UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Bacharelado em Sistemas de Informação
Walter Magno Lopes

DOS MODELOS DE REDES NEURAIS CLÁSSICOS AO MODELO TRANSFORMER: uma aplicação de processamento de linguagem natural em chatbot usando BERT

Diamantina, MG
2024

Walter Magno Lopes

DOS MODELOS DE REDES NEURAIS CLÁSSICOS AO MODELO TRANSFORMER: uma aplicação de processamento de linguagem natural em chatbot usando BERT

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Leonardo Lana de Carvalho

Diamantina, MG
2024

Dedico este trabalho à toda a minha família, com um carinho especial voltado aos meus pais e avós, Maria e João Mendes, cujo amor e apoio foram fundamentais.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me fortalecer, inspirar e guiar meus passos ao longo da minha vida com infindáveis bênçãos.

Um agradecimento especial aos meus pais, Cleuza e José, cujo apoio incondicional e incentivo foram pilares durante minha graduação e aos meus irmãos Carlos Daniel e Talita que me inspiraram no desenvolvimento deste trabalho.

Ao meu orientador, Leonardo Lana, expresso minha sincera gratidão por compartilhar valiosos conhecimentos e insights enriquecedores ao longo deste percurso.

Aos amigos que caminharam comigo, especialmente aqueles da graduação e da minha jornada profissional, agradeço por trocarmos ideias que enriqueceram minha formação acadêmica e pessoal.

Gostaria de expressar minha gratidão à Universidade Federal dos Vales Jequitinhonha e Mucuri e a todos os membros do corpo docente e também à todos os membros do corpo técnico-administrativo.

Por fim, agradeço a todos que direta ou indiretamente contribuíram para a realização desse trabalho.

RESUMO

A suplementação alimentar tem um papel muito importante para os praticantes de esporte, ajudando a potencializar os resultados com as atividades físicas. Este trabalho realiza o desenvolvimento de um chatbot focado em suplementação esportiva, utilizando técnicas avançadas de processamento de linguagem natural (NLP). Visando possibilitar a compreensão contextualizada das perguntas e dúvidas dos usuários sobre suplementação, o chatbot foi desenvolvido com base no modelo BERT, pertencente à arquitetura Transformer, uma arquitetura de modelos de *deep learning* que tem sido utilizada para problemas de NLP, incluindo a tarefa de perguntas e respostas. Foi realizado a integração do modelo BERT com uma base de dados montada com a especialidade em suplementação esportiva, também foi utilizado técnicas RAG para possibilitar a relevância das respostas. Este trabalho faz uma revisão dos principais modelos de redes neurais, apresentando a evolução destes modelos de IA até a arquitetura Transformer. É feito o detalhamento da arquitetura do sistema de chatbot, apresentamos os desafios enfrentados, as soluções implementadas e a avaliação do desempenho do chatbot.

Palavras-chave: BERT. Chatbot. LLM. NLP. RAG. Redes neurais.

ABSTRACT

Food supplementation plays a very important role for those who practice sports, helping to enhance results from physical activities. This work develops a chatbot focused on sports supplementation, using advanced natural language processing (NLP) techniques. Aiming to enable a contextualized understanding of users' questions and doubts about supplementation, the chatbot was developed based on the BERT model, belonging to the Transformer architecture, an *deep learning* model architecture that has been used for NLP problems, including the question and answer task. The BERT model was integrated with a database created with the specialty in sports supplementation, RAG techniques were also used to enable the relevance of the responses. This work reviews the main neural network models, presenting the evolution of these AI models to the Transformer architecture. The architecture of the chatbot system is detailed, we present the challenges faced, the solutions implemented and the evaluation of the chatbot's performance.

Keywords: BERT. Chatbot. LLM. NLP. RAG. Neural networks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BERT	Bidirectional Encoder Representations from Transformers
Q&A	Question and Answering
NLP	Natural Language Processing
BiRNN	Bidirectional Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
GPT	Generative Pre-trained Transformer
MLM	Masked Language Model
CLS	Classification
PAD	Padding
SEP	Separator
UNK	Unknown
RAG	Retrieval-Augmented Generation
IA	Inteligência Artificial
LLM	Large Language Model
SQuAD	Stanford Question Answering Dataset
BrWaC	Brazilian Web as Corpus
RAM	Random Access Memory
CPU	Central Processing Unit
GPU	Graphics Processing Unit
FAISS	Facebook AI Similarity Search
PDF	Portable Document Format
BM25	Best Matching 25
KNN	K-Nearest Neighbors
IDF	Inverse Document Frequency
TLS	Transport Layer Security

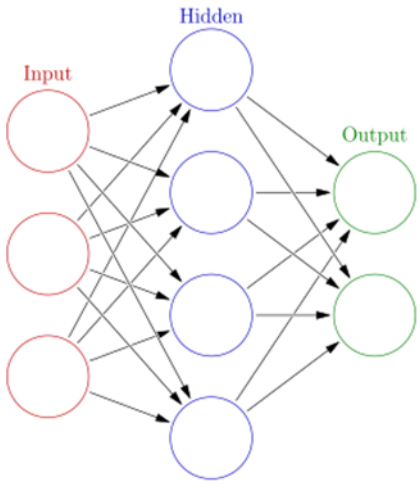
SUMÁRIO

1 INTRODUÇÃO

O *deep learning* faz parte da área de aprendizado de máquina e utiliza redes neurais artificiais dispostas em várias camadas, conforme demonstrado pela Figura 1, para entender e modelar fenômenos cognitivos complexos, possibilitando gerar sistemas de informação para diversas aplicações.

A referida subárea tem sido aplicada em cenários como geração de linguagem natural, visão computacional e reconhecimento de fala (AHMED et al., 2023). As redes neurais profundas são compostas por diversas camadas de neurônios artificiais, que são capazes de aprender representações de dados em níveis variados de abstração.

Figura 1 – Representação da Arquitetura de uma Rede de *Deep Learning*



Fonte: McCullum (2020)[1]

Um dos principais diferenciais do *deep learning* em relação a outras técnicas de aprendizado de máquina é a sua capacidade de processar dados não estruturados, como texto e imagens, sem a necessidade de pré-processamento manual intensivo (HOLDSWORTH; SCAPICCHIO, 2024)[2]. Este processo é realizado através de algoritmos de aprendizado profundo que utilizam técnicas como a retropropagação e o gradiente descendente para ajustar os pesos das conexões neurais e melhorar a precisão das previsões.

As funções de ativação, como sigmoid, desempenham um papel crucial na operação das redes neurais, permitindo a introdução de não-linearidades que aumentam a capacidade dos modelos de aprender padrões complexos de respostas (McCULLUM, 2020)[1].

[1] McCULLUM, N. *Deep learning neural networks explained in plain English*. FreeCodeCamp, 2020. Disponível em: `https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/`. Acesso em: 09 jun. 2024.

[2] HOLDSWORTH, J.; SCAPICCHIO, M. *What is deep learning?* IBM, 2024. Disponível em: `https://www.ibm.com/cloud/learn/deep-learning`. Acesso em: 09 jun. 2024.

O surgimento do pioneiro neurônio formal de McCulloch e Pitts (1943) marca o início do desenvolvimento das redes neurais explorando a ideia do "tudo ou nada"para caracterizar a atividade nervosa, usando a lógica proposicional como instrumento de modelagem. Podemos dizer que o trabalho de McCulloch e Pitts se insere no movimento da cibernética, sendo este um campo interdisciplinar de investigação da mente, do sistema nervoso e da comunicação em animais (humanos e não humanos) visando caracterizá-los como uma máquina e também de formalizar os processos dessa máquina usando a lógica e a matemática, com um interesse especial por sistemas não lineares (ROSENBLUETH; WIENER; BIGELOW, 1943; WIENER, 1961; CARVALHO; PEREIRA; COELHO, 2016). Eles demonstraram como operações lógicas podiam ser aprendidas por redes de neurônios artificiais (MCCULLOCH; PITTS, 1943).

O modelo contando com um limiar de ativação baseado no funcionamento do sistema nervoso, visando seus processos de aprendizagem, foi uma característica diferencial importante sobre modelos anteriores, como os circuitos lógicos de Shannon (1938).

Em seguida, a pesquisa de Rosenblatt sobre o Perceptron em 1957 e 1961 marcou um avanço significativo. Ele introduziu um modelo de aprendizado que poderia adaptar seus pesos sinápticos em resposta a estímulos externos, estabelecendo assim os alicerces para o aprendizado não-supervisionado e supervisionado em redes neurais (ROSENBLATT, 1957, 1962).

Em 1969, Minsky e Papert, em suas meticulosas análises, apontam as limitações estruturais intrínsecas aos Perceptrons, destacando que, apesar de inovadores no momento de sua criação, estes modelos enfrentam severas restrições ao lidar com funções não-lineares e ao manipular padrões geométricos complexos. Eles argumentam que a simplicidade dos Perceptrons, que operam primariamente através de funções lineares ajustáveis por pesos, é insuficiente para tratar da complexidade dos padrões de conectividade ou para executar tarefas de classificação que requerem uma análise contextual aprofundada dos dados (MINSKY; PAPERT, 1969). Esta perspectiva crítica desafia a visão inicialmente otimista em torno do Perceptron de Rosenblatt, evidenciando que as barreiras teóricas e práticas destes sistemas devem ser superadas para que possam efetivamente funcionar em ambientes de aprendizado supervisionado e não-supervisionado, que dependem do entendimento detalhado das interações entre múltiplos inputs.

Expandindo esse conhecimento, Hopfield em 1982 apresentou redes neurais como sistemas físicos com propriedades emergentes coletivas. A partir da aprendizagem auto-associativa e hétero-associativa já presentes no modelo de Rosenblatt, Little (1974) e Hopfield (1982) apresentam uma forma de rede neural recorrente. Elas poderiam ser usadas para criar memórias associativas, contribuindo para a compreensão de como as redes neurais podem armazenar e recuperar informações de maneira eficiente (HOPFIELD, 1982).

Em 1986, Rumelhart, Hinton e Williams apresentaram o algoritmo de retropropagação que permitiu o treinamento de maneira eficiente de redes neurais multicamadas, resolvendo o problema da representação de funções não-lineares, como no caso do XOR (RUMELHART;

HINTON; WILLIAMS, 1986). Com esse avanço, abriu-se caminho para o desenvolvimento de redes neurais profundas (*deep learning*), que possuem características importantes para desenvolvimento de modelos neurais robustos.

Esses trabalhos foram fundamentais para a compreensão e evolução das redes neurais que passaram a se destacar na capacidade de aprender, adaptar e realizar tarefas complexas de processamento de informações em diversos cenários. Essas características são essenciais no contexto emergente das redes neurais.

Os modelos mais clássicos de redes neurais, como o modelo Perceptron, ADALINE, e suas variações, serviram como base para o desenvolvimento de algoritmos avançados de *deep learning*. Sendo o Perceptron um dos algoritmos pioneiros na classificação linear, o ADALINE (Adaptive Linear Neuron) algoritmo sucessor do Perceptron, estabeleceu a possibilidade dos ajustes de pesos baseados no gradiente descendente. Os modelos clássicos de redes neurais possuíam as limitações e a incapacidade de resolver problemas não linearmente separáveis, como o problema das operações XOR. Apesar disso, os modelos foram fundamentais para o ponto de partida de pesquisas de mecanismos de aprendizado mais complexos, como o feedback e a inibição, e a descoberta de funções de ativação que permitiram o surgimento dos modelos Transformers, os quais aproveitam o paralelismo e a atenção global para capturar padrões complexos em sequências de dados.

Chatbots são uma classe de aplicações de grande valor no atendimento ao usuário e na resolução de dúvidas no que se refere a produtos ou serviços. Nesse sentido, o modelo BERT se torna ótimo para ser utilizado em chatbot para tarefas de perguntas e respostas (Q&A) devido a capacidade do modelo compreender elementos textuais e a semântica da linguagem natural com profundidade. O BERT pode analisar o texto em duas direções em uma frase sendo elas esquerda e direita, estabelecendo a compreensão contextual de palavras e frases. No contexto de chatbots, a compreensão da pergunta em diversos contextos é fundamental o que possibilita obter respostas precisas e relevantes, resultando em ótimas experiências ao usuário. O chatbot no contexto de suplementação esportiva possui relevância em oferecer respostas personalizadas às pesquisas dos usuários. O chatbot pode ter um papel importante no momento da escolha dos suplementos, dosagens recomendadas e também influenciar no desenvolvimento de uma dieta equilibrada, criação de hábitos saudáveis e pode contribuir para a maximização dos resultados no desempenho esportivo.

1.1 Objetivos

1.1.1 Objetivo Geral

Nosso objetivo foi estudar os principais modelos de redes neurais, dos clássicos aos contemporâneos, destacando a arquitetura Transformer. Baseado nos Transformers, aplicamos o BERT em processamento de linguagem natural (NLP) para criar um chatbot de perguntas e respostas (Q&A) no contexto de suplementação esportiva.

1.1.2 Objetivos Específicos

I Estudo da evolução dos modelos de redes neurais, dos modelos clássicos ao modelo Transformer.

II Estudar a arquitetura Transformer aplicada ao modelo BERT.

III Desenvolver pipelines de processamento de dados.

IV Desenvolver um protótipo de Chatbot.

2 REFERENCIAL TEÓRICO

Neste capítulo será apresentado a fundamentação teórica utilizada como base para a realização deste trabalho.

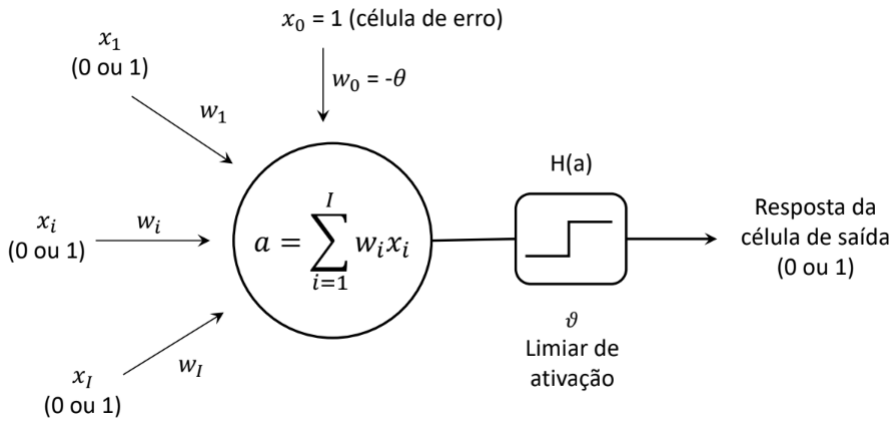
2.1 Memória Associativa

2.1.1 Memória Auto-Associativa com Aprendizagem Hebbiana

A regra de Hebb, um conceito-chave na neurociência, sugere que "neurônios que disparam juntos, conectam-se de forma simultânea", o que Rosenblatt aplicou ao ajuste dos pesos em redes neurais (ROSENBLATT, 1957). Rosenblatt, na obra "*The Perceptron*", descreve o modelo básico do Perceptron e o potencial do modelo em realizar tarefas de classificação (ROSENBLATT, 1957). Ele integra a regra de Hebb no ajuste de pesos sinápticos, uma abordagem básica para a aprendizagem de máquina.

A Figura 2 demonstra a estrutura de um neurônio, com destaque para o fluxo das informações até a geração de resposta do neurônio. Em "*Principles of Neurodynamics*", é feito um detalhamento desse conceito, explorando a capacidade das redes neurais de armazenar informações de maneira eficiente através da auto-associação (ROSENBLATT, 1962).

Figura 2 – Representação de um Neurônio Artificial



Fonte: Abdi e Valentin (2006, p. 230). [3]

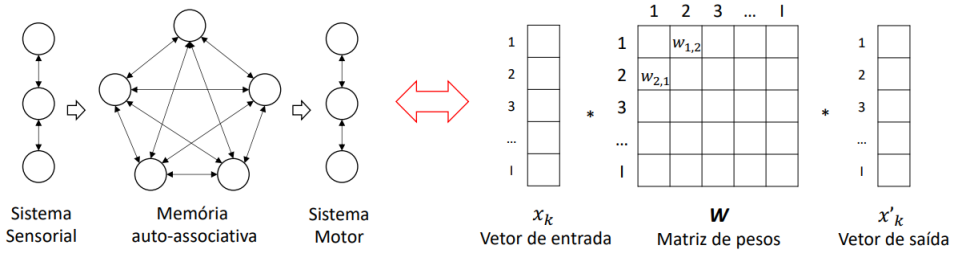
A Memória Auto-Associativa, conforme descrita por Rosenblatt, permite que a rede reconheça e lembre padrões de entrada (ROSENBLATT, 1962). A Aprendizagem Hebbiana, nesse contexto, tem o papel de reforçar conexões sinápticas que são consistentemente usadas (ROSENBLATT, 1962).

[3] Tradução de Carvalho (2022).

A obra de Rosenblatt apresenta também a forma como a memória auto-associativa pode ser afetada por erros ou entradas incompletas, sendo resistente a esses desafios (ROSENBLATT, 1962). Esta abordagem da memória tem implicações muito importantes para a compreensão da aprendizagem e de como a memória funciona em sistemas artificiais e também em sistemas biológicos (ROSENBLATT, 1962). O modelo de Rosenblatt indica a possibilidade de generalização, ao mostrar a flexibilidade do sistema (ROSENBLATT, 1962).

A junção da regra de Hebb com o Perceptron de Rosenblatt estabelece uma conexão entre a teoria neural e a computação (ROSENBLATT, 1957). A Figura 3 representa a arquitetura do modelo de Memória Auto-Associativa.

Figura 3 – Memória Auto-Associativa



Fonte: Abdi e Valentin (2006, p. 59). [4]

Para ilustrar a aplicação desses conceitos, são apresentadas a seguir algumas equações:

1. A regra de Hebb define a intensidade da conexão entre duas células i e j como proporcional à ativação das mesmas células:

$$w_{i,j} = \gamma(x_i * x_j) \tag{1}$$

   Nesta equação, $w_{i,j}$ representa o peso da conexão entre as células i e j, x_i e x_j são as ativações das células i e j, respectivamente, e γ é uma constante de proporcionalidade.

