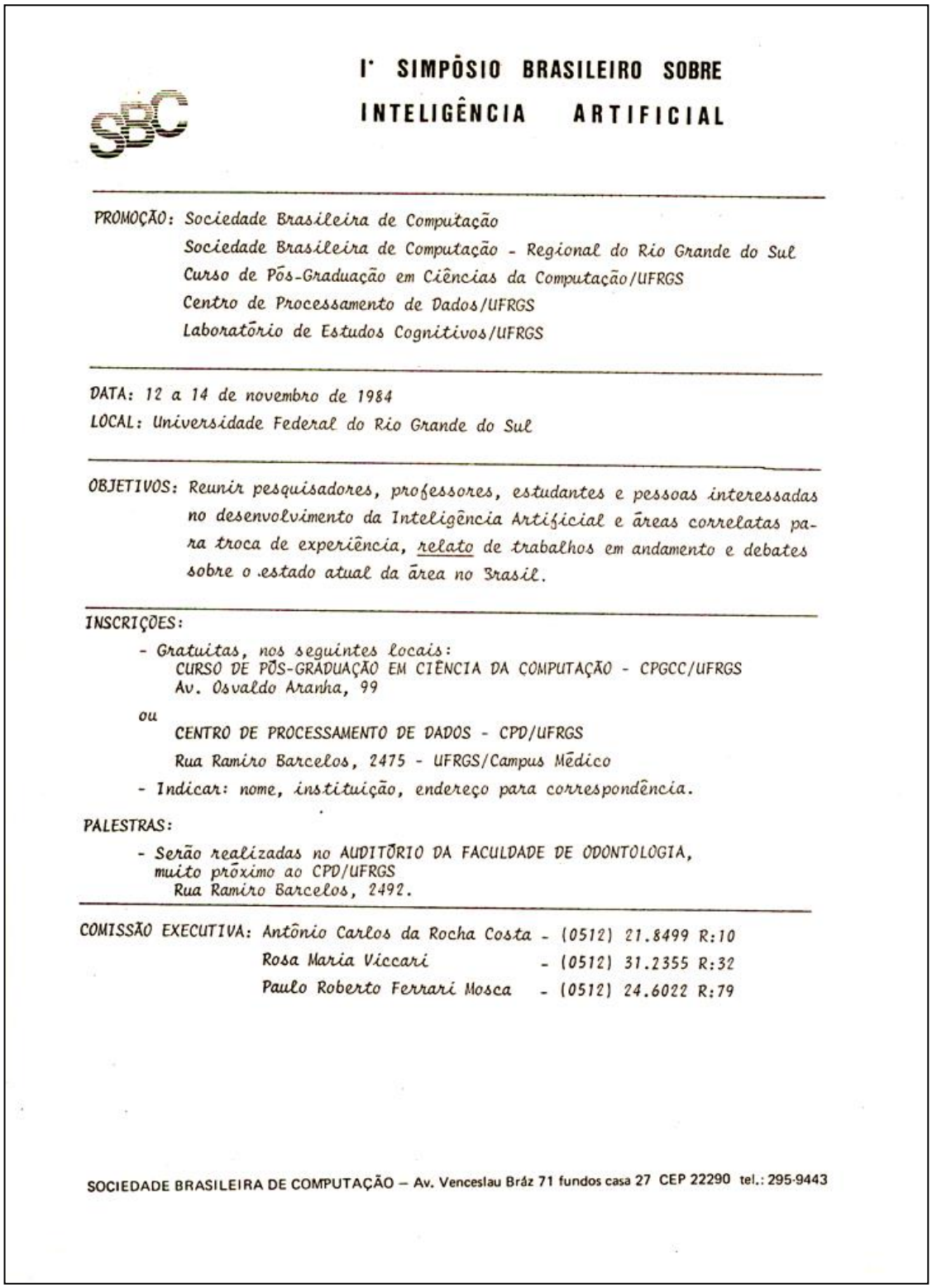
Recorrentes para dados sequenciais (HOCHREITER; SCHMIDHUBER, 1997) e *autoencoders* para aprendizado não supervisionado (HINTON; SALAKHUTDINOV, 2006). As redes neurais podem ser usadas para uma ampla variedade de tarefas, incluindo classificação, regressão, agrupamento e muito mais (BENGIO; COURVILLE; VINCENT, 2013). E têm sido usados em muitos campos diferentes, incluindo visão computacional, processamento de linguagem natural, robótica, entre outros. À medida que a pesquisa em redes neurais continua a evoluir, é provável que novas arquiteturas e técnicas continuem surgindo (GOODFELLOW; BENGIO, 2016).

2.2 Processamento de linguagem natural

O PLN surgiu praticamente ao mesmo tempo que os computadores, por volta da década de 1940, visto que a tradução automática entre línguas foi um dos primeiros problemas submetidos aos primeiros computadores. No Brasil, as pesquisas em PLN começaram timidamente ainda na década de 1970, entre acadêmicos interessados em IA. Em 1984, foi realizada a primeira edição do Simpósio Brasileiro de Inteligência Artificial, em Porto Alegre - RS, e boa parte dos trabalhos apresentados nesse evento eram da área de PLN. Naquela época, os sistemas propostos para português eram muito simples, sendo apenas estudos de caso bem elementares. Demorou mais uma década até que houvesse uma massa crítica de cientistas brasileiros dedicados ao processamento computacional do português. Em 1993 foi realizado o primeiro evento exclusivo de PLN dedicado ao português ibérico e brasileiro, em Lisboa. Tal evento se tornou o *International Conference on the Computational Processing of Portuguese*, realizado alternativamente no Brasil e em Portugal, com mais de 16 edições realizadas até este ano (2024) (CASELI; NUNES, 2024).

No âmbito nacional, em 2003 foi criado o Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana, que possui diferentes eventos satélites tratando de temas específicos para o processamento da língua, em particular o Português, na Figura 2 vemos o folheto de divulgação do primeiro simpósio que aconteceu, até esse ano (2024) houve 14 edições realizadas. E em 2007, foi criada a Comissão Especial de PLN da Sociedade Brasileira de Computação.

Figura 2 – Folheto de divulgação do 1° Simpósio Brasileiro Sobre Inteligência Artificial.



I° SIMPÓSIO BRASILEIRO SOBRE
INTELIGÊNCIA ARTIFICIAL

SBC

PROMOÇÃO: Sociedade Brasileira de Computação
Sociedade Brasileira de Computação - Regional do Rio Grande do Sul
Curso de Pós-Graduação em Ciências da Computação/UFRGS
Centro de Processamento de Dados/UFRGS
Laboratório de Estudos Cognitivos/UFRGS

DATA: 12 a 14 de novembro de 1984
LOCAL: Universidade Federal do Rio Grande do Sul

OBJETIVOS: Reunir pesquisadores, professores, estudantes e pessoas interessadas no desenvolvimento da Inteligência Artificial e áreas correlatas para troca de experiência, relato de trabalhos em andamento e debates sobre o estado atual da área no Brasil.

INSCRIÇÕES:
- Gratuitas, nos seguintes locais:
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - CPGCC/UFRGS
Av. Osvaldo Aranha, 99
ou
CENTRO DE PROCESSAMENTO DE DADOS - CPD/UFRGS
Rua Ramiro Barcelos, 2475 - UFRGS/Campus Médico
- Indicar: nome, instituição, endereço para correspondência.

PALESTRAS:
- Serão realizadas no AUDITÓRIO DA FACULDADE DE ODONTOLOGIA, muito próximo ao CPD/UFRGS
Rua Ramiro Barcelos, 2492.

COMISSÃO EXECUTIVA: Antônio Carlos da Rocha Costa - (0512) 21.8499 R:10
Rosa Maria Viccari - (0512) 31.2355 R:32
Paulo Roberto Ferrari Mosca - (0512) 24.6022 R:79

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO – Av. Venceslau Bráz 71 fundos casa 27 CEP 22290 tel.: 295-9443

Fonte: Sociedade Brasileira de Computação, 1984.

A comunidade cresceu, inicialmente graças à formação acadêmica de profissionais e pesquisadores, e posteriormente também em consequência da maior demanda de empresas tecnológicas e de outras naturezas, atraindo assim

profissionais com diferentes formações. Se antes os especialistas em PLN tinham a tarefa de explicitar e codificar conhecimento linguístico, hoje seu maior desafio é preparar dados/exemplos linguísticos para servirem de entrada aos algoritmos no treinamento de modelos computacionais capazes de transformar dados em conhecimento ou ações. E isso não é pouco. Para enriquecer os dados brutos de modo que o conhecimento adquirido seja o mais completo e correto possível, ou que a ação gerada seja a mais adequada, é preciso saber escolhê-los, prepará-los e eventualmente anotá-los com informações de várias naturezas, como: morfológicas, sintáticas, semânticas, extralinguísticas. É essa tarefa que muitos especialistas em PLN de todo o mundo têm se dedicado ultimamente. Outro papel importante está nas avaliações de qualidade dos recursos e produtos construídos, o entendimento do problema, a identificação das limitações e a proposta de melhorias. É crucial que as bases do desenvolvimento dessas tecnologias sejam fortes, bem-informadas, consistentes e éticas. A linguagem nos define, e ela faz agora parte dos nossos artefatos, a complexidade deste cenário ainda não é bem compreendida (CASELI; NUNES, 2024).

2.2.1 Tokenização

Um *corpus* é um conjunto de dados linguísticos, geralmente armazenados computacionalmente, servido como uma fonte de dados linguísticos para análise e pesquisa (KENNEDY, 2014). Bem como Caseli e Nunes (2024) diz, a utilização de *corpus* – palavra latina que significa corpo e que tem como plural a palavra *corpus* – vem de longa data nos estudos linguísticos e lexicográficos, e ganha força nos anos 1980 com a popularização dos computadores. A partir daí, e cada vez mais, *corpora* diz respeito a uma coleção de textos que pode ser processada por computadores. O material que compõe um *corpora* (os textos) é coletado com algum propósito (investigar ou explorar algum aspecto da linguagem, teórico ou aplicado) e fora produzido “naturalmente”, isto é, não estamos diante de frases artificialmente inventadas com o objetivo de construir um *corpus*.

Token é um termo que significa qualquer sequência de caracteres à qual se atribui um valor. Nas línguas europeias, a sequência consiste em caracteres delimitados por espaços gráficos, sendo que a *tokenization* é ajustada para separar sinais de pontuação. Mas, na grande maioria das línguas, a *tokenization* não opera

por espaços gráficos. Diante dessa definição, é comum associarmos token à palavra escrita. Nesse sentido, a quantidade de palavras e sinais de pontuação de uma sentença equivale à quantidade de tokens. *Type*, por sua vez, refere-se aos *tokens* únicos encontrados numa frase ou texto. A proporção *token*/*type* (divisão da quantidade de *tokens* pela quantidade de *types*) é um importante indicativo da riqueza lexical de um texto, ou seja, ela indica qual a diversidade de palavras existentes em um *corpus*, excluindo suas repetições. Mas, nessa medida, apenas as formas de palavras (as palavras diferentes) e o número total de palavras são contados. Isto é, sinais de pontuação não são considerados (CASELI; NUNES, 2024; JURAFSK; MARTIN, 2019).

2.2.1.1 Tokenização em subpalavras

Esse processo tem por objetivo reduzir o vocabulário de trabalho de um modelo de linguagem a um tamanho finito, mas que possa ser usado para representar textos onde o número de *types* seja potencialmente infinito. Dessa forma, pode-se utilizar um conjunto de treino que possua um vocabulário finito e que seja capaz de ser aplicado a qualquer texto. Abordagem de tokenização em subpalavras consiste em codificar diretamente algumas palavras mais comuns, como “de”, “fazer”, “são” e “feliz”. No entanto, palavras mais raras, como “desfazer” ou “felizmente” podem ficar de fora do vocabulário de trabalho, conhecido como *out-of-vocabulary* e, portanto, serem representadas como combinações de subpalavras respectivamente: “de” + “s” + “fazer” e “feliz” + “mente” (CASELI; NUNES, 2024).

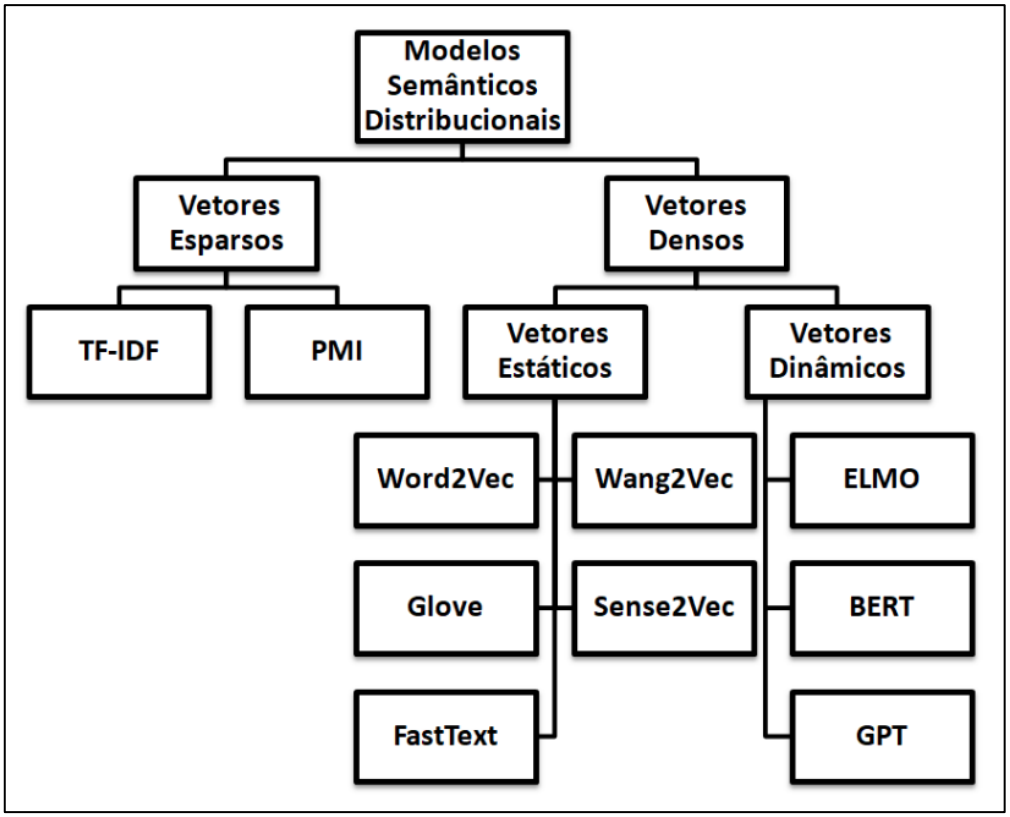
2.2.2 Semântica distribucional

Para os seres humanos é natural e relativamente fácil visualizar um texto e a partir de uma simples leitura, extrair dele determinados tipos de informação. Diferente dos humanos, os algoritmos computacionais não conseguem processar símbolos ou palavras. Ao invés disso, eles precisam de uma representação numérica de um documento ou texto a ser processado para que consigam realizar suas operações. A semântica distribucional é uma abordagem de representação do significado lexical adotada nas mais diversas tarefas do PLN, onde os itens lexicais (palavras) são representados por meio de vetores de valores reais conhecidos por

vetores semânticos que codificam o significado das palavras a partir de sua distribuição em textos (CASELI; NUNES, 2024).

