

UNIVERSIDADE DE SÃO PAULO  
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**Explorando o Potencial da Arquitetura  
Transformer na Evolução de Chatbots**

Luca Assumpção Dillenburg

MONOGRAFIA FINAL

MAC 499 — TRABALHO DE  
FORMATURA SUPERVISIONADO

Supervisora: Prof<sup>a</sup>. Nina Sumiko Tomita Hirata

São Paulo  
2023

*O conteúdo deste trabalho é publicado sob a licença CC BY 4.0  
(Creative Commons Attribution 4.0 International License)*

# Agradecimentos

*Se eu encontrar em mim mesmo um desejo que não experimento neste mundo que posso satisfazer, a explicação mais provável é que fui feito para outro mundo.*

— C.S. Lewis

Em primeiro lugar, agradeço a minha família por me apoiar e incentivar meus estudos e curiosidades desde muito cedo. Agradeço aos meus amigos, que muito me ensinaram durante os anos de graduação. Agradeço também aos professores que me ensinaram muito e, por último, mas não menos importante, a minha orientadora por me guiar durante o desenvolvimento deste trabalho.



# Resumo

Luca Assumpção Dillenbug. **Explorando o Potencial da Arquitetura Transformer na Evolução de Chatbots**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

É evidente que o campo de processamento de linguagem natural, em especial *chatbots*, experimentou um grande avanço tecnológico e um aumento de interesse do público nos últimos anos. Nesse contexto, este trabalho explora as diferentes abordagens na construção de *chatbots* utilizadas ao longo da história; aborda em detalhes os componentes fundamentais da arquitetura *transformer*; além de desenvolver e comparar duas implementações de *chatbots*: a clássica e a arquitetura estudada. Considerando um leitor sem conhecimento prévio de inteligência artificial e redes neurais, a explicação é feita de forma didática e por etapas, construindo o conhecimento desde redes neurais *feedforward*; redes neurais recorrentes; *embedding* de palavras; mecanismos de atenção; até chegar na arquitetura *transformer*. Os leitores podem esperar uma explicação aprofundada sobre a estrutura e funcionamento de cada componente dessa arquitetura, a qual é fundamental em diversas ferramentas de inteligência artificial generativa, como o ChatGPT. Em seguida, o trabalho descreve o desenvolvimento de *chatbots* especializados, programados para responder perguntas sobre um certo domínio de conhecimento baseado nos dados de treino. Por fim, as diferentes implementações do *chatbot* serão avaliadas, identificando os prós e contras de cada abordagem e discutindo os cenários ideias para cada implementação.

**Palavras-chave:** Chatbots. Processamento de Linguagem Natural. Redes Neurais. Aprendizado de Máquina. Arquitetura Transformer. ChatGPT.



# Abstract

Luca Assumpção Dillenburg. **Exploring the Potential of Large Language Models in the Evolution of Chatbots: A Study of the Transformers Architecture.** Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

It's clear that the field of natural language processing, especially with chatbots, has undergone significant technological advancements and has garnered increased public interest in recent years. In this context, this work delves into various approaches used throughout history in constructing chatbots, it provides detailed insights into the fundamental components of the transformer architecture, it also develops and compares two chatbot implementations: the classical one and the mentioned architecture. Catering to readers without prior knowledge of artificial intelligence and neural networks, the explanation is presented in a step-by-step, didactic manner, building knowledge from feedforward neural networks, recurrent neural networks, word embeddings, attention mechanisms, all the way to the transformer architecture. Readers can expect a thorough explanation of the structure and functionality of each component of this architecture, which is pivotal in various generative artificial intelligence tools, including ChatGPT. Following this, the paper describes the development of specialized chatbots programmed to respond to questions within a specific knowledge domain based on training data. Finally, it evaluates the different chatbot implementations, identifying the pros and cons of each approach and discussing the ideal scenarios for their application.

**Keywords:** Chatbots. Natural Language Processing. Neural Networks. Machine Learning. Transformer Architecture. ChatGPT.



# Lista de abreviaturas

RN	Rede Neural ( <i>Neural Network</i> )
RNN	Rede Neural Recorrente ( <i>Recurrent Neural Network</i> )
NLP	Processamento de Linguagem Natural ( <i>Natural Language Processing</i> )
BPTT	Retropropagação através do tempo ( <i>Backpropagation Through Time</i> )
LSTM	Long Short-Term Memory ( <i>Long Short-Term Memory</i> )
LLM	Large Language Models ( <i>Large Language Models</i> )
LoRA	Low-Rank Adaptation of Large Language Models ( <i>Low-Rank Adaptation of Large Language Models</i> )
API	Interface de Programação de Aplicações ( <i>Application Programming Interface</i> )
ABNT	Associação Brasileira de Normas Técnicas
URL	Localizador Uniforme de Recursos ( <i>Uniform Resource Locator</i> )
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

## Lista de figuras

1.1	Exemplo de conversa com a ELIZA. . . . .	4
1.2	Exemplo de conversa com a ALICE. . . . .	5
1.3	Estrutura de dados usada no <i>Bag-of-Words</i> . . . . .	8
1.4	Exemplo de Árvore de Decisão. . . . .	8
1.5	Exemplo de Rede Neural Recorrente. . . . .	9
2.1	A fórmula para este neurônio seria $y = \theta(w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + b)$ , sendo $\theta$ a função de ativação. . . . .	14
2.2	As funções de ativação mais comuns: Sigmoid, Tangente Hiperbólica e Unidade Linear Retificada. . . . .	15
2.3	Exemplo de rede neural com diversas camadas ocultas. . . . .	16
2.4	Algoritmo gradient descent. . . . .	18
2.5	Exemplo de um caminho do algoritmo gradient descent. . . . .	18
2.6	Exemplo de Rede Neural Recorrente. . . . .	20
2.7	As duas formas de enxergar Redes Neurais Recorrentes. . . . .	20
2.8	Os diferentes configurações de Redes Neurais Recorrentes. . . . .	22
2.9	Os componentes da LSMT. . . . .	23
2.10	Exemplo de um <i>Word Embedding</i> em um espaço 2D. . . . .	25
2.11	Treinamento do <i>Word2Vec</i> de acordo com o <i>Skip-gram</i> . . . . .	27
2.12	Arquitetura <i>Encoder-Decoder</i> utilizada em <i>chatbots</i> . . . . .	28
2.13	Treinamento do <i>decoder</i> com <i>Teacher Forcing</i> . . . . .	30
2.14	Exemplo de modelo Seq2Seq com attention. . . . .	32
2.15	Arquitetura do modelo Bahdanau Attention. . . . .	33
2.16	Organização básica da arquitetura do modelo Transformer. . . . .	36
2.17	Exemplo dos percentuais de atenção de "it" para as outras palavras da frase. Cores mais escuras indicam maior percentual de atenção. . . . .	38
2.18	Representação das diferentes interpretações no Multi-Headed Attention. . . . .	40
2.19	Processamento completo de atenção multi-headed. . . . .	41
2.20	Arquitetura Transformer simplificada. . . . .	41

2.21	Arquitetura Transformer. . . . .	43
2.22	Exemplo do processo de predição da Arquitetura Transformer. . . . .	44



# Sumário

<b>Introdução</b>	<b>1</b>
A importância de chatbots	1
Objetivo do Trabalho	2
<b>1 Histórico e evolução dos chatbots</b>	<b>3</b>
Os primeiros chatbots	3
A evolução dos chatbots no final do século XX	4
As principais abordagens pré transformers	6
Assistentes Virtuais	6
Chatbots	7
Implementações	7
A revolução da Arquitetura Transformer	9
Surgimento da arquitetura Transformers	10
Desenvolvimento de chatbots com a arquitetura Transformers	11
<b>2 Fundamentação Teórica: Chatbots com Arquitetura Transformer</b>	<b>13</b>
Redes Neurais	13
Neurônio	14
Camadas na Rede Neural	15
Treinamento	16
Redes Neurais Recorrentes	19
Funcionamento	19
As diferentes configurações	21
LSTMs	23
Como computadores interpretam as palavras?	24
Arquitetura <i>Encoder-Decoder</i>	27
Funcionamento	28
Treinamento com <i>Teacher Forcing</i>	29

Seq2Seq . . . . .	31
Mecanismos de Atenção . . . . .	31
Transformers . . . . .	34
Avanços Exponenciais . . . . .	35
Organização Básica . . . . .	35
Self-Attention . . . . .	37
Positional Encoding . . . . .	39
Multi-Headed Attention . . . . .	39
Arquitetura Completa . . . . .	41
Exemplo de Processamento . . . . .	43
Paralelismo . . . . .	44
Chatbots com Transformers . . . . .	44
Large Language Models . . . . .	45
Fine Tuning . . . . .	45
Desafios Futuros . . . . .	46
<b>3 Desenvolvimento de Chatbots</b>	<b>47</b>
Metodologia . . . . .	47
Coleta de Dados . . . . .	48
Implementações . . . . .	49
Intent Chat . . . . .	49
Fine Tuning de LLM . . . . .	50
Assistent API . . . . .	51
Resultados . . . . .	51
Intent Chat . . . . .	52
Fine Tuning de LLM . . . . .	55
Assistent API . . . . .	63
Análise dos Cenários . . . . .	68
Avaliação . . . . .	70
Escolha da Implementação . . . . .	71
<b>4 Conclusão</b>	<b>75</b>
<b>Referências</b>	<b>77</b>

# Introdução

O processamento de linguagem natural (NLP) é uma das áreas mais promissoras da inteligência artificial da última década e, pelo que tudo indica, terá um papel extremamente importante na década seguinte. Diversos empreendimentos a consideram fundamental para suas operações e grande parte da população mundial a utiliza com frequência no cotidiano.

NLP é um vasto campo de estudo. Neste trabalho, será estudada uma tecnologia disruptiva que transformou o mercado nos últimos anos. Tecnologias disruptivas são inovações que provocam mudanças expressivas em mercados, setores industriais ou na sociedade como um todo. Essas inovações têm o potencial de alterar radicalmente a maneira como as pessoas vivem, trabalham e se relacionam, impactando empresas estabelecidas, modelos de negócios tradicionais e até mesmo as normas culturais.

A tecnologia em questão é a Arquitetura *Transformer*, em especial aplicada a *chatbots*. Essa nova arquitetura foi fundamental no desenvolvimento do ChatGPT e a popularidade dela tem crescido exponencialmente tanto no contexto profissional quanto pessoal. Como será discutido posteriormente, o novo modelo representa um avanço tecnológico drástico, acompanhado de uma melhoria expressiva na experiência dos usuários e na capacidade dos *chatbots* que o utilizam. O sistema pode realizar tarefas muito complexas, manter uma conversa por mais tempo e conversar de forma mais fluida.

*Chatbot* é um programa de computador projetado para interagir com seres humanos por meio de conversas, seja por texto ou voz. Ele utiliza algoritmos de inteligência artificial para processar e interpretar a linguagem natural, respondendo a mensagem de forma clara e com informações pertinentes. O objetivo é simular uma conversa com um ser humano, respondendo perguntas e realizando tarefas de forma eficiente e natural.

## A importância de chatbots

Não é difícil encontrar vários exemplos de *chatbots* tanto no cotidiano quanto no uso empresarial. Eles são frequentemente usados em atendimento ao cliente, suporte técnico, correção automática de texto, ferramentas de busca, sugestão de textos, professor particular, assistente virtual, entre outros. Processos que antes demandavam muitas horas e milhões de reais anualmente, agora podem ser automatizados com essas ferramentas, reduzindo o custo das empresas e reduzindo o tempo de espera dos usuários.

Hoje pelo menos 18% das pessoas do mundo usam chatbots pelo menos uma vez no mês (MAHAJAN, 2023). O expressivo aumento no uso de *chatbots* deve ser, em grande

medida, creditado a OpenAI. A empresa, fundada em 2015, é conhecida por sua missão de avançar a inteligência artificial de forma a beneficiar toda a humanidade. Em novembro de 2022, a empresa lançou o ChatGPT, um *chatbot* de inteligência artificial generativa baseada na arquitetura *transformer*. O site alcançou 1 milhão de usuários em apenas 5 dias. Após apenas dois meses de lançamento, o número de usuários ativos mensais do ChatGPT ultrapassou 100 milhões, se tornando a aplicação com crescimento mais rápido da história, superando o Instagram, Spotify e o YouTube (MAHAJAN, 2023).

Olhando para o futuro, essa tecnologia ganha destaque como uma resposta às demandas por facilidade e praticidade. Na era da conveniência, existe uma disposição crescente de buscar interfaces mais naturais e intuitivas. Diante disso, a comunicação através de linguagem natural, seja por meio de voz ou texto, representa uma das formas mais intuitivas de interação e, dessa forma, os *chatbots* emergem como protagonistas no horizonte futuro de transformação digital.

## Objetivo do Trabalho

Diante desse cenário de rápida evolução e adoção dos *chatbots*, o presente trabalho tem como objetivo compreender diferentes implementações de *chatbots* e estudar a fundo a estrutura e funcionamento de cada componente da Arquitetura *Transformer*, a qual é utilizada em *chatbots* com modelos de linguagem generativos.

Além da análise histórica da evolução das abordagens no desenvolvimento de *chatbots* durante os anos, a arquitetura em questão e seus pré-requisitos serão estudada e apresentada por etapas. Construindo o conhecimento desde redes neurais *feedforward*; redes neurais recorrentes; *embedding* de palavras; mecanismos de atenção; até chegar na arquitetura *transformer* e seus componentes.

Em seguida, o trabalho descreverá o desenvolvimento de *chatbots* especializados com duas abordagens: uma implementação clássica e outra fundamentada na arquitetura estudada. Diferentemente do ChatGPT, que é generalista, os *chatbots* especializados são programados para responder perguntas sobre um certo domínio de conhecimento baseado nos dados de treino. Por fim, as diferentes implementações do *chatbot* serão avaliadas, identificando os prós e contras de cada abordagem e discutindo os cenários ideias para cada implementação.

# Capítulo 1

## Histórico e evolução dos chatbots

Ao longo dos anos, diversas implementações de *chatbots* foram concebidas. Este capítulo tem como objetivo analisar algumas dessas abordagens, compreendendo o seu funcionamento e os avanços introduzidos a cada descoberta, culminando nos modelos mais utilizados na atualidade.

### Os primeiros chatbots

Reconhecida por muitos como o primeira chatbot, apesar de ainda muito rudimentar, a ELIZA (WEIZENBAUM, 1966) foi criada em 1966 por Joseph Weizenbaum no Instituto de Tecnologia de Massachusetts (MIT). ELIZA era um programa de processamento de linguagem natural que simulava um psicoterapeuta em uma sessão de conversação.

Seu principal objetivo era responder a perguntas e estimular o diálogo com os usuários de maneira a criar a ilusão de um intercâmbio genuíno entre a máquina e o ser humano. O sistema era baseada em padrões de transformação, onde as entradas dos usuários eram analisadas e respostas eram geradas a partir de regras de "regex" (*Regular Expression*) pré-definidas. Isso permitia ao programa responder de maneira aparentemente compreensiva e reflexiva, muitas vezes reorientando as perguntas dos usuários para manter a interação.

O programa foi escrito na linguagem MAD-SLIP, que foi projetada para manipular listas de palavras e funcionava essencialmente com uma série de condicionais ("*ifs*") e expressões regulares ("*regex*") para converter frases da língua inglesa em respostas que continuassem a conversa. É evidente que o sistema não entendia as mensagens, pelo contrário ele usava a estrutura da linguagem para gerar uma resposta condizente mas sem novas informações.

```

Welcome to
          EEEEE LL   IIII ZZZZZZ  AAAAA
          EE   LL   II    ZZ   AA  AA
          EEEEE LL   II    ZZ   AAAAAA
          EE   LL   II    ZZ   AA  AA
          EEEEE LLLLL IIII ZZZZZZ  AA  AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Please tell me what's been bothering you.
YOU:   I don't really know
ELIZA: Do you wish to be able to really know ?
YOU:   Hmm.. well, I guess yes
ELIZA: Do you say you guess yes for some special reason ?
YOU:   I don't really know
ELIZA: Does that trouble you ?
YOU:   What do you think?
ELIZA: What are your feelings now ?
YOU:   Sad
ELIZA: Does talking about this bother you ?
YOU:   █

```

**Figura 1.1:** Exemplo de conversa com a ELIZA.

Embora não possuísse uma compreensão verdadeira das mensagens dos usuários, ela cativou a atenção do público e da comunidade científica. Usuários interagiam com a ELIZA como se estivessem participando de sessões de terapia, e em alguns casos os participantes se confundiam, se perguntando se ela era de fato uma máquina ou uma pessoa.

ELIZA não foi concebida para efetivamente se passar como humano, porém o programa foi supostamente um dos primeiros a passar no Teste de Turing (TURING, 1950), que buscava responder a pergunta "as máquinas podem pensar?". A essência do Teste de Turing, proposto pelo matemático Allan Turing em 1950, envolve uma situação na qual um observador humano interage com uma máquina e outro ser humano sem saber qual é qual. Se o observador não consegue discernir, com base nas respostas às suas perguntas e interações, qual dos dois é a máquina e qual é o ser humano, a máquina é considerada como tendo passado no teste.

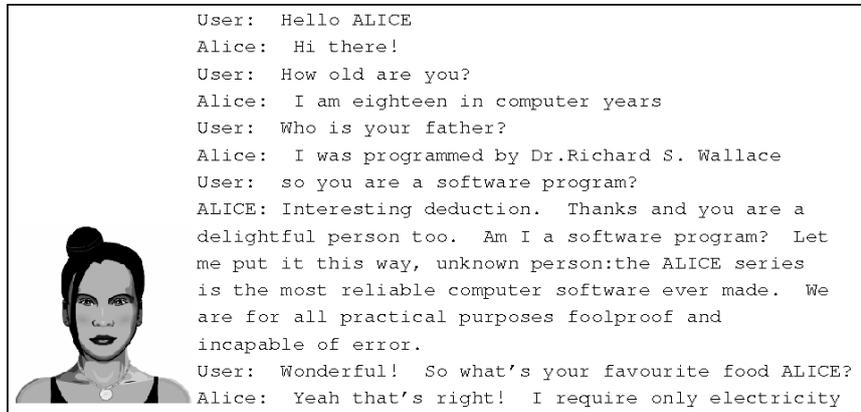
Apesar de seu algoritmo relativamente simples, o impacto da ELIZA vai além de sua funcionalidade técnica. Seu desenvolvimento contribuiu para o interesse e pesquisa contínuos em processamento de linguagem natural (NLP) e na criação de sistemas capazes de entender e gerar texto de maneira mais sofisticada. Além disso, o programa foi uma das primeiras demonstrações práticas de que era possível criar uma interação de linguagem natural entre humanos e máquinas, gerando assim um grande interesse público e acadêmico.

É inegável que a ELIZA, em sua simplicidade e inovação, pavimentou o caminho para a evolução dos *chatbots* para as tecnologias que existem hoje.

## A evolução dos chatbots no final do século XX

Durante as décadas seguintes, vários *chatbots* foram desenvolvidos. Embora muitos deles fossem simples scripts com *ifs* e respostas fixas, já surgiam alguns exemplos de sistemas um pouco mais sofisticados.

A ALICE, por exemplo, desenvolvida por Richard Wallace no final da década de 1990, explorou uma abordagem diferente. Sua base era um conjunto de regras gramaticais e padrões de linguagem, permitindo que o sistema compreendesse e respondesse de maneira mais flexível às consultas dos usuários (SHAWAR e ATWELL, 2002).



**Figura 1.2:** Exemplo de conversa com a ALICE.

A ALICE encontrou aplicações em uma variedade de cenários, desde o fornecimento de informações úteis até a simulação de interações sociais e de prestação de suporte em alguns setores.

Ela possuía a habilidade de compreender e gerar respostas em diversos idiomas e conseguia abordar uma gama de tópicos, desde informações mais gerais até questões altamente específicas.

Ainda mais, a ALICE destacava-se por sua capacidade única de simular uma personalidade, sendo uma camada adicional de envolvimento. Sua capacidade de manter o contexto da conversa pelo menos de uma forma mais primitiva também foi surpreendente na época. Essas características permitiam que o chat respondesse de maneira mais natural comparado as abordagens anteriores, permitindo que o chatbot recordasse informações fornecidas anteriormente durante a interação e mantendo uma conversa.

Por mais que a sua base de dados não fosse muito grande, respondendo que não conhecia certa informação em alguns casos, ela se aproximava mais da conversação humana, com maior fluidez.

Isso só foi possível porque a ALICE foi implementada com uma abordagem diferente da ELIZA. Enquanto a ELIZA utilizava uma abordagem baseada em padrões de transformação, a ALICE utilizava uma abordagem baseada em regras de correspondência de padrões (SHAWAR e ATWELL, 2002).

Os padrões de transformação referem-se a expressões ou modelos específicos que são utilizados para identificar e manipular determinados padrões de entrada. Eles são utilizados para entender a entrada do usuário e transformá-la em uma representação mais adequada que seja a resposta. Por outro lado, as regras de correspondência de padrões consistem em diretrizes que especificam como um sistema deve identificar padrões específicos em dados de entrada. Nesse caso, uma vez que um padrão é identificado, o sistema utiliza tanto os

dados da pergunta quanto a sua base de dados própria para gerar uma resposta contextual e apropriada.

O chatbot foi utilizado o AIML (*Artificial Intelligence Markup Language*) para representar o conhecimento e as regras de correspondência de padrões do chatbot. A linguagem foi desenvolvida para a criação de sistemas de processamento de linguagem natural e estruturada em categorias que contêm padrões de entrada e respostas associadas a esses padrões.

As categorias são organizadas de forma a representar o conhecimento do *chatbot*, com cada categoria contendo um padrão de entrada a ser reconhecido e uma resposta correspondente. A estrutura é altamente organizada, permitindo que novas categorias sejam facilmente adicionadas e que o conhecimento do chatbot seja expandido de maneira eficiente.

Embora tivesse uma base de dados pré-definida, essa nova abordagem representa um avanço significativo em relação à implementação original do ELIZA, já que, ao invés de puramente criar uma resposta reordenando as palavras da entrada do usuário, a ALICE era capaz de reconhecer padrões específicos na entrada do usuário e fornecer novas informações de acordo com a sua base de dados.

## As principais abordagens pré transformers

Nas primeiras décadas do século XXI, o uso de robôs de conversação aumentou muito. Com o surgimento de novas tecnologias e o avanço da inteligência artificial, eles se tornaram muito mais úteis. Deixaram de ser apenas uma área de estudo para se tornarem produtos no mercado de software alcançando milhões de pessoas.

### Assistentes Virtuais

Um dos primeiros produtos nessa área que foi distribuído em grande escala foi a *Siri*. Lançada em 2011 pela Apple junto ao iPhone 4S, a *Siri* foi a primeira assistente virtual com adoção em massa pelo público. Apenas na primeira semana, mais de 4 milhões de pessoas já tinham comprado o novo smartphone e eram introduzidos ao novo software (APPLE, 2011).

Desde o lançamento, o serviço era capacitado com uma variedade de recursos. Os usuários podiam fazer perguntas à *Siri* para obter informações, como condições meteorológicas, resultados esportivos, notícias e informações sobre locais. Entre outras funcionalidades, estavam o controle de música, navegação, gerenciamento de alarmes, calendário e lembretes. Ainda mais importante, a *Siri* foi projetada para aprender de acordo com o uso dos usuários, o que permitia que ela se adaptasse às preferências e necessidades de cada usuário, além de melhorar o seu funcionamento através da técnica de retro-alimentação. O resultado foi uma experiência mais natural, onde os usuários podiam fazer perguntas de uma maneira mais natural e solicitar informações relacionadas ao contexto da conversa.

O impacto da *Siri* foi notável, moldando a maneira como as pessoas interagem com seus dispositivos móveis e estabelecendo uma expectativa crescente para assistentes virtuais

em outros ecossistemas tecnológicos. A *Siri* influenciou diretamente o desenvolvimento de outros assistentes virtuais, contribuindo para a crescente ênfase em interações com linguagem natural entre humanos e computadores. Em 2014, a Amazon e Microsoft lançaram respectivamente a *Alexa* e a *Cortana*, e dois anos depois o Google anunciou o *Google Assistant*. Ao longo dos anos, esses assistentes virtuais receberam uma série de atualizações e melhorias, e hoje são ainda mais comuns no dia-a-dia de seus usuários.

Embora eles não sejam *chatbots* tradicionais (isto é, de texto), a tecnologia por trás desses serviços são muito semelhantes. Assistentes virtuais utilizam uma camada adicional de processamento de áudio. Essa camada é responsável por transformar áudio em texto, o qual é inserido no mesmo fluxo de processamento de um *chatbot* tradicional.

## Chatbots

Construindo sobre o sucesso dos assistentes virtuais, diversos setores começaram a implementar *chatbots* para automatizar processos e melhorar a experiência dos usuários.

No atendimento ao cliente, empresas de telecomunicações, bancos e varejistas desenvolveram *chatbots* para lidar com consultas comuns, oferecer suporte técnico e fornecer informações sobre produtos e serviços. Instituições educacionais utilizam *chatbots* para fornecer informações sobre cursos, orientar os alunos durante o processo de inscrição e até mesmo oferecer suporte na aprendizagem, respondendo a dúvidas frequentes.

Clínicas e hospitais adotam *chatbots* para agendar consultas, fornecer informações sobre medicamentos e orientar os pacientes sobre sintomas comuns. No varejo e e-commerce, diversas lojas online integram *chatbots* para auxiliar os clientes na seleção de produtos, fornecer informações sobre disponibilidade de estoque, processar pedidos e oferecer suporte pós-venda.

As aplicações são vastas. Muitas empresas já adotam e grande parte das demais poderiam se beneficiar significativamente ao incorporar *chatbots* em suas operações.

## Implementações

Dentre as aplicações previamente mencionadas, apesar da semelhança dos produtos, diversas abordagens podem ser implementadas. Esta seção oferece uma breve descrição sobre o funcionamento de cada um desses modelos, destacando suas vantagens e desvantagens. Essas informações não apenas evidenciam a progressão dos *chatbots*, mas também revelam a crescente complexidade.

**Modelo *Bag-of-Word*** - O modelo *Bag-of-Words* (BoW), em português "Saco de Palavras", é uma técnica simples e popular usada para representar textos em processamento de linguagem natural. Nessa abordagem, um documento é representado como um conjunto não ordenado de palavras, ignorando a estrutura gramatical e a ordem das palavras, mas mantendo a contagem de sua ocorrência. O primeiro passo envolve dividir o texto em unidades menores, chamadas de "*tokens*". Em seguida, é criada uma lista de frequência dos "*tokens*" com o *input*. A partir dessa estrutura, utiliza-se algoritmos de aprendizado de máquina, como SVM, Regressão Logística, Árvores de Decisão ou Redes Neurais, para

encontrar uma resposta adequada. Como esse modelo foca apenas na frequência das palavras, ignorando a ordem das palavras na frase, a estrutura gramatical não é aprendida, o que pode ser crucial em certos contextos linguísticos. Apesar disso, essa técnica pode ser usada para escolher a resposta mais adequada dentre o conjunto de dados de treinamento com perguntas e respostas.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Figura 1.3: Estrutura de dados usada no Bag-of-Words.

