UNIVERSIDADE FEDERAL DE JATAÍ
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGICAS (ICET)
PROGRAMA DE MESTRADO PROFISSIONAL EM MATEMÁTICA (PROFMAT)

LUCAS MEIRELES PEREIRA

A MATEMÁTICA POR TRÁS DE INTELIGÊNCIAS ARTIFICIAIS DO TIPO CHATBOT

JATAÍ-GO

2024

LUCAS MEIRELES PEREIRA

A MATEMÁTICA POR TRÁS DE INTELIGÊNCIAS ARTIFICIAIS DO TIPO CHATBOT

Dissertação apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional, Instituto de Ciências Exatas e Tecnologicas (ICET), Universidade Federal de Jataí (UFJ), como parte dos requisitos para a obtenção do título de Mestre em Matemática.

Área de concentração: Matemática do Ensino Básico

Orientador: Prof. Dr. Esdras Teixeira Costa

JATAÍ-GO

2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFJ.

Pereira, Lucas Meireles
A MATEMÁTICA POR TRÁS DE INTELIGÊNCIAS ARTIFICIAIS DO TIPO CHATBOT / Lucas Meireles Pereira. - 2024.
93 f.

Orientador: Prof. Dr. Esdras Teixeira Costa.
Dissertação (Mestrado) - Universidade Federal de Jataí, Instituto de Ciências Exatas e Tecnológicas, Jataí, PROFMAT- Programa de Pós-graduação em Matemática em Rede Nacional - Sociedade Brasileira de Matemática (RJ), Jataí, 2024.
Bibliografia. Anexos. Apêndice.
Inclui gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Redes Neurais Artificiais. 2. Inteligência Artificial. 3. Matemática de Chatbots. 4. Base Nacional Curricular. I. Costa, Esdras Teixeira, orient. II. Título.

CDU 51

UNIVERSIDADE FEDERAL DE JATAÍ

MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **35** da sessão de Defesa de Dissertação de **Lucas Meireles Pereira**, que confere o título de Mestre em **Matemática**, na área de concentração em Matemática do Ensino Básico.

No dia Quatorze de Setembro de dois mil e vinte e quatro, a partir das 10 horas, no auditório do Prédio da Pós-Graduação, da Universidade Federal de Jataí, Campus Jatobá, realizou-se a sessão pública de Defesa de Dissertação intitulada " A Matemática por trás de Inteligências Artificiais do Tipo Chatbot ". Os trabalhos foram instalados pelo Orientador, Professor Doutor Esdras Teixeira Costa (ICET-UFJ) , com a participação dos demais membros da Banca Examinadora: Professor Doutor Wender José de Souza (ICET - UFJ), membro titular interno; Professor Doutor Sadao Massago (DFQM - UFSCar) membro titular externo, de forma remota. Durante a arguição, os membros da banca n**ão fizeram** sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Esdras Teixeira Costa , Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, no dia Quatorze de Setembro de dois mil e vinte e quatro.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **ESDRAS TEIXEIRA COSTA**, **Professor do Magistério Superior**, em

14/09/2024, às 11:31, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **WENDER JOSE DE SOUZA**, **Coordenador do Programa Pós-graduação em Matemática**, em 19/09/2024, às 11:59, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Sadao Massago**, **Usuário Externo**, em 24/09/2024, às 21:03, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufj.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0333870** e o código CRC **9A812BF2**.

Referência: Processo nº 23854.007561/2024-70

SEI nº 0333870

RESUMO

PEREIRA, L.M. **A matemática por trás de inteligências artificiais do tipo chatbot**. 2024. 92 p. Dissertação (Mestrado em matemática - Mestrado Profissional em Matemática em Rede Nacional) - Instituto de Ciências Exatas e Tecnologicas (ICET), Universidade Federal de Jataí, JATAÍ-GO, 2024.

É notável que atravessamos um período de entusiasmos acerca dos potenciais da inteligência artificial (IA), haja visto a vasta gama de aplicações nas mais diversas áreas do conhecimento humano. Além disso, recentemente tem ganhado notoriedade na mídia, em especial chatbots, do tipo Generative Pre-Training Transformer (GPT), voltado para a elaboração de diálogos e textos. É importante ressaltar que as IAs necessitam de poder computacional, ao qual em sua linguagem mais básica é mantida por impulsos elétricos (Sequências de zeros e uns), o que implica que a existência de Inteligências Artificiais está atrelada a matemática desde às operações mais básicas. Além do mais, o próprio aprendizado de uma rede neural é envolto em probabilidade, álgebra linear, algoritmos, além de envolver áreas como estatística e ciência da computação, o que justifica uma busca pela compreensão de quais suportes matemáticos alicerçam uma rede neural. Assim, este trabalho busca entender a matemática por trás de inteligências artificiais do tipo chatbot, além de buscar tópicos compatíveis com a matemática básica do ensino médio, de acordo com a Base Nacional Curricular e contribuir com o acervo de dissertações do PROFMAT, onde trabalhos desta área ainda são escassos. A pesquisa teve cunho bibliográfico por meio de dissertações, teses, artigos, livros e páginas especializadas, utilizando autores como BOLDRINE, J. L. (1980), HEFEZ, A.; FERNANDEZ, C. S(2016), SHOKRANIAN, S. (2009) e STEINBRUCH, A.; WINTERLE, P. (1995), FELISBINO, R. (2012), SILVA, M. P.(2010), COZMAN, et al(2021), SILVA, M. P. (2010) e DASARADH, S.(2020). Para alcançarmos o objetivo, foi desenvolvido em linguagem Python, um código de rede neural simples, baseado no modelo “Saco de palavras” por meio do Google Colaboratory, com a função de classificar afirmações de geometria, código este que utilizamos de referência para uma análise linha a linha, com o intuito de descobrir a finalidade e a matemática por detrás de cada processo. Ao fim, foi possível localizar tópicos compatíveis com a matriz curricular do ensino médio, como funções de primeiro grau, funções de segundo grau, funções exponenciais, funções logarítmicas e linguagem de programação. Além disso, como produto dessa pesquisa, foi realizada uma sequência didática, ao qual um professor regente pode aplicar com seus estudantes.

Palavras-chave: Redes Neurais Artificiais; Inteligência Artificial; Base Nacional Curricular; Matemática de Chatbots..

ABSTRACT

PEREIRA, L.M. **The Mathematics Behind Chatbot-Type Artificial Intelligences**. 2024. 92 p. Dissertation (Mestrado em Matemática – Mestrado Profissional em Matemática em Rede Nacional) - Instituto de Ciências Exatas e Tecnologicas (ICET), Universidade Federal de Jataí, JATAÍ-GO, 2024.

It is notable that we are going through a period of enthusiasm regarding the potential of artificial intelligence (AI), given the wide range of applications in various fields of human knowledge. Moreover, it has recently gained media prominence, particularly chatbots, of the Generative Pre-Training Transformer (GPT) type, aimed at generating dialogues and texts. It is important to emphasize that AIs require computational power, which at its most basic level is maintained by electrical impulses (sequences of zeros and ones), meaning that the existence of Artificial Intelligences is intrinsically linked to mathematics, from the most basic operations. Furthermore, the learning process of a neural network itself is based on probability, li- near algebra, algorithms, and involves fields such as statistics and computer science, which justifies an exploration of the mathematical foundations underpinning a neural network. Thus, this work seeks to understand the mathematics behind chatbot-type artificial intelligences and explore topics compatible with basic high school mathematics, in accordance with the National Curriculum Guidelines (Base Nacional Curricular), and contribute to the collection of PROFMAT dissertations, where works in this area are still scarce. The research had a bibliographical nature, drawing on dissertations, theses, articles, books, and specialized websites, using authors such as BOLDRINE, J. L. (1980), HEFEZ, A.; FERNANDEZ, C. S (2016), SHO- KRANIAN, S. (2009), STEINBRUCH, A.; WINTERLE, P. (1995), FELIS- BINO, R. (2012), SILVA, M. P. (2010), COZMAN, et al (2021), SILVA, M.P. (2010), and DASARADH, S. (2020). To achieve the objective, a simple neural network code was developed in Python, based on the 'Bag of Words' model, via Google Colaboratory, with the function of classifying geometry statements. This code was used as a reference for a line-by-line analysis, aiming to uncover the purpose and mathematics behind each process. In the end, it was possible to identify topics compatible with the high school curriculum, such as linear functions, quadratic functions, exponential functions, logarithmic functions, and programming languages. Additionally, as a product of this research, a didactic sequence was developed, which can be applied by a teacher with their students.

Keywords: Artificial Neural Networks; Artificial Intelligence; National Curriculum Guidelines; Mathematics of Chatbots.

AGRADECIMENTOS

À minha família, meus amigos e a toda equipe do PROFMAT da UFJ.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

A proposta deste trabalho é estudar e procurar entender a matemática (e suas aplicabilidades) por trás de alguns processos usados em Inteligência Artificial Generativa, em especial Chatterbots baseados em Redes Neurais. Durante esse processo, procuramos perceber a importância da matemática para a área e quais são os tópicos presentes na matriz curricular do ensino básico que tem relação com a área da inteligência artificial.

Houve um empenho na busca de textos que demonstrassem e enfatizassem a forte presença da matemática, tal como sua relevância, levando em consideração as áreas da matemática que contribuem para a criação, aprimoramento e funcionamento de inteligências artificiais generativas.

Sichman (2021) afirma que atravessamos um momento de entusiasmo em relação aos potenciais benefícios da inteligência artificial, e de fato, podemos notar que a mesma têm se tornado uma área extremamente relevante, abrangendo uma vasta gama de aplicabilidades nas mais variadas áreas do conhecimento humano, acompanhada da evolução de processadores e do aprimoramento de diferentes técnicas de aprendizagem de máquina, e a grande quantidade de dados disponíveis na rede.

Podemos notar ainda a presença constante do tema nos canais e paginas de noticias, principalmente, acerca de chaterbots, do tipo Generative Pre-Training Transformer (GPT), voltado para a elaboração de diálogos e textos. A inteligência artificial é uma área relativamente jovem, com algumas poucas décadas de estudo. E embora tenha tido altos e baixos durante sua história, este campo tem crescido bastante, e ampliado seu leque de aplicações, que já abrange múltiplas áreas do conhecimento.

Recentemente, esse tipo de tecnologia recebeu bastante visibilidade em decorrência do avanço de inteligências do tipo Chatbot (Chatterbot), que de acordo com Silva (2010), trata-se de um software que interage com os usuários, fornecendo respostas para suas perguntas por meio da consulta em uma determinada base de dados.

Este tipo de software (chatbot) é apenas um dos papéis aos quais as IAs tendem a ter no futuro, haja visto, que elas têm substituído trabalhos repetitivos e auxiliado no ganho de produtividade em diversos setores. Sua aplicabilidade atual ainda é singela em relação ao que de fato essa tecnologia pode fazer, e tende a participar cada vez mais no desenvolvimento da humanidade. Outro fator relevante é que IAs necessitam de poder computacional, e conforme dito por Miranda e Mattar (2014), as informações acondicionadas em um computador nada mais são do que sequências de impulsos elétricos que em padrões humanos, podemos dizer que são sequências de zeros e uns. Assim, temos que a linguagem mais básica de um computador é regida pelo sistema binário, e por uma

área de matemática denominada Álgebra Booleana, que segundo Güntzel e Nascimento (2001) trata-se de um formalismo utilizado no tratamento sistemático da lógica. Assim, nota-se que a existência da Inteligência artificial está atrelada a matemática, desde as linguagens às operações mais básicas.

Além disso, Dasaradh (2020) afirma que, ao obter conhecimento matemático sobre redes neurais e aprendizado profundo (deep learning), pode nos proporcionar uma compreensão mais clara do funcionamento interno de uma rede neural. Tal pensamento é reforçado por Dara e al. (2022), que caracteriza o aprendizado de máquina (machine learning), como uma área combina estatística, probabilidade e álgebra linear, ciência da computação e algoritmos com o intuito de desenvolver Softwares inteligentes. Embora em nossa visão ciência da computação abranja tudo o que foi citado pelo autor, podemos perceber o quanto faz se importante, entender como se dá o processo de suporte matemático para a base de uma IA.