2. A fórmula para a assimilação e memorização de estímulos é:

$$W = \sum_{k=1}^{K} x_k x_k^T = XX^T \tag{2}$$

   Aqui, W denota a matriz de pesos, X representa o conjunto de estímulos, e k é o índice que percorre todos os estímulos K.

[4] Adaptado por Carvalho (2022).

3. Inicialmente, o algoritmo de aprendizagem de Hebb começa com todas as conexões entre células tendo valores nulos:

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

   Neste caso, W é uma matriz de pesos inicializada com zeros, indicando que não há conexões estabelecidas inicialmente.

4. A fórmula para armazenar um rosto na memória na primeira etapa é:

$$W_{[1]} = W_{[0]} + x_1 t_1^T \tag{4}$$

   Onde x_1 é a representação do rosto e $W_{[0]}$ é a matriz de pesos atualizada a partir da matriz inicial.

5. A fórmula para armazenar um rosto na segunda etapa é:

$$W_{[2]} = W_{[1]} + x_2 t_2^T \tag{5}$$

   De forma semelhante, x_2 representa o rosto e $W_{[1]}$ é a matriz de pesos atualizada a partir da primeira etapa.

6. Após a conclusão da décima etapa de armazenamento, temos:

$$W_{[10]} = W_{[9]} + x_{10} x_{10}^T \tag{6}$$

   Neste ponto, $W_{[10]}$ é a matriz de pesos após a décima atualização, e x_{10} é a representação do décimo rosto armazenado.

7. Multiplicar o vetor que representa a face pela matriz de conexão W recupera uma face da memória:

$$x_1' = W x_1 \tag{7}$$

   Onde x_1' é o vetor de saída, representando a face recuperada, e x_1 é o vetor de entrada.

8. A qualidade da resposta é avaliada através do cosseno entre x e y':

$$\cos(x_1', x_1) = \frac{x_1'^T x_1}{\|x_1'\| \|x_1\|} \tag{8}$$

   Nesta equação, a similaridade entre o vetor de entrada x_1 e o vetor de saída x_1' é medida pelo cosseno do ângulo entre eles.

9. Para reter o conjunto de faces aprendidas de uma só vez, multiplica-se a matriz X pela matriz de conexões W:

$$X' = WX \tag{9}$$

   Aqui, X é a matriz que representa todas as faces aprendidas e W é a matriz de conexões.

As equações representam um exemplo de aprendizado e recuperação das faces. Elas demonstram como a memória auto-associativa e a aprendizagem Hebbiana podem ser aplicadas para armazenar e recuperar padrões.

2.1.2 Memória Auto-Associativa com Aprendizagem de Widrow-Hoff

ADALINE é uma máquina de classificação de padrões que ilustra de forma pragmática a aprendizagem artificial. Possui a capacidade de aprendizagem e adaptação para classificação de padrões, ADALINE demonstra desempenho e a versatilidade da aprendizagem artificial. Na sua forma operacional, ADALINE atua como um circuito de comutação adaptativo onde os sinais de entrada binários são combinados linearmente e depois quantizados. Ele adapta esse comportamento para minimizar efetivamente os erros de classificação por meio de um método de gradiente iterativo cuja otimização foi introduzida por Widrow e Hoff (1960).

A base teórica para a competência do ADALINE é apresentada pela teoria da adaptação estatística. Segundo esta teoria, o termo "má adaptação"refere-se à razão entre as probabilidades de erros após a adaptação e os erros de um sistema mais adaptado que o atual (WIDROW; HOFF, 1960). Esse critério de medição permite avaliar a eficácia da aprendizagem adaptativa de forma quantitativa.

A seguir, é feito o detalhamento do processo de aprendizagem do ADALINE utilizando a regra de Widrow e Hoff (1960):

1. Inicialização da Aprendizagem

   A matriz de pesos é inicializada como uma matriz nula:

   $$W_0 = \text{Matriz Nula} \tag{10}$$

   Esta etapa visa garantir que não haja influências iniciais indesejadas nos cálculos subsequentes.

2. Atualização da Matriz de Conexões (WIDROW; HOFF, 1960)

   A matriz de conexões é atualizada iterativamente para minimizar o erro:

   $$W_{[n+1]} = W_{[n]} + \eta(x_k - W_{[n]}x_k)x_k^T \tag{11}$$

   onde W_n é a matriz de conexões na iteração n, η é a constante de aprendizagem, x_k é o estímulo escolhido e x_k^T sua transposta.

3. Aprendizagem por Pacote (Batch Learning)

   Quando utilizando aprendizado em lote, a matriz de conexões é atualizada considerando todas as entradas de uma vez:

   $$W_{[n+1]} = W_{[n]} + \eta(X - W_{[n]}X)X^T \tag{12}$$

   onde X é a matriz de entradas e $W_{[n]}$ a matriz de pesos na iteração n.

4. Lembrar os Rostos

   A saída da rede é calculada multiplicando a matriz de pesos pela matriz de entrada:

$$Y' = W \cdot X \tag{13}$$

   onde Y' é a saída da rede, W é a matriz de pesos e X é a matriz de entrada.

5. Cálculo do Erro

   O erro é a diferença entre a entrada X e a saída X':

$$E = X - X' \tag{14}$$

6. Cálculo da Matriz de Correção

   A matriz de correção é calculada como:

$$\Delta W = \eta \cdot X \cdot E^T \tag{15}$$

   onde ΔW é a matriz de correção.

7. Correção da Matriz de Pesos

   Por conseguinte, a matriz de pesos é atualizada adicionando a matriz de correção:

$$W_{n+1} = W_n + \Delta W \tag{16}$$

Essas etapas descrevem como o ADALINE ajusta seus pesos para minimizar os erros de classificação, demonstrando a eficácia e adaptabilidade do modelo na aprendizagem artificial.

2.1.3 Memória Hetero-Associativa

Hopfield (1982) discorre sobre a natureza das redes neurais em seu influente artigo. Ele discorre como as redes podem ser usadas para simular sistemas físicos e processos de memória. Por mais que o objetivo principal de Hopfield não estivesse diretamente relacionado à memória linear heteroassociativa, os princípios discutidos por ele são importantes para todas as tentativas de compreensão desse tipo de memória.

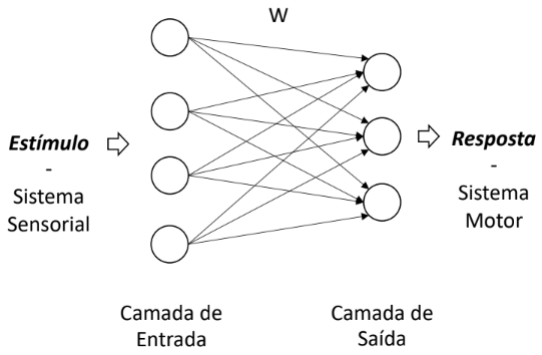
O modelo de memória em redes neurais denominado memória linear heteroassociativa - introduzido por Hopfield (1982) - permite que diferentes tipos de informação sejam associados de forma linear. Isso implica que a rede pode associar dois padrões separados de tal forma que a apresentação de um padrão produza o outro padrão. Tal forma de memória encontra uso particular em aplicações que necessitam de mapeamento de entrada para saída específica, como durante a tradução ou categorização, devido a possuir recursos únicos, ao contrário da

memória autoassociativa, que lida com padrões semelhantes, mas heteroassociativos. A memória lida com padrões distintos.

O modelo linear segundo Hopfield (1982) é tal que as transformações que são feitas pela rede nos dados de entrada são lineares, o que pode ser útil em alguns casos onde precisamos ter uma análise e um entendimento de como a rede funciona. Por outro lado, esta linearidade também pode nos limitar porque as relações entre os dados podem ser complexas ou não lineares, mas é importante para nós aprendermos sobre elas através de sua pesquisa, que forma uma base para compreender como as redes neurais podem ser configuradas para realizar tarefas sofisticadas de memória.

A rede pode associar dois padrões distintos de tal forma que quando um padrão é apresentado, o outro padrão é produzido. A Figura 4 representa a arquitetura do modelo de Memória Linear Hetero-Associativa.

Figura 4 – Representação da Memória Linear Hétero-Associativa



Fonte: Adaptado de Carvalho (2022) e Abdi e Valentin (2006)

As equações a seguir descrevem os principais passos para a representação, aprendizagem e recuperação de informações em no contexto de memória hetero-associativa:

1. Representação de rostos e nomes como vetores e matrizes:

   Os rostos e nomes são representados por vetores e matrizes, possibilitando a operação matemática dentro do sistema de memória:

$$\text{Rostos:} \quad \begin{pmatrix} 1 & -1 & -1 & 1 & -1 \\ \vdots & & & & \\ -1 & -1 & -1 & 1 & -1 \end{pmatrix} \tag{17}$$

$$\text{Nomes:} \quad \begin{pmatrix} 1 & -1 & -1 & 1 & -1 \\ \vdots & & & & \\ -1 & -1 & 1 & 1 & 1 \end{pmatrix} \tag{18}$$

Onde cada linha da matriz representa um vetor associado a um rosto ou nome específico, com valores binários (1 e -1) codificando características distintas dos rostos e nomes.

2. Regra de Hebb para aprendizagem em redes neurais:

   A regra de Hebb é onde a força das conexões sinápticas é ajustada com base na atividade simultânea dos neurônios pré e pós-sinápticos:

$$W = \gamma \sum_{k=1}^{k} x_k t_k^T \tag{19}$$

   W é a matriz de pesos sinápticos, γ é uma constante de aprendizagem, x_k é o vetor de entrada (representando, por exemplo, um rosto), e t_k^T é o vetor de saída transposto (representando o nome correspondente).

3. Lembrança de memória:

   A recuperação ou lembrança de informações armazenadas na memória é realizada utilizando a matriz de pesos ajustada durante a fase de aprendizagem:

$$t'_k = W^T x_k \tag{20}$$

   Neste caso, t'_k é o vetor de saída recuperado, W^T é a matriz de pesos transposta, e x_k é o vetor de entrada.

4. Avaliação de resposta usando similaridade de cosseno:

   Para avaliar a precisão da recuperação da memória, utiliza-se a similaridade de cosseno, que mede a semelhança entre os vetores de saída recuperado e esperado:

$$\cos(t'_k, t_k) = \frac{(t'_k)^T t_k}{\|t'_k\| \|t_k\|} \tag{21}$$

   Nesta equação, $\cos(t'_k, t_k)$ representa a similaridade de cosseno entre o vetor recuperado t'_k e o vetor esperado t_k, onde $\|\cdot\|$ denota a norma (magnitude) dos vetores. Essa medida varia de -1 a 1, indicando a proximidade angular entre os vetores no espaço multidimensional.

2.2 ADALINE e Suas Variações

2.2.1 ADALINE (Adaptive Linear Neuron)

Um modelo de classificação de padrões adaptativo, chamado ADALINE (*Adaptive Linear Neuron*), foi desenvolvido por Widrow e Hoff (1960) como um conceito fundamental em aprendizado de máquina. Esse modelo recebe padrões geométricos na fase de treinamento e aprende com eles, adaptando-se para classificar tanto padrões originais quanto distorcidos (WIDROW; HOFF, 1960).

O funcionamento do ADALINE se baseia em um circuito de comutação adaptativo com entradas e saída binárias, que combina linearmente os sinais de entrada e os quantiza. Para encontrar configurações que minimizem os erros de classificação, são usados métodos iterativos de gradiente (WIDROW; HOFF, 1960).

O ADALINE tem algumas limitações, como só poder lidar com funções de comutação linearmente separáveis. Mas também tem utilidade para reconhecimento de padrões e armazenamento de informações (WIDROW; HOFF, 1960). Além disso, redes de ADALINEs podem superar essas limitações e ser aplicadas em sistemas de reconhecimento de padrões, armazenamento de informações e sistemas lógicos auto-reparáveis (WIDROW; HOFF, 1960).

A quantização no ADALINE é feita pelo operador que emprega vários níveis de quantização para colocar os padrões em diferentes categorias, através do mesmo procedimento de adaptação (WIDROW; HOFF, 1960). A adaptação ADALINE compreende uma rotina de caça iterativa, onde o erro é identificado e todos os pesos são ajustados para minimizá-lo (WIDROW; HOFF, 1960). Este processo é também passível de descrição mecânica e pode ser automatizado em sistemas físicos.

Para melhor contextualização, serão apresentadas a seguir três equações fundamentais que descrevem o funcionamento do ADALINE:

1. Resposta do ADALINE:

   Para cada vetor de entrada x, a resposta do ADALINE é obtida multiplicando a ativação das células de entrada pelo vetor de conexões w:

   $$O = W^T X \tag{22}$$

   onde O é a resposta efetiva e W o vetor de pesos.

2. Cálculo do Erro:

   O erro E para a resposta O é a diferença entre a resposta teórica T e a resposta efetiva O:

   $$E = T - O \tag{23}$$

   onde T é a resposta teórica e E é a resposta efetiva.

3. Aprendizagem por Pacotes:

   A regra delta para ADALINE é aplicada para aprendizagem por pacotes:

   $$\Delta W = \eta (E X^T)^T = \eta X E^T \tag{24}$$

   onde η é a taxa de aprendizagem e E representa a matriz de erro. O ajuste dos pesos é calculado multiplicando a matriz de entrada X pela matriz de erro transposta E^T. Este processo iterativo ajusta todos os pesos de forma a minimizar o erro, permitindo que o ADALINE aprenda a partir dos padrões apresentados.

2.2.2 ADALINE Logística

2.2.2.1 ADALINE: Neurônio Linear Adaptativo

O ADALINE foi desenvolvido em 1960 por Bernard Widrow juntamente com seu aluno de graduação Ted Hoff. O modelo foi descrito no relatório técnico sobre circuitos de comutação adaptativos escrito por Widrow com a participação de Hoff (WIDROW; HOFF, 1960). Este modelo representa uma contribuição muito importante na história das redes neurais. O ADALINE funciona como uma rede neural de camada única com uma função de ativação linear, permitindo que o ADALINE não apenas lide com tarefas de classificação, mas também faça previsões com precisão, sendo superior ao Perceptron linear que utiliza a função degrau. A função degrau é uma função de ativação que produz uma saída binária (0 ou 1) com base em um limiar fixo.

Conforme a definição formal, a atividade a de um neurônio de McCulloch e Pitts (1943) conectado a I neurônios de entrada pode ser obtida por:

$$a = \sum_{i=1}^{I} w_i x_i = w_1 x_1 + w_2 x_2 + \ldots + w_i x_i + \ldots + w_I x_I \tag{25}$$

onde:

- x_i representando o estado de ativação dos neurônios de entrada.
- w_i é a intensidade da conexão entre o i-ésimo neurônio de entrada e o neurônio de McCulloch e Pitts (1943).

A resposta $h(a)$ é dada por:

$$h(a) = \begin{cases} 0, & \text{para } a \leq \vartheta \\ 1, & \text{para } a > \vartheta \end{cases} \tag{26}$$

Aqui, ϑ representa o valor limiar.

2.2.2.2 Função de ativação e custo em ADALINE

Como componente central, o ADALINE conta com um tipo de função de ativação: uma função de ativação linear ou uma função de custo. A função de ativação linear é uma função identidade que produz na saída o mesmo valor da ativação. Já a função sigmóide usada na regressão logística calcula a saída tendo como entrada a ativação. Esse modo de operar, bastante simples, facilita a compreensão do comportamento emergente de redes neurais de classificação mais complexas. Ao utilizar essa técnica, o ADALINE logístico tem o desempenho melhorado em comparação com o ADALINE clássico, utilizando a função logística, descrita pela Equação 27, para calcular a saída com base na ativação:

$$\text{Função logística:} \quad f(x) = \frac{1}{1 + e^{-x}} \tag{27}$$

A função de custo mede o erro como a soma dos quadrados das diferenças entre as previsões do modelo e os valores reais, permitindo que o modelo ajuste os pesos de forma a minimizar esses erros. Essa função representa um avanço em relação ao Perceptron (SURESH, 2023)[5].

2.2.2.3 Aprendizado do ADALINE

O mecanismo de aprendizagem no ADALINE possibilita adaptar pesos e minimizar a função de custo. Gradiente descendente é o nome desse processo de otimização, que forma o núcleo de muitos algoritmos de aprendizado de máquina. O valor dos pesos é alterado levando em consideração o gradiente da função de custo em relação ao peso, porque dá uma indicação de como a alteração de um peso alterará o resultado geral - uma característica que não foi considerada no Perceptron, onde apenas um simples cálculo do erro formou a base para a mudança de pesos (WIDROW; HOFF, 1960).

1. Regra de aprendizagem de Widrow e Hoff (1960) para uma ADALINE logística:

$$o = f(a) = \frac{1}{1 + e^{-a}} \tag{28}$$

   onde e representa uma constante matemática que é a base dos logaritmos naturais. É chamado de número de Euler, número de Napier, número neperiano e outros. O valor aproximado é $2,71828182845904523536028 7$.

2. O cálculo do gradiente e_k^2 é feito dessa maneira:

$$\begin{aligned} \frac{\partial e_k^2}{\partial w} &= \nabla e_k^2 \\ &= \frac{\partial e_k^2}{\partial (w^T x)} \frac{\partial f(w^T x)}{\partial (w^T x)} \frac{\partial (w^T x)}{\partial w} \\ &= -2[t_k - f(w^T x)] f'(w^T x) x^T \\ &= -2(t_k - o_k) f'(a_k) \\ &= -2(t_k - o_k) o_k (1 - o_k) x^T \end{aligned} \tag{29}$$

3. Como o objetivo é minimizar o erro quadrático e_k^2, os parâmetros do vetor w são modificados no sentido inverso do gradiente. Assim, Δw na apresentação do estímulo k é:

[5] SURESH, S. K. Understanding Adaline. 2023. Disponível em: `https://medium.com/mlearning-ai/understanding-adaline-da79ab8bbc5a`. Acesso em: 14 dez. 2023.

$$\Delta w = -\eta \nabla e_k^2 = \eta(t_k - o_k)o_k(1 - o_k)x_k^T \tag{30}$$

onde e_k^2 representa o erro quadrático associado ao estímulo k, definido como a diferença ao quadrado entre a saída desejada t_k e a saída real o_k:

$$e_k^2 = (t_k - o_k)^2 \tag{31}$$

4. Para a aprendizagem por pacote:

$$\Delta W = \eta \left[E \odot O \odot (1 - O)\right] * (X^T)^T \tag{32}$$

$$= \eta X * \left\{\left[E \odot O \odot (1 - O)\right]^T\right\} \tag{33}$$

2.2.2.4 ADALINE e a Regressão Logística

ADALINE utiliza função de ativação linear, enquanto a regressão logística usa função sigmoide. Esta divergência implica comportamentos distintos para estes modelos. A regressão logística é boa para problemas de classificação binária onde a saída é categórica enquanto a linearidade do ADALINE a torna adequada para casos onde a saída é um valor contínuo (WIDROW; HOFF, 1960).