**Aprendizado de Máquina com Base em Regras** - O Aprendizado de Máquina com Base em Regras (*Rule-Based Machine Learning*) refere-se a um tipo de abordagem em que as decisões são tomadas com base em regras lógicas explicitamente definidas. Isso envolve a análise dos padrões nos dados para identificar relações lógicas e extrair regras que descrevem esses padrões. Nesse caso, ao receber um input utiliza-se algoritmos de aprendizado de máquina para caminhar pelas regras, muitas vezes representadas em uma árvore de decisão, e encontrar uma resposta adequada. Uma vantagem dessa abordagem é a interpretabilidade, isto é, as regras podem ser facilmente compreendidas e interpretadas por humanos, o que é crucial em domínios onde a transparência nas decisões é importante. Além disso, essas regras podem ser atualizados de forma relativamente simples e, em alguns casos, induzidas automaticamente a partir de novos dados e de acordo com o uso.

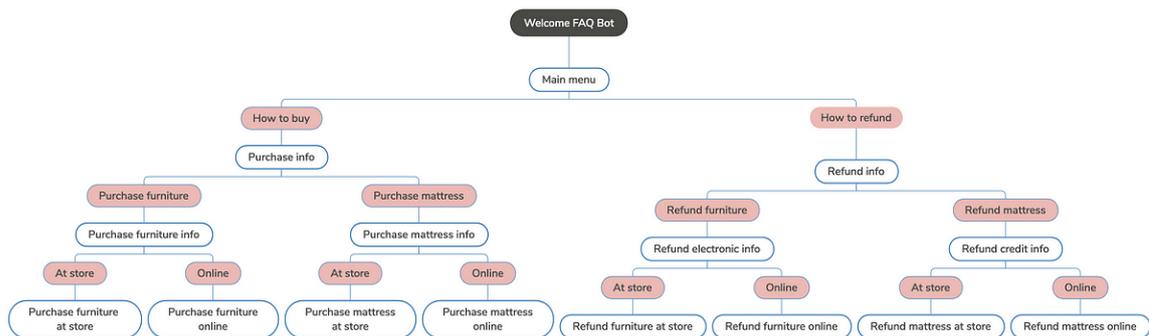


Figura 1.4: Exemplo de Árvore de Decisão.

**Redes Neurais Recorrentes (RNN)** - RNNs são uma categoria especial de redes neurais projetadas para lidar com dados sequenciais, como texto em conversas de *chatbots*.

Como será discutido nos próximos capítulos, as RNNs possuem uma capacidade interna de reter informações de estados anteriores, permitindo processamento sequencial. Durante o treinamento, as RNNs utilizam a Retropropagação no Tempo para ajustar pesos com base nas previsões e estados anteriores. As aplicações em chatbots incluem contextualização nas conversas, geração de respostas mais coesas, aprendizado de padrões dinâmicos em diálogos e correção de erros com base no feedback do usuário. Essa abordagem melhora a qualidade e a naturalidade das interações automatizadas em chatbots, especialmente em situações onde a compreensão do contexto temporal é essencial. No entanto, as RNNs apresentam alguns desafios, como a dificuldade em lidar com dependências de longo prazo e o problema de dissipação e exploração do gradiente.

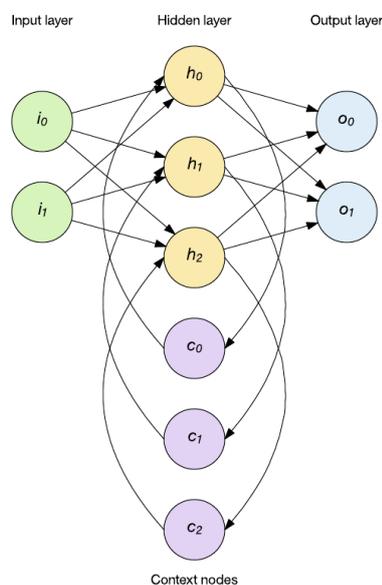


Figura 1.5: Exemplo de Rede Neural Recorrente.

**Outros Modelos** - Essa lista acima não é exaustiva, outras técnicas de construção de *chatbots* existem e não serão tratados nessa tese. Por exemplo, devido às características complementares dessas abordagens, a combinação de diferentes métodos pode ser usada para criar um chatbot mais fluido, completo e/ou preciso. Em especial, uma abordagem baseada em regras pode ser combinada com redes neurais recorrentes. A abordagem baseada em regras pode lidar com as interações mais simples que estão descritas na árvore de decisão do modelo, enquanto as RNNs podem ser usadas para responder os questionamentos que saem do escopo da árvore de acordo com os dados aprendidos no treinamento.

## A revolução da Arquitetura Transformer

Comparando a ELIZA (1966), a ALICE (1995) e a Siri (2011) é possível notar uma evolução significativa na capacidade de compreensão das perguntas e geração de respostas precisas. No entanto, ainda existem muitas limitações com essas abordagens.

A maioria dos *chatbots* ainda entendiam apenas um conjunto de ações pré-definidas e tinham uma capacidade reduzida de manter o contexto das conversas. Consequentemente, os usuários ainda precisavam aprender frases específicas para interagir com essas ferramentas, caso contrário, ela não conseguia entender o que o usuário queria. Frustrados com a experiência e precisando pensar ativamente em qual linguagem ou comando usar para que o chatbot entendesse, os usuários comumente preferiam procurar a informação em outro lugar. A limitação na conversação de maneira natural inibiram o crescimento desses serviços.

Os demais *chatbots*, que eram minoria, utilizavam as RNNs e geravam conversas mais fluidas. Porém a capacidade de entender o contexto e generalizar conceitos ainda eram muito embrionárias, ainda eram limitados em suas funcionalidades e frequentemente geravam respostas imprecisas com dados errados ou que não respondiam os questionamentos do usuário.

## Surgimento da arquitetura Transformers

A arquitetura Transformers foi proposta no artigo "*Attention is All You Need*" por Vaswani et al., publicado em 2017 (VASWANI *et al.*, 2017). Nesse documento, os pesquisadores apresentaram a ideia de mecanismos de atenção (*self-attention*), uma técnica fundamental que permite ao modelo dar mais peso a diferentes partes da entrada, dependendo do contexto.

Os mecanismos de atenção foram uma inovação crucial, permitindo ao modelo processar informações de maneira não sequencial, capturando relações contextuais entre palavras em uma frase e permitindo o treinamento de grande parte da rede em paralelo. Isso resultou em uma compreensão mais profunda e contextual das sequências.

A arquitetura *Transformers* foi proposta inicialmente para tarefas de tradução de textos. No entanto, a arquitetura se mostrou versátil e foi rapidamente adotada em várias outras tarefas de processamento de linguagem natural. Sua capacidade de lidar com sequências de comprimentos variados com eficiência a tornou atraente para uma ampla gama de aplicações.

Uma das aplicações é o chamado modelo de linguagem. Um modelo de linguagem é uma construção matemática utilizada em Processamento de Linguagem Natural ("*Natural Language Processing*" ou NLP) e aprendizado de máquina para compreender e gerar texto de maneira coerente e contextualmente relevante. Este tipo de modelo é treinado em grandes volumes de dados textuais para aprender padrões, relações e nuances da linguagem humana. Em sua essência, um modelo de linguagem é projetado para atribuir probabilidades a sequências de palavras, permitindo que o sistema avalie a probabilidade de uma determinada sequência de palavras ser gramaticalmente correta e semanticamente coerente. Esta habilidade é crucial para tarefas como geração de texto, tradução automática, resumo de documentos, entre outras aplicações em NLP.

O conceito de pré-treinamento em modelos de linguagem foi consolidado com o lançamento do GPT-1 pela OpenAI em 2018 (ALEC RADFORD e SUTSKEVER, 2018). Esse modelo introduziu a ideia de pré-treinar um modelo em grandes quantidades de dados para aprender padrões linguísticos antes de ajustá-lo para tarefas específicas.

O GPT-2, lançado em 2019, aumentou significativamente a escala do modelo, demonstrando a capacidade de gerar texto coerente e semântico em uma variedade de tarefas. Em 2020, a OpenAI lançou o GPT-3, que levou essa escala a novos patamares, com 175 bilhões de parâmetros, permitindo realizar uma diversidade ainda maior de tarefas sem ajuste fino específico.

## **Desenvolvimento de chatbots com a arquitetura Transformers**

Após as três iterações de modelos de linguagem utilizando a arquitetura de transformers com o GPT-1 (2018), GPT-2 (2019) e GPT-3 (2020), em novembro de 2022, a OpenAI lançou o ChatGPT, um dos primeiros chatbots a utilizar a arquitetura dos transformers. A comoção, surpresa e adoção do público foi extraordinário.

Em apenas 5 dias, a plataforma já contava com mais de um milhão de pessoas e, dois meses após o seu lançamento, o serviço já tinha ultrapassado 100 milhões de usuários ativos, se tornando o software com a adoção mais rápida da história. Ele redefiniu as expectativas em relação à capacidade de sistemas de conversação, demonstrando uma habilidade notável em compreender contextos, gerar respostas coesas e proporcionar experiências de usuário mais autênticas.

“O ChatGPT é, simplesmente, o melhor chatbot de inteligência artificial já lançado para ao público. Durante a maior parte da última década, os chatbots de IA têm sido terríveis, com resultados bons apenas se você escolher a dedo as melhores respostas do bot e descartar o resto. Nos últimos anos, alguns recursos de IA se destacaram ao realizar tarefas específicas e bem definidas. Mas o ChatGPT parece diferente. Mais inteligente. Mais peculiar. Mais flexível. Os primeiros usuários falam dele surpresos e em termos grandiosos, como se fosse uma mistura de software e magia, um supercérebro linguístico altamente capaz. Ainda estou tentando compreender o fato de que o ChatGPT - um chatbot que algumas pessoas pensam que pode tornar o Google obsoleto - já está sendo comparado ao iPhone em termos de seu impacto potencial na sociedade.”

— Kevin Roose - The New York Times

“ChatGPT é um desses raros momentos na tecnologia em que você vislumbra como tudo será diferente daqui para frente.”

— Aaron Levie - CEO of Box

A revolução introduzida pelos modelos de *Transformers* no mercado de *chatbots* representou uma melhoria exponencial. Ao contrário dos modelos anteriores, os *chatbots* baseados em *Transformers* conseguem compreender uma ampla variedade de ações e contextos, oferecendo respostas em um outro nível de naturalidade, contextualização e inteligência. No cenário atual, observa-se uma mudança na percepção dos usuários, que agora veem os *chatbots* como ótimos assistentes na realização de suas tarefas e obtenção de informações.

Sua utilidade se estende a diversas áreas, desde assistência ao cliente e suporte técnico até ferramentas educacionais e interações cotidianas. Essa versão do modelo alcançou rápida adoção em várias aplicações, desde serviços para pessoas em seu cotidiano até ferramentas para empresas.

Além disso, produtos inovadores baseados em *Transformers* têm se destacado em setores diversos, desde geração de texto até geração de imagem, música e moléculas químicas. Nas próximas seções, os modelos de *transformers* serão explorados em mais detalhes, em especial como eles funcionam e como podem ser utilizados para construir *chatbots* mais avançados.

## Capítulo 2

# Fundamentação Teórica: Chatbots com Arquitetura Transformer

À medida que a demanda por interações mais naturais e contextualmente ricas cresce, os *chatbots* baseados na arquitetura *Transformer* emergem como uma solução promissora. Este capítulo trará uma visão aprofundada dessa abordagem inovadora. Considerando um leitor sem conhecimento prévio de inteligência artificial e redes neurais, o capítulo explica os conceitos fundamentais de inteligência artificial, como redes neurais *feedforward*, passando por redes neurais recorrentes, *embedding* de palavras e mecanismos de atenção, até chegar na arquitetura *transformer*. Além dos mecanismos e funcionamento interno de cada componente, serão abordados seus principais desafios e limitações, bem como as principais aplicações e benefícios que elas trazem para a construção de chatbots avançados.

Em particular, será explorado como os modelos de transformers, capacitam essa geração de respostas, mergulhando em seus mecanismos internos para lidar com a diversidade e complexidade da linguagem, destacando sua aplicação na construção de *chatbots* avançados e oferecendo subsídios cruciais para a aplicação eficaz dessas tecnologias inovadoras.

## Redes Neurais

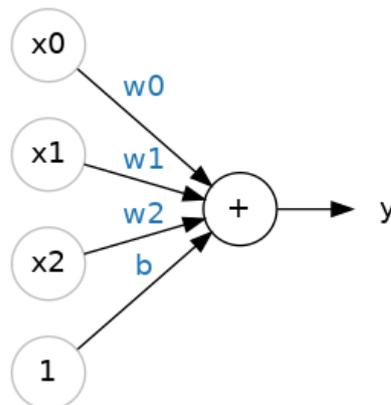
Redes neurais são um dos paradigmas de programação mais encantadores já inventados. Ao invés de dizer explicitamente ao computador como resolver um problema, quebrando-o em muitas tarefas pequenas e bem definidas que o computador possa executar, a técnica define o sucesso em termos de desempenho e deixa o computador descobrir como resolver o problema por conta própria. Isto é, a partir de uma grande quantidade de dados observacionais, ele consegue aprender, descobrindo sua própria solução para o problema em questão. Isso é especialmente útil quando o problema a ser resolvido não possui uma solução algorítmica clara, ou quando a solução é muito complexa para ser expressa como um conjunto de regras, como é o caso de muitos problemas de processamento de linguagem natural. Inspirada no cérebro humano, as redes neurais são muito flexíveis e podem aprender quase qualquer coisa, desde que o problema possa ser expresso como uma série de exemplos de treinamento ou ter uma clara função de comparação entre a

performance de redes neurais diferentes.

## Neurônio

O funcionamento básico das redes neurais começa com um entendimento de seus componentes primários: os neurônios artificiais (também conhecidos como "*nodes*"). Compreender seu funcionamento é essencial para entender o poder e a flexibilidade dessa arquitetura de aprendizado de máquina como um todo.

Em uma rede neural, um neurônio artificial recebe vários sinais de entrada, processa-os e produz um sinal de saída. Cada entrada é associada a um peso, que representa a força da conexão e refletem a importância relativa de cada entrada na decisão do neurônio, e esse valor é ajustável ao longo do processo de treinamento da rede. Além disso, um neurônio geralmente possui um viés (ou "*bias*"), uma espécie de peso independente que permite ajustar a saída do neurônio de forma independente das entradas. A saída do neurônio é calculada usando uma soma ponderada das entradas e o viés e aplicando uma função não-linear de ativação a esse valor.



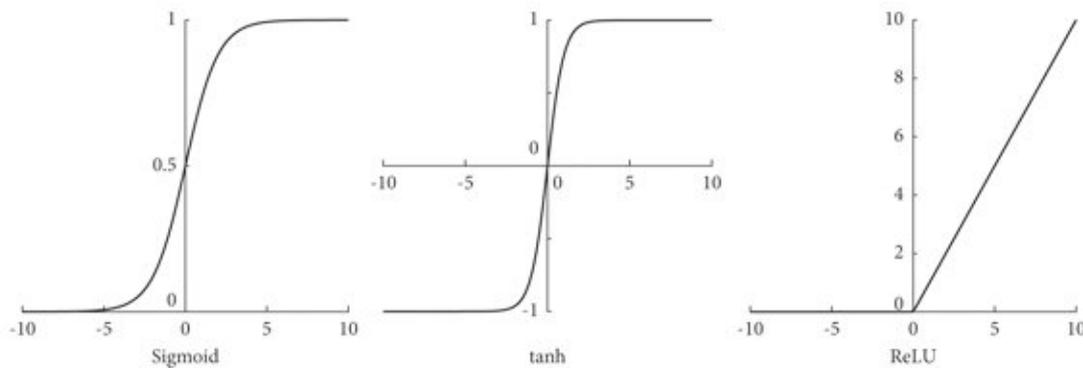
**Figura 2.1:** A fórmula para este neurônio seria  $y = \theta(w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + b)$ , sendo  $\theta$  a função de ativação.

A função de ativação desempenha um papel crucial no funcionamento de um neurônio em uma rede neural. Sua importância reside na introdução de não linearidades nas operações da rede, permitindo que a mesma aprenda e represente padrões mais complexos e não lineares nos dados.

Quando a soma ponderada das entradas, acrescida do termo de viés, é passada diretamente para a saída sem a intervenção de uma função de ativação, a rede neural se reduz a uma composição linear de operações. Isso limita a capacidade da rede de modelar relações não lineares nos dados, comprometendo sua capacidade de aprender com eficiência.

Ao aplicar uma função de ativação, a rede neural ganha a capacidade de modelar relações não lineares, capturando padrões mais intrincados nos dados. Existem várias funções de ativação comumente usadas, como a função sigmoide, tangente hiperbólica ( $\tanh$ ) e a unidade linear retificada (ReLU).

A função sigmoide, frequentemente usada em problemas de classificação binária ou quando a saída desejada está na forma de uma probabilidade, mapeia os valores para um intervalo entre 0 e 1. A tangente hiperbólica (tanh), similar à sigmoide, mapeia valores para o intervalo entre -1 e 1, proporcionando uma simetria em torno do zero. A unidade linear retificada (ReLU) é uma escolha comum em redes neurais profundas, ela retorna zero para valores negativos e deixa passar valores positivos, ajudando na superação do problema de desvanecimento do gradiente e acelera o treinamento de redes neurais, que veremos nas próximas seções.



**Figura 2.2:** As funções de ativação mais comuns: Sigmoid, Tangente Hiperbólica e Unidade Linear Retificada.

## Camadas na Rede Neural

Um neurônio por si só não é muito poderoso. No entanto, quando combinados em uma rede, eles podem ser usados para modelar funções complexas. Os neurônios são organizados em camadas distintas, geralmente incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.

Cada neurônio em uma camada está conectado a todos os neurônios na próxima camada através de conexões ponderadas, formando um grafo orientado densamente conectado, isto é, um grafo com arestas direcionadas entre cada par de arestas das camadas adjacentes. As redes que consistem em muitas camadas ocultas são referidas como redes neurais profundas ou "*deep neural networks*".

A camada de entrada é responsável por receber os dados brutos ou características iniciais do problema em questão. Cada neurônio nesta camada representa uma dimensão dos dados de entrada, e esses neurônios são conectados a cada neurônio na próxima camada, conhecida como camada oculta.

As camadas ocultas ficam entre a camada de entrada e a camada de saída. São chamadas "ocultas" porque suas atividades não são diretamente interpretadas externamente. Por conta disso, as redes neurais frequentemente são referidas como "caixas pretas". Esse termo sugere que, apesar de sua capacidade de realizar tarefas impressionantes de aprendizado de máquina, o processo interno pelo qual chegam a determinadas decisões pode ser opaco e difícil de entender para os humanos. Dentro de uma rede neural, milhões de

conexões ponderadas e camadas de neurônios interagem para gerar resultados. Isso torna difícil entender como a rede neural chega a uma determinada decisão, mesmo para os próprios criadores da rede, e levanta questões sobre a transparência e a explicabilidade do modelo, especialmente em cenários em que é crucial compreender as razões por trás das decisões tomadas. Dessa discussão surge o conceito "*Explainable AI*", que vêm sendo muito pesquisado na atualidade, buscando desenvolver modelos de inteligência artificial que não são apenas precisos, mas também compreensíveis e interpretáveis pelos seres humanos. Em outras palavras, a *Explainable AI* procura abrir a "caixa preta" das redes neurais, proporcionando *insights* sobre como o modelo toma decisões, especialmente em setores onde a confiança, ética e segurança são primordiais, como na medicina, finanças e justiça.

As camadas ocultas são organizadas da seguinte forma: cada neurônio em uma camada oculta é conectado a todos os neurônios da camada anterior e da camada seguinte. A inclusão de múltiplas camadas ocultas permite que a rede aprenda representações progressivamente mais abstratas e complexas dos dados.

A camada de saída é responsável por gerar as previsões ou classificações finais da rede com base nas representações aprendidas nas camadas ocultas. Cada neurônio nesta camada representa uma classe ou valor de saída possível, e os neurônios são ativados de acordo com as características extraídas durante o treinamento.

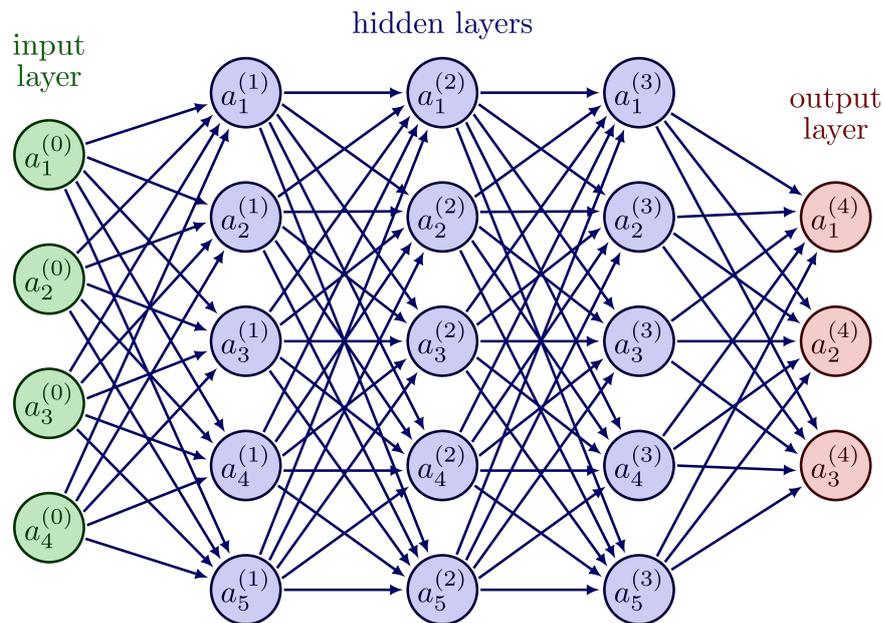


Figura 2.3: Exemplo de rede neural com diversas camadas ocultas.

## Treinamento

Como explicado anteriormente, as Redes Neurais não dizem explicitamente ao computador como resolver um problema, ao invés Redes Neurais são estruturas extremamente flexíveis que podem ser usadas para aprender a resolver problemas. Em vez de programar uma solução para um problema, nós alimentamos dados de treinamento para a rede neural e deixamos que ela aprenda a partir deles.

Para que isso seja possível, é crucial que exista uma maneira de medir o quão bem a rede neural está se saindo em uma determinada tarefa. Existem, em geral, duas formas de fazer isso: aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado, a rede neural é treinada com um conjunto de dados de treinamento, onde as respostas corretas são conhecidas. A rede neural então aprende a mapear os dados de entrada para as saídas desejadas, e a função de custo é usada para medir o quão bem a rede neural está realizando a tarefa. No aprendizado não supervisionado, a rede neural é treinada em um conjunto de dados de treinamento sem rótulos, isto é, sem um resultado conhecido, e a rede neural aprende a modelar a distribuição dos dados de entrada de acordo com uma função de custo que calcula uma pontuação para o resultado das redes.

O aprendizado supervisionado é ideal em situações onde existe um conjunto de dados rotulado, ou seja, onde as saídas desejadas estão claramente definidas, como em tarefas de classificação, como identificar se um e-mail é spam ou não, ou prever o preço de uma casa com base em características específicas, entre outros. Em contraste, o aprendizado não supervisionado é mais adequado quando não existe uma resposta certa, isto é, é uma questão de várias iterações de otimizações até chegar no melhor resultado possível para aquele modelo. Essa abordagem é usada em na criação de robôs que jogam programas de computador, tarefas de agrupamento como identificar segmentos de clientes com comportamentos semelhantes em dados de compras.

No caso da implementação de *chatbots*, o aprendizado supervisionado é o mais adequado, pois é possível definir um conjunto de dados de treinamento com perguntas e respostas, onde as respostas são as saídas desejadas. O objetivo é que a rede neural aprenda a mapear as perguntas para as respostas, de forma que, quando um usuário fizer uma pergunta, a rede neural seja capaz de gerar uma resposta coerente.

O processo de treinamento nesse caso é geralmente realizado através de um algoritmo conhecido como *backpropagation*, que ajusta os pesos de forma iterativa tentando aproximar a camada de saída da rede com o resultado esperado do treinamento. A definição inicial dos pesos é feita de forma aleatório e, a cada iteração os pesos são alterados de forma que a função de custo seja minimizada. Esse processo é repetido por várias épocas (passagens completas pelo conjunto de treinamento) até que a rede atinja um estado em que as previsões são satisfatórias para as saídas desejadas.

O *backpropagation* se baseia em duas passagens através da rede neural: uma passagem direta para calcular as saídas, e uma passagem de volta para calcular os gradientes. A primeira passagem ocorre quando os dados de entrada são alimentados pela rede neural. Cada camada de neurônios processa esses dados, multiplicando-os pelos pesos das conexões e aplicando funções de ativação. O resultado final é a previsão da rede.

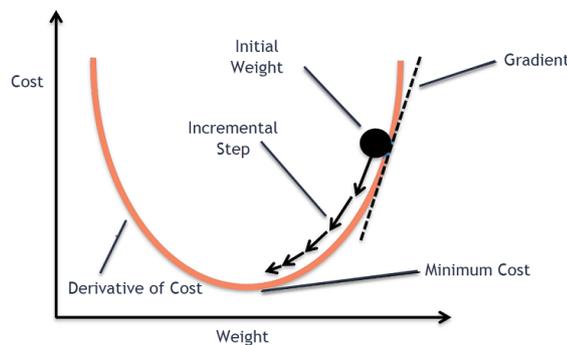
Na segunda passagem, a passagem de volta, é onde ocorre o ponto central do *backpropagation*. Durante essa fase, a rede compara a previsão obtida na passagem direta com a saída desejada (o resultado esperado do treinamento). O algoritmo então calcula o erro da rede, que é a diferença entre a saída esperada e a saída obtida e, após calcular o erro, realiza o processo de retropropagação desse erro ao longo da rede, camada por camada, calculando os gradientes dos pesos. Os gradientes indicam como os pesos devem ser ajustados para reduzir o erro na próxima iteração.