Está ancorada na Hipótese Distribucional a qual afirma que a semelhança no significado resulta em similaridade da distribuição linguística (HARRIS, 1954), por exemplo, de palavras como “ensino” e “educação” que costumam aparecer no mesmo contexto de palavras como “aluno”, “escola” e “professor”, sugerindo que existe uma similaridade entre as duas palavras em certos contextos. Fazendo engenharia reversa do processo, a semântica distribucional induz representações semânticas a partir de contextos de uso (BOLEDA, 2020), assim, as palavras são caracterizadas pelo contexto em que elas aparecem. Por se basearem em distribuição, os vetores semânticos podem ser aprendidos automaticamente a partir de textos, sem que haja supervisão de um humano (utilizando-se textos não rotulados). Os modelos que aprendem esse tipo de representação são denominados de Modelos Semânticos Distribucionais e frequentemente classificados como vetores esparsos e vetores densos (CASELI; NUNES, 2024). A figura 3 ilustra a estrutura geral dos Modelos Semânticos Distribucionais, o GPT utiliza os vetores dinâmicos que fazem parte dos vetores densos.

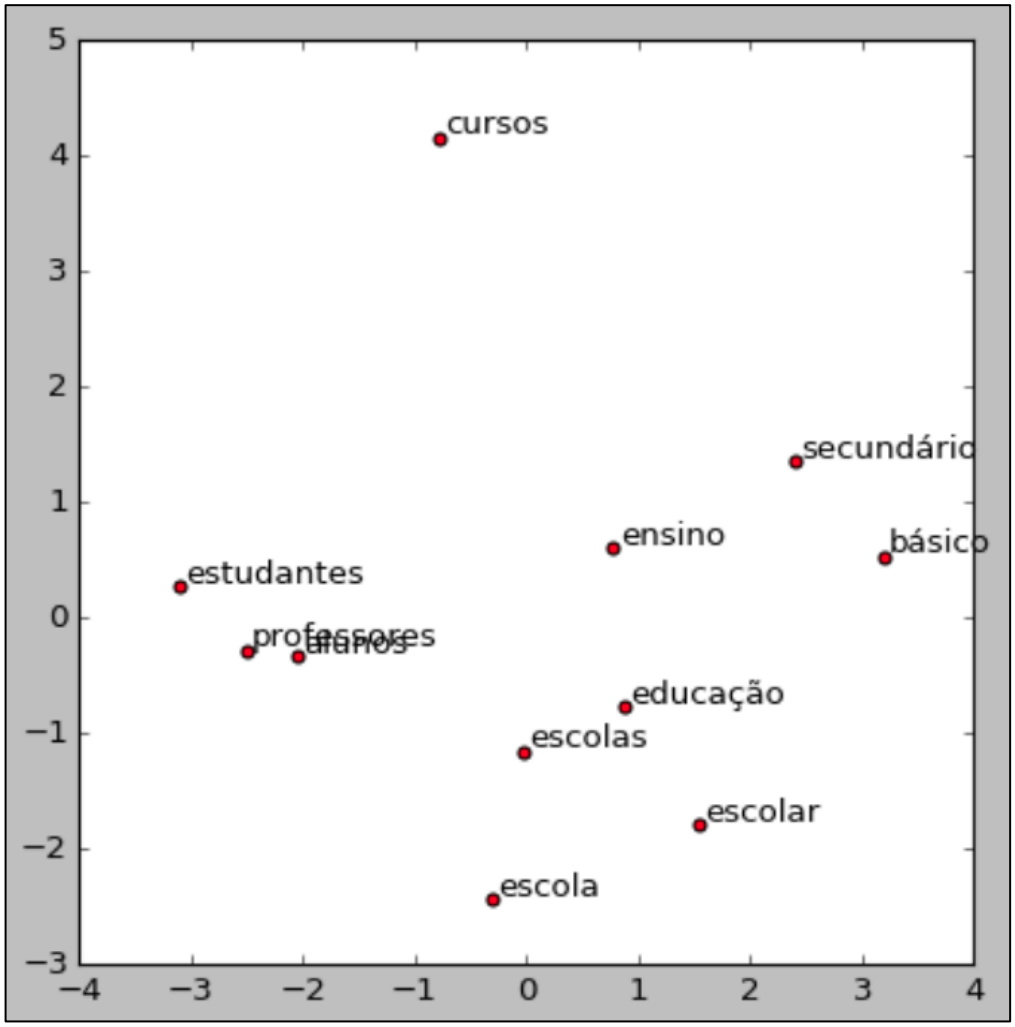
Figura 3 – Ilustração dos Modelos Semânticos Distribucionais.



Fonte: Caseli e Nunes, 2024.

A semântica vetorial se define como uma representação vetorial que retrata o significado de uma palavra a partir da distribuição das palavras que formam o seu contexto (JURAFSKY; MARTIN, 2024). Por exemplo, a Figura 4 representa o espaço vetorial semântico da palavra “ensino”. Palavras como “educação”, “estudantes”, “professores” e “alunos” são alguns exemplos de palavras que compartilham esse mesmo espaço semântico. A ideia dos vetores semânticos é representar cada palavra como um ponto em um espaço vetorial multidimensional construído a partir da distribuição de suas palavras vizinhas. Um espaço vetorial é formado por uma coleção de objetos chamados vetores (CASELI; NUNES, 2024).

Figura 4 – Representação do espaço vetorial semântico da palavra “ensino” gerada com o modelo GloVe.



Fonte: Núcleo Interinstitucional de Linguística Computacional, 2024.

Geralmente, os vetores semânticos são representados por meio de uma matriz de coocorrência (ou distribuição de coocorrência), que retrata a frequência de coocorrência das palavras. As representações matriciais mais comuns são a matriz termo-documento, onde cada dimensão (vetor) da matriz representa um documento,

e a matriz termo-contexto, em que cada dimensão representa uma palavra (JURAFSKY; MARTIN, 2024).

2.2.3 Vetores de *embeddings*

Os vetores que representam palavras são frequentemente denominados de *embeddings*, embora muitas vezes esse termo seja usado de maneira mais restrita para se referir apenas aos vetores densos. Os vetores de *embeddings* podem ser estáticos ou dinâmicos. Os *embeddings* estáticos permanecem fixos uma vez aprendidos, ou seja, eles não podem ser ajustados ou modificados para uma tarefa específica. Ao contrário desses, os *embeddings* dinâmicos podem ser ajustados em tarefas específicas se adaptando às nuances específicas da tarefa e ao contexto atual. Vários métodos desenvolvidos a partir de 2017 passaram a construir *embeddings* de forma dinâmica, considerando o contexto da sentença no momento do uso, e por isso comumente denominados de *embeddings* contextualizados. Isso quer dizer que as unidades de representação (*tokens*) podem ter *embeddings* distintos, definidos quando eles são aplicados. Considere, por exemplo:

1. Sentei no banco da praça.
2. O banco estava sem notas de R$ 200,00.
3. O banco estava super cheio hoje!

A palavra “banco” na sentença 1) evoca mais o sentido de assento, embora também seja possível pensar em outros significados; A sentença 2) evoca mais o sentido de estabelecimento comercial financeiro; A sentença 3) apesar de evocar mais o segundo sentido, também poderia estar falando de um assento cheio de pessoas. Sendo assim, uma lista estática de palavras e seus *embeddings* falharia em retornar representações distintas para estas diferentes interpretações. Considere ainda o exemplo: em frente à agência do banco de Pineapólis, existe um banco amarelo que data da década de 50, onde várias pessoas famosas já pararam para descansar e algumas vezes entoar uma melodia. Observe que a palavra “banco” aparece duas vezes na mesma sentença, com dois significados distintos, um método de geração de embeddings contextualizados deve ter a habilidade de devolver representações vetoriais distintas para os dois *tokens* (CASELI; NUNES, 2024).

Para tanto, a unidade de representação é associada a um *embedding* a partir do contexto corrente em que ela aparece, onde contexto em geral é definido nos

modelos de linguagem por uma sequência de *tokens* que aparecem antes e depois do *token* em questão. No exemplo anterior, teríamos *embedding* distintos para os diversos “bancos” mencionados. Na verdade, o *embedding* poderia diferir até mesmo para *tokens* do tipo “banco” com a mesma semântica devido aos diferentes outros *tokens* que aparecem em seus contextos. Entretanto, ainda espera-se que quanto mais próxima for a semântica do *token*, mais próximos fiquem os vetores no espaço vetorial. Os *embeddings* contextualizados tem a possibilidade de representar informação que vai além do idioma, eles são chamados de *cross-lingual* (AGIRRE, 2020). Ou seja, é possível que os embeddings associados às palavras “mãe” e “*mother*” estejam próximos no espaço vetorial, mesmo que ambas as palavras estejam em idiomas distintos. A aplicação de *embeddings* contextualizados para abordar tarefas de PLN inclui dois aspectos: a geração dos *embeddings* e a sua utilização em tarefas finais (CASELI; NUNES, 2024).

Os *Transformers* é um dos métodos para a geração de embeddings contextualizados, incluindo *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN *et al.*, 2019) e GPT.

2.2.4 Modelos de linguagem computacional

Um modelo é uma simplificação de um fenômeno complexo, uma simplificação da língua que possa ser representada por ferramentas computacionais. Embora um modelo tente capturar as nuances do fenômeno real, justamente por ser uma simplificação, ele não tem a intenção de substituir o fenômeno real, mas representá-lo para auxiliar o nosso entendimento ou resolver algumas tarefas. Porém, idealmente, o modelo deve manter alguma consistência com o fenômeno real. Por isso, um modelo de linguagem deveria respeitar os princípios léxicos, sintáticos e semânticos, componentes essenciais de qualquer linguagem, seja ela natural ou não. Também, um modelo deveria considerar o mesmo funcionamento do fenômeno real. A questão de como nosso cérebro processa e produz linguagem continua em aberto (BERWICK; CHOMSKY, 2017), nos modelos de linguagem computacionais, assume-se que um texto escrito ou falado é oriundo de um processo de completação. Em suas primeiras abordagens, definia-se que um modelo de linguagem computacional deveria ser capaz de completar a próxima

palavra em uma sequência, considerando todas as palavras que vieram antes (CASELI; NUNES, 2024).

2.2.5 Modelos de linguagem probabilísticos

Em termos computacionais, Caseli e Nunes (2024) definem a modelagem probabilística de linguagem como a tarefa que atribui uma probabilidade a uma sequência de palavras. Ou seja, o modelo assume que existe uma probabilidade associada à existência de uma sequência de palavras $p_{1:i}$, representada por $P(p_{1:i})$, onde i representa a posição da última palavra na sequência considerada. Usando a regra da cadeia da probabilidade teremos a equação 1.

$$P(p_{1:i}) = P(p_1)P(p_2|p_1)P(p_3|p_{1:2})P(p_4|p_{1:3}) \dots P(p_i|p_{1:i-1}) \quad (1)$$

Observe que na equação 1 temos uma sequência de tarefas de predição de palavra, onde o objetivo é predizer uma palavra condicionando-a às palavras precedentes. Assim, pensando na completação discutida anteriormente, assumimos que a tarefa de completar uma sequência de palavras com uma próxima palavra é definida por uma distribuição de probabilidade condicional das palavras que poderiam completar a sequência, dadas as palavras que vieram antes na sequência, ou seja:

$$P(p_i|p_i, \dots, p_{i-1}) \quad (2)$$

Onde p_i é uma palavra do vocabulário, i é a sua posição na sequência, p_1 é a primeira palavra da sequência de p_{i-1} é a última palavra da sequência.

Modelos de língua que seguem esta formulação são chamados de modelos autorregressivos ou causais e são frequentemente empregados para tarefas de envolvem geração de texto. Sendo a ideia para geração de texto:

1. Use o modelo probabilístico para escolher o próximo *token*;
2. Adicione o token gerado na sequência de entrada;
3. Repita os anteriores.

Mas no passo (1), quando falamos que um *token* é gerado pelo modelo, o que acontece, na verdade, é que um *token* é escolhido de acordo com uma distribuição de probabilidade aprendida pelo modelo. Tal distribuição de probabilidade é definida para um vocabulário, que é o conjunto de tokens que o modelo conhece (CASELI; NUNES, 2024).

2.2.6 Modelos de linguagem neurais

Modelos de linguagem computacionais gerados por redes neurais são utilizados para representar textos escritos, fala, e até mesmo especificações que não são consideradas como parte da “linguagem natural”, por exemplo, formalizações matemáticas (GEVA; GUPTA; BERANT, 2020; GONG *et al.*, 2022; LI *et al.*, 2023; PIEKOS; MALINOWSKI; MICHALEWSKI, 2021), código (Li *et al.*, 2023; WANG *et al.*, 2021), e até codificação de informações genéticas e moleculares (BRANDES *et al.*, 2022; NIJKAMP *et al.*, 2022).

Os modelos de linguagem produzidos por redes neurais tanto geram como consomem textos mapeados para representações numéricas, para tal, mapeamos os componentes da língua para vetores em um espaço semântico, seguindo a hipótese distribucional, garantido que: teremos as representações de informações essencialmente simbólicas em um formato numérico para o computador e teremos conotação semântica. Pois a hipótese distribucional tem como mote inferir significado a partir do contexto em que as palavras ocorrem (CASELI; NUNES, 2024).

Para calcularmos a probabilidade de uma sequência de palavras ao usar um modelo, mesmo que a sequência não tenha aparecido durante o treinamento do modelo, é considerar que a probabilidade associada a um modelo de linguagem é uma função e “aprender” tal função (CASELI; NUNES, 2024), uma vez que as redes neurais são métodos de aprendizado de máquina com propriedade de aproximação universal de funções (GOODFELLOW; BENGIO, 2016). Ou seja, dada uma rede neural com ao menos uma camada escondida e um número suficiente de neurônios, ela é um aproximador universal de funções contínuas no espaço de interesse (HORNIK; STINCHCOMBE; WHITE, 1989).

A ideia de usar redes neurais para aprender funções que representem modelos de linguagem pode parecer recente, mas não é. Na verdade, as primeiras tentativas datam do início da década de 90, com o trabalho de Miikkulainen Dyer (1991). Ainda na década de 90, também foram propostas técnicas baseadas em redes neurais para prever a probabilidade do próximo caractere (SCHMIDHUBER; HEIL, 1996). Os modelos que mais se assemelham aos modelos de linguagem neurais da era das neurais profundas foram propostos no início dos anos 2000, de forma independente, com os *trabalhos Can artificial neural network learn language*

models? (XU; RUDNICKY, 2000) e *A neural probabilistic language model* (BENGIO *et al.*, 2003).

Enquanto o primeiro caso usava uma forma limitada de rede neural sem camadas escondidas e limitando a predição a apenas uma palavra, ou seja, modelando apenas unigramas (palavras individuais) e bigramas (pares de palavras consecutivas), o segundo caso já apresentava várias características e fundamentos encontrados nos modelos de linguagem neurais modernos. A proposta do primeiro trabalho era aprender funções de representações distribuídas para cada palavra $P(w)$, que consideraria a vizinhança das palavras nos textos de treinamento. Mas além da probabilidade das palavras, o modelo também aprenderia de forma simultânea a função de probabilidade associada a uma sequência de palavras a partir das probabilidades das palavras. Assim, mesmo que no momento de usar o modelo aparecesse uma sequência de palavras não vista durante o treinamento, ainda seria possível obter a probabilidade da sequência a partir das palavras e sequências similares vistas durante o treinamento (CASELI; NUNES, 2024).

2.3 Trabalhos relacionados

Em Dantas (2021) é explorado os conceitos teóricos por trás da arquitetura *Transformers*, os desafios do cenário, os aprimoramentos de eficiência propostos na literatura com alternativas e variações dos *Transformers*.