2.2.3 Aplicações Práticas e Limitações

Devido à forma como foi projetado, o ADALINE pode ser usado em diversas aplicações, como reconhecimento de padrões e previsão de dados. Contudo, ser direto também tem suas desvantagens. Por exemplo, ele não consegue resolver problemas que não sejam linearmente separáveis - uma limitação que, no entanto, pode ser compensada pela introdução de múltiplas unidades ADALINE em uma rede. Isso permite a modelagem de relacionamentos não lineares mais complexos (WIDROW; HOFF, 1960).

2.3 Perceptron e Problemas Lógicos

Frank Rosenblatt deu origem ao Perceptron quando escreveu o material "The Perceptron: A Perceptive and Recognizing Automaton (Project PARA)"em 1957 - uma contribuição marcante para o campo das redes neurais. Este mecanismo auto-operacional destinado à percepção e reconhecimento é um modelo inovador na imitação das atividades cognitivas humanas e de outros animais. Ao contrário dos sistemas determinísticos baseados em regras, o Perceptron funciona com base em princípios probabilísticos; consiste em um sistema em rede que recebe informações do ambiente, aprende padrões e armazena esses padrões para reconhecê-los posteriormente, respondendo finalmente com base nos dados de padrões armazenados (ROSENBLATT, 1957).

2.3.1 Perceptron: Fundamentos e Funções Lógicas

O Perceptron consiste em três partes principais que trabalham juntas para lidar e reagir aos estímulos. A primeira parte é o S-System (ou sistema sensorial) que se parece com um raster de TV e compreende sensores que coletam estímulos externos em cada ponto sensorial para o A-System por meio de conexões que podem ser positivas ou negativas - excitatórias ou inibitórias, respectivamente (ROSENBLATT, 1957).

Uma parte vital, o A-System. Significando Sistema de Associação, ele atua como intermediário entre o Sistema S-System e o Sistema R-System, e sua função é receber impulsos do sistema sensorial – e então decidir que ação tomar. Cada unidade no A-System tem seu próprio valor limite específico que precisa ser atingido antes de acionar uma unidade no R-System.

O último sistema do modelo Perceptron é o R-System, abreviação de sistema de resposta, e consiste em poucas unidades que provocam a resposta final produzida pelo Perceptron. A ativação dessas unidades acontece quando o valor médio dos sinais recebidos do Sistema A ultrapassa um determinado nível, o que resulta na geração da saída final (ROSENBLATT, 1957).

2.3.1.1 Processamento de Informações

O Perceptron opera em três fases distintas que englobam seus principais sistemas.

Inicialmente, o S-System funciona como receptor de estímulos externos. Nesta fase crucial, os estímulos são captados e transmitidos para o A-System. A natureza das conexões entre o S-System e o A-System, positivas ou negativas, desempenha um papel significativo na ativação das unidades do A-System (ROSENBLATT, 1957).

Na sequência, o foco se volta para o A-System. Cada unidade deste sistema processa os impulsos recebidos de várias partes do S-System. Estas unidades têm um valor de limiar que, ao ser superado, aciona a ativação de uma ou mais unidades do R-System.

Por fim, o R-System entra em ação, possuindo menos unidades em comparação ao A-System. Este sistema é responsável por gerar a resposta final do Perceptron. A resposta emerge quando o valor médio dos sinais processados pelo A-System supera um nível crítico, determinando o resultado final do processamento do Perceptron.

O sistema inclui mecanismos de feedback, nos quais as unidades do R-System podem inibir a atividade de outras unidades, tanto do próprio R-System quanto do A-System. Essa interação assegura que as respostas geradas sejam mutuamente exclusivas, prevenindo respostas conflitantes ou duplicadas (ROSENBLATT, 1957).

2.3.2 O Problema XOR

O Perceptron desenvolve sua habilidade de aprender ao associar padrões de estímulo com respostas específicas. Esse aprendizado é realizado através do ajuste dos limiares nas unidades do A-System, baseando-se em experiências passadas. Com esses ajustes, o sistema aprimora

continuamente sua capacidade de responder de forma mais precisa e adequada (ROSENBLATT, 1957).

As funções lógicas que podem ser implementadas usando Perceptrons incluem:

1. OU (OR):
$$\text{a OU b} = \begin{cases} 1 & \text{se } a = 1 \text{ ou } b = 1 \\ 0 & \text{caso contrário} \end{cases}$$
2. E (AND):
$$\text{a E b} = \begin{cases} 1 & \text{se } a = 1 \text{ e } b = 1 \\ 0 & \text{caso contrário} \end{cases}$$
3. NÃO (NOT):
$$\text{NÃO a} = \begin{cases} 1 & \text{se } a = 0 \\ 0 & \text{se } a = 1 \end{cases}$$

O XOR é um problema bem conhecido para o Perceptron de camada única, que é considerado a forma mais simples de rede neural. O Perceptron de camada única funciona efetivamente quando os conjuntos são linearmente separáveis, mas não funciona no caso de XOR porque os pontos $f(0,0)$, $f(0,1)$, $f(1,0)$ e $f(1,1)$ não podem ser separados por uma linha reta. Esta questão foi destacada por Minsky e Papert que criticaram a incapacidade do Perceptron para resolver problemas que requerem abordagens não lineares como o XOR faz (RAUBER, 2005).

Representação da função XOR:

XOR (Exclusive OR):

$$\text{a XOU b} = \begin{cases} 1 & \text{se } a \neq b \\ 0 & \text{se } a = b \end{cases}$$

Para superar tal desafio, a solução envolve a incorporação de uma camada escondida na rede, uma estratégia já conhecida antes das críticas de Minsky e Papert. Esta camada adicional facilita a formação de regiões de decisão não lineares, essenciais para distinguir as classes em um cenário XOR (RAUBER, 2005). A dificuldade principal era desenvolver um algoritmo capaz de ajustar eficazmente os pesos da rede multicamada, minimizando os erros entre as saídas desejadas e as obtidas. Diferentes algoritmos de realimentação foram propostos. O algoritmo de retropropagação do erro proposto por Rumelhart, Hinton e Williams (1986) emergiu como uma solução efetiva para treinar redes de Perceptron multicamada, sendo hoje o modelo central de *deep learning*.

Este método ajusta os pesos das camadas anteriores propagando o erro da saída para trás, seguindo a regra de delta, que depende do gradiente do erro. Assim, o Perceptron multicamada se estabelece como um modelo substancialmente mais robusto e versátil que o

Perceptron de camada única, capaz de aproximar qualquer função contínua com uma camada escondida de tamanho adequado (RAUBER, 2005). Entretanto, apesar de suas vantagens, esses modelos avançados ainda enfrentam desafios, como problemas de convergência, sobreajuste, escolha da arquitetura adequada e interpretação dos resultados.

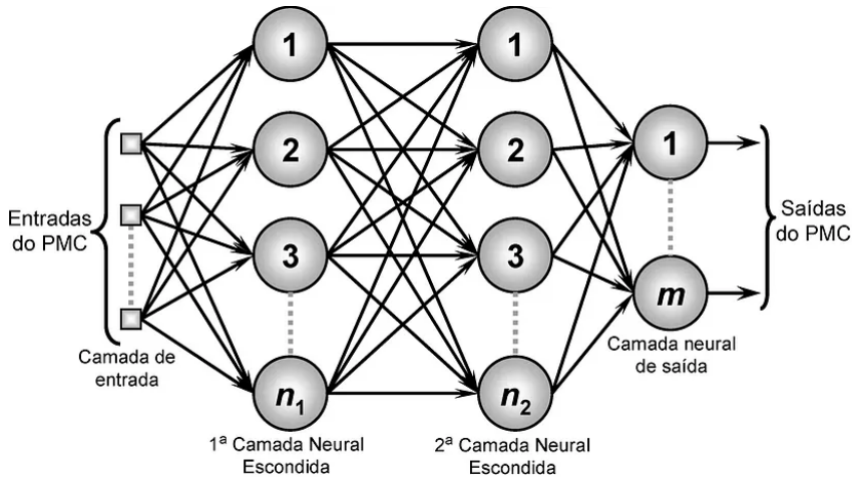
2.4 Perceptron Multicamadas e Funções Lógicas

O problema, referido no trabalho de Rumelhart, Hinton e Williams (1986), trata da otimização de pesos dentro de uma rede neural artificial com muitas camadas constituídas por unidades de processamento de informações — Perceptrons.

A estrutura de cada unidade envolve a realização de operações lineares nas entradas recebidas e depois a passagem do resultado por uma função de ativação não linear. Tendo esta estrutura configurada, a tarefa da rede é aprender um mapeamento dos padrões de entrada para os padrões de saída através da geração de exemplos.

O processo de aprendizagem é supervisionado e utiliza um algoritmo de aprendizagem baseado em gradiente descendente para coordenação de ajuste de peso durante a computação. A Figura 5 representa a arquitetura do modelo Perceptron Multicamadas.

Figura 5 – Perceptron Multicamadas



Fonte: Moreira (2018)[6]

2.4.1 Funções Lógicas

Funções lógicas, abordadas por Rumelhart, Hinton e Williams (1986), são operações binárias que geram valores verdadeiros ou falsos com base nas entradas. Funções lógicas comuns

[6] MOREIRA, S. *Rede Neural Perceptron Multicamadas*. 2018. Disponível em: `https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9`. Acesso em: 10 mar. 2024.

consistem em AND, OR, NOT, XOR e NAND que podem ser demonstradas usando tabelas verdade mostrando todas as combinações possíveis de entrada com suas saídas resultantes ou através de circuitos lógicos implementados por portas lógicas para realizar essas operações.

2.4.2 Perceptron Multicamadas

No domínio do Perceptron multicamadas descrito por Rumelhart, Hinton e Williams (1986), essas funções lógicas encontram seu lugar dentro deste modelo de rede neural. A rede pode aprender a imitar o comportamento de funções lógicas específicas com entradas binárias. Por exemplo, XOR – sendo separável não linearmente – precisa de pelo menos uma camada oculta além das camadas de entrada e saída. Esta camada oculta auxilia na criação de uma representação interna para que a camada de saída possa produzir a resposta correta com base nesta representação. A tarefa do algoritmo de aprendizagem é então ajustar os pesos da rede de modo a minimizar o erro para cada padrão entre a saída real e a desejada.

A seguir, é feito o detalhamento de como cada equação contribui para essa compreensão:

1. Função Tangente Hiperbólica:

   No Perceptron multicamadas, a função tangente hiperbólica é utilizada como função de ativação nos neurônios. Ela ajuda a incluir a não linearidade no modelo, resultando no aprendizado de padrões complexos pela rede. A função mapeia valores reais entre -1 e 1, o que normaliza a saída dos neurônios e facilita a propagação dos sinais pela rede.

   $$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{34}$$

   Nesta equação, x é a entrada para a função de ativação.

2. Função Logística:

   A função logística é utilizada como função de ativação devido à capacidade de mapear entradas reais para um intervalo entre 0 e 1. Isso é muito relevante para a interpretação probabilística das saídas dos neurônios e para a retropropagação do erro durante o treinamento da rede.

   $$f(x) = \frac{1}{1 + e^{-x}} \tag{35}$$

   Onde x é a entrada para a função de ativação.

3. Sinal de Erro para a Camada de Saída:

   O sinal de erro para a camada de saída é utilizado no processo de retropropagação para ajustar os pesos.

   $$\delta_o = o_k \odot (1 - o_k) \odot (t_k - o_k) \tag{36}$$

   Onde δ_o representa o sinal de erro da saída, o_k é a saída do neurônio, t_k é a saída desejada, e $\odot$ denota a multiplicação de Hadamard ou multiplicação elemento a elemento.

4. Sinal de Erro para a Camada Intermediária:

   O sinal de erro para a camada intermediária é calculado para propagar o erro da camada de saída de volta através da rede.

$$\delta_h = h_k \odot (1 - h_k) \odot \left(Z_{[t]}^T \delta_{o,k} \right) \tag{37}$$

   Onde δ_h é o sinal de erro para os neurônios da camada intermediária, h_k é a saída dos neurônios intermediários, $Z_{[t]}^T$ é a matriz de pesos transposta entre a camada intermediária e a camada de saída, e $\delta_{o,k}$ é o sinal de erro da camada de saída.

5. Atualização de Peso da matriz Z:

   A atualização de peso da matriz Z é uma etapa no processo de aprendizagem onde os pesos são ajustados para minimizar o erro.

$$Z_{[t+1]} = Z_{[t]} + \eta \delta_{o,k} h_k^T = Z_{[t]} + \Delta Z_t \tag{38}$$

   Onde $Z_{[t+1]}$ representa os pesos atualizados, $Z_{[t]}$ são os pesos atuais, η é a taxa de aprendizagem, $\delta_{o,k}$ é o sinal de erro da camada de saída, h_k^T é a saída transposta da camada intermediária, e ΔZ_t é o ajuste aplicado aos pesos.

6. Atualização de Peso para a matriz W:

   A atualização de peso para a matriz W segue um processo semelhante ao da matriz Z, ajustando os pesos entre a camada de entrada e a camada intermediária.

$$W_{t+1} = W_t + \eta \delta_h, k^{x_k^T} \tag{39}$$

   Onde W_{t+1} representa os pesos atualizados, W_t são os pesos atuais, η é a taxa de aprendizagem, δ_h é o sinal de erro da camada intermediária, e x_k^T é a entrada transposta.

2.5 Redes Neurais Recorrentes (RNNs)

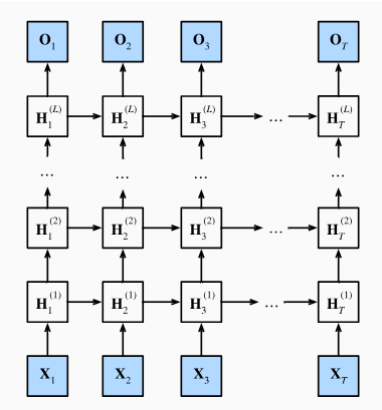
Com base nos modelos de Rumelhart, Hinton e Williams (1986), a arquitetura de RNNs permite a criação de modelos capazes de lidar com processamento de dados sequenciais. Um recurso central é a capacidade de manter uma memória interna que pode rastrear informações através de sequências de entrada – isso permite a captura de dependências temporais e padrões onde a ordem é significativa para a compreensão do processo a partir de conjuntos de dados. A Figura 6 representa a arquitetura RNN.

A definição matemática básica de um RNN pode ser vista na seguinte equação:

$$h_t = \sigma(W_{hh} h_{t-1} + W_{xh} x_t + b_h) \tag{40}$$

[7] ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. *Dive into Deep Learning*. Cambridge University Press, 2023. Disponível em: `https://D2L.ai`. Acesso em: 05 mai. 2024.

Figura 6 – Arquitetura RNN



Fonte: Zhang et al. (2023)[7]

O estado oculto h_t no tempo t, é chamado de memória da rede. Ajuda a acompanhar as dependências de longo prazo durante a sequência temporal e torna mais fácil para a rede reconhecer tais dependências nos dados de entrada.

Por outro lado, x_t denota o elemento de entrada no tempo t, que são os dados recebidos pela rede em cada momento.As matrizes W_{hh} e W_{xh} são coeficientes de peso que influenciam a relação entre os estados ocultos anteriores e as entradas atuais na atualização do estado oculto, sendo ajustados durante o treino para aprimorar a performance da rede em tarefas determinadas. O viés b_h é integrado para modular a saída da função de ativação, incrementando a adaptabilidade do modelo.

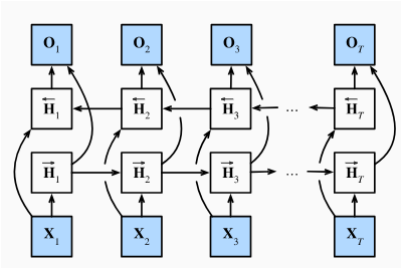
A função de ativação σ, seja ela tangente hiperbólica (tanh), é uma escolha crítica na transformação e fluxo de informações através da rede que impacta a capacidade do RNN de capturar relacionamentos sofisticados presentes em dados (SHERSTINSKY, 2018).

2.5.1 Redes Neurais Recorrentes Bidirecionais (Bi-RNNs)

As Bi-RNNs foram desenvolvidas em 1997, para processamento sequencial de dados. A Figura 7 representa a arquitetura RNN Bidirecional.

Ela processa os dados em duas direções: para frente e para trás. Isso permite que o modelo capture contextos de ambas as extremidades da sequência, um recurso útil nos casos em que tanto o contexto futuro quanto o passado desempenham um papel significativo, como reconhecimento de fala ou processamento de linguagem natural. Além de tais redes apresentarem melhores resultados que as RNNs tradicionais em diferentes tarefas de classificação e regressão (SCHUSTER; PALIWAL, 1997).