Além disso é importante salientar que esta pesquisa, também possui o intuito de identificar alguns tópicos matemáticos que sejam, de certa forma, compatíveis com o Ensino Médio, por meio do que esta previsto na Base Nacional Curricular (BNCC). Assim, ao fim desta pesquisa, faremos um levantamento de quais tópicos podem ser desenvolvidos e entender quais as habilidades seriam contempladas.

Outro fato relevante que justifica este trabalho é a escassez de pesquisas acerca do tema, visto que após uma busca pelo acervo de dissertações do PROFMAT, obtém-se poucas pesquisas relacionadas ao tema. Assim, esperasse que esta dissertação possa contribuir com o acervo do PROFMAT e para o avanço em pesquisas na área.

1 A INTELIGÊNCIA ARTIFICIAL

É notável que o desenvolvimento de inteligências artificiais, tal como seu aprimoramento, deve abrir novas perspectivas de futuro, Khakurel *et al.* (2018), trás possibilidades de participação de IAs em cinco níveis sustentáveis: Econômico; Técnico; Ambiental; Individual; Social.

No nível Econômico, o autor traz a possibilidade da indústria reduzir custos de produção, produzindo com mais eficiência energética, mais rápido e com menos mão de obra, o que pode baratear produtos, porém, pode afetar a economia negativamente, ao passo que pode deslocar ou substituir trabalhadores pouco qualificados ou oriundos de áreas substituíveis por automação.

Já no nível técnico, temos a IA como uma espécie de "individuo autodidata"capaz de aprender e ensinar a si mesma, gerando um desenvolvimento acelerado da IA, porém afetando negativamente vagas empregatícias na área de Tecnologia da Informação (TI).

A nível ambiental, poderíamos obter um melhor gerenciamento de resíduos e poluição, melhor aproveitamento dos recursos e extração, além da redução da emissão de poluentes com veículos autônomos mais eficientes, além de utilizar este tipo de software na prevenção de eventos extremos (como furacões e tsunamis). Por outro lado, tais ações poderiam gerar impacto negativo, como na extração exacerbada de recursos naturais, autoconsumo energético ou no aumento da obsolência programada.

A nível individual, o autor comenta sobre o ganho de produtividade, onde a IA auxilia em tarefas diversas, além da possibilidade de capacitar profissionalmente esses indivíduos. Por outro lado, pode afetar negativamente interações humanas, acarretando maior isolamento social. Por último, o autor retrata o nível social, ao qual a IA poderia dar assistência à comunidades, mídias sociais ou a atividades rotineiras (geralmente terceirizadas), novamente afetando vagas empregatícias.

Além destes aspectos, podemos citar também Marques (2020), que diz que a inteligência artificial há muito tempo faz parte do imaginário e cultura popular, haja visto as inúmeras obras cinematográficas como, Matrix, Ghost in The Shell, Exterminador do Futuro, O Homem Bicentenário, A.I. Inteligência Artificial, Ex Machina e muitos outros, além de livros, como “Fundação” e “Eu, robô”, de Isaak Asimov.

Todavia, embora o tema seja conhecido e já perdure por algumas décadas na comunidade cientifica, a definição do que viria a ser Inteligência Artificial não é bem esclarecida. (Cozman; Plonski; Neri, 2021, p.21) define, com ressalvas, a Inteligência Artificial como uma área voltada a construção de artefatos artificiais com comportamento inteligente. Já Schalkoff (1990 *apud* ARAGÃO, 2002, p.3), define como um campo de

estudo que busca explicar e emular comportamento inteligente em termos de processos computacionais. Porém, podemos notar que a definição do que é uma IA, é de fato complicada e ambígua, isso porque o conceito do que seria “inteligência” é subjetivo, além disso, se trata de uma área muito vasta, com diversas interpretações e conceitos que nem sempre estão alinhados.

É notável, que, embora não seja bem definida, a inteligência artificial tem estado cada vez mais presente em nosso cotidiano, com as mais diversas aplicações em múltiplas áreas do conhecimento. Além disso, no decorrer do tempo surgiram diferentes tipos de IAs, com estruturas e funcionalidades distintas.

1.1 Turbulências e triunfos: Alguns momentos da IA

A inteligência artificial é uma área relativamente jovem, Sichman (2021) relata que ela teria surgido na década de 1950, precisamente em 1956, na Darthmouth College Conference.

Porém, embora o campo da inteligência artificial parecesse promissor, passou por períodos de dúvidas, denominados “Invernos da IA”. Segundo Cozman, Plonski e Neri (2021), o mais famoso deles ocorreu na década de 1970, momento em que houve diversas críticas à área e retirada do suporte financeiro. Outro momento semelhante, teria ocorrido na década de 1990, neste caso, este período teria sido após uma expansão de investimentos em IA, o que incluía, segundo Russell e Norvig (2010) sistemas de visão, robôs, hardware e software especializados para este fim. A crise no setor, teria ocorrido em virtude do não cumprimento das expectativas sobre a tecnologia.

Todavia, Cozman, Plonski e Neri (2021), ressaltam que durante a década de 1990, foram desenvolvidas muitas técnicas computacionais "[...] em particular, houve significativo uso de probabilidades e lógica para representar conhecimento, bem como estatística para aprendizado e teoria da utilidade e de controle para tomada de decisão". Porém, poucas dessas técnicas teriam funcionado conforme o esperado.

Devido ao grande avanço no poder computacional, além da popularização e expansão da rede de computadores (Internet), tornou-se possível a utilização de novas técnicas de desenvolvimento e aprendizagem das IAs, sendo possível um novo avanço na área. Cozman, Plonski e Neri (2021) relatam que com uma quantidade maior de dados, foi possível a popularização da "mineração de dados", uma técnica desenvolvida na década de 1980 Russell e Norvig (2010), na qual acarreta em aperfeiçoamento no reconhecimento de padrões e aprendizado de máquina.

De acordo com Sichman (2021) atravessamos novamente um momento de entusiasmo com relação aos potenciais benefícios que a inteligência artificial pode promover. O autor justifica tal otimismo por meio de três fatores fundamentais:

> (i) o custo de processamento e de memória nunca foi tão barato; (ii) o surgimento de novos paradigmas, como as redes neurais profundas, possibilitados pelo primeiro fator e produzindo inegáveis avanços científicos; e (iii) uma quantidade de dados gigantesca disponível na internet em razão do grande uso de recursos tais como redes e mídias sociais. (Sichman, 2021)

1.2 A matemática das IAs

Por meio da leitura e busca de referências acerca da matemática presente na construção e funcionamento de inteligências artificiais, nos deparamos com algumas áreas da matemática que se relacionam intimamente com IAs.

Segundo Dara e al. (2022), a Álgebra Linear é utilizada para Deep learning, e tem papel na compreensão de Machine Learning, com concentração em vetores e matrizes. Segundo ele, os vetores são escritos em matrizes que podem ser acessadas por meio de índices e chaves (linhas e colunas), e a parir daí, são realizadas operações numéricas por meio do Módulo Numpy (bibliotaca Pyhon), que por sua vez, executa operações de adição, subtração, divisão e multiplicação de vetores e matrizes. O Autor ressalta, que sem a Álgebra Linear tais operações não poderiam ser realizadas, e consequentemente, não seria possível desenvolver e criar algoritmos de Aprendizado de Máquinas.

Outra área da matemática bastante relevante no campo das IAs, é a probabilidade. Ghahramani (2015), esclarecesse que a estrutura probabilística, descreve como representar e manipular a incerteza acerca de modelos e previsões, o que leva a assumir um papel central na análise de dados científicos, robótica, aprendizagem de máquina, ciência cognitiva e inteligência artificial.

Um exemplo acerca da aplicabilidade de probabilidade, ocorrem em IAs Generativas, que segundo Nelson (2021), esse modelo de inteligência depende da probabilidade conjunta, e são tipicamente empregados na estimativa de probabilidades e verossimilhança, realizando a modelagem de pontos de dados e distinguindo entre classes com base nas probabilidades associadas a eles. Conforme dito por Bell (2023), IAs generativas, são alimentadas com grande quantidade de dados, com o intuito de treinar modelos e a partir dai, produzir conteúdo. Assim, esses modelos aprendem a identificar padrões no conjunto de dados por meio de uma distribuição de probabilidade.

De acordo com Google Developers (2022), GANs (Redes adversárias generativas) tentam replicar distribuições de probabilidade. Dessa forma, utilizam de funções de perda que possibilitam refletir sobre a distância entre distribuições de dados gerados pela GAN e a distribuição dos dados reais. Vidal (2020), afirma que essa classe de Inteligência artificial, pode ser utilizada para trabalhar com otimização de imagens, geração de modelos 3D, produção de conteúdo, estimativas de séries temporais e etc.

Outra ferramenta matemática importante para IAs baseadas em redes neurais, são as funções de ativação. Cardoso (2021) define dois tipos de funções de ativação, as tradicionais e as adaptativas.

Cardoso (2021) define as funções tradicionais como funções não-lineares diferenciáveis (o que permite utilização de algoritmos de otimização). Essas funções efetuam transformações numéricas na saída de cada neurônio e possuem "a capacidade de generalização necessária para encontrar soluções de problemas não linearmente separáveis". Já as funções de ativação adaptativas, são definidas por Cardoso (2021) como aquelas que propõe uma definição dinâmica por meio de parâmetros otimizados pelo processo de treinamento da rede neural. Data Science Academy (2022) afirma que tais funções são conectadas a matemática (com o uso recorrente de gradientes, transformações lineares e não-lineares, por exemplo), e incluídas na estrutura de redes neurais artificiais com a finalidade de permitir a solução de problemas complexos.

Como IAs geralmente lidam com um grande volume de dados, faz se necessário que elas usufruam de ferramentas estatísticas. Conforme dito por Dara e al. (2022), a estatística é crucial para o campo de Machine Learning, haja visto que existe todo um processo de coleta de dados, e nesse meio, podem haver dados incorretos ou outras anomalias, que precisam de certo tratamento e filtragem, para que a informação se torne adequada e sucinta. Algumas das ferramentas elencadas pelo autor são a condensação, sumarização e conclusão, dispersão, assimetria, curtose, correlação e regressão. Além disso, Dara e al. (2022), ressalta que o aprendizado factual é o centro de qualquer aprendizado de máquina, e divide as ferramentas estatísticas em dois grupos, as descritivas (com o intuito de caracterizar e sintetizar o público-alvo, lidando com uma pequena quantidade de dados) e as inferenciais (que buscam fazer escolhas ou previsões com base em dados amostrais, trabalhando com uma grande quantidade de dados). Com base nos dados coletados e analisados, faz se necessário que a IA use estatística para auxiliar na tomada de decisões.

Cozman, Plonski e Neri (2021) subdividem a área da IA em três eixos: Representação do conhecimento; tomada de decisão e aprendizado. Tal que: a representação do conhecimento se relaciona a epistemologia e o raciocínio à lógica; a tomada de decisão à campos como psicologia, economia, engenharia e direito; o aprendizado de máquina à pedagogia e técnicas estatísticas para o processamento de dados. Cozman, Plonski e Neri (2021) afirmam ainda que, a palavra "conhecimento"é utilizada no campo da IA para se referir à um conjunto de formalismos que abrange desde fórmulas lógicas à probabilidades, tal como variações e combinações entre elas. Segundo o autor, exitem algumas tecnologias, cuja programação é baseada em restrições lógicas, que permitem representar a incerteza acerca de cenários complexos, dede que, as probabilidades sejam especificadas.

Segundo Cao *et al.* (2020) com a ampliação da escala de dados e capacidade de processamento de GPU's, surgiu-se o interesse em processar dados com domínio não

euclidiano, o que inclui gráficos e variedades, o que faz a necessidade de uma outra técnica para a aprendizagem da IA, denominada Geometric deep learning. O autor, ressalta que este tipo de aprendizado profundo, estuda principalmente gráficos e dados múltiplos, por meio de diferentes estruturas, que permitem uma serie de aplicações, como análise de redes, reconhecimento de movimento (ou padrões) humanos (por meio de analise de dados 3D), sistemas de recomendação (como Streamings e sites de compras), fluxo de tráfego, previsão de doenças e etc. Nota-se que as aplicações da geometria estão presentes na aprendizagem de Inteligências artificias, por meio de estruturas diversas, que se apoiam em áreas matemáticas, conectadas a álgebra e geometria, como teoria de grafos e topolgia, também citadas por (Cao *et al.*, 2020).