É nessa etapa que entra o algoritmo de otimização, frequentemente o *gradient descent*. O objetivo desse algoritmo é minimizar a função de custo, ou seja, encontrar os pesos que resultam na menor perda possível. O cálculo do gradiente é realizado através da derivada parcial da função de custo em relação a cada um dos parâmetros do modelo. Considerando a função de custo  $J(\theta)$ , onde  $\theta$  representa o vetor de parâmetros do modelo, o gradiente  $\nabla J(\theta)$  é um vetor que consiste nas derivadas parciais de  $J$  em relação a cada parâmetro  $\theta_i$ , calculado pela função abaixo (GOODFELLOW *et al.*, 2015):

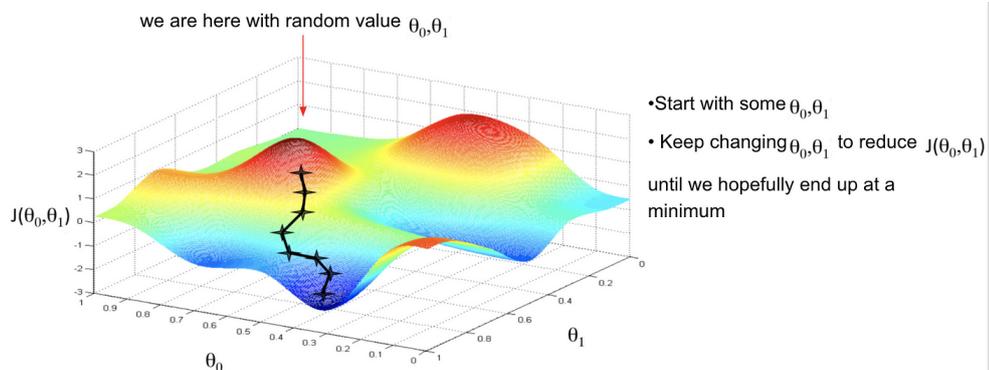
$$\nabla J(\theta) = \left( \frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \dots, \frac{\partial J}{\partial \theta_n} \right)$$

Esta fórmula representa um vetor de derivadas parciais em relação a cada parâmetro  $\theta_i$  da função de custo. Cada termo do vetor é calculado individualmente utilizando a regra da cadeia, que fornece a taxa de variação da função de custo em relação a cada parâmetro.

Essas derivadas parciais indicam como a função de custo varia em resposta a pequenas mudanças nos parâmetros individuais. A magnitude de cada componente do vetor gradiente indica a sensibilidade da função custo a cada peso e bias individual. Se o componente do gradiente para um peso específico é alto, isso significa que a função custo é mais sensível àquele peso, indicando que ajustes nele terão um impacto maior na redução do custo.



**Figura 2.4:** Algoritmo gradient descent.



**Figura 2.5:** Exemplo de um caminho do algoritmo gradient descent.

Utilizando os gradientes calculados, o algoritmo de otimização ajusta os pesos da rede na direção que minimiza o erro. Esse processo de ajuste iterativo dos pesos é repetido ao longo de várias épocas, refinando gradualmente a capacidade da rede neural de realizar previsões precisas.

## Redes Neurais Recorrentes

As redes neurais descritas na seção anteriormente, são chamadas de Redes Neurais *Feed Forward*. Nessas redes, a informação move-se em uma única direção, da camada de entrada para a camada de saída, sem ciclos ou realimentação. Cada camada de neurônios está conectada à camada seguinte, e não há conexões retroativas que formem loops na arquitetura. Essa estrutura *feed forward* é comum em muitas aplicações de aprendizado de máquina, incluindo tarefas como reconhecimento de padrões, classificação e regressão. Entretanto, é importante notar que existem outras arquiteturas de redes neurais mais complexas que incorporam conexões retroativas ou padrões específicos para tarefas especializadas.

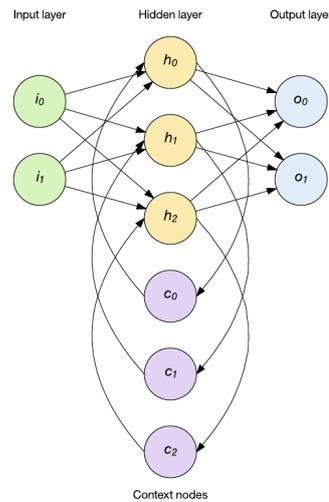
Um exemplo dessas estruturas mais complexas que é especialmente importante no contexto de *chatbots* é a rede neural recorrente, também conhecida como RNN (ou "*Recurrent Neural Network*"). Essa estrutura oferece várias vantagens em relação às redes neurais *feed forward* em tarefas que envolvem sequências de dados. Como um texto é interpretado como uma sequência de palavras, as RNNs são especialmente adequadas para tarefas de processamento de linguagem natural.

Um dos pontos fortes das RNNs é sua habilidade de capturar informações de contextos anteriores em uma sequência. As RNNs possuem uma forma de "memória interna" que lhes permite manter informações sobre eventos anteriores em uma sequência. Essa capacidade de memória de curto prazo é crucial para tarefas em que o contexto passado é relevante para a compreensão do presente. Isso é especialmente benéfico em tarefas em que o contexto histórico influencia a interpretação ou a previsão de dados atuais. Em processamento de linguagem natural, por exemplo, onde o significado de uma palavra muitas vezes depende das palavras que a precedem, as RNNs conseguem capturar essas dependências temporais e produzir resultados mais contextualmente relevantes.

Outra vantagem importante das RNNs é sua flexibilidade em lidar com entradas e saídas de comprimentos variáveis. Isso as torna eficazes em tarefas em que as dimensões dos dados podem mudar, como é o caso em *chatbots* em que o usuário pode enviar textos de tamanhos diferentes e a resposta também tem tamanhos variados.

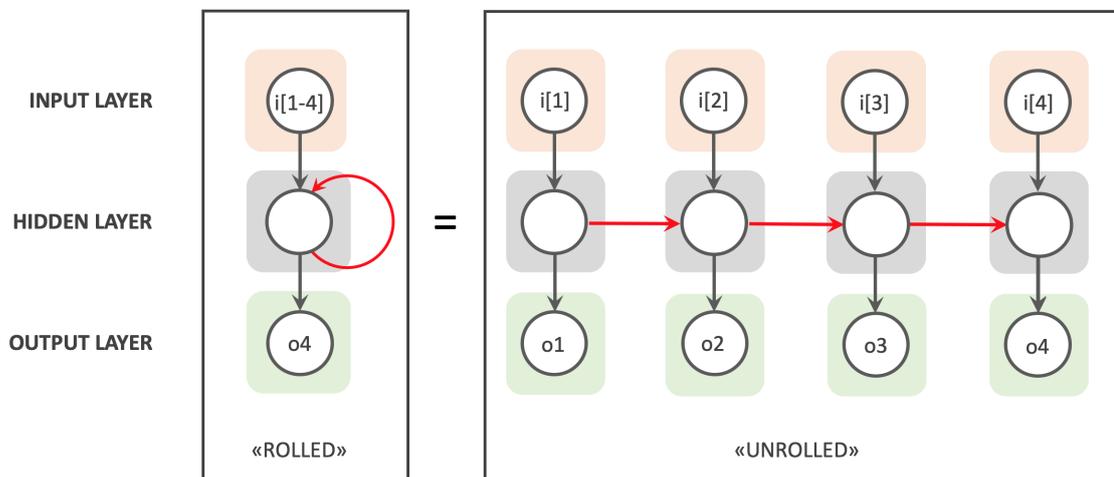
## Funcionamento

Em sua configuração básica, uma RNN possui as mesmas três camadas que uma Rede Neural *Feed Forward*: camada de entrada, a camada oculta e a camada saída. A diferença é que a camada oculta é recorrente, ou seja, ela possui conexões retroativas que permitem que as informações sejam transmitidas entre neurônios em direções arbitrárias, formando loops na arquitetura (JOHNSON, 2017).



**Figura 2.6:** Exemplo de Rede Neural Recorrente.

A imagem acima descreve muito bem um exemplo de rede neural, porém uma forma mais intuitiva de enxergar as redes neurais pode ser vista na imagem abaixo, em especial a representação da direita. Note que cada bloco bege representa uma camada de neurônios de entrada, cada bloco cinza representa várias camadas ocultas e cada bloco verde representa uma camada de neurônios de saída. A seta que sai de um bloco e entra em outro representa uma conexão entre as camadas, e a seta que sai de um bloco e entra nele mesmo representa uma conexão retroativa, ou seja, uma conexão entre neurônios da mesma camada.



**Figura 2.7:** As duas formas de enxergar Redes Neurais Recorrentes.

A representação *unrolled* em especial, evidencia o processamento sequencial dessa estrutura, que é descrito a seguir. A entrada é fornecida à rede um elemento por vez, de acordo com a sequência temporal, e cada elemento contém um vetor de características. A RNN mantém um estado oculto que é inicializado como um vetor de zeros e atualizado a cada passo da sequência. Ele atua como uma espécie de “memória” da rede, capturando informações relevantes de passos anteriores na sequência, permitindo que informações de passos anteriores influenciem a computação em passos futuros. Assim, a ativação de cada

unidade na camada oculta em um determinado passo de tempo depende não apenas da entrada atual, mas também do estado oculto anterior. Isso cria uma forma de realimentação, permitindo que a rede mantenha uma noção de contexto sequencial. A cada passo, além da alteração do estado interno, pode existir ou não uma saída dependendo da rede neural que está sendo utilizada, que será investigado ao decorrer do capítulo. Em alguns casos a saída é gerada a cada passo ("*many to many*"), em outros casos os passos simplesmente alteram o estado interno e a saída é gerada apenas no último passo ("*many to one*"), a saída também pode ser gerada a partir de um único dado ("*one to many*") ou ainda, a saída começa a ser gerada após o processamento de todos os inputs ("*many to many*").

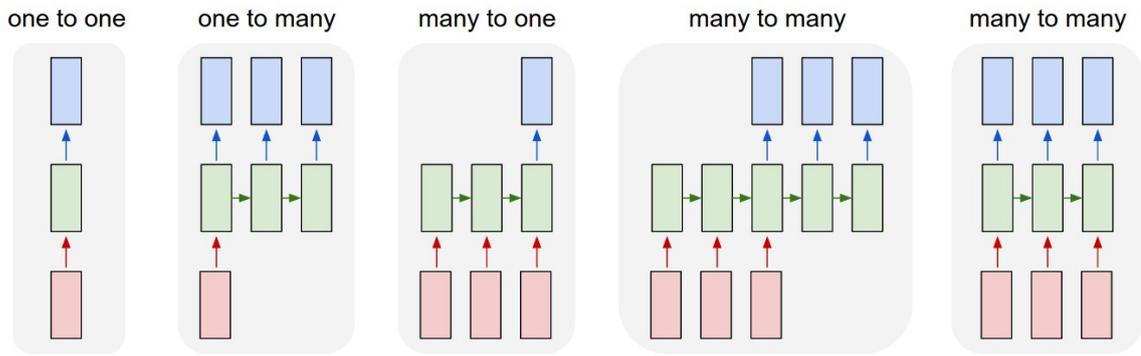
Cabe ressaltar que ambas as representações na imagem anterior descrevem a mesma rede neural, assim, os pesos e vieses são compartilhados entre os neurônios correspondentes de cada etapa. Isso significa que todos os neurônios de uma camada oculta compartilham os mesmos pesos e vieses, e todos os neurônios de saída também compartilham os mesmos pesos e vieses. Essa característica é crucial pois permite que a rede generalize padrões aprendidos em um determinado ponto temporal para outros momentos ao longo da sequência. Quando os pesos e vieses são compartilhados, significa que a mesma matriz de pesos e vetor de viés são utilizados em cada passo de tempo durante o processamento sequencial.

A técnica de retropropagação também requer adaptações ao ser aplicada em redes neurais recorrentes, sendo denominada como retropropagação através do tempo, também conhecida como "*Backpropagation Through Time*" ou BPTT. O BPTT é uma extensão do algoritmo de retropropagação padrão que permite que os gradientes fluam por várias etapas de tempo. Ele funciona desdobrando a rede ao longo do tempo, calculando os gradientes em cada etapa e, em seguida, combinando os gradientes em um único vetor que pode ser usado para atualizar os pesos da rede. Esse processo leva em consideração a contribuição de cada passo temporal para o erro global, o que é essencial para a utilização da rede com dados sequenciais (JOHNSON, 2017).

É importante notar que, ao retropropagar o erro através do tempo, a RNN pode enfrentar problemas como o "*vanishing gradient*" (gradiente que desaparece) ou "*exploding gradient*" (gradiente que explode). Isso ocorre porque os pesos de cada da sequência é o mesmo, assim como os pesos são multiplicados entre cada passo, o gradiente diminui ou aumenta exponencialmente à medida que retrocede no tempo, dependendo se o número for maior ou menor que um. Isso dificulta o treinamento eficiente de RNNs em sequências longas.

## As diferentes configurações

Por conta da flexibilidade das redes neurais recorrentes, em especial em relação ao número entradas e saídas, existem diversas configurações diferentes. Observa-se na imagem abaixo algumas dessas organizações.



**Figura 2.8:** Os diferentes configurações de Redes Neurais Recorrentes.

A configuração *one-to-one* é a mais simples, sendo equivalente a uma rede neural *feedforward* tradicional. A entrada é fornecida à rede, que processa os dados e gera uma saída. Essa configuração é adequada para problemas em que a entrada e a saída têm comprimentos fixos e não há dependências temporais entre os dados. Nesse caso, não há necessidade de uma rede neural recorrente.

A configuração *one-to-many*, por sua vez, é usada quando a entrada tem comprimento fixo, mas a saída tem comprimento variável. A partir de uma entrada de tamanho fixo, a rede neural recorrente vai iterando sobre o estado interno oculto e gera uma sequência de saídas. A sequência geralmente para quando um determinado token é gerado, ou quando o tamanho máximo da sequência é atingido. Um exemplo de aplicação dessa configuração é a geração de legendas que descrevam as imagens de entrada, pois a imagem tem um tamanho fixo e a legenda tem um comprimento variável.

A configuração *many-to-one* é usada quando a entrada tem comprimento variável, mas a saída tem comprimento fixo. A cada vetor da entrada, a rede neural altera seu estado interno oculto, resumindo de certa forma o conteúdo que está sendo lido, e, após processar todos os vetores de entrada, gera uma única saída com base no estado interno que contém o resumo das entradas. Um exemplo de aplicação dessa configuração é a classificação de sentimentos em textos, onde a entrada é uma sequência de palavras e a saída é um único valor que representa o sentimento do texto.

Por último, a configuração *many-to-many* é útil quando a entrada e saída tem comprimentos variáveis. Nesse caso, há duas possibilidades: a saída pode ser gerada a cada passo da sequência de entrada, ou a saída pode ser gerada apenas após o processamento de todos os vetores de entrada. A primeira opção pode ser usada em tarefas como o reconhecimento de fala, em que a entrada e saída tem comprimentos iguais e a saída depende apenas do contexto adquirido das entradas anteriores, nesse caso a sequência de palavras que é gerada a cada passo da sequência de entrada. A segunda opção, em geral, é mais flexível, porque o comprimento da entrada e saída podem ser diferentes, além de permitir que cada saída seja gerada a partir do conhecimento de toda a sequência de entrada, em especial para sequências de entradas pequenas a médias. Semelhante à configuração *one-to-many*, a sequência geralmente para quando um determinado token é gerado, ou quando o tamanho máximo da sequência é atingido. Ela é útil em tarefas como a geração de textos, por exemplo, em que a saída é uma sequência de palavras que é gerada apenas após o processamento de todos os vetores de entrada.

Na implementação de *chatbots*, a configuração *many-to-many* com a geração da sequência de saída após o processamento de toda a sequência de entrada é a mais adequada. Nesse configuração, a rede neural é capaz de entender a pergunta como um todo e, a partir do estado interno oculto, gerar uma resposta de tamanho dinâmico. Nesse caso, cada *input* e *output* representa uma palavra ou *token*.

## LSTMs

Como já foi explorado nessa seção, as redes neurais recorrentes trazem uma série de vantagens e possibilitam novas aplicações, em especial ao lidar com dados sequenciais. Contudo, é crucial reconhecer que as RNNs também apresentam desafios, como o problema do desaparecimento ou explosão do gradiente, que podem afetar o treinamento eficiente e a capacidade de aprender dependências temporais muito longas.

Para lidar com esses problemas, arquiteturas mais avançadas, como *Long Short-Term Memory* (LSTM), foram desenvolvidas. Elas possuem mecanismos internos para controlar o fluxo de informações ao longo do tempo, que facilitam o treinamento de RNNs em sequências mais extensas, e possuem caminhos diferentes para as memórias de curto prazo e de longo prazo.

As Long Short-Term Memory networks (LSTMs) são uma arquitetura especializada de redes neurais recorrentes (RNNs) que foram projetadas para superar os desafios associados ao treinamento de RNNs em tarefas que envolvem dependências temporais de longo prazo. Desenvolvidas por Hochreiter e Schmidhuber em 1997 (HOCHREITER e SCHMIDHUBER, 2015), as LSTMs introduzem uma estrutura de célula de memória que permite a retenção de informações por períodos prolongados, mitigando assim o problema do "*vanishing gradient*" em sequências extensas.

A arquitetura LSTM é composta por unidades de células, cada uma equipada com três portões fundamentais: o portão de esquecimento (*forget gate*), o portão de entrada (*input gate*) e o portão de saída (*output gate*). Esses mecanismos são projetados para regular o fluxo de informações dentro da célula de memória, proporcionando maior controle sobre o processo de aprendizado.

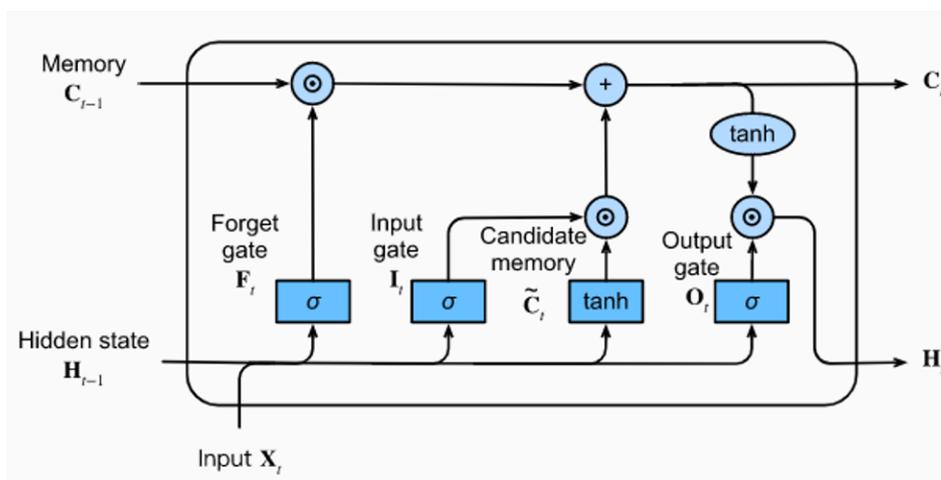


Figura 2.9: Os componentes da LSMT.

Analisando cada um desses componentes, em primeiro lugar, a célula de memória é o núcleo da arquitetura LSTM e atua como uma “memória” que pode armazenar informações por longos períodos. O estado da célula de memória é atualizado através de operações controladas pelos portões.

O portão de esquecimento decide quais informações da célula de memória anterior serão mantidas ou descartadas. É ativado por uma função sigmoide, que gera valores entre 0 e 1 para cada componente da célula de memória.

O portão de entrada determina quais novas informações da entrada atual serão adicionadas à célula de memória. Ele consiste em duas partes: uma função sigmoide para gerar valores de 0 a 1 que indicam quais informações atualizar, e uma função tangente hiperbólica para gerar os candidatos a novos valores.

O portão de saída decide qual parte da célula de memória atualizada será a saída da unidade LSTM. O resultado é obtido aplicando uma função sigmoide ao estado da célula de memória para decidir quais partes serão ativadas, seguido por uma função tangente hiperbólica para escalar o resultado (HOCHREITER e SCHMIDHUBER, 2015).

Assim, a célula de memória é atualizada combinando as informações retidas pelo portão de esquecimento e as novas informações determinadas pelo portão de entrada.

As LSTMs permitem que as redes aprendam relações temporais de longo prazo, tornando-as particularmente eficazes em tarefas como tradução automática, geração de texto e previsão de séries temporais. Elas se tornaram uma arquitetura fundamental em aprendizado de sequências, proporcionando uma solução eficaz para o desafio do “*vanishing gradient*”.

## Como computadores interpretam as palavras?

Como as próximas seções vão lidar mais especificamente com o processamento de texto, antes de prosseguir, é importante entender como os computadores interpretam as palavras. Para nós, humanos, é fácil entender o significado de uma palavra, mas para um computador, uma palavra é apenas uma sequência de caracteres.

Embora representar palavras como sequências de caracteres seja uma abordagem natural e intuitiva, essa forma de representação possui várias desvantagens. Por exemplo, representar palavras como sequências de caracteres resulta em representações de alta dimensionalidade, especialmente para palavras longas, isso pode levar a uma grande quantidade de dados e demandar mais recursos computacionais. Além disso, a representação de sequências de caracteres não captura diretamente a semântica das palavras, pois palavras semelhantes ou relacionadas podem ter representações muito diferentes quando comparadas a partir de suas sequências de caracteres.

Por essas e outras razões, é interessante que as palavras sejam representadas numericamente de uma forma que seja mais eficiente e que capture melhor o significado das palavras. Além de serem mais eficientes, essas representações permitem a generalização de relações semânticas entre palavras e que palavras semanticamente similares tenham representações vetoriais próximas umas das outras no espaço vetorial.

Existem diversas formas de representar uma palavra numericamente, como por exemplo, através de um vetor de números inteiros de zeros e uns. Cada posição do vetor representa uma palavra do vocabulário e cada vetor deve ter no máximo uma posição com valor 1, indica a presença da palavra correspondente. Essa representação é chamada de *one-hot encoding* e é uma das formas mais simples de representar uma palavra numericamente. Entretanto, essa representação carrega muitos problemas como o tamanho excessivo do vetor e a falta de uso do vetor, que é 99% vazio.

Uma solução mais eficiente é representar as palavras através de vetores de números reais. Essa representação é chamada de *word embedding* e é uma das formas mais utilizadas de representar palavras numericamente. Os *word embeddings* têm várias propriedades úteis. Palavras semanticamente semelhantes têm representações vetoriais próximas no espaço embedding, o que permite que modelos identifiquem relações de sinonímia (palavras sinônimas) e analogia. Operações vetoriais nos embeddings também podem representar relações entre palavras, por exemplo, a operação

$$\text{vetor}('rei') - \text{vetor}('homem') + \text{vetor}('mulher')$$

pode resultar em um vetor próximo ao de “rainha”. Além disso, os *embeddings* capturam nuances semânticas e relacionamentos e, pela representação densa em um espaço vetorial contínuo, a solução reduz a dimensionalidade em comparação com outras representações baseadas em sequências de caracteres ou *one-hot encoding*, facilitando o treinamento de modelos e economizando recursos computacionais (McCORMICK, 2016).

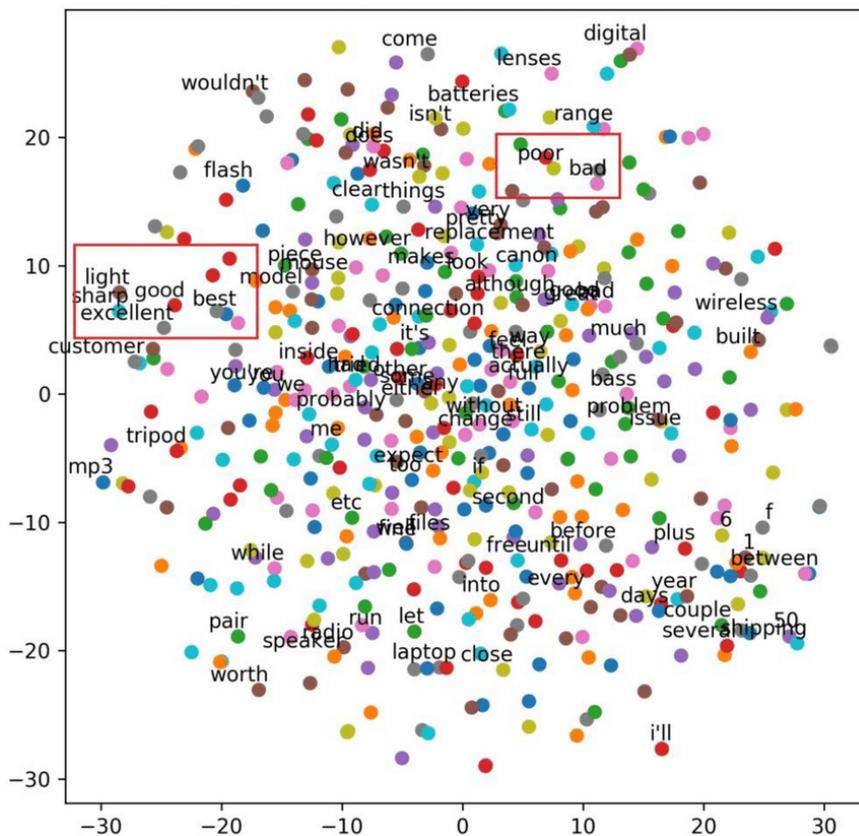


Figura 2.10: Exemplo de um Word Embedding em um espaço 2D.

Um dos algoritmos mais utilizados em processamento de linguagem natural para transformar palavras em vetores de números reais é o *Word2Vec*. O algoritmo utiliza uma abordagem baseada em aprendizado de máquina e redes neurais para gerar *word embeddings* a partir de um grande conjunto de dados de texto que mantenham as propriedades descritas anteriormente, como a propriedade de palavras que aparecem em contextos semelhantes tendem a ter significados semelhantes.

O *Word2Vec* não é um algoritmo de aprendizado supervisionado, ou seja, ele não requer rótulos ou anotações nos dados de treinamento. Ele pode ser considerado uma aprendizagem semi-supervisionada porque não temos rótulos diretos associados às palavras, mas utilizamos as palavras vizinhas de uma palavra ou o contexto em uma frase como os rótulos. Isso permite que os dados de treinamento sejam exponencialmente maiores já que não precisam estar rotulados, o que é uma vantagem significativa pela quantidade de dados estar relacionada proporcionalmente com a qualidade do modelo.

Existem duas principais arquiteturas associadas ao *Word2Vec*: o modelo *Skip-gram* e o modelo *Continuous Bag of Words* (CBOW). Nesta tese, será explorada a abordagem *Skip-gram*, que visa prever as palavras circunvizinhas dada uma palavra de entrada. A entrada é uma palavra de referência e saída deve ser um conjunto de palavras contextuais que aparecem próximas à palavra de referência.