Figura 7 – Arquitetura RNN Bidirecional

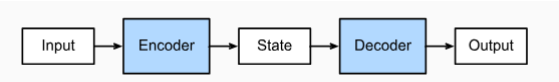


Fonte: Zhang et al. (2023)[7]

2.5.2 Arquitetura Encoder-Decoder

A estrutura codificador-decodificador é considerada uma das abordagens eficazes de tarefas sequência a sequência na NLP, que inclui tradução automática e resumo de texto. Funciona assim: primeiro um componente codificador passa pela sequência de entrada e a transforma em uma representação vetorial fixa, para que um componente decodificador possa produzir sequências de saída com base nessa representação (JURAFSKY; MARTIN, 2024a,b), (CHO et al., 2014). A Figura 8 representa a arquitetura RNN Encoder-Decoder.

Figura 8 – Arquitetura Encoder-Decoder



Fonte: Zhang et al. (2023)[7]

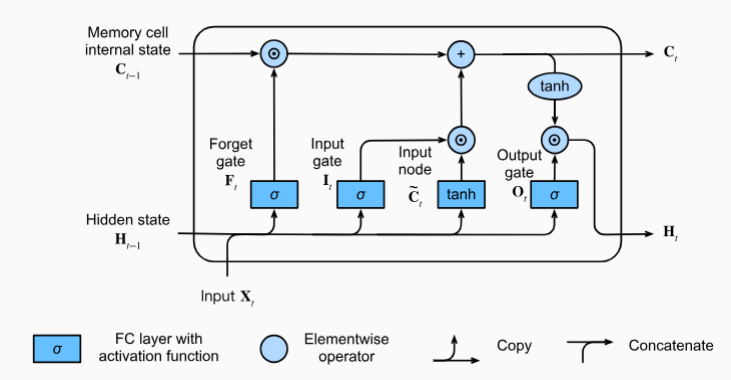
Dois tipos de modelos são frequentemente usados na arquitetura do codificador-decodificador devido à sua capacidade de resolver problemas de desaparecimento e explosão de gradiente e lidar com dependências em longos intervalos de tempo: LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit) (HOCHREITER; SCHMIDHUBER, 1997; SAXENA, 2024).

Ao usar LSTM com o modelo codificador-decodificador, a sequência de entrada $(x_1, x_2, \ldots, x_T)$ é processada pelo codificador, que gera um vetor de contexto C. Este vetor resume as informações da sequência de entrada e auxilia na geração da sequência de saída $(y_1, y_2, \ldots, y_T)$ para cada posição, baseando-se nas informações contidas em C (HOCHREITER; SCHMIDHUBER, 1997).

2.5.3 Memória Longa de Curto Prazo (LSTM)

As redes Long Short-Term Memory, também conhecidas como LSTMs, surgiram em 1997, numa tentativa de lidar com o problema do desaparecimento dos gradientes (HOCHREITER; SCHMIDHUBER, 1997). A Figura 9 representa a arquitetura da célula LSTM da classe de modelos RNN.

Figura 9 – Arquitetura de célula LSTM



Fonte: Zhang et al. (2023)[7]

Eles conseguiram isso introduzindo portas que podem regular o fluxo de informações e, assim, reter os dados por longos períodos de tempo além de atualizá-los adequadamente por meio de sequências, permitindo um aprendizado eficaz de dependências de longo prazo.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{41}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{42}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{43}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{44}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{45}$$

$$h_t = o_t \odot \tanh(C_t) \tag{46}$$

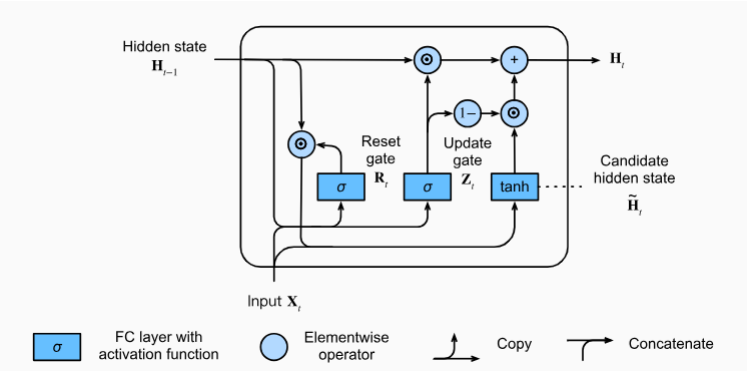
Onde i_t é o vetor da porta de entrada, f_t é o vetor da porta de esquecimento, o_t é o vetor da porta de saída, $\tilde{C}_t$ é o vetor da nova informação candidata para a célula, C_t é o vetor do estado da célula, h_t é o vetor da saída oculta, e σ e $\tanh$ são funções de ativação (SHERSTINSKY, 2018).

2.5.4 Unidade Recorrente com Portas (GRU)

Uma simplificação de redes de memória de longo prazo (LSTMs), unidades recorrentes fechadas (GRUs) foi introduzida em 2014. O objetivo principal das GRUs é mesclar

as portas de entrada e esquecimento dos LSTMs em uma, o que facilita a arquitetura da rede e reduz a quantidade de parâmetros necessários durante o treinamento conforme descrito por (CHO et al., 2014). A Figura 10 representa a arquitetura GRU pertencente a classe de modelos RNN.

Figura 10 – Arquitetura GRU



Fonte: Zhang et al. (2023)[7]

As GRUs utilizam duas portas principais: a porta de atualização e a porta de reset. A porta de atualização z_t determina a quantidade de informação do estado anterior que será carregada para o próximo estado. Já a porta de reset r_t controla o quanto da informação do estado anterior será esquecida. Com essas duas portas, as GRUs conseguem manter e atualizar a informação relevante ao longo das sequências temporais, facilitando o aprendizado de dependências de longo prazo.

Matematicamente, as equações principais de uma GRU são descritas da seguinte forma (JEEVA, 2023)[8]:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{47}$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{48}$$

$$\tilde{h}_t = \tanh(Wx_t + r_t \odot Uh_{t-1}) \tag{49}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{50}$$

Essas equações consistem nos seguintes componentes: z_t como a porta de atualização, r_t como a porta de reinicialização, $\tilde{h}_t$ como o novo estado oculto, σ que é uma função de

[8] JEEVA, C. *GRU Network*. Scaler, 2023. Disponível em: `https://www.scaler.com/topics/deep-learning/gru-network/`. Acesso em: 10 mai. 2024.

ativação sigmóide, $\tanh$ representando a função tangente hiperbólica, e $\odot$ denotando multiplicação elemento a elemento. A relação do estado anterior com o novo estado oculto é determinada por z_t, que os compõe em uma combinação. Por outro lado, o ajuste da influência do estado anterior na geração do novo estado oculto é feito através do controle estabelecido pela porta de reset que faz parte desta equação (JEEVA, 2023)[8].

2.5.5 Processamento de Linguagem Natural

Processamento de Linguagem Natural (NLP), é uma área que cresceu enormemente nas últimas décadas com base em fundamentos teóricos herdados da linguística, da ciência da computação e da estatística. Segundo JURAFSKY e MARTIN (2024c,d), o desenvolvimento de técnicas em NLP (Processamento de Linguagem Natural) visa capacitar as máquinas a compreender e gerar linguagem humana. Esta tarefa é complexa e exige a convergência de diversas abordagens devido à complexidade e ambiguidade que caracterizam as línguas naturais. Tais sistemas são essenciais para criar formas mais eficazes e intuitivas de interação com os usuários humanos.

Os desafios que o NLP enfrenta: por exemplo, ter de abordar nuances culturais e contextuais da linguagem e manter intacta a privacidade dos dados não são um acontecimento único. Trabalhos como o de Smith (2011) sublinham a necessidade de abordar estas questões éticas e técnicas – uma vez que o desenvolvimento responsável de tecnologias linguísticas de uma forma inclusiva é muito importante. O futuro da NLP teria uma interação promissoramente mais natural entre humanos e máquinas, o que virá através da constante evolução das pesquisas com adoção de novas metodologias.

Além disso, o progresso dos modelos de NLP tem sido grandemente favorecido pelo crescimento da aprendizagem automática, nomeadamente pelas redes neurais profundas (MANNING; RAGHAVAN; SCHÜTZE, 2008). Conforme indicado por Goldberg (2017), as redes neurais demonstraram uma capacidade impressionante de capturar padrões na linguagem – que são muito complexos – levando a melhorias drásticas em tarefas como tradução automática ou resumo de texto. Frequentemente, o desempenho desses modelos pode ser aumentado por meio de grandes conjuntos de textos que atuam como bancos de dados com grandes quantidades de informações a partir dos quais os algoritmos são treinados.

Ademais, o NLP enfrenta alguns problemas. Eles são chamados de “desafios contínuos” e incluem, entre outros, a necessidade de levar em conta as especificidades culturais e os detalhes contextuais do idioma – também a necessidade de abordar a privacidade dos dados.

2.5.6 Problemas das RNNs em Processamento de Linguagem Natural

Redes Neurais Recorrentes (RNNs) são eficientes na captura do relacionamento entre sequências. Há uma série de obstáculos que surgem no caminho dessas redes, ainda mais quando se trata de tarefas de Processamento de Linguagem Natural (NLP). Um dos principais

desafios enfrentados pelos RNNs é o que é conhecido como desvanecimento e explosão de gradiente (SAXENA, 2024).

Durante o processo de treinamento, os gradientes podem desaparecer exponencialmente (desaparecer) ou explodir drasticamente (explodir); esta dificuldade surge de dependências de aprendizagem em contextos de longo prazo (HOLDSWORTH; SCAPICCHIO, 2024)[2]. Este fenômeno leva a complicações na estabilidade durante o treinamento e dificulta a captura de informações contextuais em grandes sequências (HOCHREITER; SCHMIDHUBER, 1997; SARKER, 2021).

Além disso, é interessante notar que as RNNs têm capacidade limitada de reter informações de longo prazo. Nas tarefas de NLP, a informação contextual é muito importante ao longo de grandes sequências. Mas estas RNNs simples representam grandes desafios na retenção destes dados, o que compromete a compreensão do contexto e, portanto, a leitura precisa dos textos, conforme explicado por JURAFSKY e MARTIN (2024a).

A outra questão importante é que a camada RNN superior pode enfrentar uma tarefa árdua. A camada RNN superior tenta condensar todas as informações relevantes de toda a sequência em um único vetor - isso, quando falha, leva à perda de dados de contexto importantes que degradam o desempenho de toda a rede em tarefas de processamento de linguagem natural (NLP) de acordo com para Manning, Raghavan e Schütze (2008).

Os RNNs também enfrentam desafios na capacidade de capturar dependências de palavras complexas e de longo alcance que aparecem em muitas tarefas de NLP. Não é fácil modelar as dependências que se estendem por uma grande parte do texto usando RNNs tradicionais, levando a baixa precisão na tradução automática ou na análise de sentimento, conforme apontado por Cho et al. (2014). Ainda outra questão importante a ter em conta é a sobrecarga da última camada na RNN, que normalmente tem uma responsabilidade significativa acumulada sobre ela – tentar resumir todos os detalhes essenciais de uma longa sequência numa representação compacta.

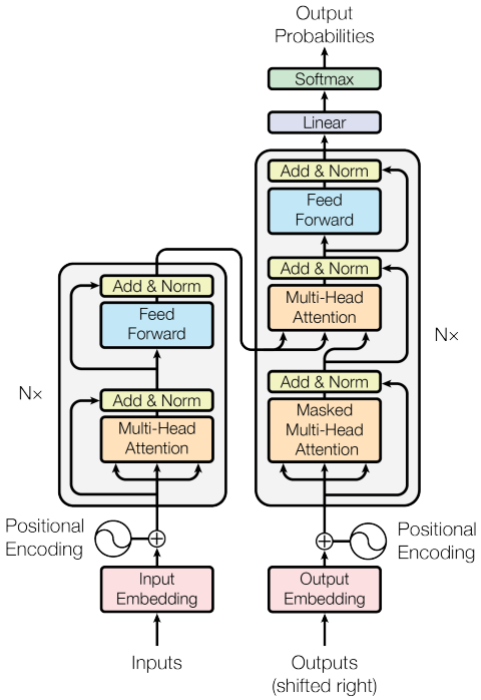
Conforme destacado por Manning, Raghavan e Schütze (2008), a omissão de pistas contextuais importantes é uma realidade que poderia ter melhorado significativamente a qualidade do trabalho em processamento de linguagem natural (PLN). Os desafios foram superados com o desenvolvimento de diferentes tipos de Redes Neurais Recorrentes (RNNs), incluindo a Long Short-Term Memory (LSTM) e a Gated Recurrent Units (GRU), que conseguem aprender dependências de longo prazo e contêm mecanismos para administrar o problema de desaparecimento do gradiente durante o treinamento (JEEVA, 2023)[8].

Ademais, os mecanismos de atenção introduzidos auxiliam em focar partes específicas de uma sequência, aumentando assim a precisão nas tarefas de NLP (VASWANI et al., 2017; HOCHREITER; SCHMIDHUBER, 1997).

2.6 Arquitetura Transformer

A revelação da arquitetura Transformer por Vaswani et al. (2017), conforme demonstrado na Figura 11, se tornou uma grande evolução na categoria de modelos de NLP. Antes dos Transformers, as redes neurais recorrentes (RNNs) e suas variantes, como LSTMs, dominavam o campo, mas enfrentavam dificuldades com dependências de longo prazo devido à característica sequencial, dificultando a paralelização dos cálculos e resultando em tempos de treinamento mais longos.

Figura 11 – Arquitetura Transformer



Fonte: Vaswani et al. (2017, p. 3).

Como diferencial e sendo um aspecto positivo, os Transformers eliminam a necessidade de recorrência, utilizando os mecanismos de atenção como componente principal, o que possibilita o treinamento paralelo mais facilitado e de maior eficiência computacional.

2.6.1 Atenção Multicabeças

A inclusão da técnica de atenção multicabeças favorece a capacidade do modelo de apreender vários tipos de associações de palavras (AHMED et al., 2023). Cada cabeça de atenção em um Transformer adquire uma própria função de atenção exclusiva, o que permite o modelo atender a diferentes partes da sequência ao mesmo tempo - divide os vetores de consulta em chave e valor em subespaços separados antes de aplicar a atenção de forma singular

e os concatena para representação final após realizar os devidos processamentos: semelhante a como as redes convolucionais usam vários filtros para uma extração de recursos mais rica (PADMANABHAN, 2024)[9].

Além disso, a atenção multicabeças favorece a elevação da capacidade do modelo de capturar diversas relações de palavras (PADMANABHAN, 2024)[9]. Cada cabeça de atenção em um Transformer aprende uma função de atenção diferente o que permite ao modelo prestar atenção a muitas partes da sequência ao mesmo tempo - dividindo os vetores de consulta, chave e valor em subdimensões separadas antes de aplicar a atenção independentemente a cada uma, e posteriormente combiná-los após concatenação para a representação final. Isso é semelhante a como as redes convolucionais usam vários filtros para uma detecção de recursos mais rica (DUFTER; SCHMITT; SCHÜTZE, 2022).

2.6.2 Codificações Posicionais

Os Transformers trouxeram outra inovação essencial: o uso de codificações posicionais. Isso ajuda a incluir informações sobre a ordem das palavras já que os mecanismos de atenção direta não lidam com isso. No trabalho de Vaswani et al. (2017) apresenta-se um método onde funções seno e cosseno de diferentes frequências são usadas para gerar codificações posicionais. Posteriormente, eles são adicionados ao *embeddings* de palavras antes do processamento por meio do modelo Transformer; dessa forma, o Transformer pode distinguir entre palavras com base nas suas próprias posições em relação a outras - preservando assim a estrutura sequencial da linguagem. Trabalhos posteriores como os de Shaw, Uszkoreit e Vaswani (2018) exploraram codificações posicionais relativas que levam em conta a distância entre palavras em uma sequência – o que levou a melhorias de desempenho em tarefas como tradução automática (AHMED et al., 2023).

2.6.3 Impacto na NLP

O campo do processamento de linguagem natural foi completamente transformado pela arquitetura Transformer. Ela inspirou modelos como BERT e GPT, que desde então passaram a ser amplamente utilizados em diferentes aplicações de NLP. O fato de os Transformers serem capazes de capturar dependências de maneira eficaz em longos intervalos e lidar bem com sequências os tornou preferidos para muitas tarefas, incluindo, mas não se limitando a, tradução de idiomas, geração de texto e muito mais (AHMED et al., 2023).

2.7 Modelo BERT

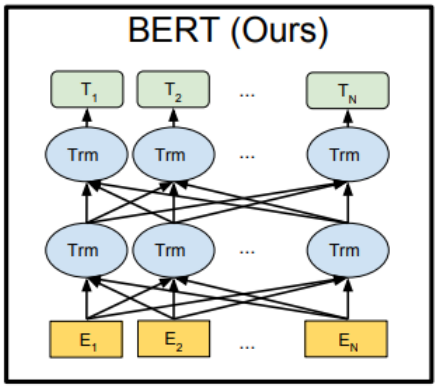
O BERT é um modelo de linguagem pertencente à arquitetura Transformer, desenvolvido em 2018. O modelo foi um dos revolucionadores do campo do NLP (DEVLIN et al.,

[9] PADMANABHAN, A. Transformer Neural Network Architecture. Version 8, 27 maio 2024. Disponível em: `https://devopedia.org/transformer-neural-network-architecture`. Acesso em: 01 mai. 2024.

2018). A Figura 12 ilustra a arquitetura do modelo BERT.

Este trabalho faz um detalhamento do modelo BERT, incluindo: pré-treinamento, ajuste fino, tokenização, tamanhos das versões base e large, e a aplicação do modelo em tarefas de perguntas e respostas (Q&A).