2 CONCEITOS MATEMÁTICOS

Para que possamos compreender alguns aspectos e funcionamento de inteligências artificias, faz-se necessário conhecermos a matemática que está envolta na IA, em especial, iremos abordar sobre tópicos de álgebra linear relevantes para a área. É importante salientar também, que um dos intuitos dessa pesquisa é encontrar tópicos de matemática compatíveis com o ensino médio, ou seja, que estejam de acordo com a Base Nacional Curricular (BNCC).

A BNCC é um documento normativo que define as habilidades essenciais que os alunos devem desenvolver ao longo da Educação Básica para garantir seus direitos de aprendizagem e desenvolvimento, conforme estabelecido pelo Plano Nacional de Educação (Educação, 2018).

Logo adiante, introduziremos os conceitos matemáticos que dão base para a compreensão de Chatboots, para mais adiante, sinalizarmos quais tópicos são aplicáveis no ensino básico.

2.1 Vetores

Baseado nas leituras de Reis e Silva (1996), Santos (2010), Neto e Neto (2023) e Hefez e Fernandez (2016), vamos discorrer acerca de alguns dos conceitos básicos de vetores. Em geral, podemos definir um vetor como um objeto que se associa os conceitos de direção, sentido e módulo. Além disso, que vetores são representados por segmentos (de retas) orientados (com sentido de percurso) no plano ou no espaço, contendo um ponto inicial em uma extremidade e um ponto final em outra. Por exemplo, o par ordenado $v = (8, 6)$ é um vetor, com ponto inicial $A = (0, 0)$ e ponto final $B = (8, 6)$.

Chamamos de módulo de um vetor $v \in R^2$, com $v = (a, b)$, o número $|v| = \sqrt{a^2 + b^2}$.

Dado $v = (8, 6)$, temos que $|v| = \sqrt{8^2 + 6^2}$, ou seja, $|v| = 10$.

Observe a representação do vetor v abaixo, na figura 1.

Além disso, com base na definição de módulo, podemos entender sobre a igualdade entre vetores. Dois vetores serão iguais, se possuem mesmo módulo, direção e sentido.

O vetor w com coordenada inicial $M = (1, 1)$ e coordenada final $N = (9, 7)$ é igual à v.

É importante ressaltar que no exemplo acima, o vetor não parte da origem. Nessas condições, dados os pontos $A = (x_1, y_1)$ e $B = (x_2, b_2)$, o vetor será dado por $AB = (x_2, y_2) - (x_1, y_1) = (x_2 - x_1, y_2 - y_1)$. Logo, voltando ao exemplo anterior, sendo $M = (1, 1)$ e $N = (9, 7)$ pontos de extremidade do vetor w, temos $MN = (9, 7) - (1, 1) = (9 - 1, 7 - 1)$,

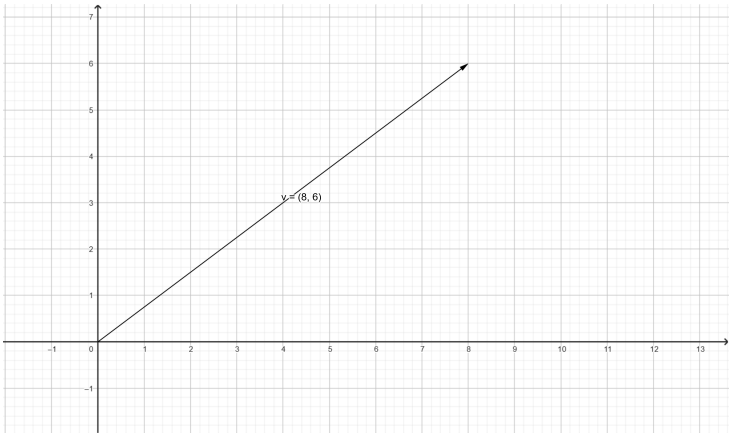

Figura 1 – Vetor v

Fonte: Elaborado pelo autor

ou seja, $MN = (8, 6)$. Portanto, como $w = MN$, temos que $w = v$.

Dados $u = (x_1, y_1), v = (x_2, y_2)$ e $\alpha \in R$, podemos definir a adição de vetores como $u+v = (x_1+x_2, y_1+y_2)$, sendo válidas as propriedades de comutatividade, associatividade e elemento neutro.

Por meio de u e v, já definidos, o autor nos mostra que a multiplicação de um vetor por um número é dada por $\alpha \cdot u = (\alpha \cdot x_1, \alpha \cdot y_1)$, onde se verificam as propriedades de distributividade, associatividade e elemento neutro.

Após o desenvolvimento da definição e algumas características de vetores, poderemos definir o conceito de Espaço Vetorial. Assim, segue a definição baseada em Pinheiro e Silva (2016).

Chamamos de espaço vetorial um conjunto, não vazio, V, munido das operações de adição (associa a cada par $u, v \in V$, o elemento $u + v \in V$ e multiplicação por escalar (que associa a cada par $\alpha \in R$, $v \in V$, o elemento $\alpha.v \in V$) e satisfaz as oito propriedades abaixo:

Quaisquer que sejam $u, v, w \in V$ e $\alpha, \beta \in R$

Em relação à adição:

- (A_1) associatividade: $(u + v) + w = u + (v + w)$
- (A_2) comutatividade: $u + v = v + u$
- (A_3) Existência de vetor nulo: $\exists \quad 0 \in V$ tal que $u + 0 = u$
- (A_4) Existência de inverso aditivo ou simétrico: $\exists \quad (-u) \in V$ tal que $u + (-u) = 0$

Em relação à multiplicação por escalar:

- (M_1) Distributividade com relação à adição de vetores: $\alpha(u+v) = \alpha u + \alpha v$
- (M_2) Distributividade com relação à adição de escalar: $(\alpha+\beta)u = \alpha u + \beta u$
- (M_3) Associatividade: $(\alpha.\beta).u = \alpha.(\beta.u)$
- (M_4) Existência do elemento neutro: $\exists \quad 1 \in R$ tal que $1.u = u$

Através da definição de espaço vetorial, podemos perceber um vetor como um objeto mais abstrato, que é apenas um elemento qualquer de um espaço vetorial. Portanto, polinômios, matrizes e funções também podem ser vetores. Mas, para o escopo deste trabalho, assim como para o ensino médio, adotaremos a definição de vetor já citada anteriormente.

2.2 Matrizes

Baseado nas leituras deBoldrine e outros (1980), Hefez e Fernandez (2016),Steinbruch e Winterle (1995), Pinheiro e Silva (2016), Leon (2010) definiremos alguns dos conceitos básicos de matrizes.

Uma matriz uma matriz pode ser definida como uma tabela com elementos dispostos em linhas e colunas. Dada uma matriz de m linhas e n colunas, dizemos que esta possui ordem $m \times n$, onde cada elemento a_{ij} (linha i e coluna j) é chamado de entrada da matriz. Podemos representar uma matriz pela notação $A = [a_{ij}]_{m\times n}$.

Dentro desta definição, existem diferentes tipos de matrizes, que variam de acordo com o número de linhas ou pelos elementos (ou posição) que os contém. Como por exemplo, a matriz linha, dada por $1 \times n$, ou a matriz coluna, $m \times 1$. Se $m = n$ teremos uma matriz quadrada, caso essa matriz tenha todos os elementos iguais a zero, com exceção da diagonal principal, ela é denominada, matriz diagonal.

2.2.1 Operações com matrizes

É possível realizar operações como soma entre matrizes, multiplicação por escalar e produto entre matrizes. Dado duas matrizes $A = (a_{ij})$ e $B = (b_{ij})$ de mesma ordem $(m \times n)$, então, existe uma matriz $C = (c_{ij})$, de tal forma que $(c_{ij}) = (a_{ij}) + (b_{ij})$, possuindo a mesma ordem de A e B.

Sejam

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} e \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Se C = A + B, então temos:

$$C = A + B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & -6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1+5 & 2+(-6) \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & -4 \\ 10 & 12 \end{pmatrix}.$$

Dadas matrizes A, B e C de mesma ordem, temos que a Adição de Matrizes é munida das seguintes propriedades:

I) Associatividade: $A + (B + C) = (A + B) + C$

II) Elemento Neutro (sendo "0"o notação para a matriz nula): $A + 0 = 0 + A = A$

III) Elemento Oposto (Sendo a matriz (-A) a oposta de A): $-A + A = A - A = 0$

IV) Comutatividade: $A + B = B + A$

Quanto à multiplicação por escalar:

Se λ é um escalar, o produto de uma matriz $A = (a_{ij})$ por escalar é uma matriz $B = (b_{ij})$ tal que: $(b_{ij}) = \lambda(a_{ij})$.

Dada uma matriz

$$A = \begin{pmatrix} 1 & 3 \\ -2 & 0 \end{pmatrix}$$

temos que

$$3 \cdot A = 3 \cdot \begin{pmatrix} 1 & 3 \\ -2 & 0 \end{pmatrix} = \begin{pmatrix} 3 \cdot 1 & 3 \cdot 3 \\ 3 \cdot (-2) & 3 \cdot 0 \end{pmatrix} = \begin{pmatrix} 3 & 9 \\ -6 & 0 \end{pmatrix}.$$

No que tange a multiplicação por escalar, dadas as matrizes A e B de mesma ordem, e os números $\lambda, \lambda_1 e \lambda_2$, temos que a multiplicação por escalar é munida das seguintes propriedades:

I) $\lambda(A + B) = \lambda A + \lambda B$

II) $(\lambda_1 + \lambda_2)A = \lambda_1 A + \lambda_2 A$

III) $0 \cdot A = 0$

IV) $\lambda_1(\lambda_2 A) = (\lambda_1 \lambda_2)A$

Dadas duas matrizes $A = (a_{ij}) \in M_{m \times p}\ (R)$ e $B = (b_{ij}) \in M_{p \times n}(R)$, definimos o produto de A por B, a matriz $AB = (c_{ij}) \in M_{m \times n}(R)$ tal que $c_{ij} = \sum_{k=1}^{p} a_{ik}.b_{kj}, i = 1, 2, ..., m; j = 1, 2, ..., n$.

Em outras palavras, podemos dizer que "o elemento (c_{ij}) [...] é obtido multiplicando os elementos da i-ésima linha da primeira matriz pelos elementos correspondentes da j-ésima coluna da segunda matriz, e somando estes produtos" (Boldrine; outros, 1980). O número de colunas de A deve ser igual ao número de colunas de B, tal fato é condicionante para a existência do produto entre duas matrizes. Além disso, a existência de um produto AB, não implica na existência do produto BA, e ainda que exita, em geral, $AB \neq BA$. Ou seja, a comutatividade no produto de matrizes não é garantida. Tal fato, pode ser exemplificado da seguinte maneira:

Dadas as matrizes:

$$A = \begin{pmatrix} 1 & -5 \\ 2 & 4 \end{pmatrix} e \quad B = \begin{pmatrix} -1 & 4 \\ 1 & -3 \end{pmatrix}.$$

Temos

$$AB = \begin{pmatrix} 1 & -5 \\ 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} -1 & 4 \\ 1 & -3 \end{pmatrix} = \begin{pmatrix} 1 \cdot (-1) + (-5) \cdot 1 & 1 \cdot 4 + (-5) \cdot (-3) \\ 2 \cdot (-1) + 4 \cdot (1) & 2 \cdot 4 + 4 \cdot (-3) \end{pmatrix}$$

$$AB = \begin{pmatrix} -1 - 5 & 4 + 15 \\ -2 + 4 & 8 - 12 \end{pmatrix} = \begin{pmatrix} -6 & 19 \\ 2 & -4 \end{pmatrix}.$$

Por outro lado, temos que

$$BA = \begin{pmatrix} -1 & 4 \\ 1 & -3 \end{pmatrix} \cdot \begin{pmatrix} 1 & -5 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} -1 \cdot 1 + 4 \cdot 2 & -1 \cdot (-5) + 4 \cdot 4 \\ 1 \cdot 1 + (-3) \cdot 2 & 1 \cdot (-5) + (-3) \cdot 4 \end{pmatrix}$$

$$BA = \begin{pmatrix} -1 + 8 & 5 + 16 \\ 1 - 6 & -5 - 12 \end{pmatrix} = \begin{pmatrix} 7 & 21 \\ -5 & -17 \end{pmatrix}.$$

Note que embora os produtos AB e BA existam, temos que $AB \neq BA$.