Já Baratto (2022) compara o desempenho do GPT-3 e BERT (empregado no SE3M) na representação de características textuais para a inferência de Estimativa de Esforço de *Software* por Analogia (EESA), uma etapa crucial no ciclo de desenvolvimento de software. A EESA é uma técnica que utiliza dados históricos de projetos para automatizar parte desse processo, especialmente com base em requisitos textuais, como histórias de usuário. Os resultados mostram que o GPT-3, mesmo sem ajuste fino, obteve resultados semelhantes aos modelos SE3M e Deep-SE, destacando-se o Erro Absoluto Médio de $3{,}80 \pm 1{,}20$. No entanto, a principal limitação do GPT-3 é o custo associado à extração da representação textual e ao ajuste fino.

3 PROCEDIMENTOS METODOLÓGICOS

Esta pesquisa é um resumo do assunto que busca explicar a área de conhecimento deste projeto, e mostra sua evolução com base nas investigações das informações obtidas, provendo entendimento de suas causas e explicações (WAZLAWICK, 2014).

Em relação aos procedimentos técnicos é uma pesquisa bibliográfica. A pesquisa bibliográfica precisa ser elaborada a partir de estudos sobre teses, artigos, sites entre outros (WAZLAWICK, 2014).

Para Gil (2017), a pesquisa bibliográfica deve ser elaborada através de materiais já publicados, possibilitando maior vantagem ao analisar uma sucessão de fenômenos. Para realizar uma pesquisa bibliográfica, de acordo com Gil (2017) é necessário seguir as etapas:

a) Escolher um tema: estudar sobre GPT.
b) Levantamento bibliográfico preliminar: realiza um levantamento bibliográfico para o pesquisador se habituar com a área de estudo escolhida, facilitando a definição do problema.
c) Formulação do problema: quais são os principais conceitos do modelo, funcionamento e as aplicações do GPT.
d) Busca das fontes: as fontes bibliográficas foram através do Google Acadêmico, arXiv, livros, Capes, site da OpenAI, dentre outros. Essas fontes ofereceram informações para responder o problema proposto, sendo consultado dissertações, periódicos científicos, obras de referência, entre outros.
e) Leitura de materiais: identificar as informações e dados disponíveis nos materiais adquiridos, criando relações com o problema proposto e analisar a coerência das informações e dados apresentados pelos autores.
f) Fichamento: Foi realizado resumo do material estudado.
g) Organização lógica sobre o assunto: foram organizadas as ideias com o propósito de realizar os objetivos da pesquisa.
h) Redação do texto: escrita da monografia do TCC 2.

4 GENERATIVE PRE-TRAINED TRANSFORMER

Fundada em 2015, por Sam Altman, Greg Brockman, Peter Thiel e Elon Musk, a OpenAI é uma empresa de pesquisa e implantação de IA (BROCKMAN; SUTSKEVER, 2024). Publicando em 2018 o GPT, um modelo de linguagem para prever a próxima palavra de uma sentença, dado todas as palavra anteriores (RAFORD *et al.,* 2018), usa parte da arquitetura *Transformer* para funcionar como um modelo autorregressivo de geração de texto.

4.1 Transformer

A geração de sequências a partir de outras sequências obedecendo princípios sintáticos e semânticos é chamada de *sequence-to-sequence* (CHO *et al.*, 2014). As tarefas de tradução automática, sumarização e respostas a consultas complexas requerem que a entrada seja um texto (uma sequência) e que a saída também seja um texto (outra sequência), na Figura 5 temos uma representação da entrada aos *Transformers* e a saída.

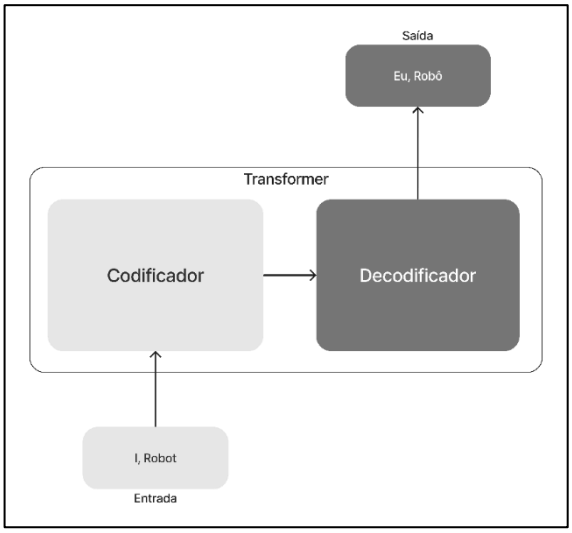
Figura 5 – Representação geral dos Transformers.

Fonte: Autoria própria.

Para resolver tarefas *sequence-to-sequence* o mais comum é considerar dois grandes componentes introduzidos por Cho *et al.* (2014); o codificador, que é responsável por processar a sequência de entrada e codificá-la como um vetor de números, chamado de vetor de contexto; o segundo componente é o decodificador, responsável por receber e processar o vetor de contexto e transformá-lo na sequência de saída. Sendo ambos uma ou várias redes neurais.

Proposta por Vaswani *et al.* (2017), a arquitetura de rede neural *Transformers* tem dois componentes principais: um componente de codificação e outro de decodificação conforme apresentado na Figura 6.

Figura 6 – Representação do codificador e decodificador nos Transformers.

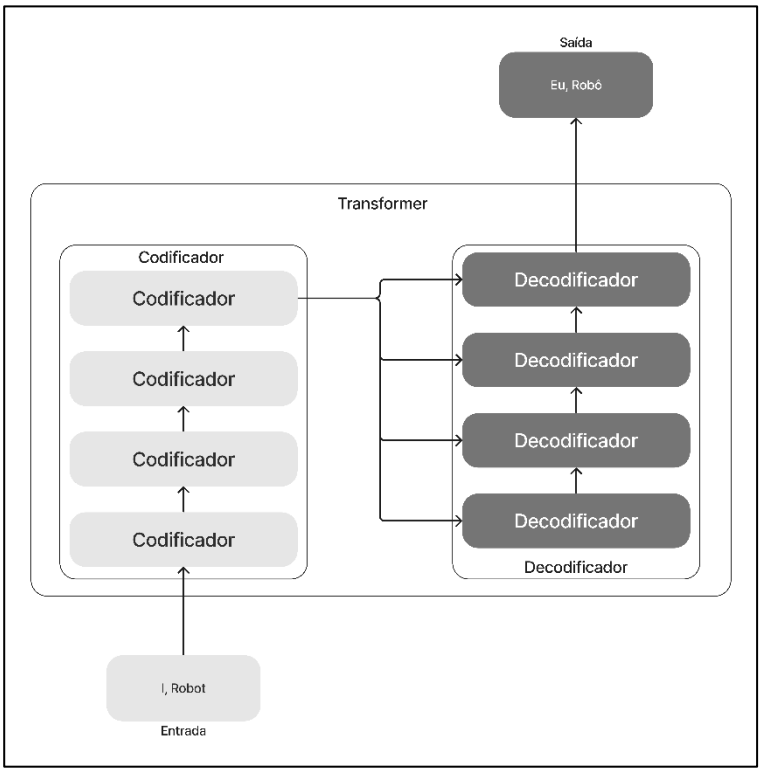


Fonte: Autoria própria.

4.1.1 Codificador e Decodificador

O codificador é uma pilha de sub-codificadores e o decodificador é uma pilha de sub-decodificadores, de qualquer quantidade, como representado na Figura 7.

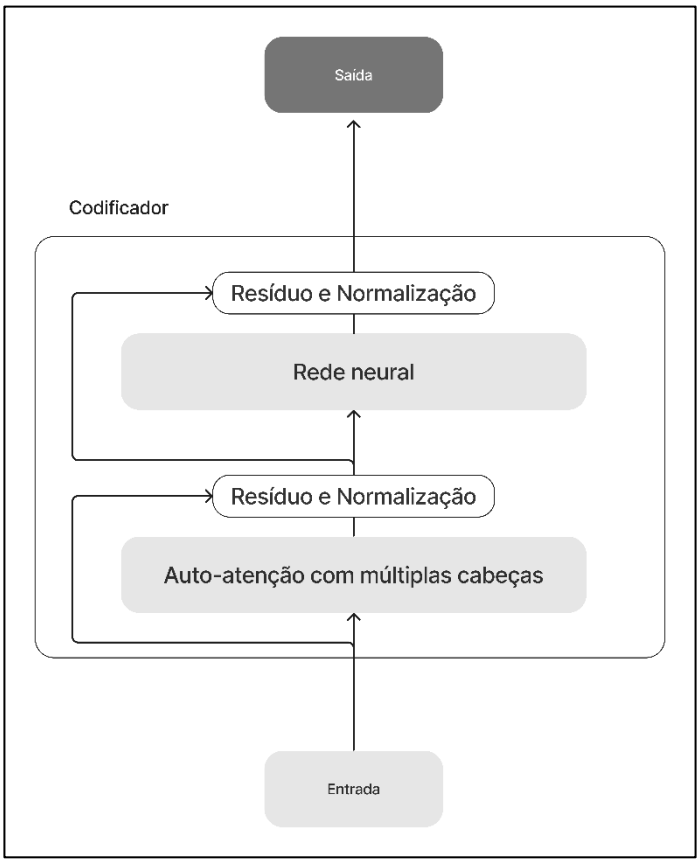
Figura 7 – Representação das pilhas do codificador e decodificador.



Fonte: Autoria própria.

Os sub-codificadores possuem estruturas idênticas e são constituídos de duas camadas: a primeiro é um mecanismo de auto-atenção com múltiplas cabeças (do inglês, *multi-head self-attention mechanism)*, que ajuda o codificador a observar outras palavras na entrada enquanto codifica uma palavra específica; a segunda é uma rede neural completamente conectada de uma camada. Também, é empregada uma conexão residual em torno de cada uma das duas subcamadas, seguidas pela normalização da camada.

Figura 8 – Representação do codificador.



Fonte: Autoria própria.

Os sub-decodificadores tem as mesmas duas camadas que os sub-codificadores, e insere uma terceira subcamada de atenção convencional que se comunica com o codificador, essa terceira camada ajuda o decodificador na concentração em partes relevantes da entrada. Também é empregada uma conexão residual em torno de cada uma das três subcamadas, seguidas pela normalização da camada.

Na Figura 8 é representado um codificador com as camadas conectadas. Ademais, vamos descrever os principais componentes e camadas nas seções seguintes, dando início pelo mecanismo de atenção, que juntamente com outros

componentes, fez dos *Transformers* e suas diversas variações o estado da arte em diversas tarefas de PLN (WOLF *et al.*, 2020).

4.1.2 Atenção

O mecanismo de auto-atenção dos *Transformers* deriva do mecanismo de atenção proposto por Bahdanau, Cho, Bengio (2015). Em primeiro lugar, o mecanismo de atenção nesta seção, consideremos o codificador e decodificador introduzidos por Cho *et al.* (2014), e não o dos *Transformers*.

O objetivo do mecanismo de atenção é que os itens mais relevantes da entrada recebam uma valoração maior no vetor de contexto, mas outros itens também podem receber algum valor. Por exemplo, suponha que você quer aprender a assar um bolo de chocolate, vamos chamar “Assar o bolo de chocolate” de consulta. Você pode pegar um livro de receitas, o livro é composto de diversas receitas, que vamos chamar de chaves. O que você quer é encontrar a receita mais adequada e para isso, todas as receitas vão receber alguma valoração. A receita do bolo de chocolate deve ter um valor maior em relação aos demais, mas uma receita de bolo de chocolate com morango também pode receber alguma relevância. Mas uma receita de purê de batata deveria ter uma relevância pequena (CASELI; NUNES, 2024).

O mecanismo de atenção segue essa ideia: a saída é a consulta, e a informação que precisa ser gerada a partir da entrada são as chaves. Para definir a chave mais relevante são calculados pesos de atenção, que definirão o vetor de contexto. Para considerar a relevância de diferentes itens na entrada, o codificador não considera que apenas o último estado escondido da rede será o vetor de contexto, mas sim todos os estados escondidos, ou seja, todos os estados obtidos após o processamento de cada item da sequência também podem participar do vetor de contexto. Mas agora o decodificador terá mais trabalho, pois ele precisará decidir o que fazer com esses vários vetores antes de gerar os itens da saída, e ainda considerando que é necessário focar nas partes mais relevantes para a resolução da tarefa. Assim, antes de gerar a saída pelo decodificador são executados os seguintes passos:

1. Computar uma pontuação para cada estado escondido, também chamada de pontuação de alinhamento, seguindo a Equação 3.

2. Passar as pontuações combinadas – por concatenação, em geral, ou o vetor resultante da equação anterior, pensando em termos de representações matriciais – por uma função de *softmax*, para capturar alguma noção de probabilidade da relevância produzindo os pesos de atenção.
3. Multiplicar cada estado escondido (lembrando que ele é representado por um vetor) pelos pesos de atenção, de forma a tornar os estados escondidos mais relevantes com valores ainda maiores e obter o efeito oposto para os estados menos relevantes. O resultado deste passo será o vetor de contexto.

$$att = \mathbf{W}_{\text{combinado}} \times \tanh\left(\mathbf{W}_{\text{decod}} \times \mathbf{H}_{\text{decod}} + \mathbf{W}_{\text{codif}} \times \mathbf{H}_{\text{codif}}\right) \quad (3)$$

Onde $\mathbf{W}_{\text{codif}}$ e $\mathbf{W}_{\text{decod}}$ representam matrizes de pesos (parâmetros) aprendidos e $\mathbf{H}_{\text{decod}}$ representam estados escondidos (CASELI; NUNES, 2024). O mecanismo de atenção calculado desta forma também é chamado de mecanismo aditivo ou mecanismo de Bahdanau, proposto em Bahdanau, Cho, Bengio (2015).

Existe um segundo tipo de atenção, proposto em Luong, Pham, Manning (2015), chamado de mecanismo de atenção multiplicativo. As diferenças principais são que o decodificador produz um estado intermediário a partir do estado escondido anterior antes de calcular os pesos de atenção, e as pontuações de alinhamento podem ser de três tipos:

I. Multiplicando os estados escondidos do codificador e decodificador apenas, ou seja, $att = \mathbf{H}_{\text{decod}} \times \mathbf{H}_{\text{codif}}$.
II. Multiplicando uma matriz de pesos aprendidos ao resultado da multiplicação em (i), ou seja, $att = \mathbf{W} \times \mathbf{H}_{\text{decod}} \times \mathbf{H}_{\text{codif}}$.
III. Somando os estados escondidos do codificador e decodificador, que são multiplicados por uma matriz de pesos e passadas pela função de ativação da tangente hiperbólica e são finalmente multiplicadas a uma matriz de pesos: $att = \mathbf{W} \times \tanh\left(\mathbf{W}_{\text{combinado}}(\mathbf{H}_{\text{decod}} + \mathbf{H}_{\text{codif}})\right)$.