Figura 12 – Arquitetura do modelo BERT



Fonte: Devlin et al. (2018)

2.7.1 Pré-Treinamento

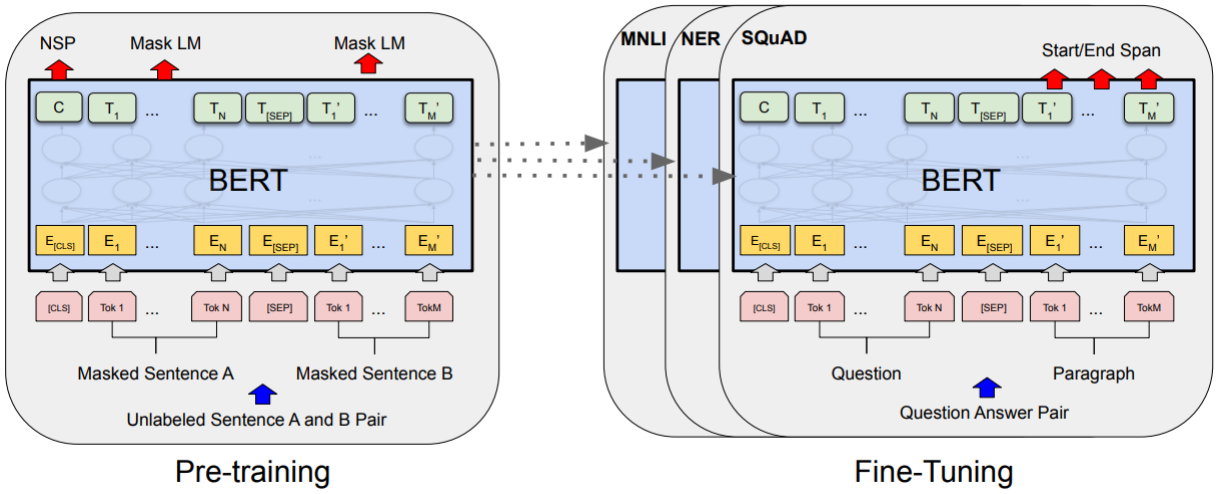
O pré-treinamento do BERT é realizado utilizando grandes corpora de texto não rotulado, como o Toronto Book Corpus e a Wikipedia (DEVLIN et al., 2018). O modelo é treinado em duas tarefas não supervisionadas: *Masked Language Modeling* (MLM) e *Next Sentence Prediction* (NSP).

No MLM, 15% das palavras em cada sequência são aleatoriamente mascaradas e o modelo é treinado para prever essas palavras mascaradas. Já no NSP, o modelo aprende a prever se um par de frases são consecutivas no texto original (DEVLIN et al., 2018). A Figura 13 mostra as etapas de treinamento do modelo BERT.

2.7.2 Ajuste Fino

O ajuste fino do BERT é o processo de adaptação do modelo pré-treinado a uma tarefa específica usando dados rotulados. Todos os parâmetros do modelo são ajustados usando os dados da tarefa final, o que permite ao BERT alcançar resultados de ponta em diversas tarefas de NLP, como classificação de texto, reconhecimento de entidade nomeada, e perguntas e respostas (JURAFSKY; MARTIN, 2024b).

Figura 13 – Etapas de treinamento do modelo BERT



Fonte: Devlin et al. (2018)

2.7.3 Tokenizador

Tokenizador é um componente fundamental em modelos de processamento de linguagem natural (NLP) como o BERT. Ele é responsável por dividir o texto de entrada em unidades menores, conhecidas como tokens. Esses tokens podem ser palavras, sub-palavras ou caracteres, dependendo do tipo de tokenizador utilizado. O tokenizador do BERT é baseado em *WordPiece* divide as palavras em sub-palavras ou tokens menores, permitindo ao modelo lidar eficientemente com vocabulários grandes e palavras raras (SONG et al., 2021).

O tokenizador do BERT possui um vocabulário de 30.000 tokens, incluindo tokens especiais como [CLS] (usado no início de cada sequência para classificação), [SEP] (usado para separar pares de sentenças) e [PAD] (usado para preenchimento, permitindo que todas as sequências em um batch tenham o mesmo comprimento) (SONG et al., 2021; MARTINS, 2021).

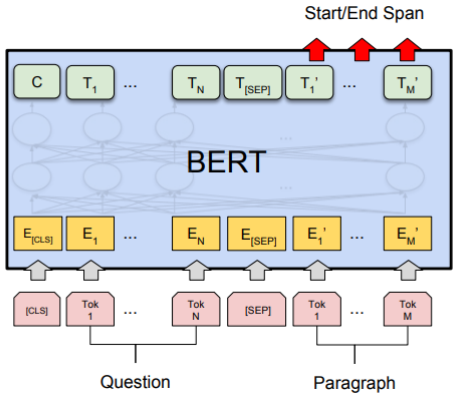
2.7.4 Tamanhos do BERT: Base e Large

O BERT é disponibilizado em duas versões principais: BERT Base e BERT Large. O BERT Base possui 12 camadas, 768 unidades de dimensão escondida e 12 cabeças de atenção, totalizando 110 milhões de parâmetros. O BERT Large, por sua vez, possui 24 camadas, 1024 unidades de dimensão escondida e 16 cabeças de atenção, totalizando 340 milhões de parâmetros (DEVLIN et al., 2018).

2.7.5 Tarefa de Perguntas e Respostas (Q&A)

O BERT tem sido amplamente utilizado em tarefas de perguntas e respostas (Q&A). Na arquitetura para Q&A, o modelo é ajustado fino usando *datasets* como o SQuAD (Stanford Question Answering Dataset). Durante o ajuste fino, o modelo aprende a prever as posições de início (p_{start}) e fim (p_{end}) das respostas dentro do texto de referência, baseando-se na pergunta fornecida (RAJPURKAR et al., 2016; DEVLIN et al., 2018). A Figura 14 mostra o modelo BERT para tarefas de perguntas e respostas.

Figura 14 – Representação de tarefas de pergunta e resposta do modelo BERT



Fonte: Devlin et al. (2018)

Para processar a entrada, o BERT utiliza tokens especiais: [CLS], [SEP], e [PAD]. O token [CLS] é adicionado no início da sequência de entrada e sua saída é usada para tarefas de classificação. No caso de Q&A, ele ajuda a agregar a informação global da entrada. O token [SEP] é usado para separar diferentes partes da entrada, como a pergunta e o contexto. Finalmente, o token [PAD] é utilizado para preencher as sequências que são menores que o tamanho máximo esperado, garantindo assim que todas as entradas tenham o mesmo comprimento (McCORMICK, 2020)[10].

A probabilidade de cada token i ser o início ou o fim da resposta é calculada da seguinte forma:

$$P_{start}(i) = \frac{e^{S_i}}{\sum_{j=1}^{N} e^{S_j}} \tag{51}$$

$$P_{end}(i) = \frac{e^{E_i}}{\sum_{j=1}^{N} e^{E_j}} \tag{52}$$

onde S_i e E_i são os *scores* produzidos pelo BERT para o início e o fim dos tokens, respectivamente, e N é o número total de tokens no texto de referência.

[10] McCORMICK, C. *Question Answering with a Fine-Tuned BERT*. 2020. Disponível em: `https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/`. Acesso em: 14 mai. 2024.

A posição de início (p_{start}) e fim (p_{end}) da resposta são então determinadas escolhendo-se os tokens com as maiores probabilidades:

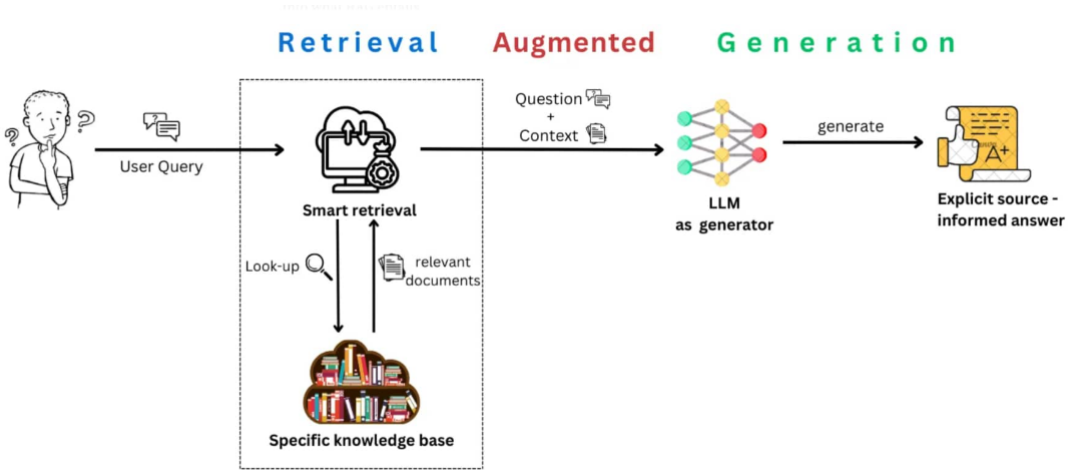
$$p_{start} = \arg\max_i P_{start}(i) \tag{53}$$

$$p_{end} = \arg\max_i P_{end}(i) \tag{54}$$

2.8 Geração Aumentada de Recuperação (RAG)

A RAG foi desenvolvida pela equipe Meta AI em 2020, é uma metodologia promissora no campo da NLP, que tem como especialidade a combinação e recuperação de informações e a geração de linguagem. Essa inovação tem por objetivo estabelecer mais assertividade aos LLMs na geração de respostas, possibilitando, assim, que os modelos de linguagem acessem fontes de informações externas sem precisar passar por novos treinamentos, sendo essa uma limitação dos LLMs que o RAG visa suprir (LEWIS et al., 2020). A Figura 15 demonstra o fluxo de processamento de informações na arquitetura RAG.

Figura 15 – Arquitetura RAG



Fonte: Zhukov (2024)[11]

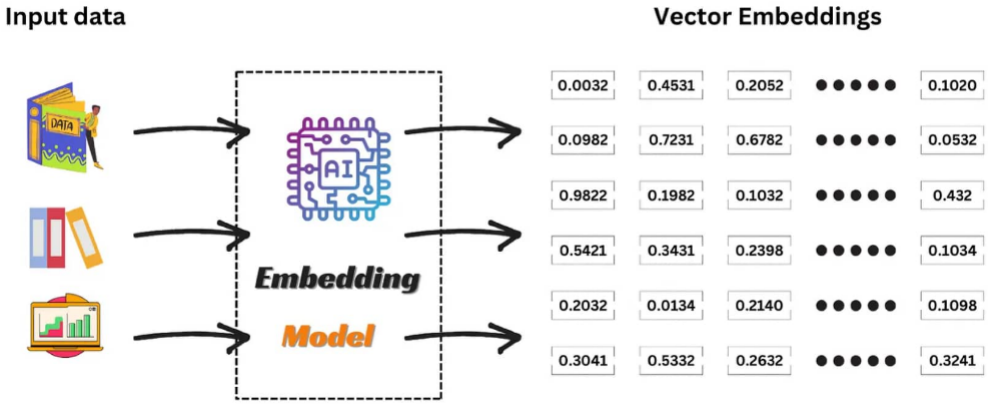
2.8.1 Estrutura e Funcionamento

A RAG é composta de duas fases principais. Na fase de recuperação, um modelo realiza uma busca por informações em bases de conhecimento externas, como bancos de dados, documentos e demais outras fontes de dados, para extrair as informações que mais se aproximam ao termo da pesquisa. Ao acessar essas fontes, de dados os dados recuperados são convertidos

em vetores de alta dimensão, classificadas de acordo com a similaridade ou proximidade com o texto utilizado na pesquisa, que pode ser uma pergunta (LEWIS et al., 2020).

Na fase de geração, as informações recuperadas são usadas como contexto para a geração de respostas, que são feitas pelo modelo de linguagem. Este processo permite que o modelo acesse e utilize informações mais recentes e específicas, permitindo a criação de respostas relevantes (LEWIS et al., 2020; KHATTAB; ZAHARIA, 2020). A Figura 16 ilustra a etapa de embeddings dos dados na arquitetura RAG.

Figura 16 – Representação do processo de incorporação de documentos em RAG



Fonte: Zhukov (2024)[11]

Ademais, a RAG pode se adaptar de forma contínua por meio das técnicas de aprendizado por reforço e ajuste fino. A medida que o sistema interage com os usuários, ele coleta feedback e utiliza essas informações para melhorar os algoritmos internos de recuperação e geração (GAO et al., 2024). Isso possibilita a melhoria da precisão das respostas ao longo do tempo e também possibilita que o sistema possa ter acesso as novas informações que surgem e, consequentemente, se atualizar. Um exemplo disso é no contexto de atendimento ao cliente, onde a RAG pode aprender com interações anteriores e oferecer soluções de maior relevância em eventuais novas pesquisas (CHANDAN, 2024)[12].

Outra vantagem da RAG é a capacidade de integrar informações de várias modalidades, o que significa que, além de textos, o sistema pode utilizar dados de outras fontes, como imagens, vídeos e gráficos, para fornecer respostas relevantes. Na esfera do comércio eletrônico, isso pode resultar em recomendações de produtos com base nas preferências textuais dos usuários e também nos elementos visuais de produtos antes vistos. Ademais, a RAG aprimora a

[11] ZHUKOV, V. *Introduction to RAG: Enhancing Language Models with External Knowledge*. IngestAI, 2024. Disponível em: `https://ingestai.io/blog/introduction-to-rag`. Acesso em: 14 abr. 2024.

[12] CHANDAN, Gokul. Retrieval-Augmented Generation (RAG): What is it? Navan.ai, 2024. Disponível em: `https://navan.ai/blog/retrieval-augmented-generation-what-is-it/`. Acesso em: 12 abr. 2024.

qualidade das interações em termos de precisão e relevância e também amplia as possibilidades de possíveis aplicações (ZHUKOV, 2024; CHANDAN, 2024)[11,12].

2.8.2 Aplicações e Benefícios

A Geração Aumentada por Recuperação tem se tornado um componente muito relevante em alguns tipos de aplicações tais como assistentes virtuais, chatbots e sistemas de perguntas e respostas. A metodologia RAG combina funcionalidades de modelos baseados em recuperação e geração, permitindo que os sistemas busquem informações relevantes de fontes externas e gerem respostas de alta qualidade (GAO et al., 2024).

A RAG pode ser utilizada principalmente no suporte ao cliente em cenários de dúvidas frequentes e pontuais, geração de conteúdo e recuperação de informações. Um exemplo, chatbots com RAG podem lidar com consultas de clientes recuperando informações específicas de bancos de dados ou documentos e gerando respostas personalizadas, possibilitando a eficiência e a satisfação do usuário (ZHUKOV, 2024)[11]. De forma semelhante, no setor de saúde, a RAG ajuda a fornecer informações médicas e recomendações de tratamento ao integrar conhecimentos médicos atualizados (ZHUKOV, 2024)[11].

Maior relevância, melhor adaptabilidade, escalabilidade e eficiência são os principais aspectos positivos da RAG. De posse de informações confiáveis e de qualidade, a RAG pode minimizar os erros e contribuir para a geração de contextos relevantes possibilitando que as LLMs gerem respostas de maneira confiável e relevante (GAO et al., 2024). Os sistemas que fazem uso da RAG podem acessar as informações mais recentes sem precisar de re-treinamento constante, o que se torna útil em campos muito voláteis, como finanças e saúde (ZHUKOV, 2024)[11].

Além do mais, as estruturas de RAG são para lidar com grandes quantidades de dados de maneira eficiente, tornando-as adequadas para aplicações em tempo real e implementações em larga escala (CHANDAN, 2024)[12]. Esses pontos positivos da RAG a tornam uma solução de relevante importância para empresas que desejam melhorar a performance das aplicações que fazem o uso da IA e aprimorar as interações com os usuários por meio de melhores respostas para o determinado cenário em específico (ZHUKOV, 2024)[11].

2.8.3 Desafios e Futuro

Apesar dos pontos positivos da RAG, a metodologia possui grandes desafios como a complexidade de integração com fontes distintas de dados simultaneamente e por ter como base uma infraestrutura robusta de computação. Além disso, a recuperação assertiva das informações continua sendo um ativo campo de pesquisa com esforços contínuos na tentativa de aprimorar técnicas de *chunking*[13] , expansão de consultas e re-ranqueamento de documentos (KHATTAB;

[13]Técnicas de *chunking* são utilizadas para dividir grandes conjuntos de texto em partes menores, isso é um facilitador no processamento e a análise de dados em linguística computacional.

ZAHARIA, 2020).

A Geração Aumentada de Recuperação representa um avanço no campo da NLP ao possibilitar que as LLMs possam acessar contextos relevantes e atualizados sem sobrecarregar a janela de contexto para modelos que possuem essa limitação como o BERT e possibilita a geração de respostas relevantes. A RAG combina recuperação de informações e geração de linguagem contribuindo para a superação dos desafios enfrentados pelos LLMs tradicionais (LEWIS et al., 2020).

2.9 Chatbot

Os chatbots se tornaram essenciais em muitos setores, facilitando a interação entre empresas e clientes por meio de respostas rápidas e eficientes. A integração de modelos de linguagem natural como o BERT possibilita que a precisão e a naturalidade das respostas fornecidas pelos chatbots se estabeleçam. BERT compreende o contexto de uma palavra baseada em todas as suas ocorrências na sentença, tanto as anteriores quanto as posteriores, permite a compreensão de maneira mais precisa e relevante a interações textuais feitas pelos usuários. Tal capacidade de compreensão contextual é a pedra angular para sistemas de Q&A, pois possibilita que as respostas fornecidas sejam relevantes, contribuindo para a melhoria da experiência do usuário (DEVLIN et al., 2018).

A precisão dos chatbots em contextos de perguntas e respostas pode ser melhorada com a aplicação de modelos como o BERT, que utilizam recursos de processamento de linguagem natural para interpretar e responder a perguntas de maneira mais natural e humanizada. O trabalho de Vaswani et al. (2017), que apresentou a arquitetura Transformer, tornou possível o desenvolvimento do BERT. Esses avanços tecnológicos possibilitam que os chatbots gerem conteúdos textuais com ótima qualidade no campo de NLP. Portanto, a combinação de chatbots com BERT ou outros modelos de linguagem avançada representam um avanço, proporcionando interações mais naturais e contextualizadas, e sendo uma solução enriquecedora para melhoria na comunicação e o atendimento ao cliente em contextos empresariais.

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados os trabalhos relacionados ao presente projeto e sua conexão com o trabalho proposto.

3.1 Introdução ao processamento de linguagem natural: Desenvolvimento de um chatbot utilizando python

A monografia de Silva (2023) aborda o desenvolvimento de um chatbot com o modelo BERT para auxiliar alunos da Unicamp (SILVA, 2023).