Caso o produto exista, temos as seguintes propriedades:

I) $A(B + C) = AB + AC$ (distributividade à esquerda da multiplicação em relação à adição);

II) $(A + B)C = AC + BC$ (distributividade à direita da multiplicação em relação à adição);

III) $(AB)C = A(BC)$ (associatividade);

IV) $AI = IA = A$ (existência de elemento identidade).

Podemos notar do item VI que $AI = IA = A$, de tal forma que chamamos I de matriz identidade.

Chamamos de matriz identidade, toda matriz I de ordem $n \times n$ de forma que $I = a_{ij}$, onde

$$a_{ij} = \begin{cases} 1 & se \quad i = j \\ 0 & se \quad i \neq j \end{cases} . \tag{2.1}$$

Para mostrar o que ocorre no ítem IV, basta observarmos o exemplo abaixo:

Sendo

$$A = \begin{pmatrix} 1 & 6 \\ 2 & -3 \end{pmatrix}.$$

e

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

temos:

$$AI = \begin{pmatrix} 1 & 6 \\ 2 & -3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$AI = \begin{pmatrix} 1 \cdot 1 + 6 \cdot 0 & 1 \cdot 0 + 6 \cdot 1 \\ 2 \cdot 1 + (-3) \cdot 0 & 2 \cdot 0 + (-3) \cdot 1 \end{pmatrix} = \begin{pmatrix} 1 & 6 \\ 2 & -3 \end{pmatrix}$$

.

A partir do conceito de matriz identidade, podemos definir a inversa de uma matriz.

Dado uma matriz A de ordem $n \times n$ é inversível se existe uma matriz B de forma que $AB = BA = I$. Assim, dizemos que B é a matriz inversa de A, também denotada como A^{-1}.

Dado

$$A = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$$

temos

$$A^{-1} = \begin{pmatrix} \dfrac{-1}{5} & \dfrac{3}{10} \\ \\ \dfrac{2}{5} & \dfrac{-1}{10} \end{pmatrix}$$

Pois,

$$A \cdot A^{-1} = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix} \cdot \begin{pmatrix} -\frac{1}{5} & \frac{3}{10} \\ \\ \frac{2}{5} & \frac{-1}{10} \end{pmatrix} = \begin{pmatrix} 1 \cdot \frac{-1}{5} + 3 \cdot \frac{2}{5} & 1 \cdot \frac{3}{10} + 3 \cdot \frac{-1}{10} \\ \\ 4 \cdot \frac{-1}{5} + 2 \cdot \frac{2}{5} & 4 \cdot \frac{3}{10} + 2 \cdot \frac{-1}{10} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

e

$$A^{-1} \cdot A = \begin{pmatrix} \frac{-1}{5} & \frac{3}{10} \\ \frac{2}{5} & \frac{-1}{10} \end{pmatrix} \cdot \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix} = \begin{pmatrix} \frac{-1}{5} \cdot 1 + \frac{3}{10} \cdot 4 & \frac{-1}{5} \cdot 3 + \frac{3}{10} \cdot 2 \\ \frac{2}{5} \cdot 1 - \frac{1}{10} \cdot 4 & \frac{2}{5} \cdot 3 - \frac{-1}{10} \cdot 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

3 SUBSÍDIOS COMPUTACIONAIS

É notável que para compreender o funcionamento de uma inteligência artificial, em especial do tipo Chatbot, se faça necessário compreender quais subsídios computacionais estão envolvidos neste processo, e em seguida, por meio de tais subsídios entender como a matemática participa nesse processo.

Além disso, fizemos uso de diversas bibliotecas que irão auxiliar no processo de compreensão de tais entes computacionais. Para ficar bem claro, o termo “biblioteca” na computação difere do significado tradicional, dado que se trata de “uma coleção de implementações de comportamentos escritos em uma linguagem e importadas no seu código” (Zanette, 2017). Basicamente, segundo o autor, alguns desenvolvedores disponibilizam bibliotecas com diversas funcionalidades já predefinidas, prontas para serem usadas. Dessa forma, outros desenvolvedores podem empregá-las, o que simplifica e acelera o desenvolvimento de trabalhos.

Nos próximos tópicos, discorreremos sobre as ferramentas computacionais e bibliotecas utilizadas nesta pesquisa.

3.1 Google Colab

Como citado anteriormente, para o desenvolvimento desta pesquisa, foram utilizadas diversas ferramentas, dentre elas o Google Colab. Tal ferramenta se tornou essencial para a escrita e execução do código de uma inteligência artificial a ser apresentada neste trabalho. Os códigos utilizados estão disponíveis publicamente por meio do link https://github.com/3sdras/sacodepalavras

No link, estão algumas das descrições acerca das funcionalidades do "Saco de palavras de geometria"para a criação do chatbot. Além disso, no github existe um link de acesso ao código na integra, contido no Google Colab, também apresentado nessa pesquisa. Neste local, podem ser realizados testes, simulações e alterações no código.

Segundo Santos (2010), o Google Colaboratory (conhecido como Google Colab), consiste em um serviço de nuvem gratuito disponibilizado pela própria Google, com o objetivo de promover a pesquisa em Aprendizado de Máquina e Inteligência Artificial.

Google (s.d.a) afirma que o Colaboratory, fornece acesso gratuito a recursos de computação, sem a necessidade de configurações prévias de GPUs e TPUs, ou instalações em sua máquina, o que descarta a exigência de um computador potente para seu uso, uma vez que a plataforma é executada via nuvem, em especial utilizando a linguagem Python. Além disso, segundo a organização, o serviço oferece compartilhamento fácil de dados, o que facilita trabalhos colaborativos.

3.2 Python

Optamos por discorrer o código em Python devido à sua praticidade e simplicidade, além de possuir bibliotecas bem documentadas. Um vez que tal linguagem conta com suporte do Google Colab e possui as ferramentas necessárias para o desenvolvimentismo de um sistema simples de inteligência artificial como o que será estudado nesta pesquisa. Dessa forma, neste tópico discorremos um pouco sobre o Python e algumas de suas características.

Paiva *et al.* (2020) afirma que a linguagem Python, foi criada pelo programador e matemático Guido Van Rossum em 1991. De acordo com o autor, o nome da linguagem foi inspirada no grupo humorístico Monty Python, porém, devido a recorrente associação de Python a serpentes, resultou na criação de uma logo com o animal, vindo a se tornar o símbolo oficial da linguagem.

Segundo Borges (2014) Python é uma linguagem de programação, geralmente definida "como uma linguagem de script orientada a objetos". Podemos complementar tal definição por meio de Luiz e Ascher (2007), que define o Python com uma linguagem de alto nível, interativa e dinâmica, que inclui estruturas como listas, dicionários, data/hora e etc.

Além disso, Luiz e Ascher (2007) afirma que se trata de um software de código aberto, sendo uma das principais linguagens no desenvolvimento de sistemas para pesquisas. O autor ressalta, que se trata de uma linguagem simples utilizada para completar tarefas e desenvolver programas rapidamente, quando comparado a outras linguagens, além de permitir a construção de programas sofisticados quando necessário.

Segundo Luiz e Ascher (2007) as principais vantagens da linguagem Python, são a qualidade do Software e a sua produtividade, uma vez que permite desenvolver códigos bem menores quando comparados a outras linguagens, o que gera praticidade. Para o autor, além de executar o código rapidamente, essa linguagem possui ótima portabilidade e uma vasta quantidade de bibliotecas com funcionalidades pré-compiladas, inclusive, consegue chamar C e C++ ou se integrar, interagir com outras linguagens de programação como o Java.

Python é uma linguagem multiplataforma, que carrega vários benefícios, que conforme dito por Sousa *et al.* (2020), transformou o Python em uma das principais linguagens no trabalho com inteligência artificial, data mininge e machine learning. Segundo Dörr e Aylon (2022), sua popularidade neste meio, se deve à presença de bibliotecas importantes,como NumPy, SciPy, Scikit-learn, Matplotlib e Keras. Várias delas podem ser utilizadas como ferramenta no Google Colaboratory, como veremos em mais detalhes na próxima seção.

3.3 Keras

No tópico anterior, citamos algumas bibliotecas da linguagem Python, porém, das citadas, teremos enfoque no Keras.

De acordo com Google (s.d.b), o Keras se trata de uma API (mecanismo que permite que dois componentes de software se comuniquem) de alto nível com o intuito de criar e treinar modelos de aprendizado profundo. Acerca da biblioteca, Keras (s.d.) possui o intuito de priorizar a usabilidade humana, por meio de práticas consistentes que reduzem as ações do usuário e fornecem mensagens de erro claras, além de possuir alta velocidade de depuração e prototipagem rápida. O Keras, pode ser utilizado para resolver diversos problemas que vão desde classificação de imagens, geração de texto, até sistemas de recomendação.

Segundo Keras (s.d.), a API é integrado ao TensorFlow, e oferece eficiência de compilação e otimizações, o que possibilita a implantação em plataformas diversas, como servidores, dispositivos móveis, navegadores e sistemas embarcados. Além disso, o autor revela que o Keras é utilizado por organizações científicas de renome, como CERN, NASA e NIH, e apresenta flexibilidade para implementar ideias de pesquisas de baixo nível de máquina, juntamente com conveniências de alto nível de máquina para acelerar experimentações.

Além disso, Melo (2019) afirma que o Keras é um biblioteca para “Deep Learning” e redes neurais. Em outras palavras, permite que os usuários criem modelos de redes neurais com poucas linhas de código, mas sem sacrificar a flexibilidade para os casos mais avançados.

3.4 Redes neurais

Para entender mais sobre redes neurais, vamos discorrer neste tópico um pouco acerca dos seus avanços no decorrer do tempo e acerca de alguns conceitos relevantes para a compreensão do tema.

É notável, que avanço nos desenvolvimentos de IAs resultou em inúmeras aplicações, algumas de acordo com Cozman, Plonski e Neri (2021) atingiram “desempenho humano ou super humano” como na detecção de rosto em fotografias e na sumarização de textos. Para o autor, o conjunto de técnicas que tornou isso possível, é denominado “aprendizado profundo”, cujo propósito é “[...] estimar, a partir de dados, uma **função** que relaciona um grande número de entradas (por exemplo, o conjunto de palavras em um texto) a um grande número de saídas (por exemplo, o conjunto de palavras que sumariza a entrada)”(Cozman; Plonski; Neri, 2021).

Felisbino (2012), afirma que muito provavelmente, a técnica de Rede Neurais Artificiais (RNAs) é a mais antiga no campo das IAs. Afirma ainda, que tal tecnologia

teria se originado na década de 40, com o intuito de fazer uma analogia entre neurônios biológicos e circuitos eletrônicos, de modo a criar uma representação que simulasse as conexões sinápticas por meio do uso de resistores variáveis e amplificadores. De acordo com Barreto (2002), tais redes, conseguem aprender por meio de exemplos (utilizando diferentes técnicas de aprendizagem), e posteriormente fazer interpolações do que foi assimilado.

De acordo com Alves (2020), Redes Neurais Artificiais (RNAs) são técnicas de aprendizado profundo que podem identificar padrões complexos em grandes conjuntos de dados. Tais redes, se inspiram na estrutura do cérebro humano e são projetadas para aprender com exemplos fornecidos durante o treinamento.

Segundo Alves (2020), as RNAs têm uma arquitetura composta por uma camada de entrada, uma camada de saída e pelo menos uma camada intermediária chamada de camada oculta. Essa camada oculta processa as informações de entrada e as transforma em um formato útil para a camada de saída. Cada uma dessas camadas é formada por neurônios artificiais que realizam cálculos para processar os dados. Um exemplo de RNA pode ser visto na figura 2.