Este último caso é o mais similar ao mecanismo aditivo, mas os estados escondidos compartilham uma matriz de pesos, diferente da Equação 3. Ao final, o vetor de contexto é concatenado com o estado do decodificador no instante anterior, para produzir uma nova saída. O mecanismo de atenção apresentado até agora é chamado de mecanismo de atenção geral, uma vez que ele tenta encontrar os

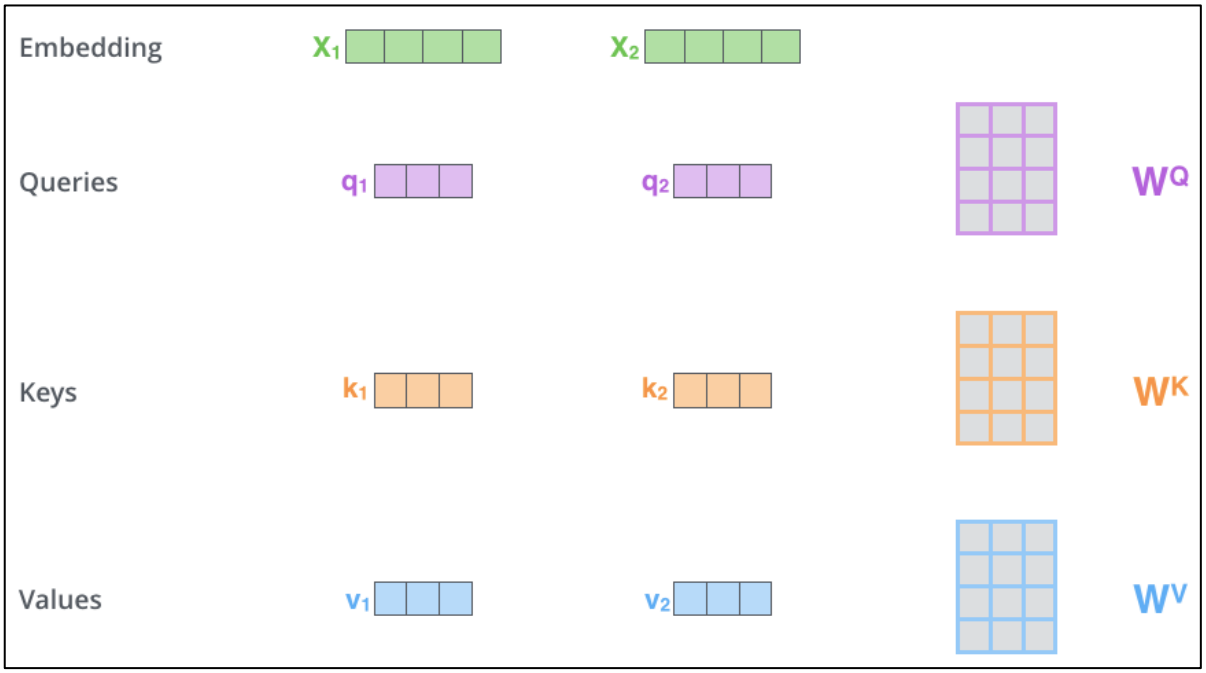
componentes da entrada que são mais relevantes para gerar a saída (CASELI; NUNES, 2024).

4.1.2.1 Auto-atenção

Transformers fazem uso de um mecanismo de atenção adicional, chamado de auto-atenção (do inglês, *self-attention*), em que a captura da relevância é feita entre os elementos de uma mesma sequência, usualmente da entrada.

Considere que cada item da sequência (um *token*, uma palavra) é representado por um *embedding*. Considerando o ponto de entrada de um *transformer* como sendo o vetor de *embeddings*, são criados três vetores a partir de cada palavra ou *token*. A implementação é matricial para fazer bom uso das *Graphics Processing Unit* (GPU).

Figura 9 – Ilustração dos vetores no mecanismo de auto-atenção para uma entrada com duas palavras.



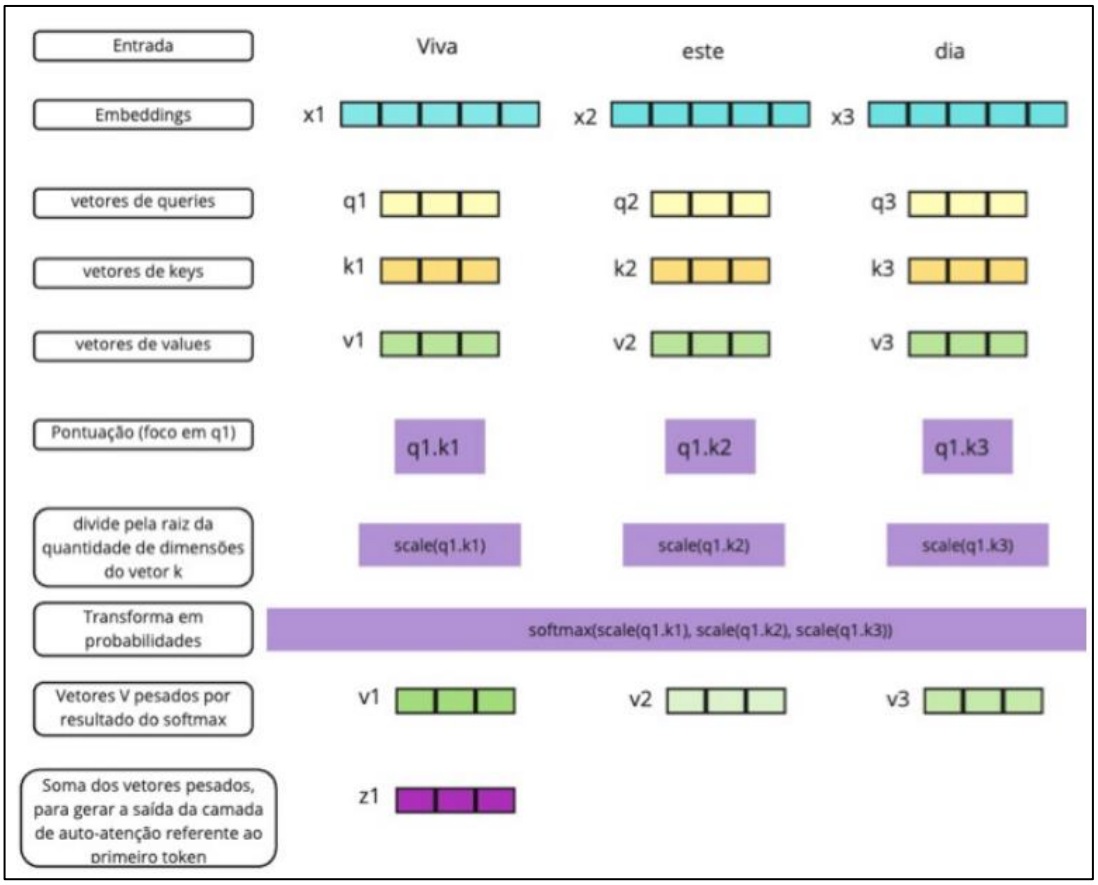
Fonte: Alammar, 2024.

Os *Transformers* fazem uso de *tokens* de subpalavras para amenizar o problema das palavras que estariam fora de um vocabulário pré-treinado. Para fim de facilitar o entendimento vamos assumir que cada palavra é um *token*, abstrair as matrizes para vetores e considerar a frase “viva este dia”, na Figura 9 temos uma ilustração para uma frase com duas palavras. Temos então três *tokens* na frase, que serão representados pelos vetores $x_1 - \text{viva}$, $x_2 - \text{este}$ e $x_3 - \text{dia}$, que são os

embeddings de cada palavra. A partir de cada um deles criamos três outros vetores, q, k, e v, de $queries$ (consultas), $keys$ (chaves) e $values$ (valores). O vetor q se refere a um item de interesse que está sendo codificado. O vetor k se refere aos demais itens da sentença. O vetor v representa a codificação do valor dado a cada item, considerando o item de interesse. Para nosso exemplo “viva este dia”, temos então os vetores q_1, k_1 e v_1 para a palavra “viva”, q_2, k_2 e v_2 para a palavra “este” e q_3, k_3 e v_3 para a palavra “dia”. Os vetores são obtidos usando matrizes de pesos aprendidos com os dados. Assim, multiplicando o vetor x_1 pela matriz de pesos associados às $queries$, $\mathbf{W}_Q$, temos o vetor q_1, isso vale para os demais itens, ou seja, para obter o vetor k_1 multiplicamos x_1 por uma outra matriz de pesos $\mathbf{W}_K$, e para obter v_1 multiplicamos x_1 por uma matriz de pesos $\mathbf{W}_V$ (CASELI; NUNES, 2024).

Para calcular o peso de atenção, digamos que estamos calculando a primeira palavra “viva” (na Figura 10 é ilustrado esse processo de cálculo considerando como entrada a frase “Viva este dia” de exemplo), precisaremos calcular o peso de cada palavra da frase de entrada em relação a esta palavra. O peso de atenção determina quanto foco colocar em outras palavras da frase de entrada à medida que codificamos uma palavra (ALAMMAR, 2024). Então, multiplica-se o seu vetor q_1 por cada uma das $keys$, k_1, k_2 e k_3. O mesmo será feito para as demais palavras. Os valores multiplicados serão divididos pela $\sqrt{d_k}$, que é raiz quadrada da dimensão dos vetores de $keys$. Essa divisão deixa os cálculos dos gradientes estáveis, podendo ter outros valores dependendo das dimensões dos vetores. A seguir, os valores serão passados por uma função de $softmax$, ou seja, vão ser normalizados para que todos os valores sejam positivos e a soma deles igual a 1, sendo assim, transformados em probabilidades. Com isso, os valores calculados pelo $softmax$ são multiplicados por cada um dos vetores de $values$, para codificar a importância das demais palavras na valoração, que tem a intuição de manter os valores das palavras que merecem foco e abafar as palavras sem importância (multiplicando-as por números pequenos). Finalmente, esses valores são somados, produzindo um valor final, chamado de z_1 para a primeira palavra, que é a saída da camada de auto-atenção, que será passado adiante para a rede neural completamente conectada.

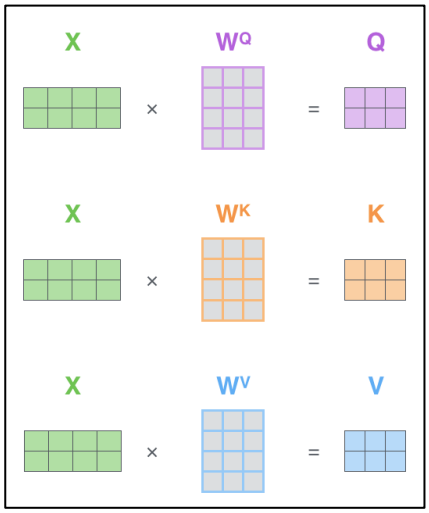
Figura 10 – Exemplo do mecanismo de auto-atenção
para codificar a frase “Viva este dia”.



Fonte: Caseli e Nunes, 2024.

Nas implementações esses cálculos são realizados em matrizes para um processamento eficiente em GPU, conforme na Figura 11 (ALAMMAR, 2024; CASELI; NUNES, 2024; VASWANI *et al*, 2017).

Figura 11 – Multiplicação em matrizes do mecanismo de auto-atenção.



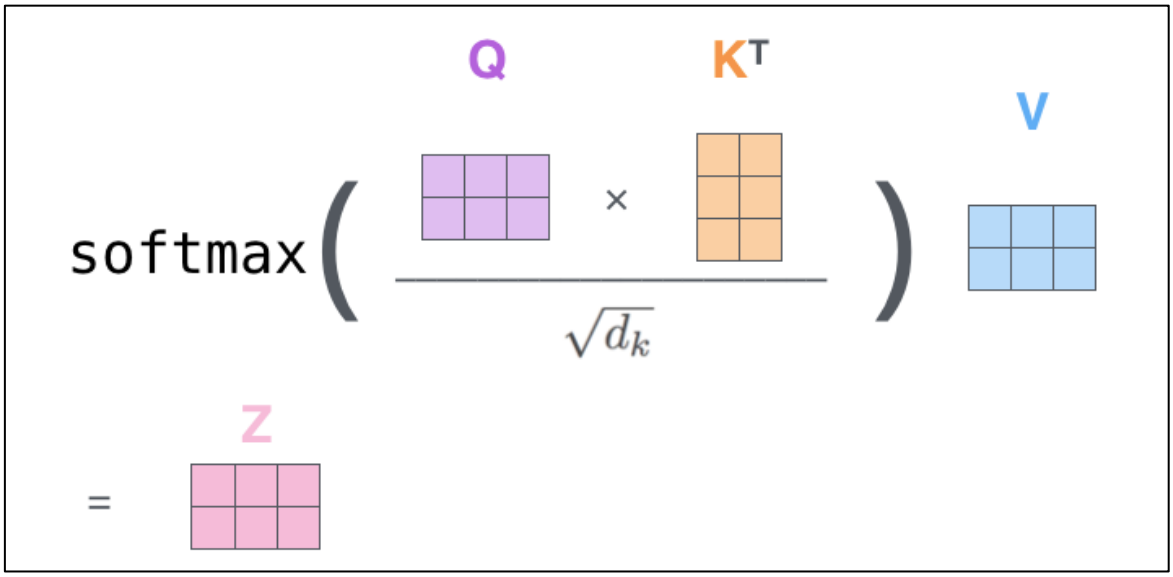
Fonte: Alammar, 2024.

Como o mecanismo de auto-atenção dos *Transformers* utiliza matrizes, temos em Vaswani *et al* (2017) o produto escalar escalonado (do inglês, *scaled dot-product attention*):

$$auto - atenção(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4)$$

Na figura 12 temos a equação 4 ilustrada visualmente.

Figura 12 – O cálculo do mecanismo de auto-atenção.



Fonte: Alammar, 2024.

Quando $d_k = d_v$, o mesmo vetor pode ser fornecido como as três entradas do mecanismo de atenção onde a consulta chave e valor, são calculados a partir de um mesmo vetor. A terminologia de auto-atenção é motivada pelo fato do mecanismo estar comparando um vetor de entrada consigo mesmo e determinando quais valores dele devem ser passados adiante. Sendo uma operação matemática que pode ser aplicada a qualquer dado que admita uma partição em consulta, chave e valor (CASELI; NUNES, 2024). Os *Transformers* são icônicos pela aplicação de atenção a dados distintos da entrada geral (DANTAS, 2021).

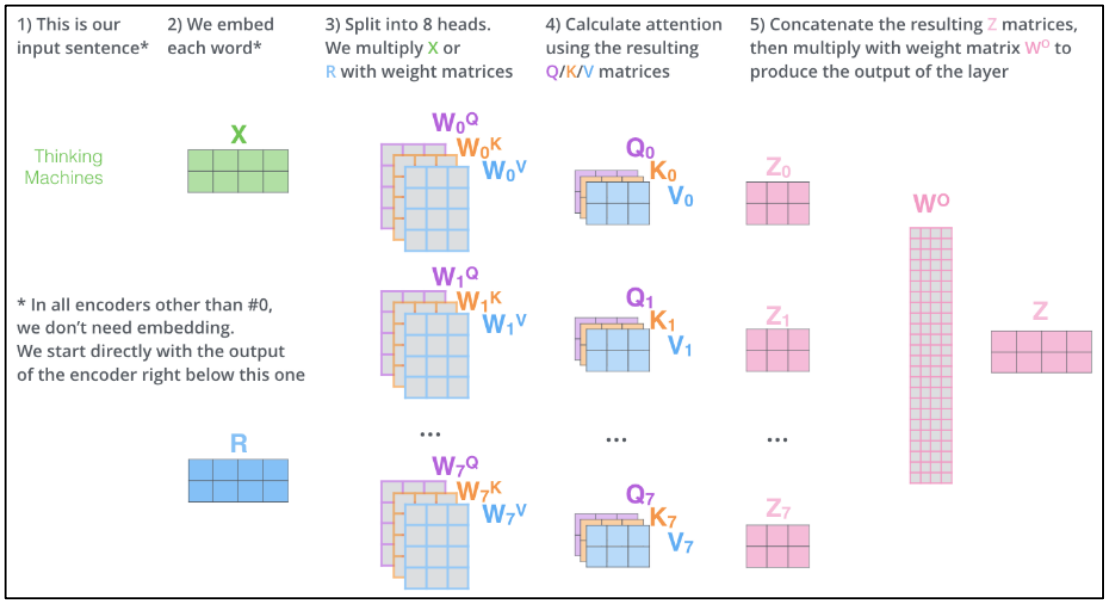
4.1.2.2 Auto-atenção com múltiplas cabeças

A linguagem é sujeita a várias complexidades que podem fazer o mecanismo de atenção não ser suficiente para representá-las, dependendo da sentença de entrada podemos ter variações na atenção. Isso é bem comum em problemas de correferência, por exemplo, na frase “Eles não levaram os livros nos compartimentos

porque eles eram muito pequenos”, o segundo “eles” pode se referir a vários outros pronomes e substantivos na frase, dependendo do contexto, as palavras podem ter vários significados, um conceito chamado de polissemia. Então, para capturar diferentes representações de uma palavra, bem como diferentes interações da palavra com os demais componentes, *Transformers* incluem um nível de paralelismo no processamento da entrada, a partir de um componente adicional de atenção com múltiplas cabeças (CASELI; NUNES, 2024).