O BERT passou por um treinamento e foi aplicado o cenário de perguntas frequentes em português. O chatbot proposto é composto pelo *framework* React para interface, API FastAPI e tecnologias de banco de dados MongoDB: as perguntas vão para a API que pesquisa o MongoDB em busca de contextos onde o BERT irá gerar respostas. A utilização do banco de dados não relacional MongoDB permitiu a rápida aquisição de informações contextuais, enquanto o BERT foi utilizado para produzir respostas.

Por mais que o modelo possua dificuldades com expressões específicas, ele se mostrou relevante na geração de respostas. No geral, os testes demonstraram que BERT gera respostas em uma amplitude de temas.

O trabalho de Silva (2023) utiliza a ideia de pipeline de recuperação de dados e geração de contexto para respostas com o modelo BERT, aplicando uma interface do usuário por meio do *framework* React, e se assemelha ao presente trabalho, pois ambos abordam a aplicação do BERT em chatbots utilizando pipeline para recuperação de contexto em base de dados.

3.2 Estudos de algoritmos de aprendizagem profunda no contexto de Processamento de Linguagem Natural para desenvolvimento de assistentes virtuais.

Ferreira (2022) compara modelos BERT pré-treinados em português e os aplica para tarefa de perguntas e respostas no âmbito educacional. No trabalho é abordado as etapas de tokenização e *embeddings* segmentados para diferenciação de pergunta e contexto, e aplica a função Softmax para definir o intervalo probabilístico da resposta em um contexto (FERREIRA, 2022).

No trabalho de Ferreira (2022), foi possível notar que o algoritmo que performou melhor nos testes realizados foi o pierreguillou/bert-base-cased-squad-v1.1-portuguese utilizando o primeiro código de implementação. Tal modelo se destacou por ter médias mais altas medições das pontuações iniciais e finais que foram em torno de 5,96 e 5,99, respectivamente – e também marcou o menor valor médio de tempo de execução aproximado de 1.131 segundos para a geração de resposta dado um contexto. Para a realização do teste comparativo dos algoritmos foram incluídos dois modelos pré-treinados distintos, sendo eles: pierreguillou/bert-base-cased-

squad-v1.1-portuguese e o mrm8488/bert-base-portuguese-cased-finetuned-squad-v1-pt, cada um experimentou dois métodos de implementação disitintos.

No primeiro método, os modelos BERT já pré-treinados e foi feito também o processo de tokenização das respectivas entrada, sendo elas o contexto e a pergunta que são concatenados com o token reservado [SEP] para segmentação os dados de entrada no modelo. Em logo em sequência, foram gerados os *scores* por meio da resposta do modelo, o que torna possível identificar a maior probabilidade de início e fim da resposta dentre os tokens, por meio da aplicação da função softmax.

No segundo método foi utilizado a função encode_plus para tokenizar as entradas, incluindo os tokens reservados [CLS] e [SEP] e mapeando os tokens com IDs específicos, truncando as sentenças para padronização de tamanho e é feito a criação de mascaras de atenção para diferenciação de reais de tokens reservados de preenchimento [PAD].

A avaliação considerou as médias dos *scores* de início e fim, além do tempo de execução para responder a um conjunto de perguntas. Esses resultados demonstram que o modelo pierreguillou/bert-base-cased-squad-v1.1-portuguese com o primeiro método de implementação é o mais eficiente e preciso para a tarefa de perguntas e respostas, confirmando a superioridade no contexto avaliado.

O trabalho de Ferreira (2022) compara e aplica modelos pré-treinados em português para perguntas e respostas, sendo semelhante ao presente estudo na aplicação de contexto e recuperação de respostas com o modelo BERT. Além disso, no presente trabalho, é utilizada a versão do BERT Large, que possui mais camadas do algoritmo em comparação com a versão BERT Base, ambos pré-treinados por Guillou (2021), que apresentaram os melhores resultados na comparação proposta.

4 METODOLOGIA

Neste capítulo, serão abordados os métodos e ferramentas utilizados para alcançar os objetivos do trabalho.

4.1 Modelo BERT

BERT é um modelo pertencente a arquitetura Transformer desenvolvido para o processamento de linguagem natural (NLP) e que possui a funcionalidade de transferência e aprendizado. Esse é um conceito trivial no desenvolvimento de modelos Transformers, sendo necessário a disponibilidade de um modelo previamente treinado com base de dados devidamente apropriada e após isso é possível realizar o ajuste fino do modelo para tarefas específicas, como *question and answering* (Q&A).

BERTimbau é um exemplo de transferência de aprendizado. Baseado no BERT, um modelo de aprendizado profundo desenvolvido pela equipe da Google conforme destacado anteriormente nesse trabalho, esse modelo foi pré-treinado especificamente para o português brasileiro e alcança ótimos desempenhos em diversas tarefas de NLP tais como Reconhecimento de Entidades Nomeadas, Similaridade Textual de Sentenças e Reconhecimento de Implicação Textual (SOUZA; NOGUEIRA; LOTUFO, 2020; DEVLIN et al., 2018).

O BERTimbau [14] foi pré-treinado pela Neuralmind utilizando o corpus BrWaC (Brazilian Web as Corpus), que representa um grande conjunto de dados extraídos da web em português (SOUZA; NOGUEIRA; LOTUFO, 2020). A etapa do pré-treinamento refere-se à aprendizagem semi-supervisionada com MLM (Masked Language Model), onde o modelo aprende as representações linguísticas gerais e, adicionalmente, aprende a prever palavras mascaradas, como por exemplo: "O gato está [MASK] no telhado". O BERTimbau aprende a prever a palavra "sentado"para preencher o token de mascaramento que representa a palavra.

Além disso, o modelo passou pela etapa de ajuste fino, onde foi utilizado o dataset SQuAD (Stanford Question Answering Dataset) versão 1.1, traduzido para o português pelo grupo Deep Learning Brasil. Esse modelo de dataset é o mais ideal para treinar modelos de Q&A; sua estrutura é composta de perguntas e respostas baseadas em trechos de texto (GUILLOU, 2021). Nessa etapa, o aprendizado é feito de forma supervisionada.

No momento em que o modelo recebe uma pergunta e o contexto o mesmo processa a entrada para gerar uma saída que são os IDs que representam as palavras da resposta no contexto fornecido e são obtidos por meio da função softmax que calcula as probabilidades das possíveis posições de resposta dentro do texto do contexto fornecido. São gerados os IDs de início e fim e, ao aplicar a softmax, tem-se a maior probabilidade da formação de um intervalo que resulta na resposta indicada pelo modelo. O Modelo de LLM utilizado nesse trabalho é o

[14] Disponível em: `https://huggingface.co/neuralmind/bert-large-portuguese-cased`

BERT large-cased [15].

Na arquitetura padrão do modelo, a quantidade máxima permitida de tokens é de 512, incluindo tanto a pergunta quanto o contexto. Este limite é um fator importante ao utilizar o modelo para aplicação em Q&A, pois informações além desse limite poderá ser truncada, podendo impactar a qualidade das respostas geradas.

4.2 Coleta dos dados

Para a criação da base de dados sobre suplementação esportiva, foram utilizados artigos da Wikipédia que abordavam esse tema em três idiomas distintos: português brasileiro, inglês e espanhol.

Para coletar os dados, foram empregadas técnicas de Web Scraping para reunir os textos dos artigos destinados a montagem da base de dados.

Foi realizado o pré-processamento dos textos de forma a deixá-los de forma bruta, removendo caracteres indesejados, tabelas, imagens e traduzindo-os para o idioma português. Após todo o processamento, a base de dados contou com um total de 23 artigos, que foram convertidos em PDFs.

4.3 Aplicação da RAG

Após o pré-processamento dos textos, foram desenvolvidas duas pipelines com o objetivo de fornecer o contexto mais relevante e com quantidade adequada de tokens para BERT, considerado também que o modelo possui uma limitação de 512 tokens, incluindo a combinação da pergunta e do contexto para a geração da resposta conforme as perguntas fossem feitas ao chatbot.

4.4 Arquitetura das Pipelines

As pipelines estão no ambiente Back-end onde recebem as requisições e processam as informações para geração de resposta as perguntas do usuário. O código fonte das pipelines [16] desenvolvidas e da interface [17] do chatbot estão disponíveis no GitHub. A Figura 17 apresenta a arquitetura do sistema de perguntas e respostas proposto nesse trabalho.

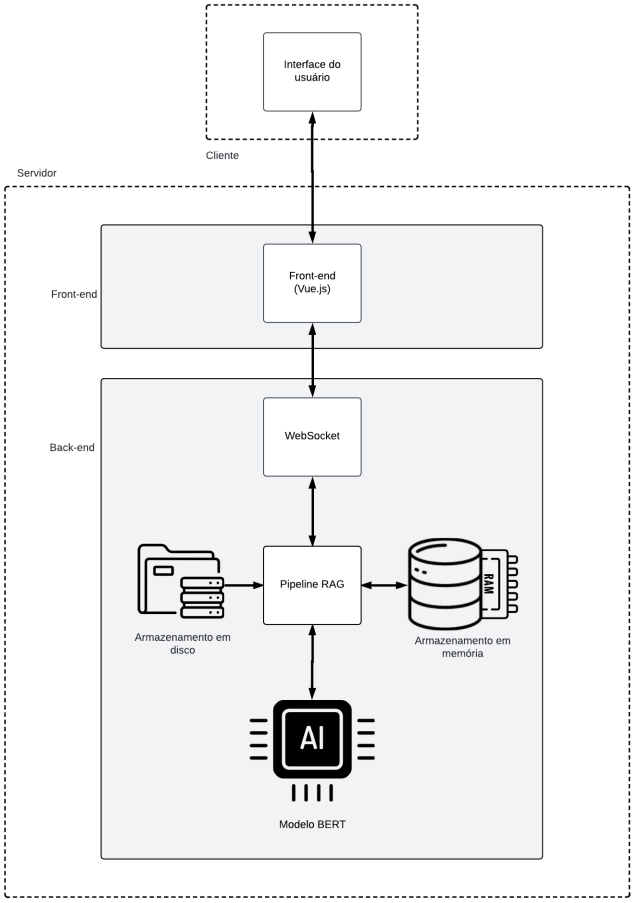
Na primeira pipeline desenvolvida, foi utilizada a blibioteca langchain juntamente com a biblioteca vectordb2. Os textos foram armazenados na memória RAM de forma agrupada e vetorizada utilizando a biblioteca faiss (Facebook AI Similarity Search) que permite a indexação e busca eficiente de *embeddings*, transformando dados textuais ou multimodais em vetores de alta dimensionalidade e utilizando algoritmos otimizados para encontrar rapidamente os vetores

[15] Disponível em: `https://huggingface.co/pierreguillou/bert-large-cased-squad-v1.1-portuguese`

[16] Disponível em: `https://github.com/WalterLops/chatbot_bert`

[17] Disponível em: `https://github.com/WalterLops/pipeline_bert_qa`

Figura 17 – Arquitetura do sistema



Fonte: Elaborado pelo autor

mais semelhantes em grandes *datasets*. Para fazer o armazenamento na memória RAM os textos foram carregados dos PDFs e agrupados em *chunks* (pedaços) com 490 tokens. Para fazer o armazenamento é utilizado o modelo sentence-transformers/multi-qa-mpnet-base-dot-v1 [18] para realizar os *embeddings* dos *chunks* fornecendo a funcionalidade de busca semântica nos textos dado a pergunta. Ao fazer as perguntas é feito uma busca semântica no método search da classe Memory da biblioteca vectordb2. E então é retornado os n melhores *chunks* da base de dados e para fazer essa busca é utilizado o algoritmo KNN para encontrar os *chunks* mais próximos da pergunta que também passa pelo processo de embedding conforme a base de dados. Os *chunks* são o contexto que são submetidos ao modelo BERT juntamente com a pergunta para gerar a resposta ao chatbot.

A segunda pipeline possui uma abordagem um pouco mais simples em comparação com a primeira. Nesta pipeline, é utilizado a biblioteca farm-haystack, onde os dados carregados dos PDFs são armazenados na memória RAM sem fragmentação, com todas as páginas de

[18] Disponível em: `https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1`

cada PDF mantidas juntas. Para tal, é utilizado a classe DocumentStore com a utilização de um algoritmo do tipo BM25 (Best Matching 25) para realização de pesquisa com recursos com base em similaridade textual. Uma função de recuperação de informações realiza a avaliação dos documentos a considerar a existência e frequência das palavras a partir de pesquisa ou pergunta. O BM25, uma variante sofisticada do modelo probabilístico de recuperação de informação, aprimora a ponderação dos documentos ao considerar a frequência de ocorrência dos termos, o comprimento dos documentos e uma série de outros atributos pertinentes. Este método, ao incorporar tais variáveis de forma intricada, permite uma recuperação de informações de maneira precisa.

Para recuperar o contexto juntamente com a pergunta, é criada uma ExtractiveQAPipeline. Nesta pipeline, é utilizado o modelo BERT, juntamente com o retriever, que acessa os dados armazenados na memória RAM.

4.5 Chatbot

O chatbot foi desenvolvido com o objetivo de ser uma interface Front-end que se comunica com o sistema onde o modelo BERT é executado, acessando uma base de dados contendo textos de artigos sobre suplementação esportiva. Para integrar o chatbot e permitir que ele faça perguntas ao modelo, foi utilizado o *framework* Vue.js. Este *framework* se conecta ao modelo BERT por meio de um servidor WebSocket.

O modelo BERT foi utilizado no ambiente Google Colab (Google Colaboratory), fazendo o uso de recursos de computação em nuvem para possibilitar o melhor desempenho a aplicação. Para tornar o servidor local da sessão do Google Colab acessível publicamente na web, foi utilizado a biblioteca pyngrok para criação um túnel com a utilização de TLS nativo. Esse túnel possibilita a conexão do servidor Vue.js com o modelo BERT hospedado no Google Colab, por meio de um link de conexão, garantindo que a comunicação com o referido chatbot seja realizada sem restrições arquiteturais.

4.6 Ferramentas utilizadas

Entre as ferramentas utilizadas para o desenvolvimento desse trabalho está a linguagem de programação Python e o *framework* Vue.js. Além disso, as principais bibliotecas utilizadas são: Transformers, langchain, vectordb2, nltk, PyMuPDF, pdfplumber, farm-haystack, flask e pyngrok.

O Google Colab, foi utilizado para a criação do ambiente de execução de código em Python diretamente no navegador. O ambiente configurado no Google Colab contou com 12,7 GB de memória RAM, GPU modelo NVIDIA-SMI, CPU Intel(R) CPU @ 2.20GHz e 79 GB de armazenamento em disco para a execução do modelo BERT Large.

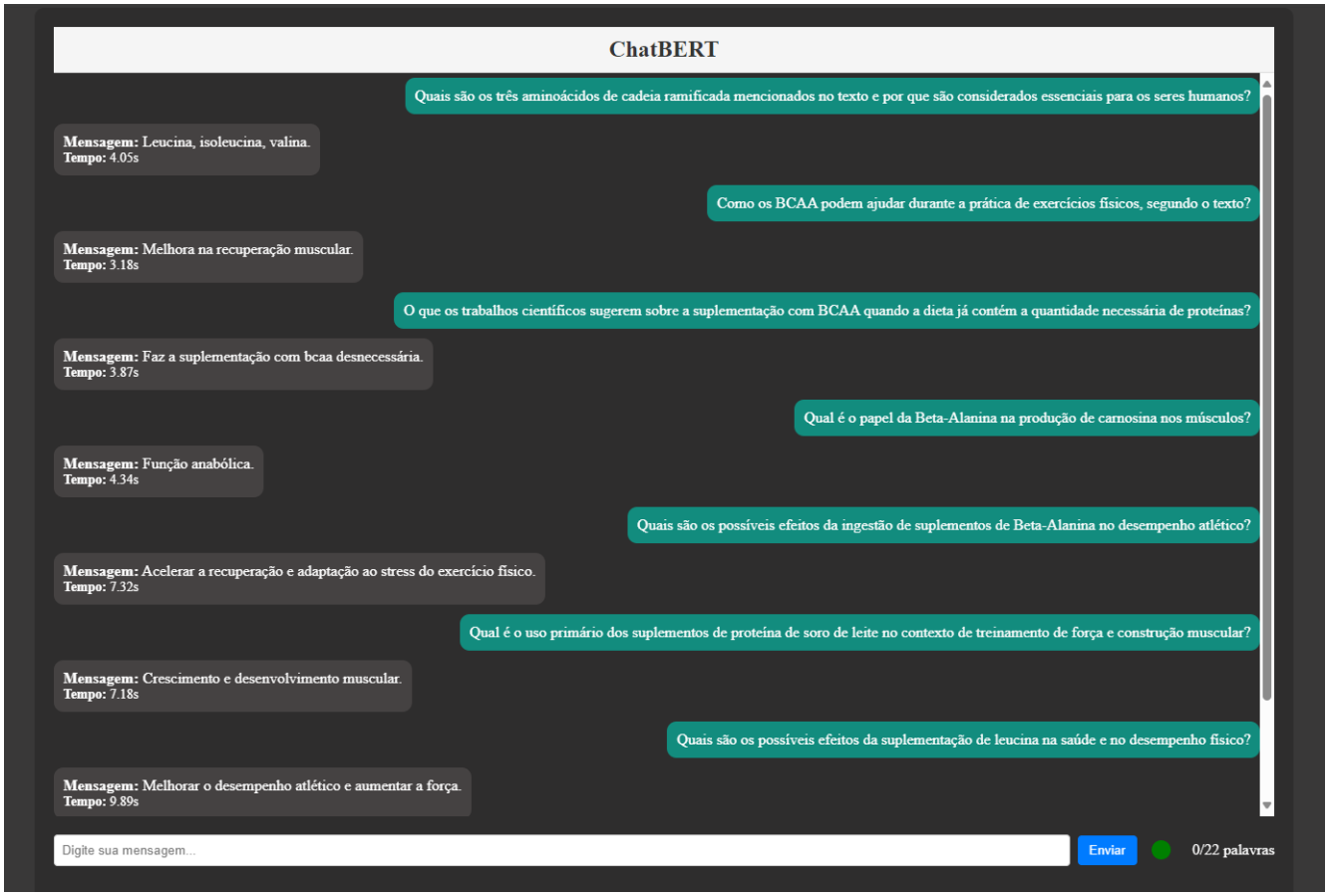
5 RESULTADOS

Este capítulo apresenta os resultados obtidos com a execução dos experimentos.