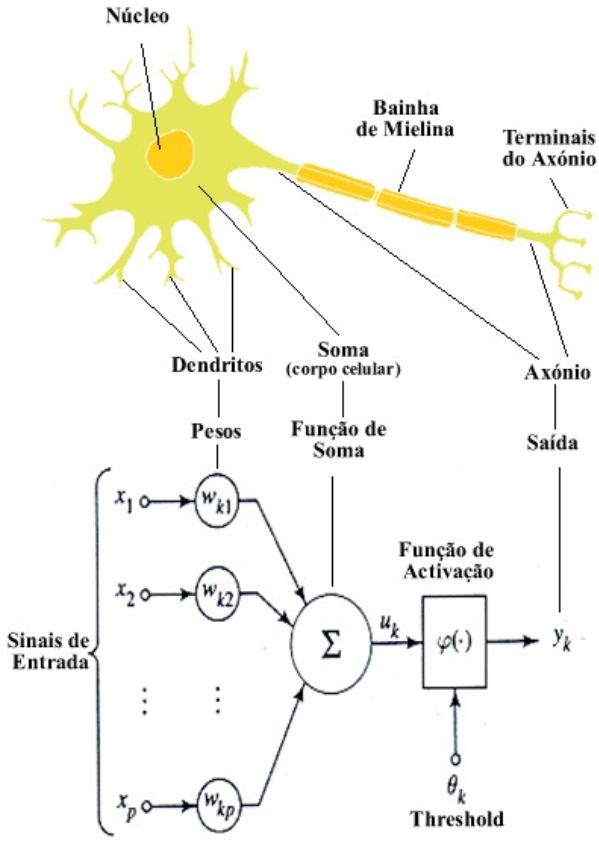

Figura 2 – Neurônio Biológico (acima) Vs Neurônio Artificial (abaixo)

Fonte: (Freitas, 2020)

De acordo com Dasaradh (2020), em 1958, Frank Rosenblatt criou o Perceptron,

que consiste em uma rede neural mais simples com uma quantidade n de entradas e apenas um neurônio de saída, de tal forma que n representa o número de recursos do conjunto de dados. Felisbino (2012) ressalta que o perceptron definido por Rosenblatt é munido de três camadas: "primeira recebe as entradas do exterior e possui conexões fixas; a segunda recebe impulsos da primeira através de conexões cuja eficiência de transmissão (peso) é ajustável e, por sua vez, envia saídas para a terceira camada (resposta)".

Aqui, podemos entender um paralelo, entre o neurônio artificial e o neurônio biológico, conforme apresentado na figura 2. Segundo Freitas (2020), da mesma maneira que neurônios humanos recebem estímulos do ambiente e órgãos sensoriais (olhos, etc.), o neurônio artificial recebe os sinais de entrada sobre a forma de **vetor**.

De acordo com Baroni e Padilha (2021):

> “um neurônio biológico recebe sinais por intermédio dos diversos dentritos, que serão examinados e encaminhados para o próximo neurônio, ou não (threshold). Nesta passagem pelo neurônio, o sinal poderá ter sua intensidade aumentada ou diminuída, dependendo do dendrito que o encaminhou, pois existe um peso associado a cada condutor, que será utilizado para multiplicar o sinal. As memórias são os pesos. ” (Baroni; Padilha, 2021)

De acordo com o autor, no decorrer de sua vida, o cérebro necessita de treinamento para cada nova tarefa, a fim de se especializar e aperfeiçoar seu desempenho. A partir daí, são definidos os pesos, ao qual chamamos de memorização. Ao passo, que as Redes Neurais Artificiais (RNAs) são estruturas compostas por camadas de nós, sendo a primeira a camada de entrada, seguida por camadas ocultas e, por fim, a camada de saída. Os neurônios em uma RNA recebem sinais de entrada, que podem vir dos dados brutos ou de neurônios em camadas anteriores, realizam cálculos e enviam sinais de saída para neurônios mais profundos na rede. Os neurônios podem ter sinapses conectadas a mais de um neurônio na camada anterior, e a partir dai efetuam cálculos e enviam sinais de saída para neurônios mais profundos por meio da sinapse. Cada sinapse tem um peso associado, determinando a importância do neurônio anterior na rede. Assim, o ajuste dos pesos é crucial no treinamento de modelos de aprendizado profundo.

Segundo Baroni e Padilha (2021), após receberem as entradas, os neurônios somam os sinais multiplicados pelos pesos correspondentes e os passam por uma função de ativação, que efetua o cálculo do valor de saída do neurônio, valor que é repassado adiante para a próxima camada da rede por meio de outra sinapse.

De acordo com Silva (2010), a função de ativação é aquela que processa o sinal gerado por meio da **combinação linear** das entradas e dos pesos das sinapses. Assim, na imagem 2, temos temos a representação de um neurônio de uma RNA, e de acordo com o autor, temos que a composição dessa função é dada "pelos sinais de entrada x_j, os pesos

associados às sinapses w_{kj}, o limiar (bias) θ_k, a função de ativação $\varphi(.)$, os sinais de saída y_k e o combinador linear da saída $\sum$"(Silva, 2010).

Em suma a autora afirma que um neurônio k pode ser descrito por meio das equações:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j + \theta_k \tag{3.1}$$

e

$$y_k = \varphi(u_k) \tag{3.2}$$

É interessante notar que Silva (2010) apresenta três das funções de ativação mais utilizadas:

> Função de limiar: a saída do neurônio é igual a zero, quando seu valor for negativo e 1, quando seu valor for positivo.
>
> $$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \tag{3.3}$$
>
> Função de limiar por partes: esse tipo de função pode ser visto como uma aproximação de um amplificador não linear.
>
> $$\varphi(v) = \begin{cases} 1 & \text{se} v \geq 0,4 \\ \frac{v}{2} & \text{se } +0,5 > v > -0,5 \\ 0 & \text{se } v \leq -0,5 \end{cases} \tag{3.4}$$
>
> Função sigmóide: é o tipo de função de ativação mais utilizado em redes neurais artificiais. É definida como uma função crescente, que apresenta um balanço entre o comportamento linear e não-linear. Um exemplo de função sigmóide é a função tangente hiperbólica, definida pela equação:
>
> $$\varphi(x) = \frac{1 - e^{-av}}{1 + e^{-av}}$$
>
> (Silva, 2010)

Em suma, a função limiar é utilizada quando a saída da rede é discreta (0 ou 1), a função limiar por partes é adequada para classificador binarico com região de incerteza e a função sigmoide em saídas contínuas e também em camadas intermediárias, mesmo em redes com saídas discretas, para facilitar o treinamento com métodos como o gradiente descendente, com o intuito de minimizar erros. Além do exemplo acima, também temos a função $\phi(x) = 1/(1 + (-x))$, como exemplo de sigmoide.

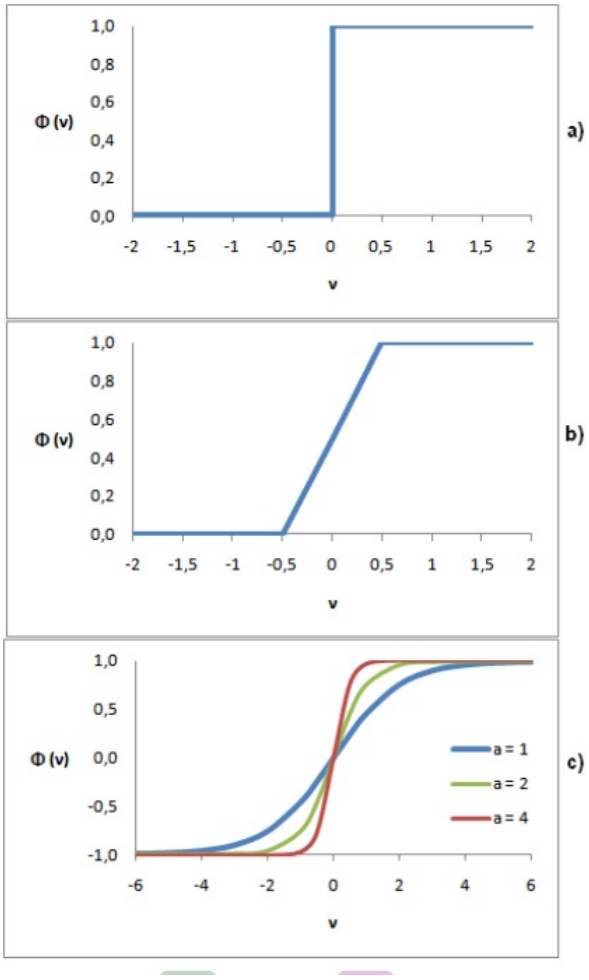

Figura 3 – (a) Função limiar, (b) Função limiar por partes e (c) Função sigmóide com parâmetro de inclinação a variável

Fonte: (Silva, 2010)

Outra função de ativação relevante é a função Tangente Hiperbólica conhecida por $Tanh(x)$, Embora possua um comportamento similar a Sigmoide, essa função projeta seus valores no intervalo [-1,1]. Por ser simétrica em torno da origem no plano de coordenadas pode ajudar o algoritmo de aprendizado a convergir mais rapidamente, sendo assim uma alternativa mais atraente do que a sigmoide para servir de ativação às camadas ocultas, visto que pode ajudar o algoritmo de aprendizado a convergir mais rapidamente.

Todavia, de acordo com Ali (2024), durante o processo de retro propagação, os gradientes da função tanh podem se tornar muito pequenos (próximos de zero), se tornando problemático em redes profundas e com muitas camadas, visto que os gradiente podem se tornar muito pequenos para fazer alterações significativas nos pesos, desacelerando o processo de treinamento e implicando em propriedades de convergência ruins. A escolha entre a Sigmoid e a Tanh, costuma depender se os valores a serem trabalhados serão positivos e negativos, ou somente positivos.

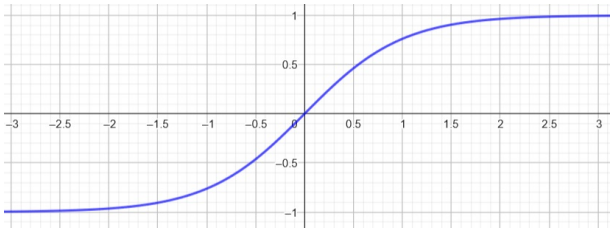

Figura 4 – Função de ativação tangente hiperbólica (Tan h (x))

Fonte: Elaborado pelo autor

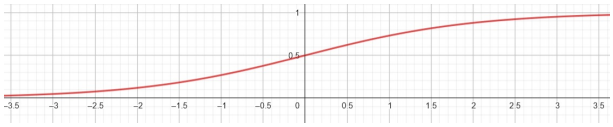

Figura 5 – Função de ativação Sigmoid ($\phi(x) = 1/(1 + (-x))$)

Fonte: Elaborado pelo autor

Outra função relevante para esta pesquisa é a max(x,0). Esta função tem o objetivo de retomar o maior valor entre x e 0. Ou seja, a intenção é truncar o valor negativo para zero, evitando assim, aparecer o valor negativo. Geralmente, utiliza-se esta função quando a saída da camada não pode ser negativa. Embora esta função seja mais rápida que do que $\phi(x)$, ela não evita que os valores aumentem com o processamento, o que pode ocasionar estouro no ponto flutuante em redes com grande dimensionalidade. Logo mais adiante, fizemos uso desta função baseado no exemplo de rede neural contido na figura 6.

Segundo Tatibana e Kaetsu (2023), arquiteturas neurais geralmente são organizadas em camadas, cujas unidades podem estar conectadas à unidades da camada subsequente. Além disso, os autores afirmam que o processo de treinamento de uma rede neural ocorre por meio de casos conhecidos, e a partir daí, executam eficientemente o processo desejado com base nos dados fornecidos. Dessa forma a rede neural torna-se capaz de extrair regras básicas por meio de dados reais, distinguindo-se da abordagem de computação programada, que requer um conjunto de regras rígidas pré-definidas e algoritmos.

Felisbino (2012) classifica as camadas em três grupos:

Camada de Entrada: Onde os padrões são apresentados à rede;

Camadas Intermediárias ou Ocultas: Onde é feita a maior parte do processamento, através das conexões ponderadas; Podem ser consideradas como extratoras de características;

Camada de Saída: Onde o resultado final é concluído e apresentado.