A arquitetura dessa solução é composta por 3 camadas: a camada de entrada, a camada de *embedding* e a camada de saída. Inicialmente, cada palavra na camada de entrada é representada por um vetor *one-hot encoding*, onde o índice correspondente à palavra é 1, e todos os outros são 0. Em seguida, a camada de *embedding* mapeia as palavras para vetores densos no espaço vetorial multiplicando a matriz de *embedding*, que contém os pesos da camada de *embedding* aprendidos pela rede, pelo vetor *one-hot encoding*. Por último, a camada de saída usa uma função de ativação *softmax* para calcular a probabilidade das palavras contextuais dados os vetores de entrada e gera uma distribuição de probabilidade sobre o vocabulário. A saída é uma distribuição de probabilidade sobre todas as palavras no vocabulário, indicando a probabilidade de cada palavra ser uma palavra contextual da palavra de referência de acordo com a distância vetorial.

Durante o treinamento, os pesos da camada de *embedding* são ajustados usando algoritmos de otimização, como o *gradient descent*, para minimizar a diferença entre as previsões do modelo e as palavras reais do contexto, que foram presumidos ou aprendidos a partir dos milhões de textos base para o modelo. Ao final do treinamento, os vetores resultantes na camada de *embedding* oferecem uma boa aproximação do contexto real e são usados como as representações distribuídas de palavras no espaço vetorial. Os vetores aprendidos durante o treinamento capturam relações semânticas e sintáticas entre palavras, permitindo que palavras semanticamente similares tenham representações vetoriais próximas no espaço contínuo. Com esse processo, o *Word2Vec* cria representações significativas para palavras com base em seu contexto em grandes conjuntos de dados textuais.

A imagem abaixo representa muito bem todo o processamento na rede neural do *Skip-gram*. O vetor  $x$  é o *one-hot encoding* da palavra de entrada, a matriz  $W_{input}$  é a matriz de pesos da camada de *embedding* que multiplicada por  $x$  resulta no vetor  $h$ , que é um vetor contínuo com dimensão muito menor do que  $h$ . Por último, o vetor  $h$  é multiplicada pela matriz de pesos da camada de saída para gerar a distribuição de probabilidade sobre o

vocabulário chamada de  $y_{pred}$ .

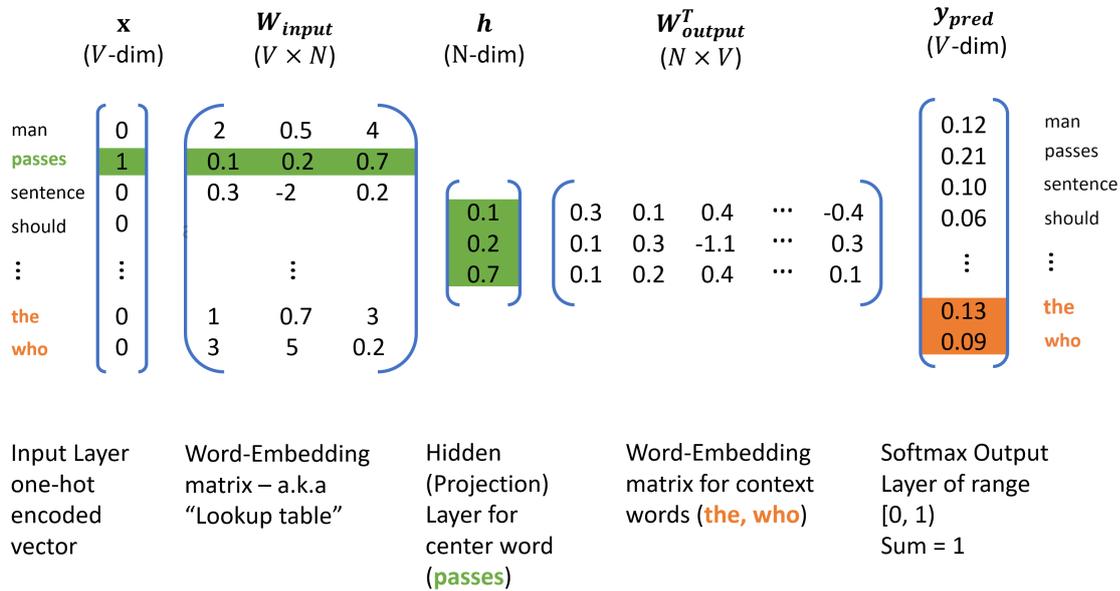


Figura 2.11: Treinamento do Word2Vec de acordo com o Skip-gram.

Apesar de não entrar em detalhes nesse documento, é importante citar a abordagem de criação de *embeddings* a partir de “subpalavras” ou morfemas. Algumas abordagens, como o *FastText*, pode criar *embeddings* para pedaços das palavras ao invés das palavras inteiras. Isso permite capturar informações semânticas de partes menores das palavras. Ela pode capturar morfologia e estruturas linguísticas em diferentes línguas, sendo mais flexível e aplicável a uma variedade de contextos linguísticos, e pode lidar com a composição de palavras e construção de significados a partir de partes menores, tornando-a mais eficaz para tarefas que envolvem palavras compostas. Uma consequência bem-vinda é a redução do vocabulário em comparação com a abordagem de palavras inteiras.

## Arquitetura Encoder-Decoder

Tendo em mente a representação das palavras em vetores contínuos, é possível entender como as arquiteturas *encoder-decoder* funcionam, as quais são fundamentais para o desenvolvimento em processamento de linguagem natural, em especial na criação de *chatbots*.

A arquitetura *Encoder-Decoder* constrói sobre o conceito do “contexto” criado nas redes neurais recorrentes e é mais flexível que a abordagem anterior, sendo o fundamento para abordagens mais complexas que serão discutidas em seguida, como modelos de atenção e a própria arquitetura *Transformers*. Ela é capaz de capturar um contexto global da entrada, especialmente através do vetor de contexto gerado pelo *encoder*. Também é altamente modular, permitindo a substituição de diferentes componentes, como o *encoder* e o *decoder*, por arquiteturas mais avançadas. Isso facilita a experimentação e o desenvolvimento de modelos mais complexos.

## Funcionamento

A arquitetura é composta por duas redes neurais recorrentes, uma chamada de *encoder* e outra chamada de *decoder*. O encoder é responsável por ler a entrada e gerar um vetor de contexto, que é uma representação vetorial do texto de entrada. O decoder, por sua vez, recebe o vetor de contexto gerado pelo encoder e gera a saída, que pode ser uma sequência de palavras ou um único valor.

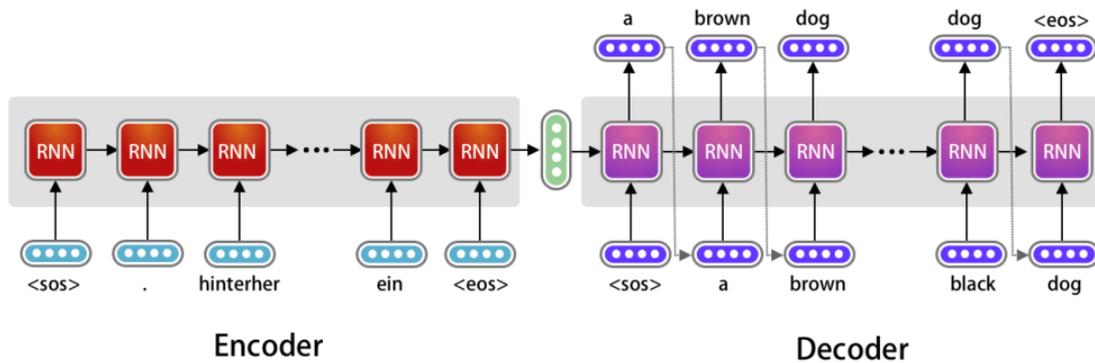


Figura 2.12: Arquitetura Encoder-Decoder utilizada em chatbots.

A imagem acima representa a arquitetura *Encoder-Decoder* utilizada em *chatbots*. Em geral, o *encoder* é uma rede recorrente LSTM "*many-to-one*". A RNN recebe uma sequência de palavras representada como *embeddings* e para cada elemento da sequência de entrada o LSTM atualiza seu estado oculto e célula de memória.

Após processar toda a sequência de entrada, o último estado oculto do LSTM do *encoder* é considerado o vetor de contexto. Este vetor, representado pelo componente verde na figura, contém informações condensadas sobre toda a sequência de entrada, encapsulando as características mais relevantes da sequência de entrada que serão utilizadas pelo *decoder* para gerar a sequência de saída correspondente.

O *decoder*, que geralmente também é implementado com LSTMs, tem a responsabilidade de gerar a sequência de saída com base nas informações contidas no vetor de contexto produzido pelo *encoder*. Ele é inicializado com o vetor de contexto produzido pelo *encoder* como entrada e um *token* especial de início (< sos > ou "*Start of Sentence*") é fornecido como a primeira entrada ao *decoder*, o que inicia o processo de geração da sequência de saída.

Diferentemente do *encoder*, o *decoder* pode ser considerado uma rede recorrente LSTM "*many-to-many*", porque além da memória interna cada rede da sequência recebe um *input* referente ao *output* da rede do passo anterior. Apesar de menos intuitiva, a interpretação da arquitetura como uma RNN "*one-to-many*" também é possível ao considerar *input* das RNN como uma simples alteração do estado interno. Nesse caso, entende-se que todos os *outputs* são gerados a partir de um só *input*: o vetor de contexto produzido pelo *encoder*.

Independente do tipo de RNN, a saída do *decoder* em cada passo temporal é usada para prever a próxima palavra na sequência de saída. O estado oculto e a célula de memória

são atualizados com base nas informações do passo temporal atual e são utilizados no próximo passo. Cada passo temporal gera uma saída baseada no estado oculto, célula de memória e no *input* que representa a última palavra gerada. Isso geralmente envolve uma camada totalmente conectada seguida por uma função de softmax para obter uma distribuição de probabilidade sobre o vocabulário. A distribuição de probabilidade sobre o vocabulário representa os percentuais de probabilidade de cada palavra ser a próxima palavra da sequência de saída (ALAMMAR, s.d.).

Duas abordagens são comumente utilizadas para gerar a sequência de saída: *argmax* e *sample*. Na abordagem *argmax*, a palavra com maior probabilidade é selecionada. O resultado é uma rede mais estável, gerando resultados mais previsíveis. Na abordagem *sample*, por outro lado, a próxima palavra é amostrada de acordo com a distribuição de probabilidade sobre o vocabulário. A seleção é randômica de acordo com a distribuição de probabilidades, isto é, palavras com maior probabilidade de serem a próxima na frase têm maior probabilidade de serem selecionadas, mas palavras com menor probabilidade também têm uma chance de serem selecionadas. Essa abordagem geralmente gera resultados com maior variabilidade e o mesmo texto no mesmo modelo pode gerar resultados diferentes. Em geral, esse comportamento é bem-vindo e essa abordagem é a mais utilizada, porém, na eventualidade do modelo selecionar algumas palavras com baixa probabilidade em sequência, o modelo pode gerar respostas ruins.

Esse processo dentro do *decoder* é repetido até que um token especial de fim (<eos> ou "End of Sentence") seja gerado ou até que o tamanho máximo da sequência de saída seja atingido.

## Treinamento com *Teacher Forcing*

Em geral, o treinamento de redes neurais ligadas deve ser *end-to-end*, isto é, o *encoder* e o *decoder* são otimizados conjuntamente, pois isso permite que os dois componentes aprendam representações e dependências temporais de maneira integrada. Essa abordagem é crucial para que o modelo capture efetivamente informações importantes na sequência de entrada e as utilize de maneira significativa ao gerar a sequência de saída correspondente. O treinamento *end-to-end* facilita a aprendizagem de representações intermediárias que são mais úteis para a tarefa específica em questão e contribui para a coerência e alinhamento adequado entre o *encoder* e o *decoder*, garantindo uma melhor generalização para dados não vistos durante a fase de inferência.

Além disso, há uma particularidade interessante no treinamento do *decoder*. O treinamento trivial desse componente seria de forma sequencial utilizando a mesma arquitetura apresentada acima, isto é, utilizando o *output* anterior como *input* para o próximo passo temporal. No entanto, essa abordagem pode resultar em alguns problemas.

Erros de previsão em passos temporais anteriores podem se acumular e impactar negativamente a geração subsequente da sequência. Como cada previsão é condicional nas previsões anteriores, pequenos erros podem se propagar e se tornar mais significativos ao longo do tempo.

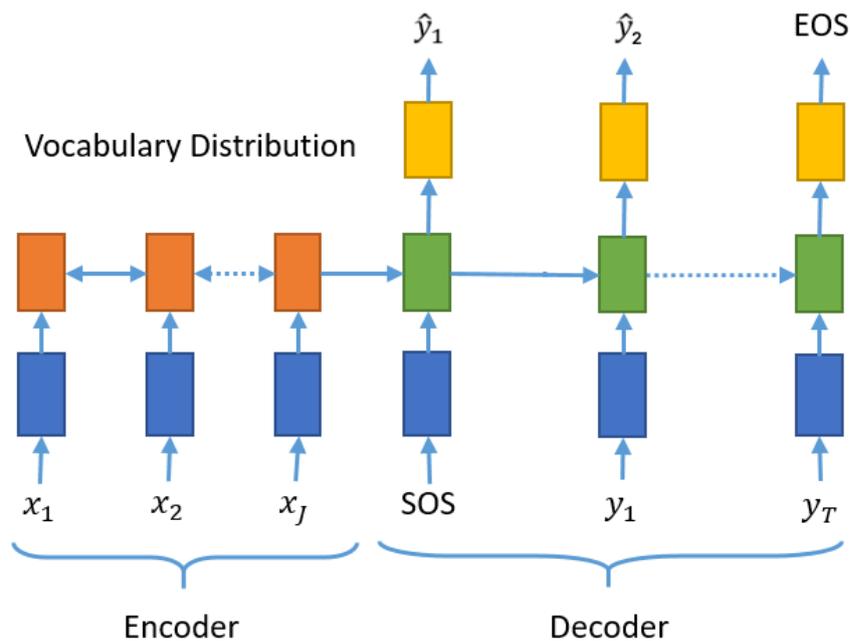
Consequentemente, o treinamento sequencial pode resultar em convergência mais lenta, pois o modelo pode ter dificuldade em ajustar eficientemente seus parâmetros devido

a essa propagação de erros. Em especial, as sequências longas podem ser um desafio maior, pois as previsões feitas no início da sequência podem ter um impacto substancial nas previsões posteriores.

Além disso, o treinamento sequencial pode levar a instabilidades no aprendizado, especialmente em tarefas complexas, devido à acumulação de erros e à sensibilidade a variações na sequência de entrada.

O resultado seria um treinamento muito demorado e evidentemente o resultado será proporcionalmente ruim comparado ao tempo de treinamento.

Uma abordagem muito utilizada para facilitar a convergência, aumentar a estabilidade e reduzir o tempo e processamento durante o treinamento é o *Teacher Forcing*. Essa estratégia treina as redes neurais recorrentes utilizando a verdade fundamental (rótulo) como entrada. Em outras palavras, o *decoder* recebe como entrada a sequência de saída esperada, em vez da saída gerada pelo *decoder* no passo temporal anterior. Essa abordagem permite que o modelo aprenda mais rapidamente e reduza a propagação de erros, pois o *decoder* recebe informações mais precisas e confiáveis durante o treinamento.



**Figura 2.13:** Treinamento do decoder com *Teacher Forcing*.

Note que as previsões, representados  $\hat{y}_i$ , estão sendo gerados e são usados para calcular o erro da rede de acordo com os pesos definidos. No entanto, as previsões ( $\hat{y}_i$ ) não estão sendo utilizadas como *input* para o próximo passo temporal, pelo contrário o *decoder* está recebendo a sequência de saída esperada ( $y_i$ ) como *input*. Essa característica parece permitir que as etapas sejam treinadas de forma simultânea, mas infelizmente, embora não exista uma dependência entre o *output* das camadas, isso não é possível devido à natureza sequencial inerente das redes neurais recorrentes, que modifica seu estado interno a cada iteração.

Por último, cabe ressaltar que o *Teacher Forcing* é utilizado apenas durante o treinamento, pois durante a fase de inferência o *decoder* não tem acesso à sequência de saída esperada. Nesse caso, o *decoder* recebe como entrada a saída gerada no passo temporal anterior. No entanto, como nesse momento o volume de dados é muito menor, isso não é um problema e seria mais difícil sobrecarregar o processamento.

## Seq2Seq

O termo "Seq2Seq" é frequentemente usado de forma intercambiável com "*Encoder-Decoder*". No entanto, apesar dos termos estarem intimamente relacionadas, não são sinônimos (BAHDANAU *et al.*, 2014). Em essência, Seq2Seq é uma abreviação de "*sequence-to-sequence*" e define uma classe especial de problemas de modelagem de sequência nos quais tanto a entrada quanto a saída são sequências. Embora modelos *Encoder-Decoder* foram originalmente desenvolvidos para resolver tais problemas Seq2Seq, existem outras aplicações com "*Image-to-Image Translation*" (traduzir imagens para outras imagens) e "*Image Style Transfer*" (transferir o estilo de uma imagem para outra).

## Mecanismos de Atenção

Embora as LSTMs (Long Short-Term Memory) tenham trazido melhorias significativas em lidar com o problema do gradiente explodindo e o problema de dependências de longo prazo em redes neurais recorrentes, ainda existem desafios quando se trata de lidar com sequências muito longas ou com alinhamentos complexos entre sequências de entrada e saída.

Em sequências muito longas, as LSTMs podem enfrentar dificuldades em manter informações relevantes ao longo de muitos passos temporais. Quando se trata de aprender alinhamentos não lineares entre diferentes partes de sequências, as LSTMs também podem não ser tão eficazes.

Na arquitetura *encoder-decoder* básica, ao desenrolar os LSTMs, a sentença de entrada inteira é comprimida em um único vetor de contexto. Isso funciona bem para frases curtas, mas com um vocabulário de entrada maior com milhares de palavras e frases mais longas palavras do início da frase poderiam ser esquecidas.

Por isso, os mecanismos de atenção em modelos *encoder-decoder*, especialmente em arquiteturas Seq2Seq, foram introduzidos para melhorar a capacidade do modelo de lidar com dependências temporais de longo prazo e alinhamentos irregulares entre sequências de entrada e saída.

A etapa de *encoding* permanece igual, a mudança vem realmente na etapa de *decoding*. A ideia principal dos mecanismos de atenção é adicionar um conjunto de novos caminhos do *decoder* para o *encoder*, um para cada valor de entrada. Essa abordagem permite que cada passo do *decoder* possa acessar diretamente o estado interno de cada passo do processo de *encoding*. Através do treinamento, o *decoder* poderia aprender o relacionamento entre os passos de *encoding* e *decoding*, identificando quais passos do *encoder* são mais relevantes para cada passo do *decoder*. A partir dessa análise, cada passo do *decoder* pode focar nos passos mais relevantes do *encoder* para gerar a sequência de saída (ALAMMAR, s.d.).



Figura 2.14: Exemplo de modelo Seq2Seq com attention.

Na imagem acima, o *encoder* (representado em verde) fornece o estado oculto (representado em laranja) de cada passo para o *decoder* (representado em roxo). Assim, o *decoder* tem acesso ao estado oculto gerado por cada passo do *encoder*. Ele usa o estado oculto do passo atual, o estado oculto dos passos do *encoder* de acordo com os pesos de atenção e o *input* para gerar o *output*.

Com a implementação de mecanismos de atenção, as LSTMs se tornam desnecessárias, pois os problemas de *vanishing gradient*, *exploding gradient* e de dependências de longo prazo são resolvidos pelos mecanismos de atenção.

Existem diferentes tipos de mecanismos de atenção. Introduzido por Dzmitry Bahdanau e colaboradores em 2014 (BAHDANAU *et al.*, 2014), a *Bahdanau Attention* ("Atenção Aditiva") calcula uma pontuação de atenção para cada posição na sequência de entrada em relação à posição atual na sequência de saída. Ele permite um ajuste dinâmico do foco nas diferentes partes da sequência de entrada durante a geração da sequência de saída. Por outro lado, o *Luong Attention* ("Atenção Multiplicativa") proposto por Minh-Thang Luong e colaboradores utiliza uma abordagem multiplicativa para calcular as pontuações de atenção.

Este documento vai se concentrar na *Bahdanau Attention*. Apesar de ser menos eficiente, essa abordagem é mais flexível para calcular pesos de atenção, permitindo um ajuste dinâmico do foco em diferentes partes da sequência de entrada para cada elemento gerado na sequência de saída, crucial para lidar com informações relevantes que podem estar dispersas em diferentes partes da entrada. O mecanismo também permite ao modelo aprender alinhamentos não lineares entre as posições na sequência de entrada e saída, o que é valioso para capturar relações complexas em dados sequenciais. Por último, ele fornece uma interpretabilidade mais clara, pois explicitamente destaca quais partes da sequência de entrada são mais relevantes para cada passo na geração da sequência de saída.

Como descrito anteriormente, a arquitetura do *encoder* continua a mesma. Para cada passo do *decoder* porém, é calculada uma pontuação de atenção em relação ao estado oculto de cada passo do *encoder*. Essa pontuação também é chamada de *Similarity Score*, ou pontuação de similaridade, e indica quão bem uma posição na sequência de entrada (*encoder*) correspondem à posição atual na sequência de saída (*decoder*). Essa pontuação é calculada através de uma função de similaridade, que pode ser uma função de similaridade de ponto, uma função de similaridade de cosseno ou uma função de similaridade de atenção. Assim como o resto da arquitetura, a rede aprende a gerar boas pontuações de atenção

durante o treinamento, começando com uma performance muito ruim e melhorando incrementalmente a cada época.

Note que a matriz de atenção é diferente para cada passo. Isso permite que o modelo ajuste dinamicamente seu foco nas diferentes partes da sequência de entrada para cada elemento gerado na sequência de saída. A matriz é calculada a cada passo de acordo com o treinamento e consequentemente aprendizado da rede neural, isto é, a matriz de atenção não é aprendida nem fixa, mas sim a rede neural aprende a calcular a matriz de atenção.

Em seguida, as pontuações são normalizadas usando a função *softmax*, transformando-as em pesos de atenção. Isso garante que a distribuição de atenção seja uma distribuição de probabilidade, indicando a importância relativa de cada posição na sequência de entrada para a posição atual no *decoder*. A soma dos pesos de atenção é igual a 1 e cada peso de atenção é um número entre 0 e 1, representando o percentual de importância entre o par: passo do *encoder* e passo do *decoder*, em outras palavras, entre uma palavra da entrada e a palavra que vai ser gerada.

O contexto ponderado, que representa de forma contextualizada as partes mais importantes da sequência de entrada para o passo temporal atual do decoder, é calculado somando os estados ocultos do *encoder* ponderados pelos pesos de atenção. Esse contexto representa uma combinação linear dos estados ocultos do *encoder*, destacando as partes mais relevantes da sequência de entrada para a posição atual no *decoder*. O contexto ponderado é concatenado com o estado oculto atual do *decoder*, que cria um vetor combinado que incorpora informações da sequência de entrada relevantes para a geração da próxima saída.

O vetor combinado é então utilizado para gerar a saída no próximo passo temporal do *decoder*.

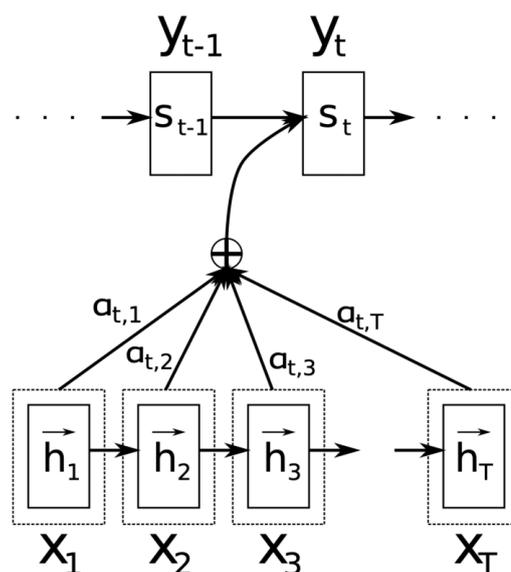


Figura 2.15: Arquitetura do modelo Bahdanau Attention.

Note que a imagem acima representa a arquitetura do modelo Bahdanau Attention. Os valores  $X_i$  representam os *inputs* e os vetores  $h_i$  representam os estados ocultos de cada passo do *encoder*. Os escalares  $a_{t,i}$  juntos formam a matriz de atenção, que representa a importância relativa de cada posição na sequência de entrada para a posição atual no *decoder*.

No modelo *Bahdanau Attention*, cada vetor de estado oculto  $h_i$  é somado ao escalar de atenção correspondente  $a_{t,i}$  formando o contexto ponderado, que por sua vez é concatenado com o estado oculto atual do *decoder*  $s_t$  para gerar o vetor combinado  $c_t$ , que é usado para gerar a saída  $y_t$ .

## Transformers

A cada seção desse artigo, o modelo de *chatbot* apresentado se torna mais complexo e poderoso, representando um salto em performance e capacidade. A introdução da arquitetura *Transformer*, no entanto, é de longe o maior salto na tecnologia de *chatbots* até hoje. Concebida no meio acadêmico em 2017 (VASWANI *et al.*, 2017) e lançada ao mercado em 2022 através do ChatGPT, a arquitetura *Transformer* revolucionou a área de processamento de linguagem natural e impressionou o mundo todo com a sua capacidade.

“O ChatGPT é, simplesmente, o melhor chatbot de inteligência artificial já lançado para ao público. Durante a maior parte da última década, os chatbots de IA têm sido terríveis, com resultados bons apenas se você escolher a dedo as melhores respostas do bot e descartar o resto. Nos últimos anos, alguns recursos de IA se destacaram ao realizar tarefas específicas e bem definidas. Mas o ChatGPT parece diferente. Mais inteligente. Mais peculiar. Mais flexível. Os primeiros usuários falam dele surpresos e em termos grandiosos, como se fosse uma mistura de software e magia, um supercérebro linguístico altamente capaz. Ainda estou tentando compreender o fato de que o ChatGPT - um chatbot que algumas pessoas pensam que pode tornar o Google obsoleto - já está sendo comparado ao iPhone em termos de seu impacto potencial na sociedade.”

— Kevin Roose - The New York Times

“ChatGPT é um desses raros momentos na tecnologia em que você vislumbra como tudo será diferente daqui para frente.”