O mecanismo de atenção produz uma matriz final Z após processar a entrada a partir das diferentes matrizes de peso Q, K e V. Como a saída do passo de atenção é submetido a uma rede neural completamente conectada, para que a rede completamente conectada consiga lidar com as várias matrizes Z geradas em paralelo, elas são concatenadas e multiplicadas por uma outra matriz de pesos adicional gerando, finalmente, uma única matriz Z que representa o resultado do mecanismo de atenção em suas várias versões. Como essa matriz de pesos adicional também é aprendida, *Transformers* dão a chance de alguma das versões do mecanismo de atenção paralelo ter mais ou menos relevância que algum outro, dependendo dos dados de treinamento (ALAMMAR, 2024; CASELI; NUNES, 2024; VASWANI *et al*, 2017).

Figura 13 – Etapas do mecanismo de auto-atenção.



Fonte: Alammar, 2024.

No artigo que apresenta a arquitetura *Transformers – Attention Is All You Need* – Vaswani *et al.* (2017), utiliza-se de oito versões ($h = 8$) paralelas do mecanismo de atenção, que no artigo original é chamado de cabeças (do inglês,

heads), assim tem oito vezes três matrizes de $queries$, $keys$ e $values$ inicializados aleatoriamente, o que permite que tais matrizes capturem diferentes aspectos da entrada. Na figura 13 vemos um exemplo do mecanismo de auto-atenção com a entrada "*Thinking Machines*".

Assim, temos adaptado de Vaswani *et al.* (2017):

$$\begin{aligned} &múltiplas\ cabeças(Q,K,V) = concatenação(cabeça_1,\dots,cabeça_h)W^O \\ &onde\ cabeça_i = auto-atenção\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{5}$$

4.1.3 Codificação posicional

Os *Transformers* possuem uma entrada de tamanho pré-definido. O tamanho pré-definido, em geral, é até bem menor do que gostaríamos para manipular textos um pouco mais longos, por questões de desempenho. Sendo possível lidar com sentenças de tamanhos distintos nos *Transformers*, adotando alguma das abordagens abaixo:

- Quando a sentença de entrada tem menos *tokens* que a quantidade de *tokens* de entrada esperada pelo modelo: esse é o caso mais fácil, quando a sentença é preenchida com valores nulos, chamado de *padding*.
- Quando a sentença de entrada tem mais tokens que a quantidade de tokens de entrada esperada pelo modelo: duas soluções podem ser adotadas, a mais simples é truncar a entrada, removendo elementos do início ou do fim da sentença e a outra forma mais elaborada é quebrar a sentença em janelas com elementos sobressalentes entre elas e passar esses pedaços ou janelas várias vezes no modelo.

Outro problema é a ordem das palavras pois exige a inclusão de um componente adicional no modelo, uma vez que a ordem é de extrema relevância para a sintaxe e a semântica, e, portanto, também para modelos que tentam aprender a resolver tarefas sintáticas ou semânticas. Assim, *Transformers* incluem um tipo especial de *embedding* chamado de codificador de posição (do inglês, *positional encoding*), para contemplar alguma informação sobre as posições dos *tokens* durante o aprendizado. O codificador posicional é um vetor a mais somado ao vetor de *embeddings* de entrada de cada *token* (CASELI; NUNES, 2024).

Embora em um primeiro pensamento possa parecer mais direto considerar um valor simples de posição como, por exemplo, um índice. Essa abordagem traria alguns problemas, o primeiro é que o valor pode ficar muito grande dependendo de quantas palavras temos, e o modelo poderia se confundir achando que esses valores altos têm alguma importância. Mesmo se o valor fosse normalizado entre 0 e 1, diferentes tamanhos de sentenças trariam diferentes valores, o que também atrapalharia a generalização do aprendizado (CASELI; NUNES, 2024).

Assim, o codificador de posição define um vetor de valores contínuos do mesmo tamanho do *embedding* de entrada do *token*, para que seja possível somá-los. Para incorporar mais uma ideia de distância entre as palavras, ou de posição relativa, do que uma ideia rígida de posição, o vetor é obtido a partir de uma função que intercala entre a aplicação de seno ou cosseno, a diferença entre o uso de seno e cosseno no codificador de posição está na periodicidade e na fase inicial das funções, a função seno e a função cosseno são ambas funções periódicas, o que significa que elas se repetem em intervalos regulares. Isso é útil para codificar a posição, pois permite que o modelo capture padrões que ocorrem em intervalos regulares nas sequências de entrada, pois elas estão desfasadas uma da outra, capturando diferentes aspectos da posição relativa dos *tokens*. Mais precisamente, para codificar a informação de posição de um *token* que está em uma posição k na sequência de entrada, considerando cada posição i do vetor posicional, fazemos:

$$p(k, 2i) = \sin\left(\frac{k}{n\frac{2i}{d}}\right) p(k, 2i+1) = \cos\left(\frac{k}{n\frac{2i}{d}}\right) \quad (6)$$

Onde d é a dimensão do vetor posicional e n é um valor pré-definido. Para posições pares do vetor de saída, aplica-se o seno e para posições ímpares, aplica-se o cosseno. (CASELI; NUNES, 2024; VASWANI *et al*, 2017).

A tabela 1 apresenta um exemplo simplificado da aplicação do codificador posicional, considerando um vetor de saída de quatro dimensões e $n = 10$.

Tabela 1 – Exemplo da computação do codificador posicional.

Token	**Índice na sentença**	**i = 0**	**I = 1**	**I = 2**	**I = 3**
viva	0	0	1	0	1
este	1	0,8415	0,5403	0,3109	0,9504
dia	2	0,9093	-0,4161	0,5911	0,8607

Fonte: Caseli e Nunes, 2024.

4.1.4 Resíduo e normalização

O último subcomponente dos *Tranformers* que precisamos falar é a inclusão de duas conexões residuais (HE *et al.*, 2016) dentro da subcamada de codificação. A conexão residual surgiu na área de visão computacional, com a motivação que redes neurais com muitas camadas podem esquecer uma informação importante de entrada após ela passar por muitos processamentos. Em geral, esse esquecimento se dá pelo problema do gradiente que vira zero depois de muitas multiplicações de valores menores que um (HOCHREITER, 1991), durante o *backpropagation*, que é o algoritmo mais utilizado para aprender os pesos de uma rede neural, usando o método de otimização de gradiente descendentes visando minimizar o erro do modelo. A conexão residual evita justamente parte dessas transformações multiplicativas, pulando algumas delas.

No caso dos *Transformers*, além de evitar que o treinamento se perca com multiplicações de valores muito pequenos, a motivação é que os *embeddings* de representação de palavras também continuem a ser aproveitados de alguma forma, trazendo uma ideia de representação local dos *tokens* para a subcamada de codificação. Ou seja, ao permitir que a informação sem ser processada pela camada de auto-atenção e a informação sem ser processada pela camada completamente conectada sejam consideradas, é como se a rede estivesse lembrando da representação original do *token*, quando necessário. Para tanto, a saída da camada de auto-atenção é somada com a entrada original, que por sua vez é somada com a saída da camada completamente conectada, preservando e repassando para a frente de alguma forma, a entrada original. Para ajudar no aprendizado dos gradientes, antes da camada de atenção e antes da camada completamente conectada, temos uma camada de normalização (CASELI; NUNES, 2024).

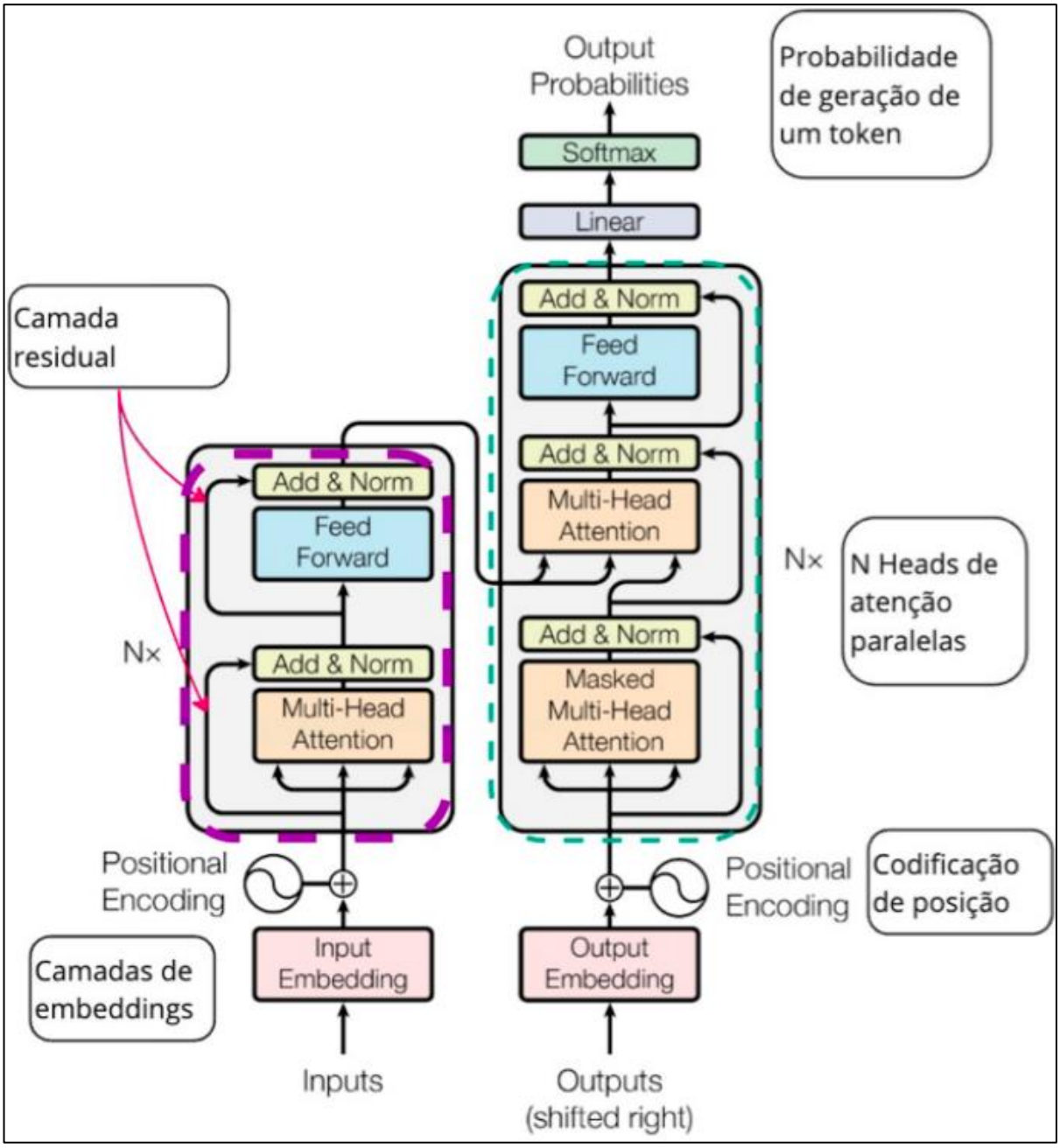
4.1.5 Arquitetura

A arquitetura *Transformer* original possui em seu componente de codificação seis subcamadas de codificadores, com as camadas internas completamente conectadas tendo 512 neurônios artificiais intermediários (na camada oculta ou escondida) e 8 cabeças de atenção. Para ajudar no aprendizado dos gradientes,

antes da camada de atenção e antes da camada completamente conectada, temos uma camada de normalização (VASWANI *et al*, 2017).

Na figura 14, temos uma visão geral da arquitetura *Transformer*, modificado a partir de (VASWANI *et al*, 2017), a qual o pontilhado violeta representa o codificador e o pontilhado verde representa o decodificador.

Figura 14 – Arquitetura Transformer.



Fonte: Caseli e Nunes, 2024.

4.1.6 Comunicação entre codificador e decodificador

Após o processo de codificação as matrizes de atenção K e V serão a entrada para a camada de atenção convencional do decodificador. Já a matriz Q vem mesmo da camada anterior, justamente para que essa camada ajude o modelo a decidir o que ele precisa considerar para gerar a saída. A camada de auto-atenção do decodificar é chamada de mascarada (do inglês, *masked multi-head attention*), uma vez que ela não tem acesso aos *tokens* que estão em uma posição posterior a um certo *token*. Assim, no decodificador, considerando o processamento de um token t_i, a camada de auto-atenção do decodificador só terá acesso aos *tokens* $\{t_o, t_1 \dots, t_{i-1}\}$ para calcular o valor da auto-atenção. Este comportamento tem a ver com o que espera-se de um decodificador: que ele gere um próximo *token*, dados os *tokens* anteriores a ele, mas sem saber o futuro de antemão (CASELI; NUNES, 2024).

Assim, o decodificador lembra muito o processo do modelo probabilístico que discutimos no referencial teórico, entretanto, nos modelos probabilísticos fica claro a existência das probabilidades, enquanto até agora só falamos em vetores. *Transformers* incluem uma última camada de rede neural, bem como na equação 7, justamente para resolver tal discrepância. Assim, a última camada da arquitetura recebe a saída do decodificar (um vetor) e a processa com uma camada linear completamente conectada. A saída da camada linear completamente conectada é um vetor de *logits* (logaritmos de probabilidades não normalizados), do tamanho do vocabulário, que representam uma pontuação associada a cada palavra do vocabulário. Finalmente, tais valores de pontuação passam por uma operação de $softmax$, para converter esses valores reais em valores que fiquem entre 0 e 1, representando a probabilidade de que o decodificador emita cada uma das palavras do vocabulário (CASELI; NUNES, 2024).

$$rede\ neural(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \tag{7}$$

4.2 Instância dos *Transformers* no GPT

Tarefas que lidam com linguagem têm sido abordados por diferentes instanciações de *Transformers*: podemos considerar a arquitetura completa, podemos considerar apenas o componente codificador ou apenas o componente

decodificador. Ainda, é possível não usar todas as camadas existentes no modelo original, mas subconjuntos (ou até mesmo superconjuntos) delas (CASELI; NUNES, 2024).