5.1 Chatbot

O chatbot pode utilizar diferentes pipelines a depender do tipo de ambiente. Se o ambiente possuir apenas CPU, então será utilizada a pipeline 1; se houver uma GPU disponível, então será utilizada a pipeline 2. O chatbot tem o seu funcionamento em um ambiente separado, onde faz requisições para o ambiente Google Colab, podendo enviar e receber mensagens através de um WebSocket que estabelece a conexão com o servidor Vue.js do chatbot. Na interface do chat, há um limitador que permite o envio de até 22 palavras para o ambiente do BERT. Além disso, como demonstrado na Figura 18 que apresenta a interface do chatbot, há um indicador de conexão com o servidor WebSocket, que mostra se o chat está desconectado (indicado pela cor vermelha) ou conectado (indicado pela cor verde).

Figura 18 – Demonstração do chatbot



Fonte: Elaborado pelo autor

5.2 Resultados das pipelines

A seguir, é apresentado o comparativo e os resultados de cada pipeline contando com perguntas de igual definição. Por questões de desempenho em tempo de execução, a pipeline 1 foi executada usando apenas CPU. A pipeline 2 foi executada usando GPU devido à necessidade de submeter o modelo a toda a base de dados o que torna as consultas muito demoradas para serem executadas apenas na CPU.

5.2.1 Pipeline 1 com a biblioteca vectordb2

Foram aplicadas 63 perguntas para a mensuração dos resultados dos procedimentos de RAG utilizados, integrados com o modelo BERT. Na tabela 1 a seguir, são apresentadas as 6 melhores respostas da pipeline 1, utilizando uma base de dados composta por artigos sobre suplementação esportiva, originados do site Wikipédia.

Tabela 1 – Melhores Perguntas e Respostas com a pipeline 1 utilizando o LangChain e vectordb2

Pergunta	**Resposta**	**Distância**	**Tempo de Resposta (s)**	**Tipo de ambiente**
Como a cafeína afeta o sistema nervoso central?	atrasa o início da fadiga muscular e da fadiga central	37.39	6.33s	CPU
Qual é a porcentagem de adultos na América do Norte que consomem cafeína diariamente?	90%	37.28	7.45s	CPU
Quais são as recomendações de ingestão diária de cafeína para adultos, crianças e adolescentes?	Associação Canadense de Saúde recomenda que crianças menores de 12 anos não devem receber mais de 2, 5 miligramas de cafeína por quilograma de peso corporal	31.56	5.25s	CPU
Qual é o uso primário dos suplementos de proteína de soro de leite no contexto de treinamento de força e construção muscular?	melhorar o desempenho atlético e aumentar a força	16.41	13.89s	CPU
Qual é o papel da fosfocreatina no músculo esquelético?	produzindo assim ATP	44.06	3.72s	CPU
Como a cafeína atua no sistema nervoso central e quais são seus efeitos no desempenho cognitivo e físico?	impede que a adenosina ative o receptor , bloqueando o local no receptor onde a adenosina se liga a ele	36.87	7.26s	CPU

Continua na próxima página

Tabela 1 – *Continuação da página anterior*

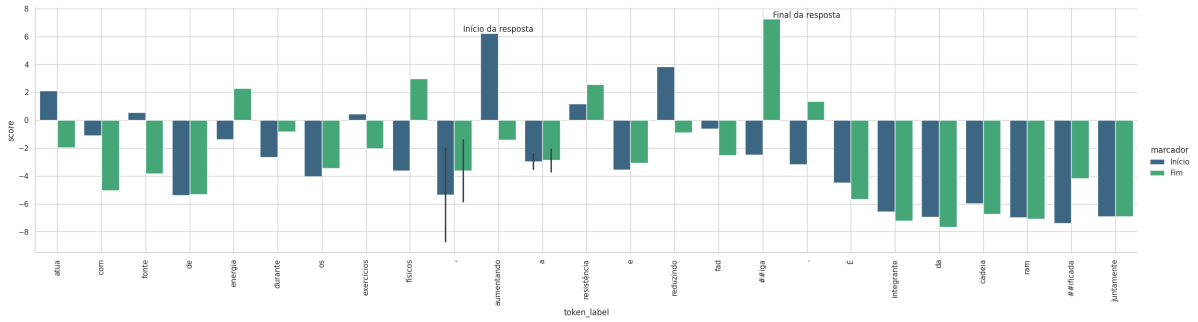
Pergunta	**Resposta**	**Distância**	**Tempo de Resposta (s)**	**Tipo de ambiente**
Como a fosfocreatina contribui para a reconstituição do ATP após um esforço muscular intenso?	ressintetizando trifosfato de adenosina (ATP) e transfere energia da mitocôndria para o citosol como "tampão espacial"fazendo assim o fornecimento rápido de energia. Esses mecanismos facilitam a homeostase do ATP no momento de turnover energético, mantendo reduzida a concentração de ADP e diminuindo a passagem de [UNK] do retículo sarcoplasmático	40.66	5.94s	CPU

Fonte: Elaborado pelo autor

Essa pipeline foi executada em um ambiente contendo apenas CPU, pois não requer muitos recursos computacionais para realizar as consultas. Foram submetidas 63 perguntas ao modelo de RAG implementado nessa pipeline, integrado ao modelo BERT, e na tabela estão as 7 melhores respostas concedidas para as respectivas perguntas.

A Figura 19 representa um gráfico que indica, em meio a um contexto, quais tokens têm maior probabilidade de iniciar e concluir a resposta para a pergunta "Quais são os possíveis efeitos da suplementação de leucina na saúde e no desempenho físico?", cuja resposta foi "aumentando a resistência e reduzindo a fadiga".

Figura 19 – Demonstração gráfica de resposta gerada pelo modelo BERT



Fonte: Elaborado pelo autor

A medida de distância retornada representa a distância vetorial entre uma pergunta e a base de dados. Ao submeter a pergunta para pesquisa por um chunk (pedaço), é feito o embedding (incorporação) da mesma e retornada uma quantidade n de *chunks*, cada um com uma medida de distância vetorial, onde quanto mais próximo de 0, mais preciso é o chunk em relação à pergunta. A distância retornada nas consultas do vectordb2 é calculado pelo FAISS (Facebook AI Similarity Search). O FAISS é chamado pelo vectordb2 para realizar a busca KNN (k-Nearest Neighbors). Como resultado, as distâncias calculadas previamente pelo FAISS são utilizadas como pontuações sendo aplicadas assim para determinar a similaridade entre o vetor da pergunta e os vetores do contexto armazenado.

5.2.2 Pipeline 2 com a biblioteca farm-haystack

Para essa pipeline também foram aplicadas 63 perguntas ao modelo para avaliar o processo de RAG, integrado com o modelo BERT. Na tabela 2, apresentam-se as 10 melhores respostas da pipeline 2, usando a base de dados que contém artigos sobre suplementação esportiva, provenientes do site Wikipédia.

Tabela 2 – Melhores Perguntas e Respostas com a pipeline 2 utilizando o farm-haystack

Pergunta	**Resposta**	***Score* da Distância**	**Tempo de Resposta (s)**	**Tipo de amiente**
Quais são os três aminoácidos de cadeia ramificada mencionados no texto e por que são considerados essenciais para os seres humanos?	Leucina, Isoleucina, Valina	0.92	4.65s	CPU e GPU
Como os BCAA podem ajudar durante a prática de exercícios físicos, segundo o texto?	Melhora na recuperação muscular	0.32	3.83s	CPU e GPU
O que os trabalhos científicos sugerem sobre a suplementação com BCAA quando a dieta já contém a quantidade necessária de proteínas?	Faz a suplementação com BCAA desnecessária	0.57	4.14s	CPU e GPU
Qual é o papel da Beta-Alanina na produção de carnosina nos músculos?	Função anabólica	0.31	5.22s	CPU e GPU
Quais são os possíveis efeitos da ingestão de suplementos de Beta-Alanina no desempenho atlético?	Acelerar a recuperação e adaptação ao stress do exercício físico	0.42	9.26s	CPU e GPU

Continua na próxima página

Tabela 2 – *Continuação da página anterior*

Pergunta	**Resposta**	***Score* da Distância**	**Tempo de Resposta (s)**	**Tipo de amiente**
Qual é o uso primário dos suplementos de proteína de soro de leite no contexto de treinamento de força e construção muscular?	Crescimento e desenvolvimento muscular	0.79	7.45s	CPU e GPU
Quais são os possíveis efeitos da suplementação de leucina na saúde e no desempenho físico?	Melhorar o desempenho atlético e aumentar a força	0.47	12.0s	CPU e GPU
Como a creatina pode influenciar o desempenho em exercícios anaeróbicos?	Melhorar o desempenho atlético	0.17	4.71s	CPU e GPU
Quais são alguns dos ingredientes comuns encontrados em suplementos pré-treino e quais são seus supostos benefícios?	Cafeína e creatina	0.98	12.2s	CPU e GPU
Como a cafeína atua no sistema nervoso central e quais são seus efeitos no desempenho cognitivo e físico?	Aumentar o fluxo sanguíneo	0.61	11.7s	CPU e GPU

Fonte: Elaborado pelo autor

Para essa pipeline, foi necessário ter a disponibilidade de uma GPU no ambiente de execução. A pesquisa tem por inicio a recuperação dos documentos utilizando como base o algoritmo BM25, o mesmo não requer muitos recursos computacionais de modo consequente possui eficiência em CPUs para buscas por similaridade textual. Desse modo, os referidos documentos são então passados para um modelo reader (BERT), que realiza a extração da resposta conforme a pesquisa realizada previamente. A inferência do reader com a utilização do modelo BERT é de elevada complexidade computacional, de tal modo sendo necessário o uso de GPUs para otimizar o processamento e propiciar respostas rápidas.

No farm-haystack, o BM25 realiza a busca inicial, enquanto o reader faz a extração da resposta. O *score* de BM25 indica a relevância textual dos documentos em relação a uma pesquisa, calculado baseando-se na frequência dos termos e métricas como IDF (Inverse Document Frequency), estima o grau de importância de um determinado termo dentro de um corpus de documentos. O *score* do reader projeta a confiança do modelo na precisão da resposta tendo como referência o texto de um determinado contexto recebido pelo modelo.

5.3 Comparação das pipelines

Na tabela 3 a seguir, são apresentadas as melhores perguntas e respostas aplicadas na pipeline 1 (P1), que serão usadas para comparar com a pipeline 2 (P2), a qual teve o melhor

resultado geral aplicado às 63 perguntas feitas sobre suplementação esportiva.

A motivação da escolha das 7 melhores perguntas como base para comparação se deve aos fatores que quantidade de perguntas respondidas e respondidas de maneira correta pela pipeline 1 ao qual teve menor quantidade de perguntas respondidas de forma satisfatória.

Tabela 3 – Comparação das pipelines

Pergunta	Resposta	Resposta	Distância	*Score*	Tempo de Resposta s	Tempo de Resposta s	Tipo de ambiente	Tipo de ambiente
	P1	P2	P1	P2	P1	P2	P1	P2
Como a cafeína afeta o sistema nervoso central?	atrasa o início da fadiga muscular e da fadiga central	ação antagonista não seletiva dos receptores de adenosina	37.39	0.36	6.33s	10.9s	CPU	CPU e GPU
Qual é a porcentagem de adultos na América do Norte que consomem cafeína diariamente?	90%	90%	37.28	0.96	7.45s	10.6s	CPU	CPU e GPU
Quais são as recomendações de ingestão diária de cafeína para adultos, crianças e adolescentes?	Associação Canadense de Saúde recomenda que crianças menores de 12 anos não devem receber mais de 2 , 5 miligramas de cafeína por quilograma de peso corporal	não mais de 400 mg	31.56	0.68	5.25s	11.0s	CPU	CPU e GPU
Qual é o uso primário dos suplementos de proteína de soro de leite no contexto de treinamento de força e construção muscular?	melhorar o desempenho atlético e aumentar a força	crescimento e desenvolvimento muscular	16.41	0.79	13.89s	8.55s	CPU	CPU e GPU
Qual é o papel da fosfocreatina no músculo esquelético?	produzindo assim ATP	agente ergogênico	44.06	0.64	3.72s	2.65s	CPU	CPU e GPU

Continua na próxima página

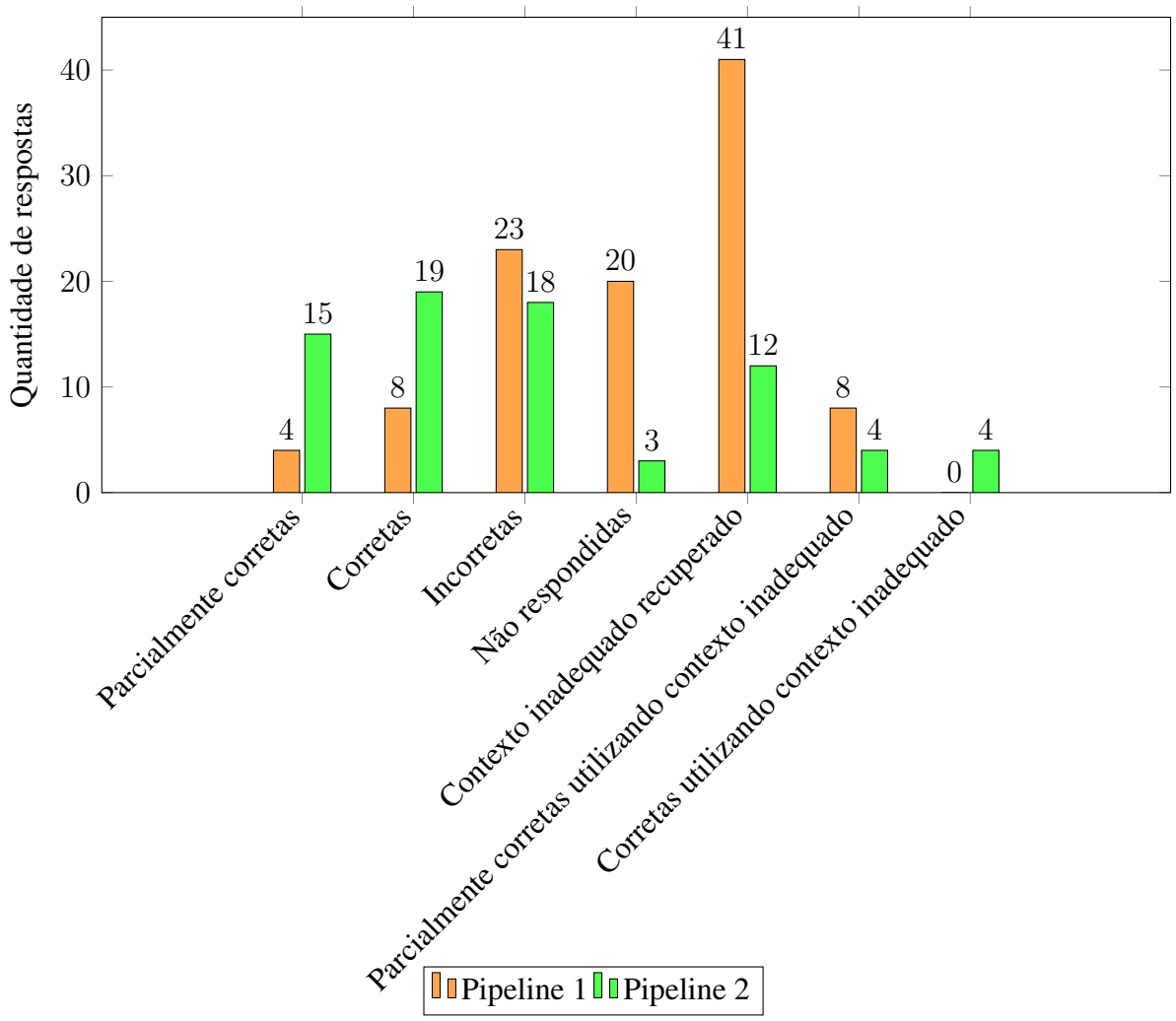
Tabela 3 – *Continuação da página anterior*

Pergunta	Resposta	Resposta	Distância	*Score*	Tempo de Resposta s	Tempo de Resposta s	Tipo de ambiente	Tipo de ambiente
	P1	P2	P1	P2	P1	P2	P1	P2
Como a cafeína atua no sistema nervoso central e quais são seus efeitos no desempenho cognitivo e físico?	impede que a adenosina ative o receptor , bloqueando o local no receptor onde a adenosina se liga a ele	aumentar o fluxo sanguíneo	36.87	0.61	7.26s	11.7s	CPU	CPU e GPU
Como a fosfocreatina contribui para a reconstituição do ATP após um esforço muscular intenso?	ressintetizando trifosfato de adenosina (ATP) e transfere energia da mitocôndria para o citosol como "tampão espacial"fazendo assim o fornecimento rápido de energia. Esses mecanismos facilitam a homeostase do ATP no momento de turnover energético, mantendo reduzida a concentração de ADP e diminuindo a passagem de [UNK] do retículo sarcoplasmático	formar creatina	40.66	0.38	5.94s	2.01s	CPU	CPU e GPU
Medianas			37.28	0.64	6.33	10.6		

Fonte: Elaborado pelo autor

Conforme demonstrado na Figura 20, no resultado geral da pipeline 1, observou-se que, das 63 perguntas aplicadas, 4 perguntas tiveram respostas parcialmente corretas devido à incompletude das respostas, enquanto 8 respostas foram completamente corretas. Nas demais perguntas, 20 não receberam resposta adequada devido ao contexto retornado pelo vectordb2 não ser apropriado. Além disso, em relação às outras 31 perguntas aplicadas, 8 foram respondidas de forma correta ou parcialmente correta, utilizando o contexto recuperado de maneira inadequada para a pergunta. Em alguns casos, a semântica do contexto era muito similar ao que estava sendo requisitado na pergunta, resultando em respostas corretas, parcialmente corretas ou incorretas. As demais respostas foram consideradas incorretas devido ao retorno inadequado de contexto ou chunk.