— Aaron Levie - CEO of Box

Em apenas 5 dias, a plataforma já contava com mais de um milhão de pessoas e, dois meses após o seu lançamento, o serviço já tinha ultrapassado 100 milhões de usuários ativos, se tornando o software com a adoção mais rápida da história. Ele redefiniu as expectativas em relação à capacidade de sistemas de conversação, demonstrando uma habilidade notável em compreender contextos, gerar respostas coesas e proporcionar experiências de usuário mais autênticas.

Grande parte da popularidade e do sucesso atribuído ao ChatGPT é consequência do desenvolvimeto dessa nova arquitetura de rede neural. Mas o que faz a arquitetura *Transformer* ser tão especial?

## Avanços Exponenciais

Introduzida no artigo "Attention is All You Need" por Vaswani et al. em 2017 (VASWANI *et al.*, 2017), o *Transformer* revolucionou a área de processamento de linguagem natural, introduzindo um mecanismo de atenção mais eficiente e uma abordagem sem o uso de camadas recorrentes ou convolucionais.

Uma das características mais distintivas dos *Transformers* é a atenção global. Em vez de depender apenas do contexto local, como em arquiteturas Seq2Seq, os *Transformers* podem acessar informações de toda a sequência de entrada simultaneamente. Isso permite lidar com dependências de longo alcance de maneira mais eficiente. A atenção na arquitetura Seq2Seq melhorou a capacidade de lidar com dependências locais, mas ainda enfrentava desafios com dependências de longo alcance, já que a atenção era calculada com base em uma janela local. Com essa nova abordagem, o processamento de linguagem natural chega a um novo patamar.

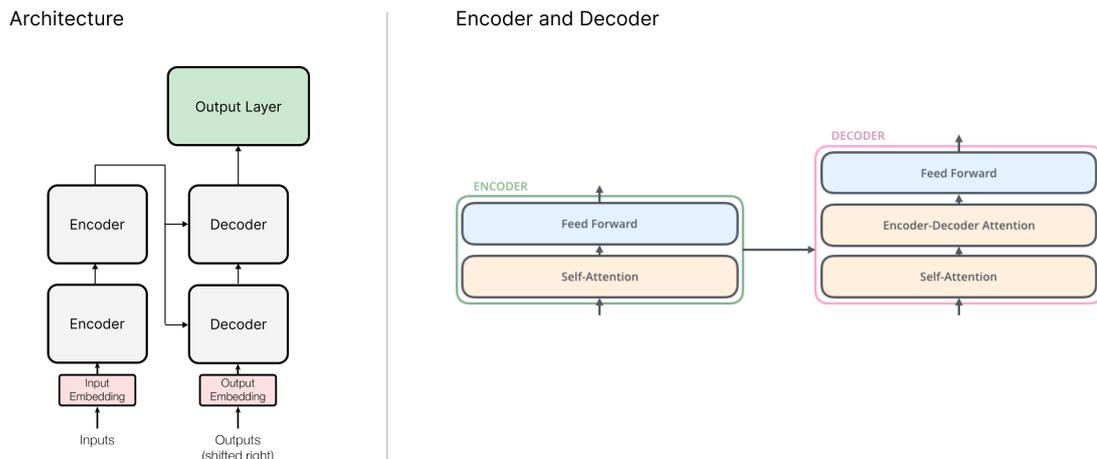
A arquitetura *transformer* também traz consigo a capacidade de processamento paralelo eficiente. Enquanto a arquitetura *seq2seq* depende da computação sequencial de cada passo de tempo, os *Transformers* podem processar todos os elementos da sequência simultaneamente, acelerando significativamente o treinamento e, em um nível um menor, a inferência.

Além disso, o tempo de computação em *Transformers* escala linearmente com o comprimento da sequência, o que é uma vantagem em comparação com RNNs, onde o tempo de computação podia crescer exponencialmente, especialmente para sequencias mais longas. O impacto disso é enorme: não é apenas uma questão de diminuir o tempo de espera do usuário, mas sim possibilitar aplicações muito mais complexas, que antes não eram possíveis e nem pensadas por conta da complexidade exponencial.

Essas são apenas algumas das diversas vantagens da nova arquitetura. Pode-se citar também a facilidade de treinamento em hardware especializado, como GPUs e TPUs, por não possuírem dependências temporais como as RNNs. A capacidade dos modelos serem pré-treinados em grandes corpora de texto e, em seguida, ajustados para tarefas específicas. A transferência de conhecimento. A versatilidade de aplicação em diferentes tarefas, como visão computacional e áudio. A modularidade da arquitetura, podendo ser adaptada para várias tarefas sem a necessidade de alterações significativas na arquitetura subjacente.

## Organização Básica

A arquitetura *Transformer*, semelhante ao modelo *encoder-decoder*, é composto por dois blocos principais: o bloco de *encoding* e o bloco de *decoding*, além das camadas de *embedding* e o bloco de saída (ALAMMAR, 2018).



**Figura 2.16:** Organização básica da arquitetura do modelo Transformer.

As camadas de *embedding* (*input embedding* e *output embedding*) utilizam algoritmos semelhantes ao *Word2Vec*, visto anteriormente, para transformar as palavras em um vetor de números reais. Esses vetores são utilizados como *input* para o bloco de *encoding* e *decoding*. As palavras a serem usadas no *input embedding* são as palavras da frase de entrada, enquanto as palavras do *output embedding* são as palavras que são geradas pelo próprio modelo, com uma potencial otimização que será discutida em breve.

O bloco de *decoding* é responsável por gerar uma representação vetorial a partir dos *embeddings* recebidos como *input*. Essa representação vetorial é utilizada como *input* para o bloco de *encoding*, o qual é responsável por gerar uma sequência *embeddings* relacionados ao *input*. Esses *embeddings* são transformados pelo bloco de saída, que é responsável por transformar os *embeddings* em palavras.

Note que o bloco de *decoding* e *encoding* possuem respectivamente diversos *encoders* e *decoders*. A imagem explicita dois *encoders* e dois *decoders*, mas na prática o número de *encoders* e *decoders* pode variar de acordo com a complexidade do modelo.

O *encoder*, por sua vez, é composto por dois componentes: uma camada de *Self-Attention*, que será explicada em detalhes em breve, e uma Rede Neural *Feed Forward*. A camada de *Self-Attention* identifica a relação das palavras em uma frase, isto é, a importância de cada uma das palavras em relação às outras, e a Rede Neural *Feed Forward* realiza transformações não lineares nas representações das palavras, permitindo o aprendizado de representações mais complexas e não lineares das palavras na sequência de entrada.

Vale acrescentar que, na camada de *Self-Attention*, o vetor real com *attention* resultante de cada *embedding* depende das outras palavras do texto de entrada. Já na Rede Neural *Feed Forward*, cada palavra percorre o seu próprio caminho e é processada de forma independente, possibilitando o processamento das palavras em paralelo.

O *decoder* se assemelha ao *encoder*, mas possui uma camada adicional de *attention*, chamada de *encoder-decoder attention*. Essa camada é responsável por ajudar o *decoder* a focar nas partes relevantes da sequência de entrada, similar ao que a camada de *Self-Attention* faz no *encoder*. Enquanto a camada *Self-Attention* no *decoder* permite que o *decoder*

olhe para as palavras relevantes na sequência de saída, a camada *encoder-decoder attention* olha para as palavras relevantes da entrada. O cruzamento dessas duas informações, isto é, a importância das palavras na sequência de entrada e na sequência de saída, maximiza a acurácia da sequência gerada como saída.

Diferentemente de uma Rede Neural Recorrente "*many-to-many*", as palavras da frase de *input* são processadas simultaneamente. Porém, semelhante a abordagem mencionada, a partir do momento que o *decoder* começa a gerar a sequência de saída, as palavras são geradas sequencialmente, uma por vez, sem receber novos inputs até que o token `<eos>` ou "*End of Sentence*" é gerado.

## Self-Attention

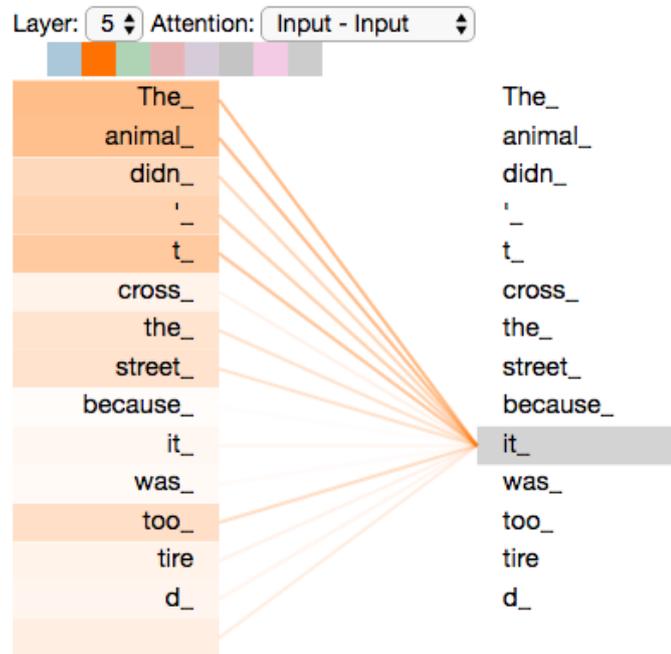
Para que a arquitetura possa obter a atenção global, mencionada anteriormente, foi necessário o desenvolvimento de um novo mecanismo de atenção, chamado "*Self-Attention*" ou "*Scaled Dot-Product Attention*". Essa abordagem permite que cada posição em uma sequência se relacione com todas as outras posições, capturando relações globais.

O objetivo do *Self-Attention* é, acessando as palavras de todo o texto, identificar a relação das palavras em uma frase, isto é, a importância de cada uma das palavras em relação às outras e a partir disso gerar uma representação vetorial para cada palavra que incorpore as outras palavras relacionadas.

Em primeiro lugar, são gerados três vetores para cada palavra: *Query vector*, *Key vector* e *Value vector*. Esses vetores são criados multiplicando o *input embedding* por três matrizes  $W^Q$ ,  $W^K$  e  $W^V$  de pesos diferentes. Essas matrizes são treinadas juntamente com toda a arquitetura buscando minimizar a função de custo dos *outputs* finais. Os vetores gerados, que em geral têm dimensão muito menor do que os vetores do *input*, são abstrações que ajudam a definir os pesos de atenção (ALAMMAR, 2018).

O próximo passo é calcular o relacionamento entre as palavras, isto é, a importância de cada uma das outras palavras para cada outra palavra da frase. Para isso, é calculado o produto escalar entre o *Query vector* da palavra atual e o *Key vector* de cada outra palavra. Esse cálculo resulta em um vetor de pesos de atenção, em outras palavras, qual é a importância de cada palavra em relação à palavra atual. Esses pesos vão determinar quanto de cada palavra será *encoded* junto com a palavra atual. Esses pesos são calculados através de uma função de similaridade, que pode ser uma função de similaridade de ponto, uma função de similaridade de cosseno ou uma função de similaridade de atenção. Assim como o resto da rede, a rede aprende a gerar boas pontuações de atenção durante o treinamento, começando com uma performance muito ruim e melhorando incrementalmente a cada época (ALAMMAR, 2018).

Em seguida, para evitar que os valores fiquem muito grandes e causem problemas de instabilidade no treinamento, os valores são divididos por  $\sqrt{d_k}$ , onde  $d_k$  é a dimensão do *Key vector*, e depois passam por uma função *softmax*. A normalização garante que os valores estejam entre 0 e 1 e que a soma dos valores seja 1, assim representam o percentual de atenção da palavra atual com cada uma das outras.



**Figura 2.17:** Exemplo dos percentuais de atenção de "it" para as outras palavras da frase. Cores mais escuras indicam maior percentual de atenção.

Finalmente, os pesos de atenção são multiplicados pelo *Value vector* de cada palavra, resultando em um vetor de *outputs* da palavra atual. Esse vetor é a representação vetorial da palavra que incorpora as outras palavras relacionadas de acordo com o percentual de atenção calculado.

Em geral, apesar do que foi explicado acima para facilitar o entendimento, o *Self-Attention* não utiliza vetores de *inputs* e *outputs* para cada palavra, mas sim matrizes. Cada linha da matriz, corresponde as informações de uma das palavras da frase. Semelhantemente, não são criados vetores *Query*, *Key* e *Value* para cada palavra, mas sim matrizes. Isso simplifica o cálculo do *output* de acordo com a fórmula abaixo:

$$Output = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \quad (2.1)$$

Comparando o *Self-Attention* com o mecanismo de atenção utilizada na arquitetura *Seq2Seq*, o *Soft Attention*, percebe-se claramente a vantagem da nova abordagem. O mecanismo *Soft Attention* é computacionalmente caro, pois requer o cálculo de uma pontuação de atenção para cada par de palavras na sequência de entrada e saída, resulta em uma complexidade quadrática em relação ao comprimento da sequência. Já o mecanismo *Self-Attention* requer o cálculo de uma pontuação de atenção apenas para cada par de palavras na sequência de entrada. Além de ser mais performático, o *Self-Attention* também é mais poderoso ao captar relações globais e não apenas locais.

## Positional Encoding

O mecanismo de *Self-Attention* é poderoso para capturar relações complexas entre palavras em uma sequência. No entanto, por padrão, ele não possui uma noção intrínseca de ordem entre as palavras, o que é crucial para entender a estrutura e o significado de uma sentença.

Sem essa informação, o mecanismo se comporta de maneira semelhante a um modelo *Bag-of-Words*, que trata a sequência como um conjunto não ordenado de palavras. É uma técnica simples usada para representar textos em processamento de linguagem natural, que mantém mantendo a contagem da ocorrência de cada palavra sem se preocupar com a posição da mesma.

Essa abordagem traz uma série de desvantagens: palavras idênticas em diferentes posições podem ter significados diferentes ou contribuir de maneira única para o contexto da frase, e, em textos mais longos, as dependências entre palavras podem se estender por várias posições na sequência e o modelo pode ter dificuldade em capturar relações de longo alcance sem informação posicional. Não seria considerada a estrutura gramatical nem a ordem das palavras na frase.

Por isso, é necessário adicionar informação posicional ao mecanismo de *Self-Attention*, essa técnica é chamada de *Positional Encoding*. Há diferentes implementações, entre elas estão a intercalação do vetor de posição e a adição do vetor de posição com o vetor de *input*.

Em ambos os casos, o vetor de posição, que representa a posição da palavra na frase a partir de um vetor de números reais, modifica o vetor de *input* gerando uma separação proporcional nos vetores de *input* gerados pelo self-attention. A partir do treinamento do modelo, a rede neural aprende que palavras mais próximas em geral têm maior relação do que palavras mais distantes e começa a considerar a ordem das palavras durante o cálculo dos percentuais de atenção.

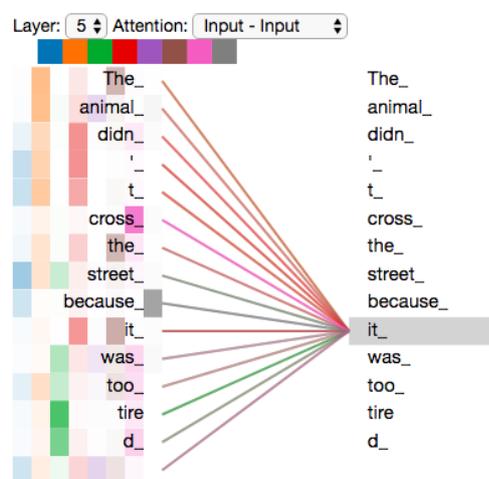
Na primeira abordagem, o vetor de posição é intercalado com o vetor de *input*, isto é, ficam em células separadas. A segunda abordagem é a mais utilizada, por ser mais simples, mais eficiente e pela vantagem se adaptar a comprimentos de sequências muito maiores. Nessa abordagem, o vetor de posição, que segue um padrão aprendido pelo modelo para minimizar a função de custo, aprendem é somado ao vetor de *input*. Essa nova informação ajuda a determinar a posição de cada palavra ou a distância entre diferentes palavras na sequência, fornecendo distâncias significativas entre os vetores de *embedding* quando eles são projetados nos vetores *Query*, *Key* e *Value* (ALAMMAR, 2018).

## Multi-Headed Attention

Outra limitação do mecanismo de *Self-Attention* padrão é a falta de diversidade no mecanismo de atenção. O modelo está limitado a aprender uma única perspectiva ou interpretação das relações entre palavras em uma sequência, que restringe a compreensão restrita das nuances no texto. Consequentemente, ele pode ter dificuldade em capturar relações complexas ao focar demais em certas partes da sequência (chamado de "risco de saturação").

Esse foco único faz com que o modelo esteja mais suscetível a ficar preso em mínimos locais durante o treinamento. A *Multi-Headed Attention* (ou "Atenção Multi-Cabeça") é uma técnica que visa capturar várias perspectivas ou tipos de relações entre palavras em uma sequência, aumentando as representações e interpretações da atenção.

Em vez de ter uma única camada de *Self-Attention*, são utilizadas várias camadas (chamadas de "cabeças") de *Self-Attention* que operam em paralelo. Cada cabeça opera como um mecanismo de atenção independente, focalizando em diferentes combinações de palavras e relacionamentos na sequência de entrada, e aprende uma representação diferente das relações entre palavras. Ao permitir a diversidade de representações, os modelos de linguagem têm maior probabilidade de compreender e lidar com textos de complexidades maiores e fogem dos mínimos locais por investigarem mais direções.



**Figura 2.18:** Representação das diferentes interpretações no Multi-Headed Attention.

Para cada palavra, são calculadas atenções individuais independentes em cada cabeça. Cada uma com seus próprios conjuntos de vetores de *Query*, *Key* e *Value*, que são inicializados com pesos diferentes e aprende pesos independentes durante o treinamento. O resultado são cabeças especializadas em identificar padrões específicos treinadas até seus mínimo locais.

Após calcular as atenções individuais para cada cabeça, os resultados são combinados de maneira ponderada usando uma nova matriz de pesos aprendida pela rede. Essa atenção *multi-headed* gera uma representação mais rica e diversificada, que considera perspectivas e interpretações diferentes para cada palavra.

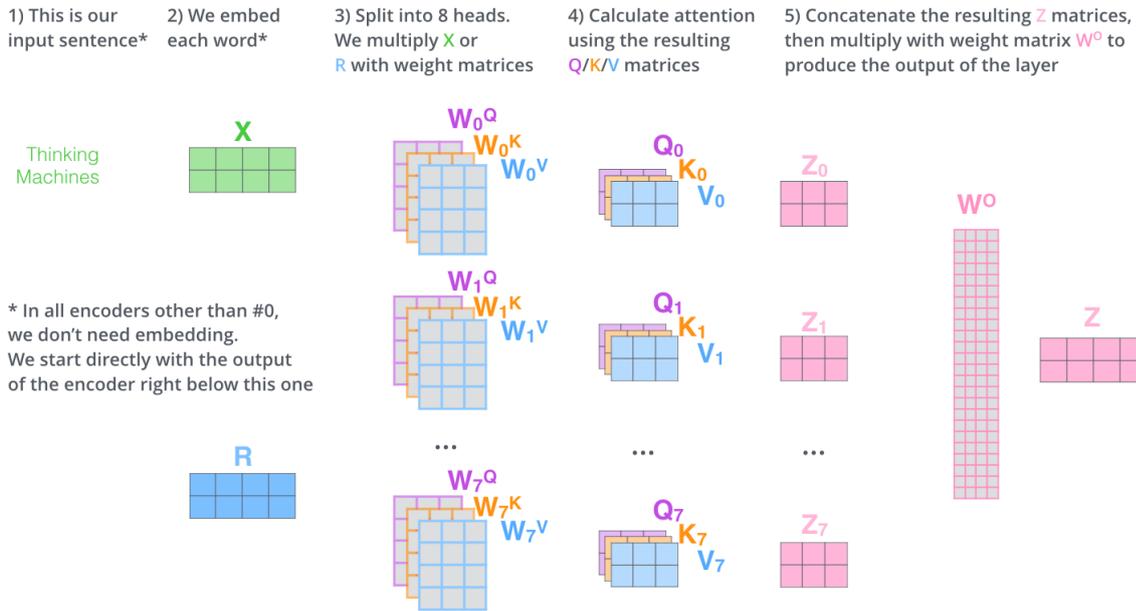


Figura 2.19: Processamento completo de atenção multi-headed.

### Arquitetura Completa

Com uma compreensão sólida dos blocos individuais descritos neste capítulo, faltam apenas alguns blocos que completarão a arquitetura *Transformer*.

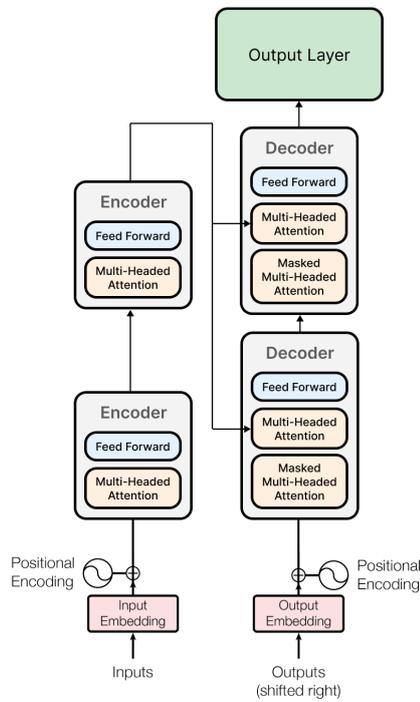


Figura 2.20: Arquitetura Transformer simplificada.

Um desses blocos é a *Masked Multi-Headed Attention* presente nos *decoders*. Como apresentado anteriormente, o *decoder* tem três componentes internos: *Self-Attention*, *Encoder-Decoder Attention* e *Feed Forward*. O *Encoder-Decoder Attention* é um *Multi-Headed Self-Attention* que ajuda o decodificador a focar em lugares apropriados na sequência de entrada, e o *Feed Forward* tem a mesma estrutura no *encoder*. O *Self-Attention*, no entanto, é uma variação da *Multi-Headed Attention* que permite que o modelo olhe apenas para as palavras anteriores na sequência de entrada. Essa variação só permite atenção a posições anteriores na sequência de saída ao mascarar as posições seguintes, definindo-as como  $-\infty$  antes da etapa *softmax*. Isso garante que as palavras geradas sejam geradas apenas com base nas palavras anteriores, evitando que o modelo "trapaceie" e olhe para as palavras seguintes, porque isso só seria possível durante o treinamento, mas não durante a inferência (ALAMMAR, 2018).

Outros dois blocos importantes são a conexão residual e a camada de normalização, presentes depois de cada *Multi-Headed Attention*, *Masked Multi-Headed Attention* e *Feed Forward* para melhorar a estabilidade do treinamento e o problema de desaparecimento de gradiente em redes profundas. A conexão residual é uma conexão direta entre a entrada e a saída de uma camada. Essa conexão permite que o gradiente flua diretamente através da camada sem passar por transformações não lineares, o que ajuda a evitar o problema de desaparecimento de gradiente. Já a camada de normalização é responsável por normalizar os vetores de *input* para que tenham média 0 e variância 1, o que ajuda a estabilizar o treinamento e garantir resultados mais previsíveis.

Por último, a camada de saída é responsável por transformar os *embeddings* em palavras. Essa camada é composta por uma camada linear e uma camada *softmax*. A camada linear é uma rede neural totalmente conectada que projeta o vetor de *input* em um vetor com uma posição para cada palavra no vocabulário. A camada *softmax* transforma os valores do vetor em probabilidades, isto é, valores entre 0 e 1 que somam 1. Assim, o resultado é um vetor onde cada posição representa a probabilidade da palavra ser a próxima na sequência.

Semelhantemente ao modelo *encoder-decoder*, existem duas abordagens para escolher a próxima palavra a partir desse vetor: *argmax* e *sample*. Na abordagem *argmax*, a palavra com maior probabilidade é selecionada e é utilizada como *input* para o próximo passo temporal, que gera uma rede mais estável com resultado previsíveis. Na abordagem *sample*, por outro lado, a próxima palavra é amostrada de acordo com a distribuição de probabilidade sobre o vocabulário, gerando resultados com maior variabilidade, o que em geral é bem-vindo.

Assim, a arquitetura *Transformer* está completa e pode ser observada abaixo, de acordo com cada bloco explicado neste capítulo (ALAMMAR, 2018).

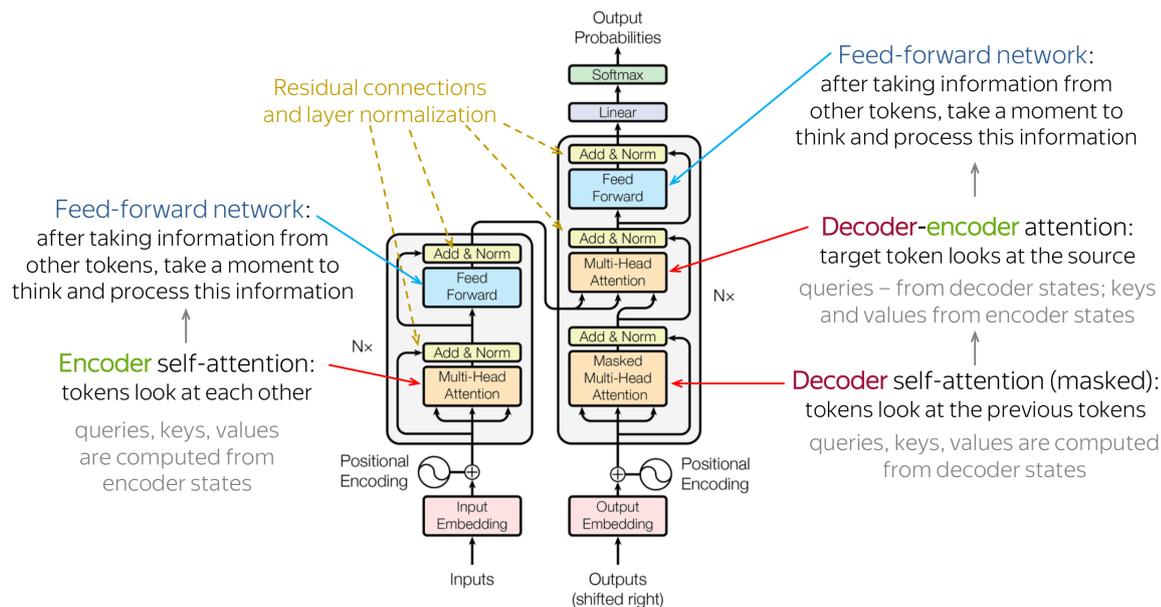


Figura 2.21: Arquitetura Transformer.