Os *Transformers*, o aumento das capacidades computacionais e a disponibilidade de dados de treinamento em larga escala impulsionaram avanços significativos em modelos de linguagem, permitindo a criação de Modelos de Linguagem Grandes (do inglês, *Large Language Models* (LLM)), os quais têm demonstrado notáveis capacidades em tarefas de processamento de linguagem natural e, além disso, são usados em uma variedade de aplicações como: tradução, resumo, recuperação de informações e interações conversacionais (CHEN *et al.*, 2023).

O GPT usa blocos decodificadores da arquitetura *Transformers* para funcionar como um modelo autorregressivo de geração de texto. Sem o componente codificador, o GPT não tem o mecanismo de atenção tradicional como parte das entradas do decodificador.

Em sua primeira versão, o GPT (RADFORD *et al.*, 2018) era similar ao componente decodificador do *Transformer*, sendo composto de 12 subcamadas de decodificadores com 12 cabeças (do inglês, heads) de auto-atenção mascaradas de dimensão 768, camadas escondidas completamente conectadas de 3.072 dimensões e 117 milhões de parâmetros (que se referem aos pesos e vieses das camadas da rede neural). A tokenização também é baseada em subpalavras, mas segue o algoritmo *Byte Pair Encoding*, que é inspirado em técnicas de compreensão de dados e busca representar como subpalavras os *tokens* mais frequentes. No treinamento não supervisionado pré-treinado foi usado o *corpus BookCorpus*, *Stanford Natural Language Inference*, *RACE* e *Quora*, seguido de um ajuste fino com treinamento supervisionado adaptando o modelo para tarefa discriminativa com dados rotulados.

A segunda versão, GPT-2 (RADFORD *et al.*, 2019) veio em quatro versões de tamanhos variados, descritos na Tabela 2. A camada de normalização passou a estar na entrada de cada subcamada e adicionou-se uma outra camada de normalização após o último bloco de auto-atenção. As mudanças arquiteturais do GPT para GPT-2 não foram profundas, exceto pelo tamanho e consequentemente quantidade de parâmetros treinados. Porém, em Radford *et al*. (2019) começou-se a vislumbrar um modelo mais geral, que pudesse executar várias tarefas (aprendizado

de múltiplas tarefas ou agnóstico de tarefas (COLLOBERT; WESTON, 2008)), mesmo sem ser treinado novamente para cada uma delas (configuração (ROMERA-PAREDES; TORR, 2015)), e usando apenas a geração de texto como uma abstração de qualquer outra tarefa mais específica. O pré-treinamento em um conjunto grande e diverso de textos seria suficiente para que o modelo pudesse lidar com problemas com os quais não havia sido explicitamente treinado. Algumas sentenças nos textos usados para o pré-treinamento já eram exemplos de tradução de uma língua para a outra, o que faria o modelo aprender a traduzir naturalmente. No treinamento foi usando *corpus* com 8 milhões de páginas web, totalizando 40 GB, e um conjunto de dados *WebText* criado a partir do rastreamento de todos os links do *Reddit*.

Tabela 2 – Versões do GPT-2.

Versão do GPT-2	**Subcamadas de decodificadores**	**Dimensão dos *embeddings***	**Parâmetros**
GPT-2 *small*	12	768	124 milhões
GPT-2 *medium*	24	1.024	335 milhões
GPT-2 *large*	36	1.280	774 milhões
GPT-2 *extra large*	48	1.600	1.5 bilhões

Fonte: Adaptado de Radford *et al.*, 2019.

Este foi o principal motivador para o desenvolvimento do GPT-3 (BROWN *et al.*, 2020). A ideia seria que durante o pré-treinamento o modelo consegue desenvolver indiretamente habilidades de geração de texto que poderiam ser usadas para resolver diversas tarefas, como tradução e resposta a perguntas. Tais habilidades poderiam ser resgatadas em tempo de execução de acordo com a tarefa pedida, um processo chamado de aprendizado em um contexto (DONG *et al.*, 2023).

Três configurações foram discutidas por Brown *et al.* (2020), que já eram objeto de estudo de outros trabalhos voltados para o aprendizado a partir de poucos exemplos:

1. *Zero-shot* que explora cenários em que o modelo recebe como contexto uma descrição da tarefa (que até poderia ser opcional, dependendo da tarefa).
2. P*rompt* que é um texto em linguagem natural que especifica uma instrução do que deve ser feito, e espera-se que o modelo responda a

partir destes dois componentes, sem nenhum tipo de ajuste nos seus pesos.

3. *Few-shot* que é mais de um exemplo fornecido para o modelo ter como base.

Intuitivamente, a descrição da tarefa e o *prompt* direcionarão o modelo para certos pesos que ativarão as distribuições de probabilidade de geração de textos para o ponto correto. É como se ele estivesse sempre completando textos que tenham alguma coerência com o que foi visto antes. De acordo com o que o GPT é, um modelo gerativo, que são desenhados para predizer o próximo token a partir dos tokens anteriores (CASELI; NUNES, 2024).

Em termos de arquitetura, no GPT-3 foi usada a mesma do GPT-2, mas com uma alteração no mecanismo de atenção para fatorizar de forma esparsa as matrizes de atenção (CHILD *et al.,* 2019) e reduzir a complexidade do mecanismo de atenção de $O(n^2)$ para $O(n\sqrt{n})$. Foram treinados oito modelos de diferentes tamanhos, variando de 12 a 96 camadas e dimensão de *embeddings* de 768 a 12.288. O maior deles, com 175 bilhões de parâmetros foi o que obteve os melhores resultados, em geral, e é o GPT-3, treinado com o *corpora* descrito na tabela 3.

Tabela 3 – *Corpora* de treinamento do GPT-3.

Corpus	**Quantidade de tokens (bilhões)**
CommonCrawl	410
WebText2	19
Books1	12
Books2	55
Wikipedia	3

Fonte: Adaptado de Brown *et al.*, 2020.

A OpenAI começou gradativamente a disponibilizar versões melhoradas do GPT-3, introduzindo novas formas de treinamento que se valem de cada vez mais textos e modelos cada vez maiores, o GPT-3 foi treinado em cerca de 45 TB de dados de texto puro (BROWN *et al.*, 2020). A OpenAI disponibilizou novas versões e *application programming interfaces* para os modelos que estenderam o GPT-3, chamados de *text-davinci-003* e *code-davinci-002*, que passaram a ser genericamente chamados de GPT-3.5. O GPT-3.5 é a base para o ChatGPT, um agente de conversão genérico e que foi disponibilizado em 2022 (CASELI; NUNES, 2024).

O GPT-4 foi lançado em 2023 e diferente dos modelos anteriores, ele é multimodal que aceita imagens e textos como entradas e emite textos como saídas. Apresentou desempenho de nível humano em vários testes profissionais e acadêmicos, como o exame da Ordem dos Advogados dos EUA, onde passa em um exame simulado com uma pontuação entre os 10% dos melhores. O desenvolvimento do GPT-4 incluiu a reconstrução da pilha de aprendizado profundo da OpenAI e o projeto de um supercomputador em conjunto com a *Azure*. Como resultado, a execução de treinamento do GPT-4 foi estável, tornando-se o primeiro modelo cujo desempenho de treinamento foi possível prever com precisão (OPENAI *et al.*, 2023).

Em uma conversa casual, a distinção entre GPT-3.5 e GPT-4 pode ser sutil. A diferença surge quando a complexidade da tarefa aumenta, onde o GPT-4 é mais confiável, criativo e capaz de lidar com instruções muito mais matizadas e múltiplas línguas do que o GPT-3.5. Esforços significativos foram feitos para melhorar a segurança e o alinhamento do GPT-4, incluindo a seleção e filtragem de dados de pré-treinamento e a inclusão de um sinal de recompensa de segurança durante o treinamento com *Reinforcement Learning from Human Feedback* (RLHF). Essas medidas reduziram a tendência do modelo de gerar conteúdos prejudiciais (OPENAI *et al.*, 2023). No artigo original não é informado a quantidade de camadas e parâmetros.

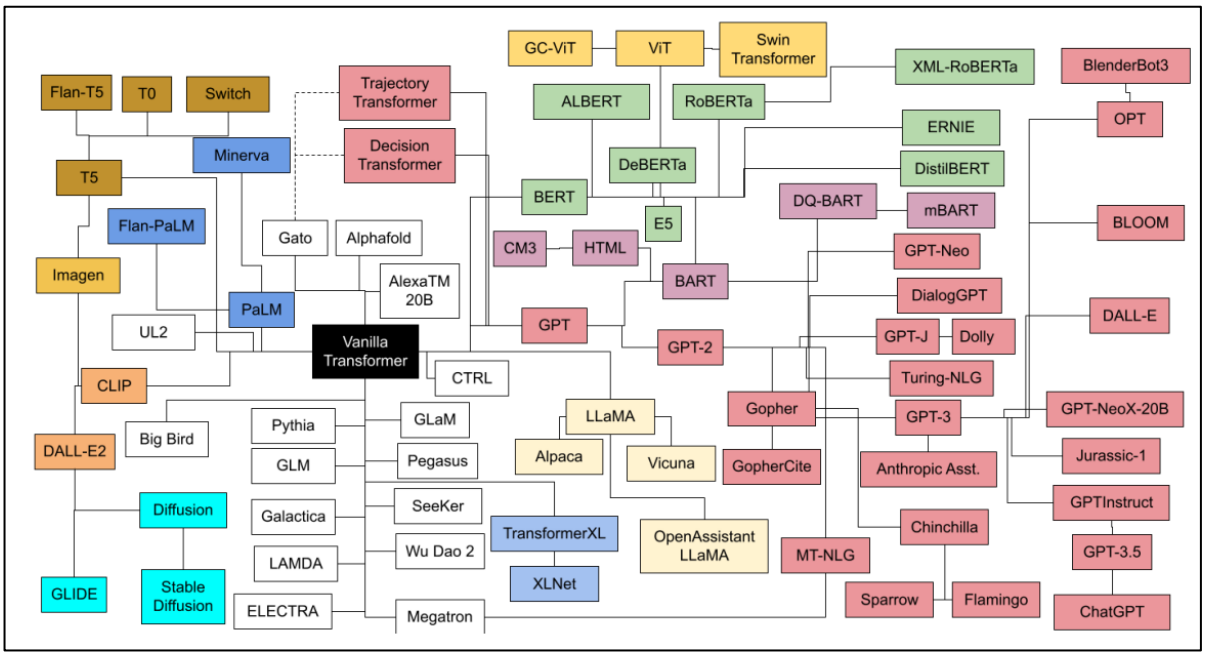
Depois foi lançado GPT-4 *Turbo* em 2024, sendo uma versão aprimorada do GPT-4 com um limite de contexto de 128 mil *tokens*, que é equivalente a 300 páginas de texto em um único *prompt*, limite de conhecimento atualizado até abril de 2023 com melhor desempenho em texto de idiomas não ingleses e códigos de programação, maior velocidade e capacidades aprimoradas de compreensão em visão computacional e áudio (OPENAI, 2024a).

O GPT-4o (“o” de “*omni*”) lançado em 2024 é um avanço do GPT-4 em direção a uma interação mais natural entre humanos e computadores. Ele aceita como entrada qualquer combinação de texto, áudio, imagem e vídeo e gera qualquer combinação de saídas de texto, áudio e imagem. Ele pode responder a entradas de áudio em apenas 232 milissegundos, com uma média de 320 milissegundos. O GPT-4o foi treinado de ponta a ponta em texto, visão computacional e áudio, o que significa que todas as entradas e saídas são processadas pela mesma rede neural. Em termos de avaliações de modelo, o GPT-4o atinge o desempenho do GPT-4

Turbo em texto, raciocínio e inteligência de codificação, enquanto melhora nas capacidades multilíngues, de áudio e de visão. Passou por uma extensa equipe de avaliação externa com mais de 70 especialistas de domínios como psicologia social, viés, justiça e desinformação para identificar riscos que são introduzidos ou amplificados pelas novas modalidades adicionadas (OPENAI, 2024b).

Na figura 15 visualizamos diversas instâncias do *Transformers*, inclusive o GPT e algumas derivadas a partir dele.

Figura 15 – Árvore Genealógica dos Transformadores.



Fonte: Amatriain *et al*., 2023.

4.2 Pré-treinamento

Podemos recuperar *embeddings* contextualizados tanto para *tokens* como para combinações de *tokens*, o que inclui sentenças e textos. *Embedding* de palavra é quando uma combinação de *tokens* resulta em uma palavra, é um *embedding* de palavra. Quando recuperamos os *embeddings* de uma frase, é um *embedding* de sentença. Modelos baseados em *Tranformers* permitem recuperar *embeddings* de sentenças de duas formas: construindo os *embeddings* a partir das médias dos *embeddings* de cada *token* na sentença, usando uma ou mais camadas da arquitetura (em geral, as quatro últimas camadas), ou usando os *embeddings* de saída do primeiro *token* (CASELI; NUNES, 2024).

Com *Transformers*, além dos *tokens* influenciarem uns aos outros com o mecanismo de atenção, também temos o codificador de posição, que influenciará na saída final. Uma outra possibilidade é treinar modelos que consigam devolver *embeddings* de sentenças de entrada, o que é feito costumeiramente tendo como base a tarefa de similaridade semântica e arquiteturas siamesas, como no modelo *sentence-transformers* (REIMERS; GUREVYCH, 2020) ou treinamentos contrastivos, como a abordagem SIM-CSE (GAO; YAO; CHEN, 2021).

Esse passo inicial de treinamento, que é parte dos modelos disponibilizados em arcabouços como *TensorFlow* e *HuggingFace*, é chamado de pré-treinamento. O pré-treinamento refere-se a uma técnica de treinamento de redes neurais profundas, que no caso de modelos de linguagem usa uma quantidade expressiva de textos sem nenhum rótulo ou anotação, com o intuito de gerar um modelo de propósito geral capaz de “entender” linguagem (CASELI; NUNES, 2024).

Segundo Howard e Ruder (2018), o estágio de pré-treinamento resulta em grandes melhorias no desempenho de um modelo neural e possibilita o aprendizado usando pouco exemplos, o GPT foi umas das primeiras aplicações da técnica a *Transformers* e constatou os ganhos com uso do pré-treinamento.