Figura 20 – Resultados das respostas das Pipelines 1 e 2



Fonte: Elaborado pelo autor

Já na pipeline 2, das 63 perguntas aplicadas, foram geradas 19 respostas corretas e 15 parcialmente corretas devido à incompletude das respostas. Nas 29 perguntas restantes, 3 não receberam resposta, e 18 foram respondidas de forma incorreta devido à recuperação insatisfatória de contexto. Utilizando o contexto recuperado de maneira inadequada, observou-se um resultado de 4 perguntas respondidas corretamente e 4 parcialmente corretas.

Na tabela 3, foram apresentadas todas as melhores perguntas da pipeline 1 em comparação com a pipeline 2 para o entendimento da consistência e utilização de recursos por cada uma das pipelines. Para verificar se as respostas geradas estavam corretas, foram observados os metadados de cada resposta retornada via console por cada pipeline como demonstrado na figura 21, onde é possível identificar o nome do documento e qual parte do mesmo foi utilizada para gerar o contexto. As perguntas também foram elaboradas considerando exclusivamente a base de dados construída previamente.

Portanto, como resultado, foi observado que a pipeline 2 teve o melhor resultado na qualidade de resposta. Entretanto, exige mais recursos computacionais para poder executar,

Figura 21 – Demonstração da resposta gerada por console pela pipeline 1

```
Digite a sua pergunta: Qual é o papel da fosfocreatina no músculo esquelético?
Documento: {'meta': {'chunk': '1'}, 'source': 'file: "Creatina  Wikipédia a enciclopédia livre.pdf"'}
Resposta: produzindo assim ATP
Distâcia: 44.0699462890625
```

Fonte: Elaborado pelo autor

devido à utilização do próprio modelo BERT para gerar as respostas finais por meio do reader. Essa pipeline utiliza similaridade textual por meio do algoritmo BM25, que demonstrou maior precisão para esse tipo de texto de suplementação esportiva.

Por outro lado, a pipeline 1 executou de forma mais rápida, não exigindo um ambiente com GPU para gerar as respostas. Um dos principais motivos de a pipeline 1 não ter tido boas precisões ao selecionar os *chunks* se deve à necessidade de um bom algoritmo de embedding para ser utilizado pela biblioteca vectordb2, especialmente para reconhecer as palavras do campo de suplementação esportiva.

Ao utilizar o modelo sentence-transformers/multi-qa-mpnet-base-dot-v1, que é um algoritmo de geração de *embeddings* de elevada medida de dimensionalidade otimizado para busca semântica e similaridade vetorial, é utilizado de forma integrada a arquitetura MPNet (Masked and Permuted Language Model), que realiza a combinação de técnicas de mascaramento e permutação de tokens para criar representações contextuais na recuperação das informações textuais.

O modelo de embedding não obteve bons resultados em textos de suplementação alimentar. O principal motivo de isso ter ocorrido se deve ao fato de o algoritmo não ter sido treinado com uma base de dados específica para esse tipo de conteúdo. Os *embeddings* gerados para palavras parecidas, como creatina, glutamina e albumina, não foram muito precisos, o que impossibilitou o retorno de *chunks* mais relevantes e consequentemente a falta de respostas para diversas perguntas.

Para obter uma melhor qualidade nas respostas, ao utilizar o sistema, é importante que a base de dados textual seja de ótima qualidade. Evitar termos repetidos em assuntos diferentes, como, por exemplo, repetir um conceito em diversos artigos, pois isso pode levar a pipeline de RAG a recuperar contexto incorreto e indesejado. A pipeline 1 demonstrou mais dificuldade em recuperar contextos de maneira mais assertiva nesse tipo de caso, com repetição de conceitos em artigos distintos. Outro ponto importante é ter o texto bruto sem tabelas e elementos visuais, com textos claros, concisos e boas estruturas gramaticais.

6 CONSIDERAÇÕES FINAIS

Neste trabalho, foi apresentada a aplicação do modelo BERT a um chatbot por meio de técnicas RAG, permitindo a utilização de uma base de dados sobre suplementação alimentar para responder perguntas feitas ao modelo. A NLP tem avançado cada vez mais com o uso de técnicas de RAG, que possibilitam a integração de documentos como base para a geração de respostas em LLMs. Durante o desenvolvimento deste trabalho, um dos principais desafios foi superar a limitação da janela de tokens do modelo BERT. Para a pipeline 1, a primeira a ser desenvolvida. Foi necessário montar os *chunks* considerando 490 tokens utilizando o tokenizador do BERT e deixar reservados 22 tokens para a pergunta. Outra limitação observada na pipeline 1, que utiliza a biblioteca vectordb2 como componente principal, foi a precisão da busca por similaridade vetorial, que não se mostrou muito precisa com os tipos de textos utilizados no trabalho.

Outra limitação encontrada refere-se à arquitetura de extração de informações do BERT. A mesma pode enfrentar alguns desafios ao responder perguntas compostas com múltiplas partes ou tópicos entre si, podendo ocasionar no retorno de respostas parciais. Um dos principais motivos de tal fato ocorrer é devido ao BERT necessitar do contexto fornecido, podendo dificultar a geração de respostas mais assertivas em situações de pesquisas mais complexas.

A pipeline 2 demonstrou bons resultados por meio da utilização da biblioteca farm-haystack com o algoritmo BM25 para busca por similaridade textual e recuperação de contexto e para geração de respostas no modelo BERT.

Por meio dos experimentos realizados, foi possível notar que a qualidade das respostas geradas é diretamente proporcional à qualidade da base de dados usada pelo modelo BERT por meio da utilização da RAG. Dados precisos e contextualizados resultam em respostas melhores e mais completas sendo geradas para as pesquisas dos usuários.

Um sistema de perguntas e respostas com BERT e RAG no campo de suplementação esportiva pode contribuir com avanços significativos. Apesar de BERT ter limitações em fornecer respostas detalhadas e responder perguntas compostas, a capacidade de entender o contexto das perguntas, combinada com a recuperação de informações do RAG, pode possibilitar o acesso rápido a dados confiáveis, desde que os dados sejam de qualidade. Isso contribui para o suporte educacional dos usuários sobre o uso de suplementos.

No dia a dia, esse tipo de aplicação pode oferecer suporte a dúvidas dos usuários sobre suplementação, de forma a auxiliar na tomada de decisões, permitindo combinar diferentes suplementos. Profissionais como nutricionistas podem se beneficiar com a otimização do atendimento ao cliente, direcionando melhor o foco a casos mais complexos e individualizados.

Para maior confiabilidade na recuperação de informações utilizando a técnica RAG, é muito importante desenvolver de forma integrada estratégias de curadoria contínua de dados, verificação por especialistas e mecanismos de feedback. Com essas práticas, é possível melhorar a precisão da aplicação e possibilitar que o sistema ofereça suporte educacional de qualidade,

resultando em um melhor uso dos suplementos e impactando positivamente a experiência dos usuários no campo da suplementação esportiva.

Para trabalhos futuros, recomenda-se o pré-treinamento de algoritmos de embedding, especialmente do tipo dot-product, para busca por similaridade vetorial de forma específica para o tipo de texto a ser utilizado, como, por exemplo, suplementação esportiva, além de expandir a base de dados, o que poderá resultar em ótimos resultados.

REFERÊNCIAS

ABDI, Hervé; VALENTIN, Dominique. **Mathématiques pour les sciences cognitives : avec des applications aux réseaux de neurones, au traitement du signal, à l'imagerie cérébrale et à la statistique**. Grenoble: Presses Universitaires de Grenoble, 2006.

AHMED, S.; NIELSEN, I. E.; TRIPATHI, A.; SIDDIQUI, S.; RAMACHANDRAN, R. P.; RASOOL, G. Transformers in Time-Series Analysis: A Tutorial. **Circuits, Systems, and Signal Processing**, v. 42, n. 12, p. 7433–7466, 2023. Disponível em: http://dx.doi.org/10.1007/s00034-023-02454-8. Acesso em: 19 mar. 2024. ISSN 1531-5878. DOI: 10.1007/s00034-023-02454-8.

CARVALHO, L. L. **Processos Cognitivos, Redes Neurais & Matrizes: material do curso de Inteligência Artificial**, p. 142, 2022.

CARVALHO, L. L.; PEREIRA, D. J.; COELHO, S. A. Origins and evolution of enactive cognitive science: Toward an enactive cognitive architecture. **Biologically Inspired Cognitive Architectures**, v. 16, p. 169–178, 2016.

CHO, K.; MERRIENBOER, B. van; GÜLÇEHRE, Ç.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. **CoRR**, abs/1406.1078, 2014. Disponível em: http://arxiv.org/abs/1406.1078. Acesso em: 19 mar. 2024. eprint: 1406.1078.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **CoRR**, abs/1810.04805, 2018. Disponível em: https://arxiv.org/abs/1810.04805. Acesso em: 04 abr. 2024. arXiv: 1810.04805.

DUFTER, P.; SCHMITT, M.; SCHÜTZE, H. Position Information in Transformers: An Overview. **Computational Linguistics**, v. 48, n. 3, p. 733–763, set. 2022. Disponível em: https://doi.org/10.1162/coli_a_00445. Acesso em: 08 abr. 2024. eprint: https://direct.mit.edu/coli/article-pdf/48/3/733/2040503/coli_a_00445.pdf.

FERREIRA, M. P. **Estudos de algoritmos de aprendizagem profunda no contexto de Processamento de Linguagem Natural para desenvolvimento de assistentes virtuais**. Jul. 2022. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/49001/1/EstudosDeAlgoritmosDeAprendizagemProfunda_Ferreira_2022.pdf Acesso em: 15 dez. 2023.

GAO, Yunfan; XIONG, Yun; GAO, Xinyu; JIA, Kangxiang; PAN, Jinliu; BI, Yuxi; DAI, Yi; SUN, Jiawei; WANG, Meng; WANG, Haofen. **Retrieval-Augmented Generation for Large Language Models: A Survey**. 2024. Disponível em: https://arxiv.org/abs/2312.10997?context=cs.LG. Acesso em: 25 abr. 2024. arXiv: 2312.10997 [cs.CL].

GOLDBERG, Y. **Neural Network Methods for Natural Language Processing**. Cham: Springer, 2017. (Synthesis Lectures on Human Language Technologies). Disponível em: https://doi.org/10.1007/978-3-031-02165-7. Acesso em: 14 mai. 2024. ISBN 978-3-031-01037-8. DOI: 10.1007/978-3-031-02165-7.

GUILLOU, P. **NLP | Modelo de Question Answering em qualquer idioma baseado no BERT base (estudo de caso em português)**. 2021. Medium. Disponível em: https://medium.com/@pierre_guillou/nlp-modelo-de-question-answering-em-qualquer-idioma-baseado-no-bert-base-estudo-de-caso-em-12093d385e78. Acesso em: 12 mar. 2024.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-term Memory. **Neural computation**, v. 9, p. 1735–1780, dez. 1997. DOI: 10.1162/neco.1997.9.8.1735.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences of the United States of America**, v. 79, n. 8, p. 2554–2558, 1982.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. In: 3rd ed. draft: Cambridge University Press, 2024. Chapter 9: RNNs and LSTMs. Disponível em: https://web.stanford.edu/~jurafsky/slp3/. Acesso em: 14 mai. 2024.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. In: 3rd ed. draft: Cambridge University Press, 2024. Chapter 10: Encoder-Decoder and Attention. Disponível em: https://web.stanford.edu/~jurafsky/slp3/. Acesso em: 14 mai. 2024.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. In: 3rd ed. draft: Cambridge University Press, 2024. Chapter 1: Introduction. Disponível em: https://web.stanford.edu/~jurafsky/slp3/. Acesso em: 14 mai. 2024.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. In: 3rd ed. draft: Cambridge University Press, 2024. Chapter 2: NLP Tasks and Applications. Disponível em: https://web.stanford.edu/~jurafsky/slp3/. Acesso em: 14 mai. 2024.

KHATTAB, O.; ZAHARIA, M. **ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT**. In: PROCEEDINGS of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020. Disponível em: https://dl.acm.org/doi/10.1145/3397271.3401075. Acesso em: 02 abr. 2024.

LEWIS, P. S. H.; PEREZ, E.; PIKTUS, A.; PETRONI, F.; KARPUKHIN, V.; GOYAL, N.; KÜTTLER, H.; LEWIS, M.; YIH, W.; ROCKTÄSCHEL, T.; RIEDEL, S.; KIELA, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. **CoRR**, abs/2005.11401, 2020. Disponível em: https://arxiv.org/abs/2005.11401. Acesso em: 05 abr. 2024.

LITTLE, W. A. The existence of persistent states in the brain. **Mathematical Biosciences**, v. 19, p. 101–120, 1974.

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **An Introduction to Information Retrieval**. Cambridge University Press, 2008. Disponível em: http://www.informationretrieval.org/. Acesso em: 14 mai. 2024. ISBN 9780521865715.

MARTINS, G D. **Named Entity Recognition com BERT para reconhecimento de entidades nas políticas operacionais do Banco Central do Brasil**. 2021. Disponível em: https://medium.com/@gdutramartins/named-entity-recognition-com-bert-para-reconhecimento-de-entidades-nas-politicas-operacionais-do-16439aa67a3b. Acesso em: 30 abr. 2024.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of mathematical biophysics**, University of Chicago Press, v. 5, n. 4, p. 115–133, 1943. Acesso em: 22 set. 2023.

MINSKY, Marvin; PAPERT, Seymour. **Perceptrons**. Cambridge, MA: MIT Press, 1969.

RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. **CoRR**, abs/1606.05250, 2016. Disponível em: https://arxiv.org/abs/1606.05250. Acesso em: 20 mai. 2024.

RAUBER, T. W. **Redes Neurais Artificiais**. 2005. Disponível em: <https://www.researchgate.net/publication/228686464_Redes_neurais_artificiais>. Acesso em: 12 dez. 2023.

ROSENBLATT, F. **Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms**. Washington, DC, USA, 1962.

ROSENBLATT, F. **The perceptron: a perceiving and recognizing automaton (Project PARA)**. Buffalo, N.Y., jan. 1957.

ROSENBLUETH, A.; WIENER, N.; BIGELOW, J. Behavior, Purpose and Teleology. **Philosophy of Science**, v. 10, n. 1, p. 18–24, 1943.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Institute for Cognitive Science, University of California; Department of Computer Science, Carnegie-Mellon University**, v. C-015, p. 10–21, 1986. Disponível em: <https://www.cs.utoronto.ca/~bonner/courses/2016s/csc321/readings/Learning%20representations%20by%20back-propagating%20errors.pdf>. Acesso em: 22 set. 2023.

SARKER, I. H. **Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions**. 2021. v. 2, p. 420.

SAXENA, S. **Introduction to Gated Recurrent Unit (GRU)**. 2024. Disponível em: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/. Acesso em: 10 mai. 2024.

SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. **IEEE Transactions on Signal Processing**, v. 45, n. 11, p. 2673–2681, 1997.

SHANNON, Claude E. A Symbolic Analysis of Relay and Switching Circuits. **Transactions of the American Institute of Electrical Engineers**, Massachusetts Institute of Technology, Cambridge, MA, v. 57, paper number 38–80, 1938. Presented at the AIEE summer convention, Washington, D.C., June 20-24, 1938. Manuscript submitted March 1, 1938; made available for preprinting May 27, 1938. Disponível em: <https://www.cs.virginia.edu/~evans/greatworks/shannon38.pdf>.

SHAW, Peter; USZKOREIT, Jakob; VASWANI, Ashish. **Self-Attention with Relative Position Representations**. In: PROCEEDINGS of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018. P. 464–468. DOI: 10.18653/v1/N18-2074. Disponível em: <https://aclanthology.org/N18-2074>.

SHERSTINSKY, A. **Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network**. abs/1808.03314. 2018. Disponível em: <http://arxiv.org/abs/1808.03314>. Acesso em: 25 jun. 2024. Disponível em: <http://arxiv.org/abs/1808.03314>.

SILVA, R. R. **Introdução ao processamento de linguagem natural: Desenvolvimento de um chatbot utilizando python**. 2023. Disponível em: https://www.ime.unicamp.br/~mac/db/2023-1S-120094.pdf Acesso em: 10 mai. 2024.

SMITH, Noah A. **Linguistic Structure Prediction**. Morgan & Claypool Publishers, 2011.

SONG, X.; SALCIANU, A.; SONG, Y.; DOPSON, D.; ZHOU, D. **Fast WordPiece Tokenization**. In: PROCEEDINGS of the 2021 Conference on Empirical Methods in Natural Language Processing. Online e Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. P. 2089–2103. Disponível em: https://aclanthology.org/2021.emnlp-main.160. Acesso em: 30 abr. 2024. DOI: 10.18653/v1/2021.emnlp-main.160. Disponível em: <https://aclanthology.org/2021.emnlp-main.160>.

SOUZA, Fábio; NOGUEIRA, Rodrigo; LOTUFO, Roberto. **BERTimbau: Pretrained BERT Models for Brazilian Portuguese**. Edição: Ricardo Cerri e Ronaldo C. Prati. Cham: Springer International Publishing, 2020. P. 403–417. ISBN 978-3-030-61377-8.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention Is All You Need. **CoRR**, 2017. Disponível em: http://arxiv.org/abs/1706.03762. Acesso em: 08 nov. 2023.

WIDROW, B.; HOFF, M. E. **Adaptive Switching Circuits**. Stanford, 1960. P. 96. Disponível em: <https://www.academia.edu/81564188/Adaptive_Switching_Circuits>. Acesso em: 14 dez. 2023.

WIENER, N. **Cybernetics: Or Control and Communication in the Animal and the Machine**. 2. ed. Cambridge, MA: The MIT Press, 1961. Disponível em: https://uberty.org/wp-content/uploads/2015/07/Norbert_Wiener_Cybernetics.pdf Acesso em: 15 abr. 2024.