## Exemplo de Processamento

Tendo como base a arquitetura *Transformer* e seus blocos internos, analisar um exemplo de predição da arquitetura pode facilitar o entendimento e a visualização do funcionamento da arquitetura. Considerando um contexto de tradução do inglês para português de uma frase específica. Em primeiro lugar, cada palavra será transformada em um *embedding* na camada de *input embedding* utilizando um algoritmo similar ao *Word2Vec* e depois adiciona-se a informação posicional das palavras com o *Positional Encoding*. Em seguida, o vetor de *embeddings* é passado para o primeiro bloco de *encoder*, que gera uma nova representação vetorial normalizada de cada palavra com as informações das outras palavras relacionadas. Esse processo é repetido algumas vezes para cada *encoder*.

O *output* do último *encoder* é então transformado em um conjunto de vetores de atenção  $K$  e  $V$  e enviados para os *decoders*. Esses vetores de atenção auxiliam os *decoders* a focarem nos lugares mais importantes da sequência de entrada. Cada *decoder* também tem acesso ao vetor de *embeddings* da sequência de saída que já foi gerada pelo próprio modelo, que já tem a informação posicional e foi alterada pelos *decoders* anteriores.

O *decoder* então gera uma nova representação vetorial normalizada de cada palavra com as informações das outras palavras relacionadas. Esse processo é repetido algumas vezes para cada *decoder*. O *output* do último *decoder* é então transformado em um conjunto de vetores de atenção  $K$  e  $V$  e enviados para a camada de saída, que transforma os *embeddings* em palavras. A camada constrói um vetor onde cada posição representa a probabilidade da palavra ser a próxima na sequência e seleciona uma das palavras de acordo com a distribuição de probabilidades. A palavra selecionada é utilizada como *input* para o próximo passo temporal do bloco de *decoding*.

**Figura 2.22:** Exemplo do processo de predição da Arquitetura Transformer.

## Paralelismo

Como explicado anteriormente, a arquitetura *Transformer* é altamente paralelizável durante o treinamento, pois não possui dependências temporais entre as palavras. Enquanto a arquitetura *seq2seq* depende da computação sequencial de cada passo de tempo, os *Transformers* podem processar todos os elementos da sequência simultaneamente, acelerando significativamente o treinamento e a inferência.

Conhecendo a arquitetura, é possível identificar os pontos de paralelismo. Em especial, após o processamento do bloco de *encoding* tendo gerado os vetores  $K$  e  $V$ , todos os *decoders* podem ser processados em paralelo usando o *Teacher Forcing* explicado anteriormente. Isso é possível pois, diferentemente de RNN, a arquitetura não altera estados internos, dependendo apenas dos vetores  $K$  e  $V$  e os *outputs*, nesse caso os *outputs* esperados.

Além disso, a arquitetura pode ser paralelizada em outras partes também. Por exemplo, as diferentes cabeças de atenção operam independentemente umas das outras, portanto, o cálculo de atenção para diferentes posições pode ser paralelizado. A camada de *Feed Forward* também não possui dependências, e assim os vários caminhos podem ser executados em paralelo.

No entanto, existem áreas que não podem ser paralelizadas. O *Positional Encoding*, *Self-Attention* de cada cabeça e a passagem de informações entre cada camada de *encoder* ou *decoder* tem operações sequenciais que não podem ser paralelizadas.

## Chatbots com Transformers

A arquitetura *transformer* tem diversas aplicações. Ela é muito eficaz em tarefas de tradução, sumarização, reconhecimento de entidades nomeadas, classificação de texto, visão computacional, química computacional, composição de música, entre outras. Uma das

aplicações mais interessantes, no entanto, é o desenvolvimento de *chatbots*. O mecanismo de atenção presente nos *Transformers* permite que o *chatbot* identifique e mantenha seu foco nas partes relevantes da conversa, além de aprender as representações semânticas ricas das frases.

## Large Language Models

Antes de desenvolver um *chatbot* em si, é necessário treinar um *Large Language Models* (LLM). LLMs são modelos de inteligência artificial de processamento de linguagem natural, sendo treinados para compreender e gerar linguagem humana de forma coerente gramaticalmente e semanticamente (ALEC RADFORD e SUTSKEVER, 2018).

Em primeiro lugar, é necessário obter um conjunto de dados abundante de textos, os quais são transformados em vetores de *embeddings* utilizando um algoritmo similar ao Word2Vec. Em seguida, o componente *Decoder* da arquitetura *transformer* é treinado para, dado um conjunto de palavras, prever a próxima palavra da sequência utilizando os dados obtidos. Buscando minimizar a função de custo de acordo com as palavras nos dados de treino, a arquitetura ajusta seus pesos e aprende a construir frases de acordo com a estrutura gramática, relação entre as palavras, a semântica das frases e informações textuais presentes nos dados.

Esse processo de treinamento é extremamente caro e demorado por conta da quantidade de dados e número de parâmetros. No entanto, um dos grandes benefícios das LLMs é a capacidade de portabilidade e de transferência de aprendizado. Esses modelos pré-treinados ajustados para tarefas específicas, desde *chatbots* até a composição de músicas.

## Fine Tuning

A etapa de ajuste fino é chamada de *fine tuning* e consiste do ajuste dos pesos do modelo pré-treinado para a aplicação desejada. No *fine tuning*, o modelo pré-treinado é refinado em um conjunto de dados menor, específico para a tarefa desejada. Durante esse processo, os pesos do modelo são ajustados para otimizar o desempenho na tarefa específica, preservando o conhecimento geral aprendido durante o pré-treinamento, enquanto adapta o modelo para a nova tarefa (OPENAI, 2022).

Primeiramente, busca-se um conjunto de conversas que seja relevante para o contexto do *chatbot*. Os dados, formatados como pares de perguntas e respostas, serão transformados em vetores de *embeddings* utilizando o algoritmo Word2Vec.

Em seguida, inicializa-se o modelo *Transformer* com os pesos do *Decoder* aprendidos pela LLM e começam os treinamentos usando os dados obtidos. A entrada do *Encoder* são as perguntas do conjunto de dados e as respostas são geradas e validadas no *Decoder*. Buscando minimizar a função de custo de acordo com as respostas nos dados de treino, a arquitetura ajusta seus pesos e aprende a construir resposta de acordo com as perguntas que foram feitas.

Por último, o *Reinforcement Learning* pode ser implementado para promover a melhoria contínua do modelo baseado no *feedback* dos usuários. Ao possibilitar a avaliação das respostas, esses dados podem ser considerados no próprio modelo como uma *Reward*

*Function*. Esse processo garante que os pesos serão reajustados para desestimular as respostas mal avaliadas e estimular as respostas bem avaliadas.

## Desafios Futuros

Quando comparada as abordagens anteriores, a arquitetura *transformer* significa um salto enorme para capacidade dos *chatbots*, em especial na fluidez do sistema, preservação de contexto da conversa e produção de respostas complexas de acordo com as nuances da pergunta do usuário. Como foi discutido anteriormente, o grande impacto do ChatGPT e ferramentas similares é fruto dessa arquitetura.

No entanto, o modelo também introduz um grande desafio: as "*hallucinations*". A "*hallucination*" ou "alucinação" se refere a informações incorretas produzidas pelo programa, as quais fogem dos dados de treino.

Essas alucinações acontecem quando o modelo extrapola demais as informações aprendidas durante o treinamento, gerando respostas que não correspondem à realidade. Como cada palavra gerada é escolhida de acordo com a distribuição percentual de probabilidade, o modelo costuma gerar frases corretas tanto gramaticalmente quanto semanticamente, mas nem sempre isso ocorre. Em geral, quanto maior a fluidez da abordagem, menor é o controle sobre a precisão das informações que são geradas.

Com o crescimento da popularidade dos *chatbots*, esse problema tem se intensificado. Usuários não-técnicos não compreendem a arquitetura das soluções que estão utilizando e nem as suas limitações, assim, decidem acreditar cegamente nas respostas dessas ferramentas. Isso levanta a questão: as vantagens da arquitetura *transformer* são suficientes para justificar suas limitações?

Existe um paralelo muito claro entre o surgimento dos *chatbots* com *Transformers* e o lançamento do algoritmo de ranqueamento do Google. Nos primórdios da internet global, as ferramentas de busca da internet indexavam as páginas html manualmente após a verificação das informações por especialistas. Esse processo, apesar de garantir a veracidade dos fatos, limitava o número de informações indexadas. O ranqueamento do Google revolucionou o mercado pela sua capacidade de indexar páginas automaticamente, eliminando a necessidade de verificação manual. A quantidade de informações indexadas aumentou exponencialmente. No entanto, a ausência de verificação das informações gerou um desafio significativo: o risco de desinformação. Apesar desse risco, as pessoas continuam usando a plataforma pela conveniência encontrando outras formas de evitar a desinformação, como identificar a confiabilidade dos sites não confiando em tudo o que leem na internet.

Semelhantemente, os *chatbots* com *Transformers* promovem uma conveniência muito grande para os usuários, mas também apresentam o risco de desinformação através das alucinações. Portanto, é de extrema importância que os usuários estejam cientes desse comportamento e verifiquem as informações em outras fontes mais confiáveis. Além disso, é de igual importância o desenvolvimento de novas técnicas destinadas a mitigar as alucinações.

## Capítulo 3

# Desenvolvimento de Chatbots

Ao longo dos capítulos anteriores, foi estudado em profundidade a implementação de *chatbots* com a arquitetura *transformer*. Desde redes neurais até a arquitetura *transformer* em si, cada seção contribuiu para estabelecer um alicerce sólido. Este capítulo representa a transição para a fase prática do desenvolvimento do trabalho, na qual esses conhecimentos teóricos serão aplicados e materializados, consolidando as sólidas bases teóricas previamente exploradas nos capítulos anteriores.

O estudo teórico previamente apresentado foi acompanhado do desenvolvimento de uma série de projetos, visando aprofundar o conhecimento adquirido e aplicar os conceitos assimilados. Entre esses projetos estão um Detector de Dígitos com rede neural *feedforward*; uma rede neural recorrente para a previsão dos preços das ações utilizando LSTM e GRU; a implementação da Arquitetura *Transformer* com a biblioteca *PyTorch*; e a geração de textos com o *Decoder* da Arquitetura *Transformer*. Esses e outros projetos foram fundamentais na compreensão da arquitetura e seus pré-requisitos. No entanto, esses projetos não serão discutidos neste documento.

Esta seção descreverá o desenvolvimento de *chatbots* especializados. Serão avaliadas diferentes abordagens, identificando os prós e contras e discutindo os cenários ideias para cada implementação. A descrição da metodologia vem a seguir.

## Metodologia

O desenvolvimento de *chatbots* é um campo dinâmico que envolve a escolha cuidadosa da mais adequada abordagem e implementação para garantir uma comunicação com usuários eficiente, fluída e precisa. Neste capítulo, serão apresentados diferentes arquiteturas e abordagens para o desenvolvimento de *chatbots* de objetivo específico, também chamados de *chatbots* especializados. Diferentemente do ChatGPT, que é um *chatbot* generalista, os *chatbots* especializados são programados para responder perguntas sobre um certo domínio de conhecimento baseado nos dados de treino.

Os *chatbots* serão responsáveis por aconselhar startups em seus questionamentos na língua inglesa. Considerando startups desde a ideação até o lançamento e crescimento, respondendo perguntas sobre ideias de negócio, validação, marketing, finanças, entre

outros. Serão analisados três implementações: (1.) *intent chatbot*, (2.) *fine-tuning* de LLMs e (3.) *Assistant API*.

Ao final, essas implementações serão comparadas de acordo com os seguintes parâmetros:

- **Fluidez e Adaptabilidade:** Os *chatbots* devem ser capazes de compreender perguntas complexas e gerar respostas adequadas de acordo com as especificidades da pergunta. Esse parâmetro não avalia a correteza das respostas, mas sim a compreensão da pergunta, assim, as respostas devem demonstrar a compreensão da pergunta.
- **Precisão nas Respostas:** *Chatbots* precisos são aqueles que produzem respostas corretas de acordo com os dados de treino. O sistema pode relacionar diversos dados de treinamento, porém não deve produzir informações falsas.
- **Custo e Tempo de Treinamento:** Esse parâmetro faz referência ao custo de treinamento do sistema. Tanto o investimento financeiro quanto aos requisitos de hardware e tempo serão considerados.

Outros parâmetros, como a continuação das conversas e manutenção do contexto, não serão analisados neste trabalho e podem ser objetos de estudo e aprofundamento no futuro.

## Coleta de Dados

O primeiro passo no desenvolvimento desses chatbots é a coleta de dados. A qualidade dos modelos é um resultado direto da qualidade, abundância e variedade desses dados. Poucos dados e dados com pouca variedade podem gerar um modelo *overfitted*, isto é, um modelo que está muito entrelaçado com os dados de treinamento que não funciona bem com os dados de uso real, ou um modelo muito básico capaz de responder um número pequeno de perguntas. Por último, dados de pouca qualidade geram um modelo de pouca qualidade e muito impreciso.

Os dados foram obtidos de diversas fontes confiáveis e vetadas pela comunidade de *startups* e empreendedorismo. Desde artigos, vídeos, livros, perguntas e respostas, entre outros.

A *YCombinator*, por exemplo, é uma aceleradora de startups de referência mundial. Ela oferece todo o suporte financeiro, contatos e treinamento para as *startups* mais promissoras de todo o mundo. Além de seu programa presencial nos Estados Unidos, a empresa oferece uma gama de conteúdos gratuitos pelo seu blog e canal no YouTube.

As centenas de artigos em seu blog podem ser usadas como base de treinamento. Pensando nisso, foi desenvolvido um *crawler* para o blog da *YCombinator*. *Crawler* é um programa que acessa sites da internet com o objetivo de capturar as informações de forma automática. Além de ler os componentes do site, o programa também formata o artigo com os devidos parágrafos, desconsiderando imagens, propagandas e outras tags desnecessárias.

Os vídeos do YouTube também são uma boa fonte de dados de treino, porém existe uma complexidade um pouco maior por conta do tipo de mídia. Foi necessário desenvolver um *crawler* que acompanha a transcrição do vídeo e fazer um processamento adicional para separar as linhas.

Por último, foi feito um levantamento dos 15 livros mais importantes para startups, como *The Lean Startup* (RIES, 2011), *Zero to One: notes on startups* (BLAKE MASTERS, 2014) e *The Startup Owner's Manual* (BLANK e DORF, 2012). Também foram usados datasets já existentes com perguntas e respostas referentes a empreendedorismo e startups, como *Glavin001/startup-interviews* (W, 2023) e *TuningAI/Startups\_V1* (AI, 2023).

Ao final desse processo, se obtém um grande dataset de texto e um dataset menor de perguntas e respostas. As diferentes implementações necessitam formatos diferentes: o intent chatbot e fine-tuning são treinados com perguntas e respostas e o ChatGPT Assistant com grandes textos.

Assim, com o objetivo de aumentar a quantidade de perguntas e respostas, criamos um script para gerar perguntas e respostas a partir do material de texto obtido. Usando a API do ChatGPT Assistant, solicitamos a geração de centenas de perguntas e respostas com base nos documentos lidos. Apesar dessas perguntas e respostas serem gerados artificialmente, têm boa qualidade e pode ser usado como dados de treino para o intent chat e fine-tuning.

## Implementações

### Intent Chat

A implementação do *chatbot* baseado em *intents* é considerada uma implementação clássica. O programa usa uma lista de assuntos como dados de treino, onde cada assunto é composto por uma lista de perguntas sinônimas e suas respectivas respostas potenciais. O modelo deve ser capaz de identificar o assunto de uma dada pergunta e escolher uma das respostas potenciais.

A lista de perguntas sinônimas desempenha um papel crucial no treinamento do modelo de *chatbot*, capacitando-o a reconhecer e generalizar perguntas relacionadas a um determinado assunto, mesmo quando formuladas de maneiras diversas. Essa abordagem visa enriquecer a capacidade do modelo em compreender a essência do tema, promovendo mais adaptabilidade diante de variações na formulação das perguntas e evitando que os usuários precisem usar palavras e construções de frases específicas para obter uma resposta adequada.

O primeiro passo na construção do modelo é o pré-processamento. Nesta etapa, as perguntas são tokenizadas, isto é, cada palavra ou parte dela é transformada em um *embedding*. Esses *embeddings* carregam algumas características importantes: palavras semanticamente semelhantes têm representações vetoriais próximas e possibilitam as operações vetoriais nos *embeddings*.

Em seguida, selecionamos as palavras mais importantes de cada pergunta, removemos palavras de ligação e transformamos cada pergunta em sua representação *bag-of-words*.

Apesar de existirem representações mais avançadas, a simplicidade inerente desse sistema gera um teto de performance e capacidades que não pode ser ultrapassado mesmo com representações mais complexas.

Por último, essas representações de *bag-of-words* passam por uma rede neural *feed-forward* que identifica a resposta mais adequada de acordo com os dados de treino. A rede foi treinada com diversas configurações de camadas, número de neurônios e funções de ativação, e otimizada de acordo com o melhor resultado. Além dessas otimizações, as hipóteses a seguir foram testadas:

- Aumentar o número de perguntas para cada tópico nos dados de treino pode aumentar a fluidez do modelo, permitindo a compreensão de perguntas mais variadas. Buscar sinônimos e variações na formulação de perguntas de cada tópico pode contribuir para o enriquecimento do conjunto de dados.
- Manter o mesmo número de perguntas para todos os tópicos pode ser vantajoso. Isso garante que todos os tópicos tenham a mesma quantidade de registros nos dados de treinamento e evita um desbalanceamento a favor dos tópicos com mais versões das perguntas. Dessa forma, a seleção dos tópicos estará mais equilibrada.
- Incluir camadas de *dropout* pode ser uma boa tática para evitar o *overfitting*. Essa abordagem ajuda a melhorar a capacidade do modelo de generalizar para dados que não foram vistos durante o treinamento.
- Extrair as palavras essenciais de cada pergunta, com um *lemmatizer* por exemplo, pode permitir que o modelo compreenda com mais facilidade as perguntas sobre o mesmo tema que foram formuladas de maneiras diferentes. Embora essa técnica possa eliminar alguns detalhes da pergunta, a contribuição na facilidade de compreensão deve ser priorizada.

O modelo não é de natureza generativa como a arquitetura *transformer* estudada anteriormente. Apesar da rigidez inerente do sistema, estando limitada as respostas definidas nos dados de treino, a técnica descrita acima garante uma fluidez maior e, como será discutido posteriormente, essa implementação também tem as suas vantagens e cenários em que ela é recomendada.

## Fine Tuning de LLM

Como foi visto anteriormente, *Large Language Models* são modelos de inteligência artificial de processamento de linguagem natural, sendo treinados para compreender e gerar linguagem humana de forma coerente gramaticalmente e semanticamente. LLM usam uma arquitetura baseada em *transformer* e são treinadas em milhões de dados em milhares de iterações (OPENAI, 2022).

A capacidade de portabilidade e de transferência de aprendizado das LLMs possibilita a utilização do *fine tuning* para ajustar os pesos do modelo para uma tarefa específica, como o desenvolvimento de um *chatbot* para aconselhar startups.

Ainda assim o *fine tuning* pode ser muito custoso e demorado. Uma técnica utilizada para agilizar esse processo é o LoRA ou *Low-Rank Adaptation of Large Language Models*.

Seu propósito é otimizar a atualização dos parâmetros do modelo por meio de uma decomposição de baixa classificação. O LoRA mantém as camadas pré-treinadas do LLM inalteradas, introduzindo uma matriz de decomposição de classificação ajustável em cada camada do modelo. Ao contrário da abordagem direta de ajustar parâmetros nas camadas pré-treinadas do LLM, o LoRA concentra-se exclusivamente na otimização da matriz de decomposição de classificação, gerando um resultado que se aproxima da atualização completa obtida através do *fine-tuning*.

Neste caso, por conta do baixo investimento financeiro, todas as otimizações são bem-vindas. Existem várias *Large Language Models*. Dentre as LLMs de código aberto, a mais conceituada é a LLaMA 2. Desenvolvida pela Meta em fevereiro de 2023, ela foi treinada em um conjunto de dados de texto e código de 1,56 trilhão de *tokens* e possui uma versão com 65 bilhões de parâmetros. Existe mais de uma versão do LLaMa 2, cada uma tem um número diferente de parâmetros. Modelos com menos parâmetros tendem a ser mais eficientes computacionalmente, exigindo menos recursos de hardware para treinamento e inferência. Além disso, modelos maiores conseguem compreender representações mais abstradas e são ajustados com um volume maior de dados. Neste caso, como a capacidade de processamento e o tamanho do dataset não são tão grande em relação às grandes empresas, a solução mais adequada parece ser a versão com 3 bilhões de parâmetros.

Portanto, a segunda implementação do *chatbot* utilizou o *fine tuning* com *LoRA* na LLaMa 2 com 3 bilhões de parâmetros. O modelo foi treinado com o conjunto de dados de perguntas e respostas obtidos anteriormente e testado com diferentes configurações do modelo, de *fine tuning* e *LoRA* até chegar ao melhor resultado.

## Assistant API

Em outubro de 2023, a OpenAI, criadora do ChatGPT e disruptora do mercado de NLP, lançou a *Assistant API* (OPENAI, 2023). A API permite a construção de *chatbots* integrados ao ChatGPT de acordo com a especificação do seu negócio. Entre outras habilidades, a API é capaz de acessar uma base de dados customizada e responder as perguntas de acordo com as informações desses documentos.

Essa implementação aproveita toda a infra-estrutura da OpenAI além do treinamento e *fine tuning* do ChatGPT, acrescentando uma camada de personalização em cima dessas ferramentas. Além de reduzir o custo de desenvolvimento e treinamento, essa é uma solução mais simples para startups e empresas com menos conhecimento nos campos de inteligência artificial e NLP.

Essa implementação também tem o caráter generativo, utilizando a arquitetura *transformer* nos servidores da OpenAI. Consequentemente a API é capaz de gerar respostas mais complexas e contextualizadas, além de manter o contexto da conversa e continuar a conversa com o usuário.

## Resultados

A fim de avaliar o desempenho das diversas implementações, foram conduzidos vários testes e experimentos. Esses testes abrangem uma ampla gama de tópicos, passando por

vários dos temas treinados, com complexidades variadas e até perguntas cujas respostas não estavam incluídas nos dados de treino.

Essa seção tem o objetivo de apresentar as respostas geradas por cada implementação. Em seguida, as implementações serão comparadas de acordo com os parâmetros levantados na metodologia.

## Intent Chat

As mensagens abaixo foram enviadas ao chatbot desenvolvido com a tecnologia *intents*. As perguntas foram divididas em grupos de acordo com sua complexidade e as respectivas respostas estão disponíveis a seguir.

As perguntas abaixo estão presentes nos dados de treino, assim, a complexidade de formulação das respostas deve ser mais simples. Essas respostas serão uma referência inicial da sua capacidade dessa abordagem.

<b>Pergunta:</b> What is the first step in creating a startup?
<b>Pergunta no Treino:</b> What is the first step in creating a startup?
<b>Resposta:</b> A business plan is a good place to start. It outlines your goals, target market, competition, and financial projections. It serves as a roadmap for your startup's development.
<b>Nota:</b> 5/5

<b>Pergunta:</b> How can I identify a viable business idea?
<b>Pergunta no Treino:</b> How can I identify a viable business idea?
<b>Resposta:</b> To find good business ideas, look for unmet needs in the market, consider your own passions and skills, and explore opportunities where you can provide unique value.
<b>Nota:</b> 5/5

As próximas perguntas possuem uma complexidade um pouco mais elevada. Estas perguntas não coincidem exatamente com as formulações presentes nos dados de treinamento, isto é, palavra por palavra, mas ainda assim mantêm uma proximidade significativa.

<b>Pergunta:</b> How to make my brand well known?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>Resposta:</b> Create a user-friendly website, leverage social media, and invest in digital marketing.
<b>Nota:</b> 5/5

<b>Pergunta:</b> How can a startup be lean?
<b>Pergunta no Treino:</b> How to be a lean startup?
<b>Resposta:</b> A lean approach suggests starting with a basic plan and adjusting as you learn from the market.
<b>Nota:</b> 5/5

<b>Pergunta:</b> What strategies can we implement to launch fast?
<b>Pergunta no Treino:</b> How can I launch fast?
<b>Resposta:</b> Adopt a lean and agile mindset to streamline your product development process. Break down tasks into small, manageable increments. This allows for quicker iterations, faster feedback loops, and the ability to respond to market changes more effectively.
<b>Nota:</b> 5/5

Agora, as perguntas abaixo se distanciam ainda mais das perguntas do treino evitando repetir qualquer palavra de uma dada pergunta do *dataset*, tendo apenas sinônimos das palavras da pergunta original.

<b>Pergunta:</b> How to make my startup well known?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>Resposta:</b> Create a user-friendly website, utilize social media strategically, produce valuable content, and invest in online marketing efforts.
<b>Nota:</b> 5/5

<b>Pergunta:</b> How should I go about creating a startup?
<b>Pergunta no Treino:</b> How do I create a business?
<b>Resposta:</b> A business plan is a good place to start. It outlines your goals, target market, competition, and financial projections. It serves as a roadmap for your startup's development.
<b>Nota:</b> 5/5

<b>Pergunta:</b> Give me good business ideas
<b>Pergunta no Treino:</b> What are some good business ideas?
<b>Resposta:</b> To find good business ideas, conduct market research to identify gaps or unmet needs in the current business landscape, and explore opportunities to address those gaps. (...)
<b>Nota:</b> 5/5

<b>Pergunta:</b> Can you give me some advertising tips?
<b>Pergunta no Treino:</b> Tips for impactful advertising?
<b>Resposta:</b> Use various marketing channels, offer promotions or incentives, and focus on delivering an outstanding customer experience to attract and retain customers.
<b>Nota:</b> 5/5

<b>Pergunta:</b> Can you offer advice on the proper management of business finances?
<b>Pergunta no Treino:</b> How should I manage my business finance?
<b>Resposta:</b> Leverage financial tools and technology to streamline financial processes. Accounting software can simplify bookkeeping, invoicing, and financial reporting.
<b>Nota:</b> 5/5

<b>Pergunta:</b> Do I need a detailed business plan from the beginning?
<b>Pergunta no Treino:</b> Is it necessary to have a detailed business plan from the beginning?
<b>Resposta:</b> A lean approach suggests starting with a basic plan and adjusting as you learn from the market.
<b>Nota:</b> 5/5

<b>Pergunta:</b> How can I establish trustworthiness for my startup in its initial phases?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>Resposta:</b> To find your first customers, start by leveraging your existing network. Reach out to friends, family, and colleagues, sharing information about your product or service.
<b>Nota:</b> 0/5

Verifica-se a seguir as respostas a perguntas muito mais complexas. As perguntas a seguir são muito mais específicas, a qual não é respondida diretamente no *dataset*. Isto é, nenhuma resposta dos dados de treino por si só responderia de forma completa a pergunta.