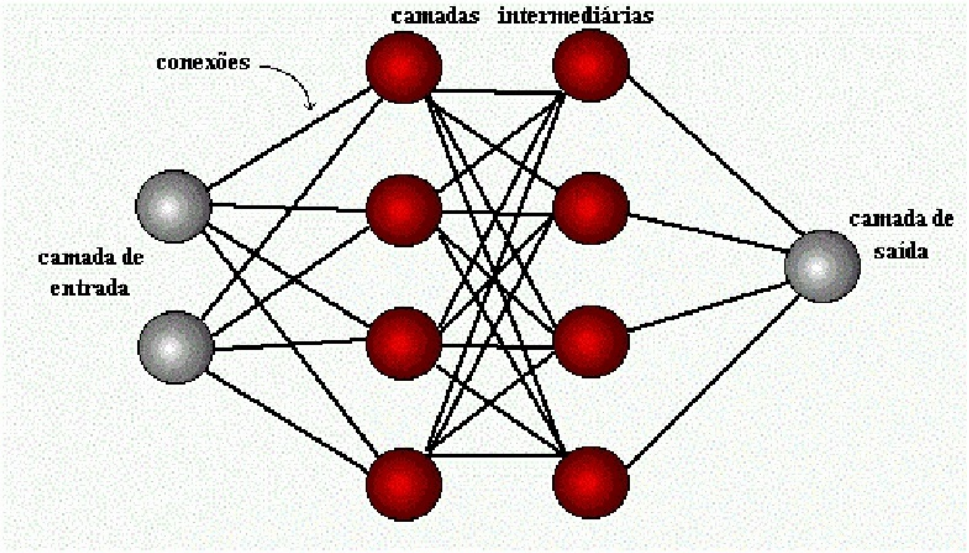

Figura 6 – Organização em camadas

Fonte: (Tatibana; Kaetsu, s.d.) APUD (Felisbino, 2012)

De acordo com Silva (2010), o treinamento ocorre por meio da aplicação sequencial dos vetores de entrada (por vezes os de saída), ao passo que os pesos da rede são ajustados de acordo com um processo de treinamento pré-determinado, ao qual os pesos da rede, gradualmente, convergem para determinados valores de tal forma que os vetores de entrada produzem as saídas necessários.

Ao descrever o funcionamento de uma rede neural, podemos perceber que alguns tópicos podem ser pouco compreendidos pela sua complexidade, assim, vamos trazer um exemplo básico de uma rede neral simples, para que o leitor posso compreender um pouco mais acerca do tema. Observe a imagem 3.

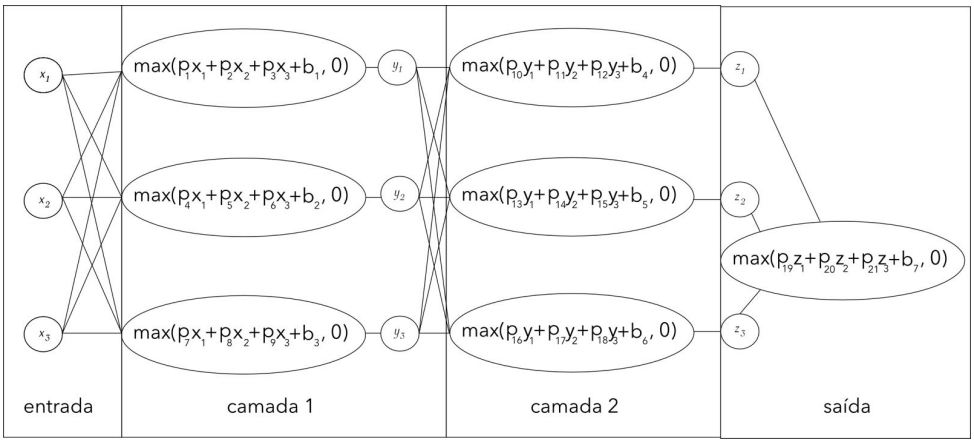

Figura 7 – Exemplo de uma Rede Neural simples

Fonte: Elaborado pelo autor

Podemos observar que neste exemplo, a rede neural possui três entradas, duas camadas ocultas e uma saída. Após o pré processamento do texto, a rede neural atribui valores para os pesos e bias, seguindo os atributos pré definidos de x_1, x_2 e x_3. Aqui, os pesos são representados por $p_1, p_2, p_3, ..., p_{21}$ e as bias por $b_1, b_2, ..., b_7$. Para entendermos como são efetuados os cálculos no interior dessa rede neural, vamos supor que após o pré processamento de dados, obtivemos os seguintes valores:

Primeira camada:

$p_1 = 0,5;\ p_2 = -0,2;\ p_3 = 0,8;\ b_1 = 0,1\ p_4 = -0,4;\ p_5 = 0,6;\ p_6 = -0,7;\ b_2 = 0,2\ \ p_7 = 0,3;$

$p_8 = -0,9;\ p_9 = 0,5;\ b_3 = -0,1$

Segunda Camada:

$p_{10} = 0,7;\ p_{11} = -0,3;\ p_{12} = 0,4;\ b_4 = 0,3\ \ p_{13} = -0,5;\ p_{14} = 0,2;\ p_{15} = -0,6;\ b_5 = 0,1\ \ p_{16} = 0,8;$

$p_{17} = -0,1;\ p_{18} = 0,9;\ b_6 = -0,2$

Camada de saída:

$p_{19} = 0,5;\ p_{20} = -0,4;\ p_{21} = 0,3;\ b_7 = 0,4$

Para as entradas, vamos supor $x_1 = 0,6;\ x_2 = -0,3$ e $x_3 = 0,8$

Suponhamos ainda, a função $max(X, 0)$, para evitar valores inferiores a zero, tal que X é a expressão estabelecida para cada neurônio no interior de cada camada respectivamente. Tomando como base o exemplo da imagem e os valores descritos, vamos efetuar os cálculos de saída da primeira camada. Para o primeiro neurônio temos:

$max(p_1x_1 + p_2x_2 + p_3x_3 + b_1, 0)$

$max(0,5\ .\ 0,6 - 0,2.(-0,3) + 0,8\ .\ 0,8 + 0,1\ \ 0)$

$max(1,1\ ;\ 0) = 1,1$

Para o segundo neurônios temos:

$max(p_4x_1 + p_5x_2 + p_6x_3 + b_2, 0)$

$max(-0,4\ .\ 0,6 + 0,6\ .\ (-0,3) - 0,7\ .\ 0,8 + 0,2\ ;\ 0)$

$max(-0,78\ ;\ 0) = 0$

Como nossa função de ativação possui $max(x\ ;\ 0)$ e -0,78 é um valor negativo, temos que o máximo é zero.

Para o terceiro neurônio:

$max(p_7x_1 + p_8x_2 + p_9x_3 + b_3\ ;\ 0)$

$max(0,3\ .\ 0,6 - 0,9.(-0,3) + 0,5\ .\ 0,8 - 0,1\ ;\ 0)$

$max(0,75\ ;\ 0) = 0,75$

Note, que temos pelo diagrama contido na imagem 3, temos que as saídas da primeira camada se tornam entradas da segunda, assim, temos que os valores de y_1, y_2 e y_3 são os valores obtidos nos máximos das funções anteriores, ou seja, $y_1 = 1,1; y_2 = 0$ e $y_3 = 0,75$.

Aplicando os valores na função de ativação da segunda camada, temos, para o primeiro neurônio:

$max(p_{10}y_1 + p_{11}y_2 + p_{12}y_3 + b_4\ ;\ 0)$

$max(0,7\ .\ 1,1 + (-0,3)\ .\ 0 + 0,4\ .\ (0,75) + 0,3\ ;\ 0)$

$max(1,37\ ;\ 0) = 1,37$

Quanto ao segundo neurônio dessa camada:

$max(p_{13}y_1 + p_{14}y_2 + p_{15}y_3 + b_5\ ;\ 0)$

$max(-0,5\ .\ 1,1 + 0,2\ .\ 0 + (-0,6)\ .\ 0,75 + 0,1\ ;\ 0)$

$max(-0,9\ ;\ 0) = 0$

Para o terceiro neurônio temos:

$max(p_{16}y_1 + p_{17}y_2 + p_{18}y_3 + b_6\ ;\ 0)$

$max((0,8\ .\ 1,1) + (-0,1)\ .\ 0 + 0,9\ .\ 0,75 + (-0,2)\ ;\ 0)$

$max(1,355\ ;\ 0) = 1,355$

Note novamente, que as saídas da segunda camada são entradas para a última. Dessa forma temos que, os valores de $z_1, z_2\ e\ z_3$ são os valores obtidos pela função na ordem disposta anteriormente, sendo $z_1 = 1,37\ ;\ z_2 = 0$ e $z_3 = 1,355$.

Assim, esses valores são aplicados na última camada, que em nosso esquema, produz uma única saída.

Temos que a função de ativação é dada por

$max(p_{19}z_1 + p_{20}z_2 + p_{21}z_3 + b_7\ ;\ 0)$

$max(0,5\ .\ 1,37 + (-0,4)\ .\ 0 + 0,3\ .\ 1,355 + 0,4\ ;\ 0)$

$max(1,4915\ ;\ 0) = 1,4915$

3.4.1 Redes Neurais Convolucionais

De acordo com Madeleine (2021), a rede neural convolucional, é formada por uma sucessão de blocos que extraem características que discriminam a classe pertencente à imagem de outros. Neste caso, como estamos trabalhando com chatbots, a classe pertenceria a palavra. Para o autor, uma rede neural convolucional possui ao menos 3 camadas, a

camada Convolucional (responsável por processar os dados de um campo receptor), a camada ReLU (com referência a função de ativação) e a camada Pooling (responsável por compactar as informações, reduzindo a dimensionalidade da imagem intermediaria).

Segundo Rodrigues (2018a), matematicamente, a convolução é um produto (uma operação binária) que recebe duas funções como entrada e retorna uma terceira função originária do somatório da multiplicação da função kernel na região superposta da função alvo pelo deslocamento do kernel sobre ela. Rodrigues (2018a) Assim, a função kernel é o filtro que está sendo aplicado à função de entrada através da operação de convolução. De acordo com o autor a fórmula é dada por:

$$h(k) = (f * g)(k) = \sum_{i=0}^{k} f(i).g(k-i) \tag{3.5}$$

De forma que f e g são sequências numéricas com tamanhos iguais ou variados, e a função retorna ao k-ésimo elemento do somatório da multiplicação. Em suma, $(f * g)(k)$ é uma especie de média ponderada (em caso de ser móvel, pode considerar tanto os valores anteriores como os posteriores, diferentemente da equação acima que considera apenas os dados anteriores) dos valores de g na vizinhança de k, e pesos determinados por f. O intuito é obter relação entre valores na vizinhança de k, juntando com informações fornecidas por f.

E importante salientar, que nossa pesquisa, baseia-se em uma rede convolucional. Além disso, de acordo com Luz, Salvatore e Finger (2018), redes convolucionais demonstram eficiência em tarefas que exigem identificação de padrões linguísticos, independentemente de sua posição. Por exemplo, em classificação de documentos, análise de sentimentos, entre outras.

Paiva e Pereira (2022), também afirma que além da convolução ser útil em visão computacional, também é prática para a extração de características importantes em dados sequenciais, uma vez que os dados pertencentes a uma sequência textual também compartilham características com seus elementos vizinhos. Então, de acordo com o autor, assim como em uma imagem um pixel compartilha algumas características comuns com pixels vizinhos, as palavras de um texto também armazenam algum tipo de relação com as palavras nas proximidades. Porém, os dados são obtidos por meio do Processamento de Linguagem Natural (PLN), definido por Amazon Web Services (2023) como "uma tecnologia de machine learning que oferece aos computadores a capacidade de interpretar, manipular e compreender a linguagem humana". Esse processamento, implica em dados unidimensionais (dimensão temporal), utilizando a convolução 1D, enquanto que no processamento de imagens, é utilizada a 2D são unidimensionais (dimensão temporal).

No entanto, como visamos apenas entender a matemática por trás do Chatboot, e procurar tópicos que podem ser aplicados no ensino médio, não iremos aprofundar muito

neste tópico, haja visto que aqui a matemática já alcança níveis mais complexos para um estudantes de nível básico.