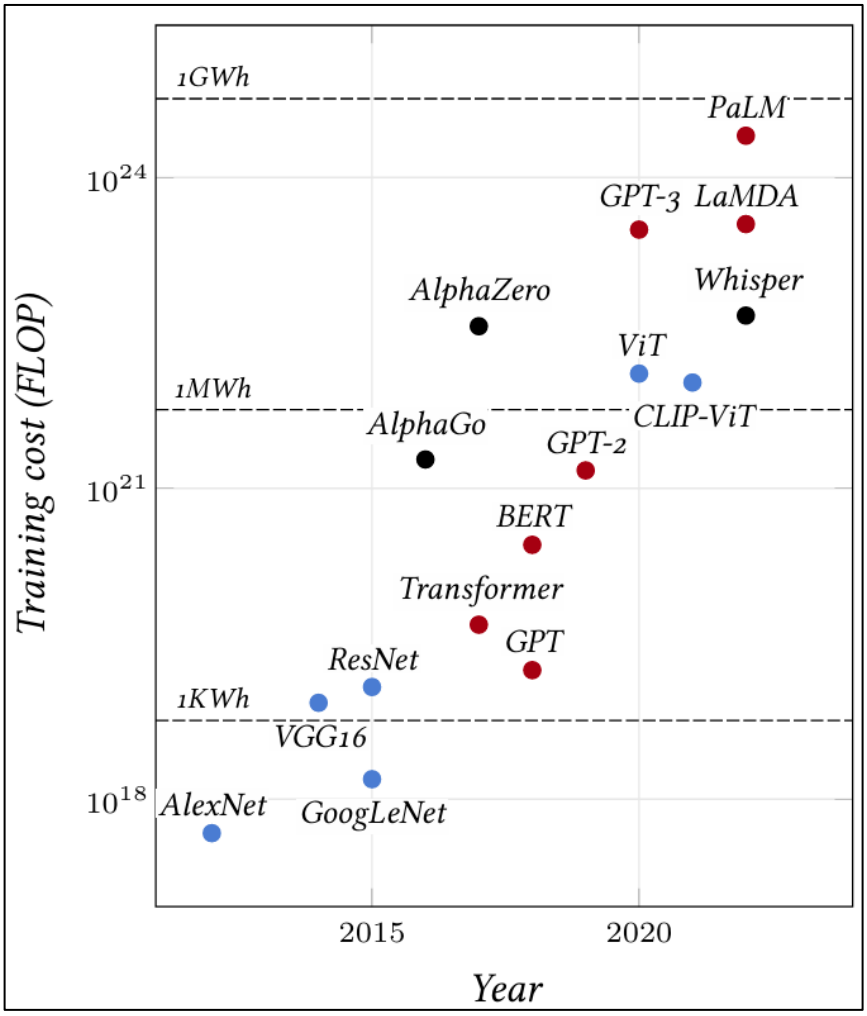
Durante o pré-treinamento de um modelo de linguagem, uma ou mais funções objetivo ou tarefas intermediárias são utilizadas para guiar o aprendizado do modelo a gerar texto, ou, de forma mais genérica, a predizer partes do texto que estejam faltando. O intuito é que o modelo passe a ter uma compreensão estatística da(s) língua(s) em que foi treinado, ressaltando detalhes de suas arquiteturas e *corpora* usado para treinar os pesos iniciais de cada um dos modelos (CASELI; NUNES, 2024).

Porém, tem a necessidade do uso intensivo de uma quantidade significativa de recursos computacionais por longos períodos, resultando em alto consumo de energia. Na Figura 16 é representado os custos de treinamento em número de *floating point operations per second* de alguns modelos, as cores indicam os domínios de aplicação: visão computacional em azul, PLN em vermelho e outros em preto. As linhas tracejadas correspondem ao consumo de energia usando A100s SXM da *NVIDIA* em precisão de 16 *bits*. Para referência, o consumo total de eletricidade nos EUA em 2021 foi de 3.920 TWh.

O fato dos textos não terem rótulos ou anotações é o que permite usar uma quantidade enorme de textos, pois anotar exemplos é uma tarefa custosa e que

requer um tempo precioso de especialistas. Apesar de os modelos de linguagem serem treinados usando coleções vastas e diversas de textos, eles podem se tornar obsoletos, uma vez que essas coleções são estáticas. Com o tempo, o modelo pode não ser capaz de gerar e reconhecer textos sobre eventos atuais (CASELI; NUNES,

Figura 16 – Custos de energia de treinamento dos modelos.



Fonte: Fleuret, 2023.

2024). Uma das formas encontradas para atualizar modelos de linguagem é o que chamamos de treinamento continuado (do inglês, *continued pre-training*) (GURURANGAN *et al.*, 2020; JIN *et al.*, 2022; KE *et al.*, 2023).

Modelos pré-treinados podem ser ajustados de acordo com um domínio ou uma tarefa específica com ajuste fino (do inglês, *fine-tuning*), que, assim como o pré-treinamento, é um processo de treinamento auto-supervisionado atuais (CASELI; NUNES, 2024).

Diante dos resultados, Radford *et al.* (2019) e Brown *et al.* (2020) propõem a redução das diversas tarefas de PLN a mera modelagem de linguagem.

Argumentando que um modelo suficiente pré-treinado será capaz de realizar tarefas tipicamente alcançadas através de treino supervisionado direto.

4.3 ChatGPT

O ChatGPT é um agente de conversão *online* que usa o GPT para geração de texto, foi desenvolvido pela OpenAI e lançado em novembro de 2022. Usa os modelos GPT fundamentais, especificamente GPT-3.5, GPT-4 e GPT-4o, foi ajustado para aplicações conversacionais usando uma combinação de técnicas de aprendizado supervisionado e de reforço com RLHF (OPENAI, 2024c).

O RLHF torna possível o alinhamento entre a saída do modelo de linguagem e a intenção do usuário, se a saída for adequada, a recompensa é positiva, caso contrário, devolve-se uma penalidade (CHRISTIANO *et al.*, 2023).

Teve 1 milhão de usuários em cincos dias e mais de 100 milhões de usuários em março de 2023, tendo o crescimento mais rápido da história dos aplicativos (GARFINKLE, 2024).

Agentes de conversão como o ChatGPT enfrenta algumas limitações, Caseli e Nunes (2024) diz que são poucas as línguas que tais modelos conseguem lidar, se compararmos com a quantidade de línguas que temos no mundo, e que existe um viés social negativo embutido em tais modelos.

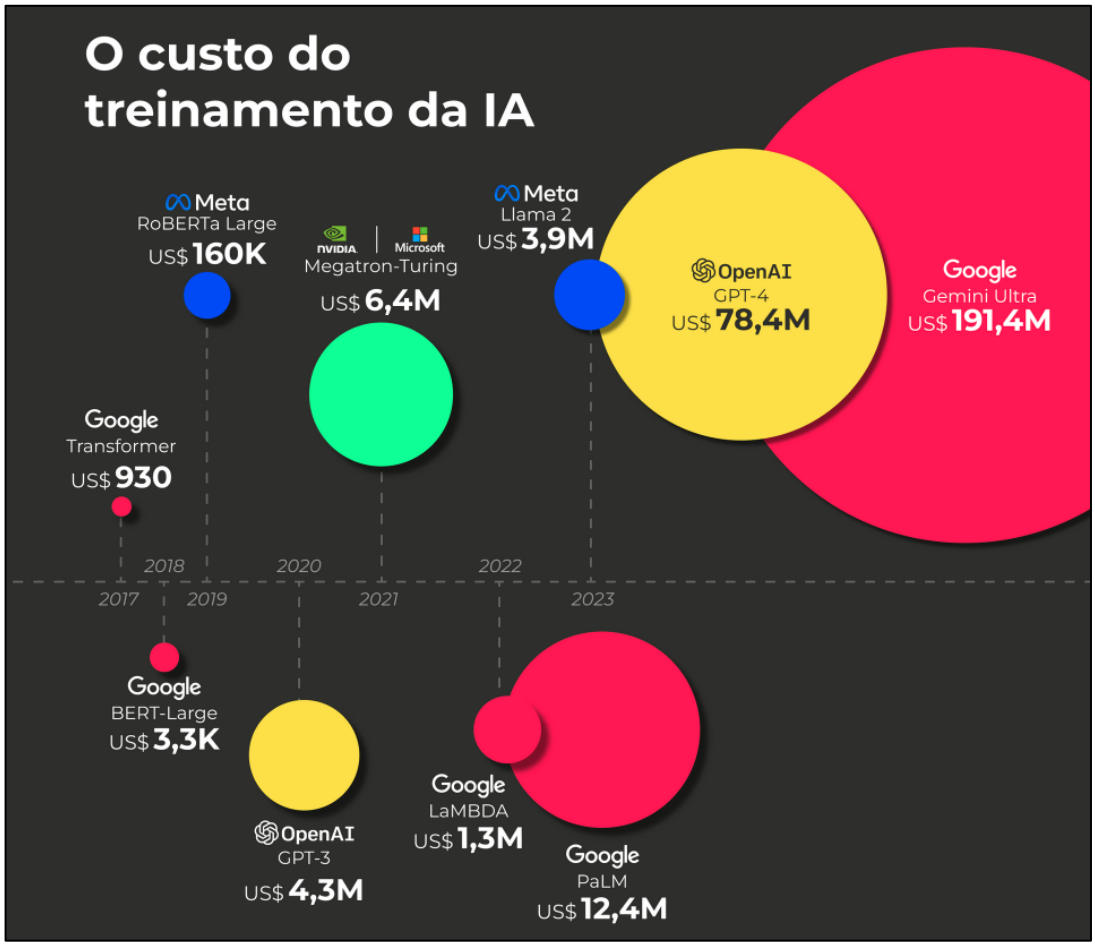
4.5 Limitações

O GPT enfrenta os mesmo desafios que os *Transformers*, Dantas (2021) destaca dois desafios; o consumo crescente de recursos de memória e a complexidade computacional quadrática (com relação ao tamanho da sequência).

Modelos cada vez maiores em busca de desempenho resulta em um crescimento vertiginoso de parâmetros nos *Transformers*, conforme mostrado no crescimento de parâmetros do GPT-n. Os modelos baseados na arquitetura são cada vez mais difíceis de construir e executar devido à escassez de recursos computacionais, pelo custo de treinamento desses modelos e por saber o desempenho de fato do modelo somente após o treinamento. Segundo o *AI Index Report* 2024 (MASLEJ *et al.*, 2024), as empresas de IA não costumam abrir os custos envolvidos no treinamento de seus modelos, mas acredita-se que eles

estejam na casa dos milhões de dólares, o que tem dificultado as universidades na pesquisa com os grandes modelos de IA. O CEO da *OpenAI*, Sam Altman, disse que

Figura 17 – Custos dos treinamentos dos modelos em dólares americanos.



Fonte: Snaq, 2024.

o custo de treinamento do GPT-4 foi superior a 100 milhões de dólares. Na Figura 17 temos esses custos estimados a partir do *AI Index Report* 2024 (MASLEJ *et al.*, 2024).

A complexidade computacional é outro desafio, mecanismos de atenção representam componentes essenciais dos *Transformers*. Tanto blocos de codificador como de decodificador contêm mecanismos de atenção. No entanto, segundo Vaswani *et al.* (2017), o mecanismo de auto-atenção possui complexidade temporal de $O(n^2 \cdot d)$, onde n é o tamanho da sequência e d a dimensão de representação.

Tal complexidade surge do cálculo da pontuação de similaridade, mais especificamente do produto das matrizes de chaves e consultas. Portanto, quanto maior a entrada de um *Transformer*, mais computacionalmente exigente será o seu processo de treino e a inferência no momento de aplicação. Tal característica

impõem um obstáculo notável para o uso de *Transformers* para processamento de texto.

Já Dziri *et al.* (2023) analisa as limitações dos *Transformers* com treinamento de perguntas e respostas; a qual desempenho dos modelos *Transformers* pode ter um limite atribuído à falta de dados específicos da tarefa durante o pré-treinamento. Mesmo após o ajuste fino exaustivo com pares de perguntas e respostas, os modelos ainda têm dificuldade em generalizar para casos fora do domínio. A capacidade de resolução sistemática de problemas não emerge apenas do treinamento em dados específicos da tarefa. Também que os modelos ainda não aprendem as operações componentes de maneira geral.

A característica autorregressiva dos *Transformers*, que os faz abordar problemas sequencialmente, apresenta um desafio fundamental que não pode ser resolvido apenas instruindo o modelo a gerar soluções passo a passo. Dziri *et al.* (2023) testou-se que o treinamento além do ponto de *overfitting* levaria a uma melhor generalização (fenômeno conhecido como *grokking*). No entanto, mesmo após treinamento extensivo, não houve melhoria significativa na generalização para casos fora do domínio. A dificuldade da tarefa pode ser um fator que impede a aprendizagem de uma representação bem estruturada.

5 CONSIDERAÇÕES FINAIS

Os *Transformers* e seu mecanismo de atenção impactou o PLN, neste trabalho descrevemos o seu conceito e funcionamento. Que diante de outros LLMs, o GPT-n demonstrou o grande potencial da arquitetura *Transformers*. Foi descrito os conceitos de IA generativa do GPT-n que é capaz de criar textos, áudios e imagens por meio de autorregressão usando o decodificador da arquitetura *Transformers* para gerar a probabilidade da próxima palavra. O pré-treinamento GPT-n possibilitou que o modelo fosse capaz de realizar diversas tarefas através do uso de grandes *Corpora*, treino supervisionado direto, aprendizado contínuo, ajustes finos e RLHF, o que em consequência eleva os custos de treinamento. A geração da probabilidade da próxima palavra considerando as palavras anteriores resultou em diversas aplicações nos campos de PLN com bons resultados. A partir dos aprimoramentos do GPT-n, temos a aplicação dele ao *ChatGPT*, um produto de sucesso com milhares de usuários e melhorias constantes.

No entanto, *Transformers* e consequentemente o GPT-n possuem limitações como: a complexidade computacional, número elevado de parâmetros, questões éticas, dificuldades com línguas não inglesas, altos custos e limites no pré-treinamento que representam desafios significantes para a arquitetura e modelo. Ainda assim, o progresso dos *Transformers* não parece ter chegado ao seu limite, novas propostas continuam a surgir, enquanto outras antigas são reinventadas.

Diante de tantas novidades e conceitos, a pesquisa atual apresenta os avanços nos modelos de linguagem e ferramentas que se tornaram populares, evidenciando os “limites que continuam a ser desafiados e a responsabilidade ao se usar e aplicar modelos cujas saídas ainda fogem da nossa compreensão” (CASELI; NUNES, 2024).

Para pesquisas posteriores, o aprofundamento nas descrições dos conceitos de agentes de conversação como o ChatGPT se faz necessário para o ampliamento do conhecimento, além de comparar outras instâncias da arquitetura *Transformers* com o GPT-n, e por fim pesquisar alternativas e soluções para as limitações.

REFERÊNCIAS

AGIRRE, E. Cross-Lingual Word Embeddings. **Computational Linguistics**, v. 46, n. 1, p. 245–248, mar. 2020.

AMATRIAIN, Xavier *et al.* Transformer models: an introduction and catalog. **Arxiv**, [S.L.], v. 4, n. 230207730, p. 1-67, 2023. DOI: http://dx.doi.org/10.48550/ARXIV.2302.07730. Disponível em: https://arxiv.org/abs/2302.07730v4. Acesso em: 06 maio 2024.

BARATTO, Gabriel Junges. **COMPARAÇÃO ENTRE OS MODELOS PRÉ-TREINADOS GPT-3 E BERT NA ESTIMATIVA DE ESFORÇO DE SOFTWARE POR ANALOGIA A PARTIR DE REQUISITOS TEXTUAIS**. 2022. 96 f. TCC (Graduação) - Curso de Engenharia da Computação, Universidade Tecnológica Federal do Paraná, Pato Branco, 2022. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/30619/1/comparacaogptbertsoftware.pdf. Acesso em: 10 maio 2024.

BAHDANAU, D.; CHO, K.; BENGIO, Y. **Neural Machine Translation by Jointly Learning to Align and Translate**. (Y. Bengio, Y. LeCun, Eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Anais...San Diego, California.: 2015. Disponível em: http://arxiv.org/abs/1409.0473. Acesso em 01 maio 2024.

BENGIO, Yoshua; COURVILLE, A.; VINCENT, P. Representation Learning: a review and new perspectives. **IEEE Transactions On Pattern Analysis And Machine Intelligence**, [S.L.], v. 35, n. 8, p. 1798-1828, ago. 2013. DOI: http://dx.doi.org/10.1109/tpami.2013.50. Disponível em: https://ieeexplore.ieee.org/document/6472238. Acesso em: 09 mar. 2023.

BERWICK, R. C.; CHOMSKY, N. **Por que apenas nós? Linguagem e evolução**. [S.L.] SciELO-Editora UNESP, 2017.

BISHOP, Christopher M.. **Pattern Recognition and Machine Learning**. New York: Springer New York, 2006.