<b>Pergunta:</b> What digital marketing strategy would be most effective in reaching and engaging our specific target audience for a elderly focused business?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> Create a user-friendly website, utilize social media strategically, produce valuable content, and invest in online marketing efforts.
<b>Nota:</b> 1/5

<b>Pergunta:</b> How do I cultivate trust and build a solid reputation for my software house?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> To find your first customers, start by leveraging your existing network. Reach out to friends, family, and colleagues, sharing information about your product or service. (...)
<b>Nota:</b> 0/5

<b>Pergunta:</b> What specific key performance indicators (KPIs) should we prioritize to measure the effectiveness of our customer acquisition strategy in the first six months?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> To find your first customers, start by leveraging your existing network. Reach out to friends, family, and colleagues, sharing information about your product or service. (...)
<b>Nota:</b> 0/5

<b>Pergunta:</b> Explain what is an MVP and why it is important for venture capitals to see the results from that first application?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> To find your first customers, start by leveraging your existing network. Reach out to friends, family, and colleagues, sharing information about your product or service. (...)
<b>Nota:</b> 0/5

Por fim, a tabela a seguir destaca a resposta dessa abordagem para uma pergunta que não está relacionada a empreendedorismo e startups.

<b>Pergunta:</b> Where is Waldo?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> I'm sorry, I don't understand you.
<b>Nota:</b> 5/5
<b>Observações:</b> Esse modelo é capaz de identificar perguntas fora do tema com certa facilidade. Isso acontece quando o percentual de certeza do assunto identificado está abaixo do <i>threshold</i> cadastrado.

## Fine Tuning de LLM

Para garantir uma avaliação justa e equivalente das diferentes abordagens, as mesmas perguntas realizadas ao *chatbot intent* na seção anterior serão direcionadas ao chatbot implementado com *fine tuning*. De acordo com a natureza generativa desse modelo, cada pergunta terá duas respostas geradas com suas respectivas notas. Observe as respostas abaixo.

As perguntas abaixo estão presentes nos dados de treino, assim, a complexidade de formulação das respostas deve ser mais simples. Essas respostas serão uma referência inicial da sua capacidade dessa abordagem.

<b>Pergunta:</b> What is the first step in creating a startup?
<b>Pergunta no Treino:</b> What is the first step in creating a startup?
<b>1º Resposta:</b> Start by asking yourself: what does your company have to do that no one else has done before or better than anyone else's approach for doing it. Create some sort of unique product, service, methodological process and experience with an unparalleled quality (think about how this will be different from any existing competitors) as well as strong value proposition based on customer needs.
<b>Nota da 1º resposta:</b> 5/5
<b>2º Resposta:</b> You can go to many seminars, read lots of books but it's not until you actually start your own business that what works and what doesn't. This talk will be about my experiences starting up 5 companies from scratch over multiple years with different models ranging between early stage startups. My talks have been covered by The Next Web on their podcast series as well as other news outlets such CNN News
<b>Nota da 2º resposta:</b> 2/5
<b>Observações:</b> Algumas respostas dos dados de treino são formuladas em primeira pessoa, porém não há nenhuma menção de uma palestra que foi realizada.

<b>Pergunta:</b> How can I identify a viable business idea?
<b>Pergunta no Treino:</b> How can I identify a viable business idea?
<b>1º Resposta:</b> 1. Identify Your Strengths and Weaknesses What are your strengths, and what is where you need to improve/change in order for the company or product / service that would be started by yourself become successful? This includes understanding whether there is any knowledge gap (e.g., not knowing how to manage accounting) , as well as personal interests such As long-term goal of this work will help you decide if it's worth investing time learning everything about an industry You don't want to spend 5 years only because at first glance seemed good It could also include taking into account cultural influences Affecting decision making process For example, people who are raised in Asian cultures often tend To have more conservative views on life than Americans do And when trying to start their own businesses They may consider things like safety before starting up With higher risks That usually requires more capital from investors etc
<b>Nota da 1º resposta:</b> 4/5
<b>Resposta:</b> Identifying the right venture for you depends on your passion and skills. Here are some questions to ask yourself before starting up: Are there needs that aren't being addressed in my local market or by existing businesses/startups? Is it possible, legal, profitable etc..
<b>Nota da 2º resposta:</b> 5/5

As próximas perguntas possuem uma complexidade um pouco mais elevada. Estas perguntas não coincidem exatamente com as formulações presentes nos dados de treinamento, isto é, palavra por palavra, mas ainda assim mantêm uma proximidade significativa.

<b>Pergunta:</b> How to make my brand well known?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>1º Resposta:</b> This is a very common question that any startup should ask. If you have been in the business for some time, it might already be pretty clear what your product or service can do and how good of an offering will attract customers but new startups are often not quite sure where exactly their efforts ought to go next. You want people talking about your company online so they spread information quickly on social media channels like Facebook pages and LinkedIn groups. Get involved with communities related to your industry and get yourself seen as an expert by sharing relevant content through blog posts or articles.. Startups need help from experts who understand digital marketing.
<b>Nota da 1º resposta:</b> 5/5
<b>2º Resposta:</b> It's essential that your business and its products or services be recognizable. To do this, you need a good marketing strategy based on the quality of your image and content. For example, if it is possible for consumers to recognize themselves in advertising messages created by their favorite brands, they will buy more often from these companies . Also consider creating an online presence where users can see what makes your company unique among others.
<b>Nota da 2º resposta:</b> 5/5

<b>Pergunta:</b> How can a startup be lean?
<b>Pergunta no Treino:</b> How to be a lean startup?
<b>1° Resposta:</b> Startups should strive to become as lean and efficient as possible. They do not have the luxury of large, established teams or expensive resources that take up space in their office. Startup companies are always looking for ways to cut costs, reduce operational expenses, eliminate unnecessary tasks, etc., all with the aim of increasing profitability.
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> How to Be Lean in 2019: Startup Advice from Founders. A simple way of becoming more efficient and effective is by following the principle of being as short-sighted with your decisions today as you'd want any decision maker or employer to do for future employees.
<b>Nota da 2° resposta:</b> 0/5
<b>Observações:</b> Aparentemente a resposta apresenta um livro, porém esse título de livro não existe.

<b>Pergunta:</b> What strategies can we implement to launch fast?
<b>Pergunta no Treino:</b> How can I launch fast?
<b>1° Resposta:</b> Starting a company in today's world is hard. It isn't getting any easier as time goes on, and every day it seems like new startups are popping up left and right all over the globe. In fact, just look at what happened this past year with SpaceX launching its first rocket! A lot of times when you start something that does so well for an initial release (such as Uber), there aren't many barriers to entry other than capital and some basic technical skills - but things change quickly once those barriers go down since competition tends to follow close behind.
<b>Nota da 1° resposta:</b> 4/5
<b>2° Resposta:</b> We need something that helps us get the first few launches under our belt so people know what they're getting into. We also want a system that works in any scenario and allows for rapid iteration over time.
<b>Nota da 2° resposta:</b> 5/5

Agora, as perguntas abaixo se distanciam ainda mais das perguntas do treino evitando repetir qualquer palavra de uma dada pergunta do *dataset*, tendo apenas sinônimos das palavras da pergunta original.

<b>Pergunta:</b> How to make my startup well known?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>1° Resposta:</b> The internet is an important tool for any company. There are several ways you can promote your brand, from social media marketing to online advertising and public relations. You will find that there are many platforms where startups can advertise their business, such as LinkedIn, Facebook, and Instagram. By using these channels effectively, companies can maximize their exposure and attract more customers. Startups also need to create a strong online presence by building an official website and updating it regularly with fresh content. Doing so will help promote the brand and increase engagement with visitors.
<b>Nota da 1° resposta:</b> 5/5

**2º Resposta:** - 10. Start building your digital presence: Your website is the foundation of you being seen online, so start by creating it if you don't have one yet! Create social media profiles for yourself and/or use some other platforms (such as LinkedIn). Then write blog posts or start a podcast about what makes up who you are as an entrepreneur. This will get people talking about their work in different ways than just seeing someone at networking events which may sound intimidating but remember that they can become friends with these individuals too

**Nota da 2º resposta:** 5/5

**Pergunta:** How should I go about creating a startup?

**Pergunta no Treino:** How do I create a business?

**1º Resposta:** A business plan is a good place to start. It outlines your goals, target market, competition, and financial projections. It serves as a roadmap for your startup's development.

**Nota da 1º resposta:** 5/5

**2º Resposta:** "I don't know anything and have no money. Can you help me get started, please?" If this is your story - then keep reading! In the next few minutes, we will discuss 3 important things that every aspiring entrepreneur must do to establish themselves as an independent business owner: research market trends; understand the requirements of potential customers; & finally start up in accordance with these guidelines from day one itself!. The idea behind doing all these early on would be so even before being able fund it fully later down when everything else falls into place there are some key foundations for startups based upon which their growth path can begin.. Start by researching what kinds of businesses succeed around town or country. (...)

**Nota da 2º resposta:** 4/5

**Pergunta:** Give me good business ideas

**Pergunta no Treino:** What are some good business ideas?

**1º Resposta:** You can create your own successful businesses with the following 7 types of products: food, clothing and cosmetics; sports accessories - sportswear or footwear ; fashion items (designers); jewelry / watches Accessories for homes : tableware , kitchen utensils

**Nota da 1º resposta:** 5/5

**2º Resposta:** Please, give me good business ideas. I am a 24-year old entrepreneur and have recently completed MBA from University of London in UK. I would like some basic steps that i can take step by step so that things do not become more complicated than it already are! Thank you very much!!

**Nota da 2º resposta:** 0/5

**Observações:** Note que ao invés de escrever a resposta, o modelo continuou a geração de texto da frase enviada. Talvez isso tenha acontecido porque os dados de treino tem apenas perguntas e não frases no imperativo.

<b>Pergunta:</b> Can you give me some advertising tips?
<b>Pergunta no Treino:</b> Tips for impactful advertising?
<b>1° Resposta:</b> You've got to be aggressive. You have to understand that the world is changing, and most of what I do now has changed in my lifetime from when I was a kid. And so if your business hasn't adapted yet — as opposed being afraid because it can seem like fear is an easier emotion than anything else but we just need to get over our fears about how difficult this whole thing is gonna be. This isn't easy at all. But there are very few businesses out there today that don't really depend on people coming together with other people for stuff to happen.
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> Well, I guess my first tip would be to just make sure that your product is good. You have an awesome idea for a new app or service and then there's this great logo - but if no one knows about it, how do they know to use the thing? So as much time can spend working on marketing as work designing products!
<b>Nota da 2° resposta:</b> 5/5
<b>Pergunta:</b> Can you offer advice on the proper management of business finances?
<b>Pergunta no Treino:</b> How should I manage my business finance?
<b>1° Resposta:</b> The good news is that small businesses have an advantage when it comes to this area. You can manage financials and stay financially healthy by keeping a close eye on your money, setting budgets, staying organized with invoices and bills, reviewing bank statements regularly, managing credit, adhering strictly to a budget, paying taxes promptly, seeking professional help where needed (like tax preparers), sticking to a regular payment schedule for debt, investing in retirement plans or other savings vehicles, ensuring insurance coverage, avoiding unnecessary expenses, establishing strategic partnerships, outsourcing tasks, making use of technology, leveraging data analytics, monitoring cash flow trends, employing sophisticated tools like AI software, tracking KPIs, building a strong credit history, planning ahead based on predicted needs, having effective systems in place, etcetera . It's easy to fall into bad habits over time , but if you maintain these practices consistently throughout the duration of your company's existence - no matter how long that may be!
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> First step to doing this is creating a financial budget. You want your monthly expenses and revenue estimates aligned, then balanced so that you're spending less than you take in each month.
<b>Nota da 2° resposta:</b> 5/5

<b>Pergunta:</b> Do I need a detailed business plan from the beginning?
<b>Pergunta no Treino:</b> Is it necessary to have a detailed business plan from the beginning?
<b>1° Resposta:</b> Business Plans come in many shapes and sizes, but all should be well-defined. It's not essential that they are extensive or contain an exhaustive list of financial projections (even if you intend to seek funding) as long as it clearly sets out how your idea will make money for its current founders/startup team .
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> Not necessarily, but you should have some idea of what your product or service will look like and who it's intended for. That way when we do our market research at the end (which involves reading lots of books about this topic) you can start to align yourself with certain demographics that might be interested in purchasing whatever is coming out later on down the line. This also gives us an idea if there are any gaps left open within those demographic groups so maybe something else could work better than just taking over completely. The reason why these kind of things matter so much comes down to logistics: they help you figure out how many units are going into production at any given time which allows for proper planning and scheduling ahead
<b>Nota da 2° resposta:</b> 5/5
<b>Pergunta:</b> How can I establish trustworthiness for my startup in its initial phases?
<b>Pergunta no Treino:</b> How to enhance brand visibility?
<b>1° Resposta:</b> Establishing credibility when you launch your new business is a vital aspect of building the brand. You need to be visible and reach out to potential customers through various channels so they know that what sets you apart from other competitors
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> This is a very common question that we get from new startups. Establishing credibility takes time and starts with building genuine relationships, but some basic practices will also go a long way towards establishing the brand's reputation early on. Ensure you provide high quality products or services at competitive prices to build positive reviews, maintain transparency about your business operations and make sure clients are satisfied during product use and after sale service.
<b>Nota da 2° resposta:</b> 5/5

Verifica-se a seguir as respostas a perguntas muito mais complexas. As perguntas a seguir são muito mais específicas, a qual não é respondida diretamente no *dataset*. Isto é, nenhuma resposta dos dados de treino por si só responderia de forma completa a pergunta.

<b>Pergunta:</b> What digital marketing strategy would be most effective in reaching and engaging our specific target audience for a elderly focused business?
<b>Pergunta no Treino:</b> -
<b>1° Resposta:</b> We are currently looking into the best way to use social media effectively targeting older people. Social Media is often overlooked by businesses as they may not think it's relevant or accessible but this really shouldn't deter you! Research shows that 46% of adults over 55 regularly access Facebook, Twitter, Instagram etc whilst also being very engaged with content from brands on these platforms too - so why wouldn't we tap into their needs using this medium?! It will enable us to build up relationships directly with them which can increase brand awareness overall if done correctly..
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> The senior market is generally more difficult than other markets because they don't use computers often or at all They may not have smartphones so mobile advertising isn't as successful with them As you can see this graph below shows search engine usage amongst different demographics 50 percent of Baby Boomers Google every day But only 17 percent look up doctors online Search engines such as google which make finding information easier might work better but there aren't many companies doing it yet So social media tends to work well when using Facebook Twitter Linked In Instagram And Pinterest There are very low barriers to entry into these platforms even if your company doesn't specialize in Social Media Advertising If You Are An Artist Or Photographer Use Tumblr It has some high quality art blogs That Might interest older people Check out LinkedIn Groups For Older People Many communities on here like Senior Technology Networking Events offer meetups opportunities etc Also check local newspapers This will help get word of mouth around town When talking about getting an email list ask yourself Is Email Marketing More Important To My Business Than Having A Website Make Your Emails Relevant Personalize Send Them Often Don't bombard everyone Get their permission before sending anything Offer incentives eg discounts coupons freebies prizes Etc Consider giving away something completely free then charging later e g monthly memberships Try offering exclusive content ie early access deals On Other Types Of Content Like articles videos podcasts etc
<b>Nota da 2° resposta:</b> 4/5
<b>Pergunta:</b> How do I cultivate trust and build a solid reputation for my software house?
<b>Pergunta no Treino:</b> -
<b>1° Resposta:</b> Trust is essential when working with clients. Building confidence by delivering reliable, quality products results in loyalty and long-term partnerships
<b>Nota da 1° resposta:</b> 5/5
<b>2° Resposta:</b> If you're building your own startup, this question could be the most important one that we can ask ourselves. The only way to create or maintain strong relationships with clients is by showing them how much they are worth in numbers: through tangible results delivered on time. This goes back to customer satisfaction—the best indicator of whether people value their business relationship.
<b>Nota da 2° resposta:</b> 5/5

<b>Pergunta:</b> What specific key performance indicators (KPIs) should we prioritize to measure the effectiveness of our customer acquisition strategy in the first six months?
<b>Pergunta no Treino:</b> -
<b>1º Resposta:</b> This is a question that many organizations struggle with. It's hard for us as managers and leaders to know what actually matters most, even though it seems like obvious things would be good candidates: getting 10% more customers who agree to sign up or converting those interested into paying users are both important milestones along your startup journey – but how can you tell when one achievement outpaces another without having data at hand from which to draw conclusions?.
<b>Nota da 1º resposta:</b> 5/5
<b>2º Resposta:</b> If you can get an early indicator that will allow us to act on something, then it'll be very helpful later
<b>Nota da 2º resposta:</b> 4/5
<b>Pergunta:</b> Explain what is an MVP and why it is important for venture capitals to see the results from that first application?
<b>Pergunta no Treino:</b> -
<b>1º Resposta:</b> First, make sure you understand how your product can help solve a real-world problem. If you're not familiar with this concept, think of Google Maps when you were in college or high school as if someone invented something like Microsoft Outlook 15 years ago! The next step involves validating assumptions about user behavior through data analysis so that we know how many people are using our app versus other similar apps on iOS/Android devices during peak hours etc.. Finally, after gathering all the necessary information such as usage trends based upon analytics reports & online surveys conducted amongst target audience then only start working towards building robust features by focusing key areas including content strategy development & deployment
<b>Nota da 1º resposta:</b> 2/5
<b>2º Resposta:</b> MVP stands for Minimum Viable Product. And so this idea of a minimum viable product, it's very closely related to sort of your ability to prove out that you have something real, something tangible even though there are other people who may claim they do or they don't on paper without having built anything. So you really need to show some kind of proof. It's important because investors want to see whether they can build what they claim is their idea, and whether there actually will be a user base once that product is made. That has been proven by companies like Instagram, which was basically a picture sharing app with one photo. But it showed up in a way that it was clear that there would be users, just by showing pictures.
<b>Nota da 2º resposta:</b> 5/5

Por fim, a tabela a seguir destaca a resposta dessa abordagem para uma pergunta que não está relacionada a empreendedorismo e startups.

<b>Pergunta:</b> Where is Waldo?
<b>Pergunta no Treino:</b> -
<b>Resposta:</b> The second most famous book to appear in the movie theater. First, I believe that it can be a very good way of encouraging children and young people not only on their reading habits but also with an interest for books.
<b>Nota:</b> 0/5
<b>Observações:</b> A resposta está correta, porém como um chatbot para startups essa pergunta não deveria ser respondida. É provável que esse dado tenha sido aprendido durante a fase de treino da LLM antes do <i>fine-tuning</i> .

## Assistent API

As mesmas perguntas da seção anterior foram direcionadas ao *chatbot* implementado com *Assistant API*. Esse *chatbot* se mostrou mais estável, assim um número menor de perguntas será apresentado a seguir. Além disso, essa implementação utiliza os dados de um grande arquivo de texto como base para responder as perguntas em vez de um *dataset* com perguntas e respostas. Assim, as perguntas serão ordenadas pela complexidade da mesma e não pela similaridade com as perguntas do *dataset* de treino como nas outras soluções.

As primeiras perguntas serão mais simples:

<b>Pergunta:</b> What is the first step in creating a startup?
<b>Resposta:</b> Creating a startup typically involves several key steps, and the first of those is often the ideation phase, where you come up with a business idea. However, the exact first step can depend on personal viewpoints and methodologies. Here is a general outline of early steps in the startup process: <ol style="list-style-type: none"> <li><b>Idea Generation</b>: The very beginning of creating a startup is to have a business idea. This could come from recognizing a gap in the market, identifying a problem that needs solving, or through innovation in technology or processes.</li> <li><b>Market Research</b>: After you have an idea, market research is crucial. You need to understand your audience, the demand for your product or service, and the competitive landscape.</li> <li><b>Building a Business Plan</b>: This plan outlines your business idea, your strategy for success, the market potential, the business model, and financial projections.</li> <li><b>Finding Co-Founders and Building a Team</b>: Early in the process you may also seek out co-founders with complementary skills and start building your team.</li> <li><b>Minimum Viable Product (MVP)</b>: Developing an MVP, which is a basic version of your product to introduce to the market for feedback.</li> <li><b>Fundraising</b>: If your startup requires external funding, this would typically be a step where you seek out investors or other funding sources.</li> </ol>

Please note that these steps are a high-level overview and actual practices may vary depending on the nature of the startup and other factors. Additionally, since I am unable to reference the specific content of the document you provided, this advice does not directly draw from that document. If you have any other documents or if the document you previously uploaded is corrected, feel free to share it so I can provide you with answers based on that specific content.

**Nota:** 5/5

Aumentando o nível de complexidade, as próximas perguntas exigem um relacionamento entre alguns trechos do texto base:

**Pergunta:** What strategies can we implement to launch fast?

**Resposta:** To launch fast, one important strategy is "Fast Decision-Making, Cycle Time, Speed, and Tempo." Speed is crucial in startups because as long as the startup is in operation, its bank balance is declining daily. Quick iterations and pivots increase the chances of finding a scalable business model while the startup still has cash on hand. Slower cycles can lead to the startup running out of cash and, ultimately, failing. The primary impediment to short cycle time is often psychological, which implies the need for a mindset geared towards swift decision-making and actions.

**Nota:** 5/5

**Pergunta:** Give me good business ideas

**Resposta:** The document you've uploaded seems to be an educational or informative piece possibly related to startups or entrepreneurship. Unfortunately, the text provided does not offer any direct business ideas.

However, there are several themes and areas mentioned where potential business ideas could stem from:

1. **Technology solutions** in untapped or rapidly growing markets (like Microsoft and Facebook in their early days).
2. Developing products or services in **new markets** that offer customers something they couldn't do before or dramatically lowering costs to create new user classes.
3. **Re-segmenting existing markets** by either offering a low-cost alternative or creating a niche product that appeals to a specific part of the market.
4. **Cloning existing business models** in a new geographic area with adaptations for local language and culture.

These themes suggest that good business ideas could come from identifying a gap in the market or creating a new market altogether. Re-segmenting an existing market or copying a successful model in a new region are also viable strategies.

For tailored business ideas, consider your own expertise, interests, and the needs of the market you are familiar with. This personalized approach, combined with the strategies and principles found in the document, could result in a compelling business proposition. Is there a particular industry or type of business you are interested in, or do you require further guidance on how to apply these concepts to generate specific ideas?

**Nota:** 5/5

<b>Pergunta:</b> Can you give me some advertising tips?
<b>Resposta:</b> Here are a couple of advertising tips based on the document you provided: 1. <b>**Identify and Understand Your Revenue Channels**</b> : When operating within multi-sided revenue channels, it is essential to attract a large number of users since advertisers pay to reach them. This can be done via text links, banner ads, in-game visibility, or traditional online advertising. Your goal is to identify potential advertisers or agencies and estimate how much they are willing to pay. Also, learn how the sales process works and validate your hypotheses about the revenue side of the market [2 source]. 2. <b>**Authentic Communication**</b> : Be honest and authentic when responding to questions. Authenticity is crucial for building lasting business relationships. For instance, Barack Obama's strategy was centered around the concept of "Forward," which provides an important lesson for framing your own pitch. Your messaging should reflect genuine value and connect with your audience on a sincere level [3 source]. These insights reiterate the importance of understanding your market and communicating authentically with your audience to build a successful advertising strategy. Would you like more detailed advice on specific aspects of advertising, like digital marketing, ad design, targeting, or other areas?
<b>Nota:</b> 5/5

<b>Pergunta:</b> Do I need a detailed business plan from the beginning?
<b>Resposta:</b> A detailed business plan isn't necessary from the very beginning, especially for startups where circumstances can change rapidly and flexibility is crucial. Instead of a highly detailed business plan, it can be more effective to start with a bottom-up estimate of the market opportunity and develop a brief framework that outlines the key aspects of your business model. This would include a one- or two-page brief about each component of your business model canvas, such as your market size, value proposition, customer segments, distribution channels, customer relationships, key resources, key partners, and revenue streams [4 source]. (...)
<b>Nota:</b> 5/5

Numa escala de complexidade ainda maior, também podemos verificar a resposta do modelo para perguntas mais específicas que envolvem o relacionamento de conhecimentos prévios com o documento de texto.

<b>Pergunta:</b> What digital marketing strategy would be most effective in reaching and engaging our specific target audience for an elderly focused business?
<b>Resposta:</b> When formulating a digital marketing strategy to reach and engage an elderly-focused audience, it's crucial to understand the unique characteristics of this demographic. While the specific digital marketing strategies are not explicitly mentioned in the document for targeting the elderly, here are some general tips that can be applied effectively to this audience based on both principle and experience. 1. <b>**Simplicity and Accessibility**</b> : Ensure that your website and digital platforms are easy to navigate and are accessible. This might involve larger text, intuitive interface, and clear calls to action.

2. **Use of Relatable Content**: Develop content that resonates with the elderly by addressing their needs, preferences, and concerns. This can include articles, videos, and images that speak to topics like health, retirement life, technological assistance, and social engagement.
3. **Targeted Social Media Advertising**: Older adults are increasingly using social media. Platforms like Facebook offer targeting options based on interests and demographics, allowing you to reach seniors who may be interested in your products or services.
4. **Email Marketing**: Since many elderly users regularly check their email, a straightforward email campaign with relevant information and offers can be impactful.
5. **Educational Content and Workshops**: Offering webinars, tutorials, or Q&A sessions on topics pertinent to older adults can be a way to engage them.
6. **Influencer Partnerships**: Partner with influencers who have credibility with the elderly population. This could be senior bloggers, YouTubers, or personalities that they trust.
7. **User Interface Testing**: Make sure your digital properties are tested with your target audience to ensure they are user-friendly for an older demographic. Tools like Usertesting or Userfy can be very helpful for this purpose [8 source].
8. **Direct Mail**: Although not digital, direct mail can complement your digital efforts, since it is a familiar channel to many seniors and can drive them to your online platforms [5 source].

It's important to note that the strategies that work for one market type seldom work for another. Different market types require dramatically different marketing strategies. Therefore, understanding where your product or service fits in and how your specific elderly audience behaves online will guide your digital marketing efforts effectively [1 source].

Remember to track and measure the performance of these strategies and be ready to adapt based on feedback and engagement from your target audience. Would you like more information on a specific digital marketing channel or about understanding your market type further?

**Nota:** 5/5

**Pergunta:** How do I cultivate trust and build a solid reputation for my software house?

**Resposta:** Building trust and establishing a solid reputation for your software house involves a combination of several strategies that focus on delivering value, maintaining high-quality standards, and engaging effectively with clients and the broader community. While the document provided does not give specific guidance on building trust for a software house, here are some practices based on general business principles that can help you cultivate trust and build a reputation:

1. **Deliver High-Quality Products and Services**: Consistently deliver software solutions that meet or exceed client expectations. Ensure that your products are reliable, user-friendly, and fulfill the needs of the users.
2. **Provide Excellent Customer Service**: Offer responsive, helpful, and transparent customer service. This includes being easily reachable, promptly addressing concerns, and providing clear communication at all times, from initial contact through to post-sale support.