4 MERGULHO DE PALAVRAS

Neste tópico iremos iniciar o estudo do código de uma inteligência artificial do tipo chatbot, produto desta pesquisa. Inicialmente, deveremos compreender o significado do termo "Bag of Words".

4.1 O modelo “Saco de palavras”

De acordo com Ferreira (2019), o modelo “saco de palavras” (Bag of Words - BoW) é um modelo de representação que busca simplificar dados textuais, ignorando a ordem e a gramática das palavras. Tal modelo, organiza as palavras em uma especie de “saco” de palavras, contabilizando a frequência de cada termo. Segundo o autor, este modelo é muito comum no Processamento de Linguagem Natural (PNL). Vale ressaltar, que PNL “é um ramo da inteligência Artificial que tem por objetivo interpretar e gerar textos em uma língua natural (e.g., Português, Inglês, Francês, Espanhol, etc.)” Robin (s.d.). Dessa forma, Massoni (2021) afirma que método Bag of Words, utiliza como covariáveis palavras que aparecem no texto, de forma que tais covariaveis assumam valores binários: 1 se a palavra apareceu no texto e 0 em caso contrário. É interessante ressaltar que quando o autor utiliza o termo "covariavel", cada palavra é tratada como uma variável que pode assumir diferentes categorias (as próprias palavras) e que essas variáveis são usadas para prever a variável resposta de interesse.

A seguir, baseada na tabela apresentada por Massoni (2021) em sua pesquisa, montamos o quadro abaixo para ilustrar a situação.

Texto 1: "Quem não tem cão caça com gato"

Texto 2: "Quem não arrisca, não petisca".

Palavras:	**Quem**	**não**	**tem**	**cão**	**caça**	**com**	**gato**	**arrisca**	**petisca**
Texto 1:	1	1	1	1	1	1	1	0	0
Texto 2:	1	1	0	0	0	0	0	1	1

Massoni (2021) afirma que existem variações de utilização de bag of words, como considerar a frequência em que aparece certas palavras (unigramas), ou considerar uma sequência com n palavras (n-gramas). Porém, segundo a autora, o bag of words possui certas limitações, uma vez que não consegue capturar relações semânticas entre as palavras e inversões de sentidos de sentido, como na palavra "não", por exemplo. Para a construção de uma rede neural mais sofisticada, podemos fazer a inclusão de outras metodologias como o Word Embedding.

O mergulho de palavras (Word Embedding) é abordagem de representações vetoriais de processamento de linguagem natural (PLN) que visa representar palavras como vetores numéricos contínuos em um espaço de alta dimensão. Segundo Fonseca (2021), O mergulho de palavras consegue criar formas eficientes e significativas para palavras, permitindo que modelos de processamento de linguagem natural capturem relações semânticas e sintáticas mais ricas e precisas, sendo possível expandir as dimensões para capturar mais informações sobre uma palavra. Uma forma de utilizarmos dessa ferramenta, é por meio do Word2vec, utilizado no produto desta pesquisa.

4.2 O modelo Word2vec

Word2vec, representa uma técnica de aprendizado de máquina que desenvolve um modelo de linguagem utilizando aprendizado profundo, de cordo com Paranhos (2023). O Word2vec, de acordo com Mikolov *et al.* (2013), é um modelo de rede neural que se fundamenta na hipótese linguística de que palavras semelhantes compartilham de contextos semelhantes, conhecido como similaridade distributiva, que amplia o poder de predição da rede neural, quando comparado com bag of words.

Ao inserir o corpo de um texto como entrada, o word2vec produz vetores de palavras como saída. Inicialmente, constrói um vocabulário com base nos dados do texto de treinamento, posteriormente, aprende a representação vetorial das palavras, assim o arquivo resultante torna-se útil para diversos aplicativos de processamento de linguagem natural e aprendizado de máquina, conforme Google (2013).

Além disso,segundo Kathuria *et al.* (2019), tal modelo é subdividido em dois tipos de arquitetura, Continuous bag-of-words (CBOW), que faz uso dos contextos (entrada) para prever a palavra central baseada nas que a cercam (saída), e Skip-gram (SG), de forma "inversa", visto que, utiliza a palavra central como entrada da dede neural com o intuito de prever os contextos (saída).

Para entender melhor o contexto, vamos tomar como exemplo a frase "O gato mia". Utilizando o Continuous bag-of-words (CBOW), treinaríamos a rede neural para deduzir a palavra "Gato" com base no contexto, ou seja "O.....mia". No caso do Skip-gram (SG) treinaríamos a rede neural para deduzir o contexto por meio da palavra, ou seja ".... gato...." para deduzir "o" e "mia".

Uma vez que fazemos uso da biblioteca Keras, torna-se importante entender algumas de suas ferramentas que estão inclusas em nosso código. Além disso,de acordo com Brownlee (2019), o keras, é uma ótima API para preparação de textos, ainda que para um grande volume de documentos. Segundo Brownlee (2019), o Keras fornece a classe tokenizer, cuja função é preparar documentos de textos para aprendizado profundo.

4.3 Tokenização

De acordo com Kathuria *et al.* (2019), a tokenização é o processo pelo qual há a quebra de strings (textos) em diferentes partes (denominadas tokens), como frases, palavras, palavras-chaves, símbolos e etc. Segundo Kathuria *et al.* (2019), alguns caracteres como sinais de pontuação, são considerados como irrelevantes, e são descartados, enquanto que os tokens se transformam na entrada para o próximo processo, a análise e mineração de texto, alimentando a rede neural.

Basicamente, ao utilizar as frases do quadro anterior "Quem não tem cão caça com gato"e "Quem não arrisca não petisca"no processo de tokenização teríamos essas frases quebradas em 9 tonkens onde cada um seria uma palavra: Quem; não; tem; cão; caça; com; gato; arrisca; petisca.

Chen (2023) afirma que a Tokenização assume que os tokens dos textos de entrada são delimitados por espaços em branco. Segundo o autor, a partir dai a ferramenta "keras.preprocessing.text.Tokenizer", consegue criar um dicionario paro todo o corpo do texto, mapeando os tokens por meio do "Tokenizes.fit_on_text()"

De acordo com Sharma (2021), o método fit_on_texts é utilizado para atualizar o vocabulário interno da lista de textos. Segundo o autor O fit_on_texts pode ser utilizado com os atributos Word_Counts, Word_index, "word_docs"e document_count. É importante dizer, que neste conceito, o termo documento representa unidades de textos ou sequências de caracteres que estão sendo processadas pelo tokenizer, podendo ser frases ou parágrafos inteiros por exemplo. Neste caso, consideraremos que cada documento representará uma frase.

Para compreendermos melhor o método fit_on_texts, iremos nos basear nas definições propostas por Sharma (2021) e tomar como base de texto, as duas frases utilizadas no exemplo anterior. Assim, conseguiremos descrever e exemplificar cada um dos atributos.

O Word_counts, age como um dicionário de palavras e efetua sua contagem. Assim, no exemplo citado poderíamos ter a seguinte representação: ("Quem", 2), ("não", 2), ("tem", 1), ("Cão", 1), ("caça", 1), ("com", 1), ("gato", 1), ("arrisca", 1), ("petisca",1).

Quanto ao Word_index, é responsável por atribuir números inteiros de forma individual para cada palavra. Ou seja, neste caso, poderiamos representar: ("Quem", 1), ("não", 2), ("tem", 3), ("Cão", 4), ("caça", 5), ("com", 6), ("gato", 7), ("arrisca", 8), ("petisca",9). Note, que nestas condições poderíamos representar a frase "Quem não arrisca não petisca"utilizando a sequência [1,2,8,2,9].

Já o Word_docs, elenca as palavras e indica em quantos documentos cada uma apareceu. Dessa forma teríamos ("Quem", 2), ("não", 2), ("tem", 1), ("Cão", 1), ("caça", 1),

("com", 1), ("gato", 1), ("arrisca", 1), ("petisca",1). Note que a contagem neste exemplo coincidiu com o apresentado no Word_counts. Porém, é apenas uma coincidência, visto que efetuam a contagem de itens diferentes.

O Document_count, efetua a contagem do número total de documentos. Tal função, é útil para ajustes no tokenizer. Como consideramos cada frase como um documento, teremos que essa contagem resultará em 2, visto que temos duas frases no banco de dados desse exemplo.

Após estes ajustes com os dados de treinamento, o Tokenizer poderá ser utilizado na codificação de documentos no treinamento, ou ainda, testar o conjunto de dados Brownlee (2019).

Outra função utilizada no Tokenizer é a texts_to_matrix(), cujo papel é criar um vertor por documento de entrada, onde o comprimento dos vetores é o tamanho do vocabulário Brownlee (2019). Dessa forma, tal função oferece uma variedade de padrões de codificação de texto para modelos de saco de palavras, os quais podem ser especificados por meio de um argumento de modo na função. Segundo Brownlee (2019), os modos incluem binary (se uma palavra está ou não no documento), count (indica a contagem de cada palavra no documento), tf-idf (a pontuação de texto e a frequência inversa para cada palavra no documento) e freq (indica a frequência de cada palavra e sua proporção em cada documento). Por fim, temos o print(encoded_docs) que exibe a matriz resultante da contagem de palavras gerada.

Logo, podemos perceber que o Tokenizer do Keras é uma ferramenta valiosa para realizar o processamento e tokenização de documentos de texto. Essa ferramenta possibilita a criação de matrizes de recursos, considerando a contagem de palavras, frequência das palavras ou outras métricas, conforme configurado no modo especificado pela função texts_to_matrix().

5 ALGUMAS FERRAMENTAS UTILIZADAS NO PROCESSO DE ESTUDO

Esta seção irá explicar detalhes acerca das ferramentas empregadas no algoritmo descrito no próximo capítulo. Daremos inicio falando um pouco sobre a biblioteca NumPy, contida no keras.

5.1 NumPy

Segundo Mulinari (2021) o NumPy (abreviação de Numerical Python) foi criado em 2005 por Oliphant, baseado nos projetos Numeric e Numarray com o intuito de a reunir a comunidade em um único sistema de processamento de arrays. Além disso, de acordo com a autora, o NumPy se trata de uma biblioteca com código aberto cujo intuito era realizar operações em arrays multidimensionais (ndarray). Já os arrays são estruturas de dados que possuem a capacidade de armazenar valores de mesmo tipo, o que permite operações eficientes em um grande volume de dados Awari (2023c) ou seja, são o equivalente computacional das matrizes.

De acordo com Awari (2023c), a biblioteca NumPy é bastante popular, uma vez que consegue realizar operações matemáticas (em suas mais diversas áreas) de forma rápida e eficiente, e oferece recursos úteis para modelagem matemática. Tal ferramenta, incluí uma vasta gama de funções matemáticas, como exponencial, trigonométrica e logarítmica. Outra vantagem, é sua capacidade de se integrar à outras bibliotecas Python, como o Matplotlib e o Pandas, o que facilita a análise de dados e a visualização de resultados, permitindo também a criação de gráficos e tabelas Awari (2023c).

Além disso, Mulinari (2021) afirma que essa biblioteca utiliza pouca memória, e diferentemente de outros objetos Python, ela armazena dados em um bloco contínuo de memória, assim, consegue acessar e modificar tais dados de forma eficiente, conceito conhecido como localidade de referência na ciência da computação. A autora afirma ainda que o NumPy, é capaz de realizar processamentos complexos, com agilidade, de 10 a 100 vezes mais rápidas em relação a outras aplicações nativas do Python, e é bastante utilizada na área de aprendizagem de máquina e processamento de imagens e rotinas matemáticas. Logo, devido a tais características, a biblioteca NumPy é uma boa pedida para compor uma rede neural baseada no Keras.

5.2 Ferramentas da biblioteca Keras

Outra ferramenta de grande importância é o keras.preprocessing.sequence, que fornece ferramentas para pré-processamento de sequências no keras. De acordo com TensorFlow (2023b), este módulo é responsável por fornecer três funções, make_sampling_-

table (Gera uma tabela de amostragem probabilística baseada em rank de palavras), pad_-sequences (cria Sequências de mesmo comprimento) e skipgrams(Gera pares de palavras skipgram).