BOLEDA, Gemma. Distributional Semantics and Linguistic Theory. **Annual Review Of Linguistics**, [S.L.], v. 6, n. 1, p. 213-234, 14 jan. 2020. Annual Reviews. DOI: http://dx.doi.org/10.1146/annurev-linguistics-011619-030303. Disponível em: https://www.annualreviews.org/content/journals/10.1146/annurev-linguistics-011619-030303. Acesso em: 30 abr. 2024.

BRANDES, N. *et al*. ProteinBERT: a universal deep-learning model of protein sequence and function. **Bioinform**, v. 38, n. 8, p. 2102–2110, 2022.

BROCKMAN, Greg; SUTSKEVER, Ilya. **Introducing OpenAI**. Disponível em: https://openai.com/blog/openai-lp. Acesso em: 01 abr. 2024.

BROWN, Tom B. *et al*. Language Models are Few-Shot Learners. **Arxiv**, [S.L.], v. 4, n. 200514165, p. 1-75, jun. 2020. DOI:

http://dx.doi.org/10.48550/ARXIV.2005.14165. Disponível em: https://arxiv.org/abs/2005.14165. Acesso em: 02 fev. 2024.

CASELI, H.M.; Nunes, M.G.V. (org.) **Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português**. 2 ed. BPLN, 2024. Disponível em: https://brasileiraspln.com/livro-pln/2a-edicao. Acesso em: 10 mar. 2024.

CHEN, Yi *et al*. Exploring the Use of Large Language Models for Reference-Free Text Quality Evaluation: an empirical study. **Arxiv**, [S.L.], v. 3, n. 230400723, p. 1-46, 2023. DOI: http://dx.doi.org/10.48550/ARXIV.2304.00723. Disponível em: https://arxiv.org/pdf/2307.06435. Acesso em: 06 maio 2024.

CHILD, Rewon *et al*. Generating Long Sequences with Sparse Transformers. **Arxiv**, [S.L.], v. 1, n. 190410509, p. 1-10, abr. 2019. DOI: http://dx.doi.org/10.48550/ARXIV.1904.10509. Disponível em: https://arxiv.org/abs/1904.10509. Acesso em: 17 abr. 2024.

CHO, Kyunghyun *et al*. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. **Proceedings Of The 2014 Conference On Empirical Methods In Natural Language Processing (EMNLP)**, [S.L.], v. 3, n. 1, p. 1724-1734, 2014. DOI: http://dx.doi.org/10.3115/v1/d14-1179. Disponível em: https://arxiv.org/abs/1406.1078v3. Acesso em: 17 abr. 2024.

CHRISTIANO, Paul *et al*. Deep reinforcement learning from human preferences. **Arxiv**, [S.L.], v. 4, n. 170603741, p. 1-17, fev. 2023. DOI: http://dx.doi.org/10.48550/ARXIV.1706.03741. Disponível em: https://arxiv.org/abs/1706.03741. Acesso em: 18 maio 2024.

COLLOBERT, Ronan; WESTON, Jason. A unified architecture for natural language processing. **Proceedings Of The 25Th International Conference On Machine Learning**, [S.L.], v. 5, n. 19, p. 160-167, jul. 2008. DOI: http://dx.doi.org/10.1145/1390156.1390177. Disponível em: https://dl.acm.org/doi/10.1145/1390156.1390177#sec-cit. Acesso em: 03 abr. 2024.

DANTAS, Leonardo Santana. **Transformers**: teoria e viabilização. 2021. 57 f. TCC (Graduação) - Curso de Matemática Aplicada, Escola de Matemática Aplicada, Fundação Getulio Vargas, Rio de Janeiro, 2021. Disponível em: https://repositorio.fgv.br/items/4e2ef1af-2425-46cd-be66-02776ba6ddb9. Acesso em: 12 maio 2024.

DEVLIN, J. *et al*. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. (J. Burstein, C. Doran, T. Solorio, Eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019. Anais... Minneapolis, MN, USA: Association for Computational Linguistics, 2019. Disponível em: https://doi.org/10.18653/v1/n19-1423. Acesso em 09 mar. 2024.

DONG, Qingxiu *et al*. A Survey on In-context Learning. **Arxiv**, [S.L.], v. 3, n. 230100234, p. 0-21, jun. 2023. DOI: http://dx.doi.org/10.48550/ARXIV.2301.00234. Disponível em: https://arxiv.org/abs/2301.00234. Acesso em: 04 mar. 2024.

DZIRI, Nouha et al. Faith and Fate: limits of transformers on compositionality. **Arxiv**, [S.L.], v. 3, n. 230518654, p. 1-40, out. 2023. DOI: http://dx.doi.org/10.48550/ARXIV.2305.18654. Disponível em: https://arxiv.org/abs/2305.18654. Acesso em: 10 maio 2024.

FLEURET, François. **The Little Book of Deep Learning**. Geneva: University Of Geneva, 2023.

GAO, Tianyu; YAO, Xingcheng; CHEN, Danqi. SimCSE: simple contrastive learning of sentence embeddings. **Proceedings Of The 2021 Conference On Empirical Methods In Natural Language Processing**, [S.L.], p. 6894-6910, nov. 2021. DOI: http://dx.doi.org/10.18653/v1/2021.emnlp-main.552. Disponível em: https://aclanthology.org/2021.emnlp-main.552/. Acesso em: 22 maio 2024.

GARFINKLE, Alexandra. **ChatGPT on track to surpass 100 million users faster than TikTok or Instagram: UBS**. Disponível em: https://finance.yahoo.com/news/chatgpt-on-track-to-surpass-100-million-users-faster-than-tiktok-or-instagram-ubs-214423357.html. Acesso em: 19 maio 2024.

GEVA, M.; GUPTA, A.; BERANT, J**. Injecting Numerical Reasoning Skills into Language Models**. (D. Jurafsky *et al.*, Eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. Anais. Association for Computational Linguistics, 2020. Disponível em: https://doi.org/10.18653/v1/2020.acl-main.89. Acesso em 06 mar. 2024.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.L.]: MIT Press, 2016.

GUO, Taicheng *et al*. What can Large Language Models do in chemistry? A comprehensive benchmark on eight tasks. **Arxiv**, [S.L.], v. 3, n. 1, abr. 2023. DOI: https://doi.org/10.48550/arXiv.2305.18365. Disponível em: https://arxiv.org/abs/2305.18365. Acesso em: 03 mar. 2024.

HARRIS, Z. S. Distributional Structure. **Word**, v. 10, n. 2-3, p. 146–162, 1954.

HE, K. *et al*. **Deep Residual Learning for Image Recognition**. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas,

NV, USA, June 27-30, 2016. Anais... IEEE Computer Society, 2016. Disponível em: https://doi.org/10.1109/CVPR.2016.90. Acesso em: 01 maio 2024.

HINTON, G. E.; SALAKHUTDINOV, R. R.. Reducing the Dimensionality of Data with Neural Networks. **Science**, [S.L.], v. 313, n. 5786, p. 504-507, jul. 2006. DOI: http://dx.doi.org/10.1126/science.1127647. Disponível em: https://www.science.org/doi/10.1126/science.1127647. Acesso em: 10 abr. 2023.

HOCHREITER, S. Untersuchungen zu dynamischen neuronalen Netzen. **Diploma, Technische Universität München**, v. 91, n. 1, p. 31, 1991.

HORNIK, K.; STINCHCOMBE, M. B.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989.

HOWARD, Jeremy; RUDER, Sebastian. Universal Language Model Fine-tuning for Text Classification. **Proceedings Of The 56Th Annual Meeting Of The Association For Computational Linguistics: Long Papers**, [S.L.], v. 1, n. 18-1031, p. 328-339, jun. 2018. DP:. http://dx.doi.org/10.18653/v1/p18-1031. Disponível em: https://aclanthology.org/P18-1031/. Acesso em: 16 maio 2024.

JENSEN Huang NVidia CEO NTU Commencement Speech. Taiwan: National Taiwan University, 2023. Son., color. Legendado. Disponível em: https://youtu.be/__Ewkal7s3g. Acesso em: 05 mar. 2024.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. [S.L.]: Stanford University, 2024.

KAPLAN, Andreas; HAENLEIN, Michael. A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. **California Management Review**, [S.L.], v. 61, n. 4, p. 5-14, jul. 2019. DOI: https://doi.org/10.1177/0008125619864925. Disponível em: https://journals.sagepub.com/doi/abs/10.1177/0008125619864925. Acesso em: 20 abr. 2023.

KENNEDY, Graeme. **An Introduction to Corpus Linguistics**. London: Routledge, 2014.

LECUN, Y. *et al*. Gradient-based learning applied to document recognition. **Proceedings Of The IEEE**, [S.L.], v. 86, n. 11, p. 2278-2324, nov. 1998. DOI: http://dx.doi.org/10.1109/5.726791. Disponível em: https://ieeexplore.ieee.org/document/726791. Acesso em: 10 abr. 2023.

LI, R. *et al.* StarCoder: may the source be with you! **CoRR**, v. abs/2305.06161, 2023.

LUONG, T.; PHAM, H.; MANNING, C. D. **Effective Approaches to Attention based Neural Machine Translation**. (L. Màrquez *et al*., Eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. Anais... The Association for Computational Linguistics, 2015. Disponível em: https://doi.org/10.18653/v1/d15-1166. Acesso em: 01 maio 2024.

MCCANN, B. *et al*. **Learned in Translation: Contextualized Word Vectors**. Proceedings of the 31st International Conference on Neural Information Processing Systems. Anais... NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017.

MIIKKULAINEN, R.; DYER, M. G. Natural Language Processing With Modular Pdp Networks and Distributed Lexicon. **Cognitive Science**, v. 15, n. 3, p. 343–399, 1991.

NG, Andrew. **Machine learning yearning**. [S.L.]: Deeplearning.Ai, 2018. Disponível em: https://info.deeplearning.ai/machine-learning-yearning-book. Aceso em: 25 mar. 2023

NIJKAMP, E. *et al.* ProGen2: Exploring the Boundaries of Protein Language Models. **CoRR**, v. abs/2206.13517, 2022.

NÚCLEO INTERINSTITUCIONAL DE LINGUÍSTICA COMPUTACIONAL. **Repositório de Word Embeddings do NILC**. Disponível em: http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc. Acesso em: 30 abr. 2024.

OPENAI (San Francisco) (org.). **GPT-4 Turbo in the OpenAI API**. 2024. Disponível em: https://openai.com/index/hello-gpt-4o/. Acesso em: 15 maio 2024a.

OPENAI (San Francisco) (org.). **Hello GPT-4o**. 2024. Disponível em: https://openai.com/index/hello-gpt-4o/. Acesso em: 20 maio 2024b.

OPENAI (San Francisco) (org.). **GPT-4 is OpenAI's most advanced system, producing safer and more useful responses**. 2023. Disponível em: https://openai.com/gpt-4. Acesso em: 03 mar. 2024c.

OPENAI *et al*. GPT-4 Technical Report. **Arxiv**, [S.L.], v. 6, n. 230308774, p. 1-100, 2023. DOI: http://dx.doi.org/10.48550/ARXIV.2303.08774. Disponível em: https://arxiv.org/abs/2303.08774. Acesso em: 15 maio 2024.

PRINCE, Simon J.D. **Understanding Deep Learning**. The Mit Press, 2023. Disponível em: http://udlbook.com. Acesso em: 15 mar. 2024.

RADFORD, Alec *et al.* **Improving Language Understanding by Generative Pre-Training**. 2018. Disponível em https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. Acesso em: 28 abr. 2024.

RADFORD, Alec *et al*. **Language Models are Unsupervised Multitask Learners**. 2019. Disponível em: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Acesso em: 09 abr. 2023.

REIMERS, Nils; GUREVYCH, Iryna. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. **Proceedings Of The 2020 Conference On Empirical Methods In Natural Language Processing (Emnlp)**, [S.L.], v. 1, n. 1, p. 4512-4525, nov. 2020. DOI. http://dx.doi.org/10.18653/v1/2020.emnlp-main.365. Disponível em: 10.18653/v1/2020.emnlp-main.365. Acesso em: 25 mar. 2024.

ROMERA-PAREDES, Bernardino; TORR, Philip H. S.. Recurrent Instance Segmentation. **Arxiv**, [S.L.], v. 1, n. 151108250, p. 1-24, nov. 2015. DOI: http://dx.doi.org/10.48550/ARXIV.1511.08250. Disponível em: https://arxiv.org/abs/1511.08250. Acesso em: 05 mar. 2025.

RUSSELL, Stuart Jonathan; NORVIG, Peter. **Artificial Intelligence**: a modern approach. 4. ed. New Jersey: Pearson, 2020.

SCHMIDHUBER, J.; HEIL, S. Sequential neural text compression. **IEEE Transactions on Neural Networks**, v. 7, n. 1, p. 142–146, 1996.

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Simpósio Brasileiro Sobre Inteligência Artificial**. 1984. Disponível em: https://comissoes.sbc.org.br/ce-ia/pg/historico/base/SBIA-1984/Folheto_de_Divulgacao.png. Acesso em: 12 abr. 2024

SRIVASTAVA, Nitish *et al*. Dropout: a simple way to prevent neural networks from overfitting. **Journal Of Machine Learning Research**, Toronto, Ontario, v. 15, n. 56, jun. 2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html. Acesso em: 05 mar. 2024.

SUTTON, Richard S.; BARTO, Andrew G.. **Reinforcement Learning**: an introduction. Cambridge: MIT Press, 2018.

TURING, Alan Mathison. Computing Machinery and Intelligence. **Mind**, [S.L.], v. 1, n. 236, p. 433–460, out. 1950. DOI: https://doi.org/10.1093/mind/LIX.236.433. Disponível em: https://academic.oup.com/mind/article/LIX/236/433/986238. Acesso em: 20 mar. 2023.

VASWANI, Ashish *et al*. Attention Is All You Need. **Arxiv**, [S.L.], v. 7, n. 1, jun. 2017. DOI: https://doi.org/10.48550/arXiv.1706.03762. Disponível em: https://arxiv.org/abs/1706.03762. Acesso em: 08 mar. 2023.

YENDURI, Gokul *et al*. Generative Pre-trained Transformer: a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. **Arxiv**, [S.L.], v. 2, n. 1, abr. 2023. DOI: https://doi.org/10.48550/arXiv.2305.10435. Disponível em: https://arxiv.org/abs/2305.10435. Acesso em: 05 mar. 2024.

WANG, Y. *et al.* **CodeT5: Identifier-aware Unified Pre-trained Encoder Decoder Models for Code Understanding and Generation**. (M. F. Moens *et al.*, Eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11, Novembro, 2021. Association for Computational Linguistics, 2021. Disponível em: <https://doi.org/10.18653/v1/2021.emnlp-main.685> Acesso em: 06 mar. 2024.

WOLF, Thomas *et al*. HuggingFace's Transformers: state-of-the-art natural language processing. **Arxiv**, [S.L.], v. 5, n. 191003771, p. 1-8, 2019. DOI: http://dx.doi.org/10.48550/ARXIV.1910.03771. Disponível em: https://arxiv.org/abs/1910.03771. Acesso em: 04 maio 2024.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 2. ed. São Paulo: Elsevier Editora Ltda, 2014. 146 p.

XIONG, R. *et al*. **On Layer Normalization in the Transformer Architecture**. Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Anais... Proceedings of Machine Learning Research.PMLR, 2020. Disponível em: http://proceedings.mlr.press/v119/xiong20b.html. Acesso em: 01 maio 2024.