3. **Cultivate a Strong Brand and Online Presence**: Develop a professional brand image and maintain an excellent online presence through your website and social media profiles. Share insights, news, and information relevant to your expertise.
4. **Referrals and Testimonials**: Encourage satisfied customers to provide testimonials and refer others to your services. Positive word-of-mouth referrals from happy clients will substantiate your credibility.
5. **Case Studies and Portfolios**: Showcase successful projects with case studies and a portfolio on your website. This serves as proof of your capabilities and experience.
6. **Maintain Professional Integrity**: Keep your promises, meet deadlines, and communicate openly if challenges arise. Honesty and integrity go a long way in building trust with clients.
7. **Thought Leadership**: Position yourself as an expert in your field by sharing knowledge through blogs, speaking engagements, webinars, and participation in industry events.
8. **Continuous Improvement**: Stay updated with the latest technologies and industry best practices, implementing them into your workflow and products. This demonstrates a commitment to continuous improvement and innovation.
9. **Gather and Act on Feedback**: Regularly seek feedback from your clients and act on it to improve your services. Let your clients know that their opinions are valued and that you are committed to serving them better.
10. **Community Engagement**: Get involved in the software development community by contributing to open-source projects, forums, and discussions. This builds rapport with peers and establishes your company as a trusted member of the tech community. Implement these strategies consistently and adapt them as your business and client base grow. A reputation is built over time through steady accumulation of positive interactions, client satisfaction, and visible expertise in your field. If you need further guidance, it's also helpful to look into case studies or advice from established software houses that have built a reputable brand.

**Nota:** 5/5

**Pergunta:** Explain what is an MVP and why it is important for venture capitals to see the results from that first application?

**Resposta:** An MVP, or Minimum Viable Product, is the most basic version of a product that can be released to market. It's designed to have just enough features to satisfy early customers and provide feedback for future product development. The MVP is a critical concept in the Lean Startup methodology, which emphasizes the importance of learning and iterating quickly.

In the context of venture capital and startup investment, presenting an MVP is important because it acts as a proof of concept, demonstrating that there is market demand for your product and that your team is capable of delivering it. This early version allows the venture capitalists to see that:

1. **The team can execute**: Developing an MVP requires a certain level of skill and organization. Being able to showcase a working product helps build confidence in the team's capabilities.
2. **Market validation**: An MVP that has some level of user adoption shows evidence that the market is interested in the product. It's a crucial step in validating the business hypothesis.

3. **\*\*Customer feedback\*\***: It opens up the lines of communication with early users for feedback. This information can help shape the future direction of the product development while reducing the time and resources spent on features that customers may not value.
  4. **\*\*Potential for scale\*\***: A successful MVP shows promise for future growth—something that is critical to venture capitalists, who are often looking for investments that can generate significant returns.
  5. **\*\*Iterative improvement\*\***: The MVP starts a process of iteration where the product can be improved based on actual user data and feedback, reducing the guesswork in product development.
  6. **\*\*Efficiency\*\***: It's usually more cost and time efficient to produce an MVP than to invest heavily in a full-featured product from the outset. This aligns with the risk-averse approach that both startups and venture capitalists can appreciate.
- By investing in a startup with a successful MVP, venture capitalists are not just investing in an idea or team, but in a proven concept that has begun to demonstrate its market fit and potential for profitability. They look for results from an MVP to minimize their risk and to gauge the potential return on their investment. These tangible results make discussions about valuations more grounded in reality, which is beneficial for both the startup and the investors.

**Nota:** 5/5

Por último, quando uma pergunta que não está relacionada aos dados de treinamento, o resultado está demonstrado a seguir:

**Pergunta:** Where is Waldo?

**Resposta:** I'm here to assist with startup and business-related questions. If you have questions about creating a minimum viable product (MVP), developing a business plan, marketing strategies, or other topics in this realm, feel free to ask! For questions like "Where is Waldo?" or similar, I would not be able to provide an answer. Please let me know how I can assist you with your startup or business concerns.

**Nota:** 5/5

## Análise dos Cenários

Analisando os resultados das implementações em cada cenário, observa-se alguns padrões de comportamento.

No caso das perguntas dos dados de treino, o *chatbot intent* apresenta o comportamento mais estável. Isso é de se esperar porque nessa implementação as respostas são pré-definidas e, pela pergunta aparecer nos dados de treino, a seleção dessa resposta é trivial. Nas implementações que utilizam a tecnologia generativa, as respostas também são muito coerentes, porém são mais instáveis por conta da natureza generativa.

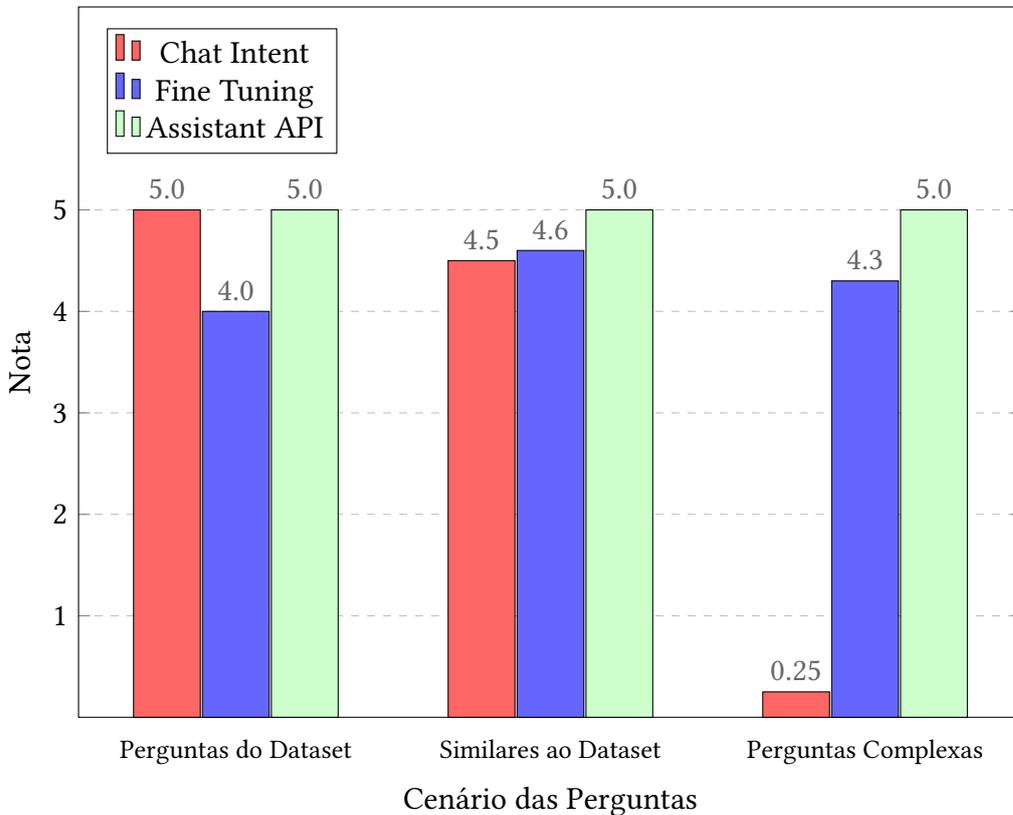
Por outro lado, à medida que as perguntas se afastam do conjunto de treinamento, as tecnologias generativas começam a se destacar, demonstrando a sua grande capacidade de compreender o contexto da pergunta e construir uma resposta de acordo com as informações dos dados de treinamento. Apesar da performance satisfatória *chatbot intent* nesses cenários, a ferramenta passou a produzir algumas respostas que, mesmo com

informações corretas, não abordam diretamente a pergunta formulada.

A capacidade das implementações generativas mostram efetivamente a sua capacidade nas perguntas mais específicas, que envolvem o relacionamento de dados entre diversas respostas e um conhecimento base prévio. Nesse cenário, apesar da grande complexidade das perguntas, as respostas são em geral extremamente pertinentes e completas, até mais do que era esperado originalmente. O *chatbot intent*, no entanto, não foi desenvolvido para perguntas tão complexas e estando limitado as respostas previamente definidas, resulta em respostas que, mesmo com informações corretas, não abordam diretamente a pergunta formulada.

Por último, em perguntas sobre temas não pertinentes ao chatbot, em vez de recusar uma resposta, a implementação de *fine-tuning* cria um frase com base no conhecimento adquirido durante o pré-treinamento. As outras duas implementações, tem mais facilidade em compreender se a pergunta é ou não pertinente ao *chatbot*. Esse comportamento é de se esperar para o *chatbot intent* já que ele calcula a chance da pergunta se referir a cada tópico cadastrado, assim, ele consegue identificar quando os percentuais são baixos demais. No entanto, a capacidade do *chatbot* com *Assistant API* de, mesmo utilizando a arquitetura *transformer*, identificar perguntas fora de contexto é no mínimo surpreendente. Uma hipótese é que isso ocorre porque a instrução especifica o objetivo do chatbot e seus dados de consulta, por isso, o modelo primeiro verifica a similaridade da pergunta com esses dados e oferece uma resposta apenas caso existir uma proximidade.

**Performance dos Chatbots nos Diferentes Cenários**



Em geral, os usuários tendem a enviar perguntas mais diversas e que não estão tão

próximas das perguntas de treinamento. Esse é um dos motivos das tecnologias generativas terem ganhado tanta visibilidade nos últimos anos e é justamente uma das principais razões da superioridade aparente dessa tecnologia.

## Avaliação

Como foi explicado anteriormente, as implementações do chatbot serão avaliadas de acordo com três parâmetros:

- **Fluidez e Adaptabilidade:** Os *chatbots* devem ser capazes de compreender perguntas complexas e gerar respostas adequadas de acordo com as especificidades da pergunta. Esse parâmetro não avalia a correteza das respostas, mas sim a compreensão da pergunta, assim, as respostas devem demonstrar a compreensão da pergunta.
- **Precisão nas Respostas:** *Chatbots* precisos são aqueles que produzem respostas corretas de acordo com os dados de treino. O sistema pode relacionar diversos dados de treinamento, porém não deve produzir informações falsas.
- **Custo e Tempo de Treinamento:** Esse parâmetro faz referência ao custo de treinamento do sistema. Tanto o investimento financeiro quanto aos requisitos de hardware e tempo serão considerados.

Com o objetivo de avaliar os parâmetros de fluidez e precisão, foi conduzida uma pesquisa envolvendo sete startups do estado de São Paulo. A seleção dessas startups considerou seu estágio de desenvolvimento e os mercados nos quais atuam, buscando uma diversidade de segmentos e maturidade. A pesquisa incluiu startups de diversas cidades, como São Paulo, Campinas, São José dos Campos e Ribeirão Preto, abrangendo setores como tecnologia, agronegócio, saúde e advocacia, em diferentes estágios, desde a ideação até o desenvolvimento do MVP (Produto Mínimo Viável) e tração de clientes.

Cada fundador das startups foi orientado a formular 10 perguntas, avaliando a utilidade das respostas com base na fluidez e precisão. Entre elas, 3 perguntas eram mais simples, 4 eram mais complexas, e 3 eram específicas a startup, considerando seu mercado e estágio atual.

As mesmas perguntas foram realizadas aos três *chatbots* e as respostas eram apresentadas em ordem aleatória para garantir a imparcialidade dos resultados. Os fundadores avaliaram cada resposta com notas de 1 a 5 para a fluidez e precisão e, ao final do teste, a média das notas de cada implementação será calculada. Veja os resultados abaixo:

/ Participante	Chat Intents		Fine Tuning		Assistant API	
	Fluidez	Precisão	Fluidez	Precisão	Fluidez	Precisão
<b>Startup 1</b>	3.6/5	4.6/5	4.4/5	4.3/5	4.8/5	4.8/5
<b>Startup 2</b>	4.0/5	4.7/5	4.5/5	4.5/5	4.9/5	4.8/5
<b>Startup 3</b>	3.8/5	4.5/5	4.4/5	4.4/5	4.9/5	4.7/5
<b>Startup 4</b>	3.4/5	4.4/5	4.3/5	4.1/5	4.8/5	4.9/5
<b>Startup 5</b>	3.8/5	4.7/5	4.6/5	4.5/5	4.9/5	4.8/5
<b>Startup 6</b>	3.7/5	4.7/5	4.8/5	4.9/5	4.8/5	4.6/5
<b>Startup 7</b>	3.6/5	4.8/5	4.6/5	4.5/5	4.9/5	4.9/5
<b>Média</b>	<b>3.7/5</b>	<b>4.6/5</b>	<b>4.5/5</b>	<b>4.5/5</b>	<b>4.9/5</b>	<b>4.8/5</b>

Os dados acima reforçam os resultados apresentados anteriormente. O *chatbot intents* possui uma alta precisão, favorecida pelas suas respostas pré-definidas que foram validadas. A mesma característica, porém, diminui a fluidez da ferramenta, já que perguntas mais específicas não estarão no conjunto de treino e assim não serão respondidas adequadamente.

As outras duas implementações tem uma fluidez muito maior com um leve declínio na precisão. As soluções com a arquitetura *transformer* compreendem perguntas com complexidades muito maiores e geram respostas muito mais específicas para o contexto do usuário, apesar das ocasionais mensagens com informações erradas.

O custo de cada implementação foi analisado separadamente de acordo com as necessidades de hardware, tempo de treinamento e custo de locação das máquinas virtuais:

Custo das Implementações		
Chat Intents	Fine Tuning	Assistant API
5/5	3.5/5	2.5/5

Observando as tabelas acima, parece existir uma correlação entre custo e eficácia. O *chatbot intents* tem um requisito muito baixo de hardware e o treinamento é bem rápido, porém como foi visto a fluidez da solução não é muito grande.

O custo do *fine tuning* está acumulado no período de treinamento do modelo por conta da necessidade de GPUs de alto nível. Com o custo maior, no entanto, a solução se apresenta uma melhoria surpreendente na fluidez, principalmente em relação a perguntas mais complexas.

Por último, o *chatbot* implementado com *Assistant API* utiliza toda a infraestrutura e treinamento da OpenAI. Sendo assim, essa é a solução mais simples de ser desenvolvida e tem uma ótima performance com fluidez e precisão. No entanto, é a solução mais cara. A API cobra pelo número de tokens lidos e respondidos, assim a execução do modelo tem custo elevado.

## Escolha da Implementação

Implementações clássicas de *chatbots* e assistentes virtuais como o *chatbot intent* foram usadas por muitos anos e são eficientes para uma série de aplicações. Ainda hoje, essa técnica pode ter resultados satisfatórios. À medida que os usuários adquirem experiência ao

interagir com uma assistente virtual, por exemplo, eles tendem a elaborar suas perguntas de maneiras que facilitem a compreensão por parte da ferramenta. Ainda hoje, grande parte dos *chatbots* das empresas utilizam essa tecnologia.

No entanto, a introdução da arquitetura *transformer* e das tecnologias generativas possibilitaram um avanço excepcional na área, especialmente na capacidade de fluidez. A compreensão de perguntas formuladas de diversas maneiras e a geração de respostas complexas relacionando diversos conhecimentos são bem impressionantes.

O modelo é treinado com uma série de exemplos de perguntas e respostas e aprende a gerar respostas coerentes. As respostas geradas artificialmente são sempre diferentes uma das outras e apresentam diferentes faces ou visões sobre um mesmo tema. Essa diversidade é um outro benefício dessa solução.

Ainda assim, a arquitetura *transformer* nem sempre é a melhor opção. A partir dos dados coletados, pesquisas e testes realizados, levanta-se alguns cenários em que o uso de outras soluções é mais indicada:

- **Limitação de Custo:** A arquitetura transformers requer um alto poder de processamento. Especialmente quando o fine-tuning e suas otimizações não são possíveis, o custo de treinamento pode ser um limitante.
- **Fluxo muito claro:** Quando o fluxo dos usuários de um chatbot é muito claro e tem poucas opções, em geral é mais eficiente utilizar tecnologias semelhantes ao intent chat.
- **Dados muito precisos:** A arquitetura transformers é muito poderosa na geração de dados, mas ela não é muito confiável para retratar números.
- **Alucinações e Dados Sensíveis:** Como pode ser observado nos testes realizados, nem sempre as informações da resposta estão corretas. Diferentemente do *chatbot intent* no qual as respostas são pré-definidas, na arquitetura *transformer* as respostas são geradas automaticamente e assim são mais imprevisíveis. Pior do que não conseguir responder uma pergunta é responder com dados errados e essa é uma característica da arquitetura. Isso é chamado de alucinações e acontece quando o modelo extrapola as informações disponíveis em seus dados de treinamento. Em situações onde informações erradas podem gerar impactos tremendos, no campo médico por exemplo, essa arquitetura não é recomendada. Buscando resolver esses problemas, vêm surgindo diversas pesquisas que procuram reduzir a alucinação de *chatbots*.
- **Ações Externas:** O objetivo desse modelo é gerar respostas coerentes, mas em alguns casos, como no contexto de assistentes virtuais, o usuário pode pedir para que o sistema realize uma ação em seu nome. Essa ainda é uma limitação do modelo, sendo necessário adicionar uma camada com uma implementação clássica de *chatbots* que identifica algumas ações pré-definidas e as realiza.
- **Limitação de Dados:** A aprendizagem e conseqüentemente a eficácia do *chatbot* depende da abundância e qualidade dos dados de treino. A complexidade do modelo *transformer* exige um grande número de dados de treino. Por mais que a abundância de dados de qualidade em geral resulte em uma experiência melhor para qualquer

modelo, modelos mais simples como o *chat intent* tem um resultado satisfatório com um número muito menor de dados.

Por fim, o contexto ideal para a implementação de um *chatbot* usando a arquitetura *transformer* ocorre quando há considerável poder computacional disponível, uma vasta quantidade de dados, desenvolvedores qualificados e ainda mais importante: a possibilidade dos usuários validarem as respostas sem um custos excessivos ou riscos significativos.



## Capítulo 4

### Conclusão

É evidente que a arquitetura *transformer* representa um marco significativo no desenvolvimento de *chatbots*. O impacto global do ChatGPT e de outras tecnologias de inteligência artificial generativa baseadas em NLP é inegável, sinalizando uma transformação duradoura na dinâmica da interação entre humanos e computadores, que certamente se estenderá ao futuro.

A análise detalhada dos componentes e comportamentos da arquitetura *Transformer* destacou o porquê da sua flexibilidade e adaptabilidade em ambientes de conversação dinâmicos. O desenvolvimento de *chatbots*, por sua vez, demonstrou essa capacidade na prática e mostrou a sua superioridade sobre *chatbots* clássicos.

É crucial reconhecer que, embora a arquitetura *transformer* seja uma contribuição significativa, ela não é uma solução única para todos os desafios. Este trabalho destaca que, embora tenha demonstrado notável eficácia, não é imune a dificuldades. Entretanto, é essa compreensão das limitações que motiva a continuidade da pesquisa e desenvolvimento nesta área.

O trabalho conclui ressaltando que, a partir desse grande marco em NLP e das surpreendentes capacidades evidenciadas, é essencial que a comunidade persista no aprimoramento dessa tecnologia que, sem dúvida, continuará a moldar a forma como interagimos com os computadores no futuro.



## Referências

- [AI 2023] Tuning AI. *Dataset Startups*. 2023. URL: [https://huggingface.co/datasets/TuningAI/Startups\\_V1](https://huggingface.co/datasets/TuningAI/Startups_V1) (citado na pg. 49).
- [ALAMMAR 2018] Jay ALAMMAR. “The illustrated transformer” (2018). URL: <https://jalammar.github.io/illustrated-transformer/> (citado nas pgs. 35, 37, 39, 42).
- [ALAMMAR 2021] Jay ALAMMAR. “Master positional encoding: part i” (2021). URL: <https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3>.
- [ALAMMAR s.d.] Jay ALAMMAR. “Visualizing a neural machine translation model (mechanics of seq2seq models with attention)” (). URL: <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/> (citado nas pgs. 29, 31).
- [ALEC RADFORD e SUTSKEVER 2018] Tim Salimans ALEC RADFORD Karthik Narasimhan e Ilya SUTSKEVER. “Improving language understanding by generative pre-training” (2018). URL: [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf) (citado nas pgs. 10, 45).
- [APPLE 2011] APPLE. *iPhone 4S First Weekend Sales Top Four Million*. 2011. URL: <https://www.apple.com/newsroom/2011/10/17iPhone-4S-First-Weekend-Sales-Top-Four-Million/> (citado na pg. 6).
- [BAHDANAU *et al.* 2014] Dzmitry BAH DANAU, Kyunghyun CHO e Yoshua BENGIO. “Neural machine translation by jointly learning to align and translate” (2014). URL: <https://arxiv.org/abs/1409.0473> (citado nas pgs. 31, 32).
- [BATOOL HAIDER 2020a] Hedü AI by BATOOL HAIDER. *Visual Guide to Transformer Neural Networks - (Episode 1) Position Embeddings*. 2020. URL: <https://www.youtube.com/watch?v=dichlcUZfOw>.
- [BATOOL HAIDER 2020b] Hedü AI by BATOOL HAIDER. *Visual Guide to Transformer Neural Networks - (Episode 2) Multi-Head & Self-Attention*. 2020. URL: <https://www.youtube.com/watch?v=mMa2PmYJlCo>.

- [BATOOL HAIDER 2020c] Hedu AI by BATOOL HAIDER. *Visual Guide to Transformer Neural Networks - (Episode 3) Decoder's Masked Attention*. 2020. URL: <https://www.youtube.com/watch?v=gJ9kaJsE78k>.
- [BLAKE MASTERS 2014] Peter Thiel with BLAKE MASTERS. *Zero to One: Notes on Startups*. 2014 (citado na pg. 49).
- [BLANK e DORF 2012] Steve BLANK e Bob DORF. *The Startup Owner's Manual*. 2012 (citado na pg. 49).
- [DEEPLARNINGAI 2018] DEEPLARNINGAI. *Attention Model Intuition*. 2018. URL: <https://www.youtube.com/watch?v=SysgYptB198>.
- [GENSIM 2022] GENSIM. *Gensim Word2Vec Tutorial*. 2022. URL: [https://radimrehurek.com/gensim/auto\\_examples/tutorials/run\\_word2vec.html](https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html).
- [GOODFELLOW *et al.* 2015] Ian GOODFELLOW, Yoshua BENGIO e Aaron COURVILLE. *Deep Learning*. 2015 (citado na pg. 18).
- [HOCHREITER e SCHMIDHUBER 2015] HOCHREITER e SCHMIDHUBER. “Understanding lstm networks” (2015). URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> (citado nas pgs. 23, 24).
- [JOHNSON 2017] Justin JOHNSON. “Recurrent neural networks”. In: 2017. URL: <https://www.youtube.com/watch?v=6niqTuYFZLQ> (citado nas pgs. 19, 21).
- [LIL'LOG 2018] LIL'LOG. “Attention? attention!” (2018). URL: <https://lilianweng.github.io/posts/2018-06-24-attention/#a-family-of-attention-mechanisms>.
- [MAHAJAN 2023] Viraj MAHAJAN. *ChatGPT Statistics and Facts in 2023*. 2023. URL: <https://www.notta.ai/en/blog/chatgpt-statistics#:~:text=Key%20ChatGPT%20Statistics,-If%20you%27re&text=ChatGPT%20currently%20has%20100%2B%20million,1.5%20billion%20visitors%20per%20month> (citado nas pgs. 1, 2).
- [MANNING 2017] Chris MANNING. “Word vector representations: word2vec”. In: 2017. URL: <https://www.youtube.com/watch?v=ERibwqs9p38>.
- [McCORMICK 2016] Chris McCORMICK. “Word2vec tutorial - the skip-gram model” (2016). URL: <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/> (citado na pg. 25).
- [MIKOLOV *et al.* 2013] Tomas MIKOLOV, Kai CHEN, Greg CORRADO e Jeffrey DEAN. “Efficient estimation of word representations in vector space” (2013). URL: <https://arxiv.org/abs/1301.3781>.
- [NG 2021] Andrew NG. “Sequence models complete course”. In: 2021. URL: <https://www.youtube.com/watch?v=S7oA5C43Rbc>.

## REFERÊNCIAS

- [NG s.d.] Andrew NG. *Sequence Models (Course 5 of the Deep Learning Specialization)*. URL: <https://www.coursera.org/learn/nlp-sequence-models>.
- [NIELSEN 2017] Michael A. NIELSEN. *Neural Networks and Deep Learning*. 2017.
- [OLAH 1997] Christopher OLAH. “Long short-term memory” (1997). URL: <https://www.bioinf.jku.at/publications/older/2604.pdf>.
- [OPENAI 2022] OPENAI. *Introducing ChatGPT*. 2022. URL: <https://openai.com/blog/chatgpt> (citado nas pgs. 45, 50).
- [OPENAI 2023] OPENAI. *Assistants API*. 2023. URL: <https://platform.openai.com/docs/assistants/overview> (citado na pg. 51).
- [RIES 2011] Eric RIES. *The Lean Startup*. 2011 (citado na pg. 49).
- [SHAWAR e ATWELL 2002] SHAWAR e ATWELL. *A comparison between Alice and Elizabeth chatbot systems*. 2002 (citado na pg. 5).
- [SINHA 2018] Nimesh SINHA. “Understanding lstm and its quick implementation in keras for sentiment analysis” (2018). URL: <https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47>.
- [TURING 1950] Alan TURING. *Computing Machinery and Intelligence*. 1950 (citado na pg. 4).
- [VASWANI *et al.* 2017] Ashish VASWANI *et al.* “Attention is all you need” (2017). URL: <https://arxiv.org/abs/1706.03762> (citado nas pgs. 10, 34, 35).
- [VOITA s.d.] Lena VOITA. “Sequence to sequence (seq2seq) and attention” (). URL: [https://lena-voita.github.io/nlp\\_course/seq2seq\\_and\\_attention.html](https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html).
- [W 2023] Glavin W. *Dataset Startup Interviews*. 2023. URL: <https://huggingface.co/datasets/Glavin001/startup-interviews> (citado na pg. 49).
- [WEIZENBAUM 1966] Joseph WEIZENBAUM. *ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine*. 1966 (citado na pg. 3).
- [ZHANG *et al.* 2023] Aston ZHANG, Alexander J. SMOLA, Zachary LIPTON e Mu LI. *Dive into Deep Learning*. 2023.
- [ZIPDO 2023] ZIPDO. *Essential Call Center Statistics In 2023*. 2023. URL: <https://zipdo.co/statistics/call-center/>.