A função mais relevantes nessa pesquisa é pad_sequences. De acordo com Zhu e Chollet (2023),uma vez que após o processamento de uma sequência de dados, é comum que amostras individuais tenham comprimentos diferentes, o que pode ser um empecilho para a rede neural, visto que os dados de entrada do modelo de aprendizado deve possuir um mesmo tamanho. Assim, conforme Zhu e Chollet (2023), a função pad_sequences faz com que as amostras mais curtas que um ítem mais longo, sejam preenchidas com valores nulos no inicio ou fim da sequência, ao passo que aos amostras muito longas também podem ser truncadas (reduzidas retirando-se informações excedentes no inicio ou fim da sequência) antes deste processo.

Para ilustrar este processo, imagine que após o processo de tokenização e vetorização de três documentos de um certo banco de dados obtivéssemos o seguinte:

$$[[10, 15, 32] \quad [8, 18, 5, 2] \quad [41, 9, 11, 35, 17]]$$

Como vimos em um tópico anterior, cada número inteiro representa uma palavra. Aplicando o pad_sequences teríamos:

$$[[10, 15, 32, 0, 0] \quad [8, 18, 5, 2, 0] \quad [41, 9, 11, 35, 17]]$$

Note, que neste exemplo o preenchimento ocorreu no fim da sequência. Caso o pad_sequence estivesse padronizado para sequências de comprimento máximo igual a 4, a última sequencia seria truncada para possuir o comprimento desejado. Assim teríamos obtido:

$$[[10, 15, 32, 0] \quad [8, 18, 5, 2] \quad [41, 9, 11, 35]]$$

Em nosso código também fizemos uso do Keras.models, objeto que agrupa camadas e pode ser treinado com dados TensorFlow (2023a). Segundo Kazmi (2023), existem dois tipos de modelos keras, o sequential e o API Model in Keras. Assim, de acordo com o autor, uma das possibilidades no keras.models é o modelo “sequential”, configurado para funcionar em uma pilha linear de camadas e com um único tensor de entrada e saída, permite a criação de modelos de aprendizagem profunda utilizando camadas sequenciais, ou seja, cada camada é adicionada uma após a outra. Porém, de acordo com Kazmi (2023), o sequential não suporta a criação de modelos com múltiplas entradas e saídas.

Quanto ao API Model, TensorFlow (2023a) afirma que é utilizado em arquiteturas mais robustas, uma vez que se trata de uma forma de criar modelos mais flexíveis em relação ao keras.sequential, além de poder lidar com modelos não lineares, camadas

compartilhadas, múltiplas entradas e saída, e permitir a construção de gráficos arbitrários de camadas ou subclasses para escrever modelos do zero.

Assim, TensorFlow (2023a) afirma que a classe tf.keras.model, é munida de métodos integradas de treinamento e avaliação, sendo eles: tf.keras.Model.fit (responsável por treinar o modelo para um número fixo de épocas); tf.keras.Model.predict (cujo papel é geral previsões de saída para as amostras de entrada); tf.keras.Model.evaluate (configurado pelo método tf.keras.model.compile, retoma valores de perca e métricas do modelo).

5.3 Camadas de uma rede neural

Outro fator crucial para a existência de uma rede neural, é a existência de diversas camadas, ou keras.layers. De acordo com Muzaffar (2023), as camadas Keras, são fundamentais no aprendizado profundo, pois elas definem a arquitetura e funcionalidade de cada modelo de rede neural. Segundo o autor, cada camada executa uma operação específica nos dados de entrada e produz uma saída de tal forma, que pode servir de entrada para uma próxima camada. Assim, uma camada transforma os dados de entrada, por meio de operações matemáticas com o intuito de gerar resultados significativos, utilizando a entrada da camada anterior e passa adiante para a próxima . Observe a ilustração trazido pelo autor, acerca das interações entre as camadas.

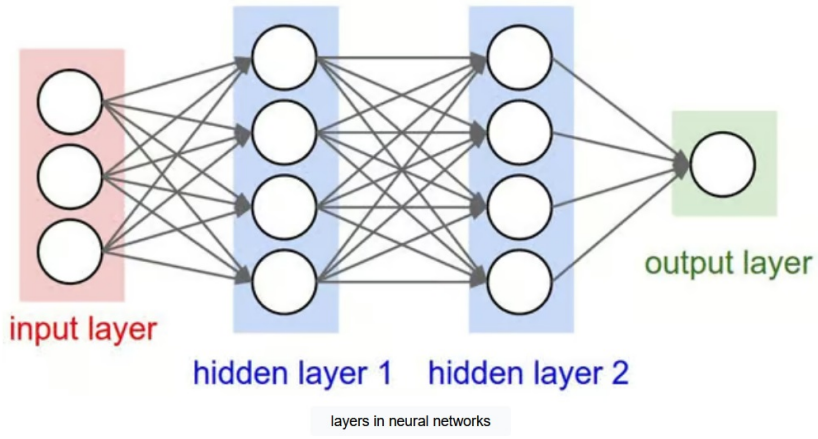

Figura 8 – Camadas em redes neurais

Fonte: (Muzaffar, 2023)

Na imagens, podemos notar o processo descrito, a camada input layer recebe os dados de entrada e os distribui para as camadas ocultas (hidden layer), que realizam as transformações nos dados de entrada à medida que passam pela rede, e entregam os dados à output layer, saída final da rede neural. Vale ressaltar, que redes mais profundas (mais camadas) podem aprender representações mais complexas. Há diversos tipos de camadas no

keras, porém iremos focar apenas em algumas delas: Dense, Embedding, Flatten, conv1d, maxpooling1.

Awari (2023a) afirma que na camada Dense, cada neurônio está conectado a todos os neurônios da camada anterior, o que permite que as informações se propaguem livremente pela rede, ampliando a capacidade de aprendizado. Além disso, essa camada, também gera uma saída para todos os neurônios da próxima camada. De acordo com o autor, esse tipo de camada, é bastante utilizada no desenvolvimento de redes neurais para aprendizado profundo (deep learning).

Já a camada de embedding, segundo TensorFlow (2023c), transforma inteiros positivos (índices) em vetores densos (embeddings). Ou seja, de início, cada palavra no vocabulário é representada por um vetor de números (os pesos) de forma aleatória, e durante o treinamento são gradualmente ajustados por retropropagação. Em suma, a medida que a camada de embedding é treinada, os pesos são ajustados para aprender representações semânticas das palavras. O que implica que palavras semanticamente semelhantes terão vetores mais próximos no espaço de incorporação.

La (2019) afirma que o processo de retropropagação descobre os pesos que serão aplicados aos nós de uma rede neural, por meio de comparação das saídas atuais da rede com as saídas desejadas, obtida pela função de perda, ou custo. De acordo com o autor, essa função, representada abaixo, informa o nível de precisão da rede neural, efetuando previsões para uma determinada entrada, minimizando a diferença entre o valor calculado e o correto. Esse algoritmo é representado pela soma dos quadrados das diferenças.

$$f(m, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Segundo La (2019), os pesos e desvios, geralmente iniciam com valores aleatórios e um valor alto de perda, e posteriormente, o algoritmo retorna e ajusta os pesos e desvios após efetuar o calculo de uma resposta, quanto menor a perda da rede, melhor sua precisão. Cada retomada de ciclo de propagação é denominado época.

Outra camada relevante em nosso código é a Flatten. De acordo com Viceri-Seidor (2020), essa camada prepara os dados para que entrem em uma camada densa, coletando a matriz resultante das camadas convolucionais e poolings anteriores e efetua o redimensionamento dessa matriz para uma forma linear, de uma única dimensão. Ou seja, ela possui o papel de "achatar" uma estrutura de dados multidimensional e converte-la em unidimensional.

É Interessante saber que camadas covolucionais (Conv1D em nosso caso) são responsáveis por extrair features da entrada, representações numéricas que descrevem as propriedades distintivas dos dados de entrada, extraindo características importantes e

discriminativas presentes nos dados que a rede neural (por meio de filtros covulacionais reduzidos que percorrem os dados de entrada) tenta aprender e usar para realizar tarefas específicas, conforme dito por Rodrigues (2018b). Segundo Rodrigues (2018b), após o uso dessa camada, é comum a utilização de camada de pooling, cuja função é reduzir o tamanho dos dados de entrada, o que diminui o uso de memoria de processamento e amplia o aprendizado da rede. De acordo com o autor, uma técnica bastante utilizada para executar o pooling é denominada de max-pooling.

Aqui, utilizamos o Max Pooling 1D, uma operação geralmente utilizada para processamentos de sequências, como tarefas de PNL, que reduz a dimensionalidade da representação da sequência e mantém suas características mais importantes. A operação aplica elementos em uma sequência (processo de percorrer cada elemento individualmente na sequência de entrada e aplicar uma operação específica a eles), onde a janela se move ao longo da dimensão temporal, e em cada janela, o valor máximo é mantido enquanto os demais são descartados, criando uma sequência reduzida. De acordo com Carneiro (2020), este processo extrai o maior valor do vetor de características, ou seja, a representação da palavra que teve maior destaque na frase.

6 UM ALGORITMO PARA O MODELO BAG OF WORDS

Agora que já entendemos as ferramentas utilizadas na construção do código, estamos aptos a entender cada uma de suas linhas. Assim, neste tópico iremos destrinchar o código.

6.1 Detalhes de um exemplo de código

Este código tem o intuito de trazer um exemplo de rede neural e entender quais os mecanismos matemáticos por trás de suas linhas. O código criado tem a função de verificar afirmações matemáticas, classificando-as como verdadeiras ou falsas, ou seja uma rede neural de uma única saída. A plataforma utilizada para o seu desenvolvimento foi o Google Colaboratory, por meio de linguagem python e em especial, bibliotecas como o Keras e Numpy. A rede neural foi treinada por meio de 20 afirmações e propriedades de figuras geométricas.

```
1  !pip install keras
2
3  from keras.preprocessing.text import Tokenizer
4  from keras.preprocessing.sequence import pad_sequences
5  from keras.models import Sequential
6  from keras.layers import Dense, Embedding, Flatten, Conv1D,
       MaxPooling1D
7
8  import numpy as np
9
10 # Define sample documents
11 $docs = ['Um triângulo tem 3 lados. ',
12           'Um quadrilatero tem 4 lados. ',
13           'Um triângulo pode ser escaleno, isósceles ou
                equilátero',
14           'Um triângulo escaleno tem 0 lados iguais',
15           'Um triângulo isósceles tem 2 lado iguais',
16           'um triângulo equilátero tem 3 lados iguais',
17           'Um quadrado é um quadrilátero',
18           'Um quadrado é um losango',
19           'Um quadrado é um ret\^{a}ngulo',
20           'Um retângulo tem 4 ângulos retos',
21           'Um triângulo é um quadrilátero',
22           'Um quadrilátero é um triângulo',
23           'Um quadrado é um triângulo',
24           'Um retâgulo é um triângulo',
```

```
25          'Um losango é um triângulo',
26          'Um quadrado tem 0 lados iguais',
27          'Um triângulo tem 2 ângulos retos',
28          'Um quadrado tem 0 ângulos retos',
29          'Um triângulo isósceles é escaleno',
30          'Um triângulo isósceles é equilátero']
31
32  # Assign labels
33  labels = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0]
34
35  labels = np.array(labels)
36
37  # Create tokenizer
38  tokenizer = Tokenizer()
39  tokenizer.fit_on_texts(docs)
40
41  # Convert docs to sequences
42  sequences = tokenizer.texts_to_sequences(docs)
43
44  # Pad sequences
45  padded = pad_sequences(sequences, padding='post', maxlen=5)
46
47  # Define model
48  vocab_size = len(tokenizer.word_index) + 1
49  embed_dim = 8
50  model = Sequential()
51  model.add(Embedding(vocab_size, embed_dim, input_length=5))
52  model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
53  model.add(MaxPooling1D(pool_size=2))
54  model.add(Flatten())
55  model.add(Dense(10, activation='relu'))
56  model.add(Dense(1, activation='sigmoid'))
57
58  # Compile and fit
59  model.compile(optimizer='adam', loss='binary_crossentropy',
        metrics=['accuracy'])
60  model.fit(padded, labels, epochs=500, verbose=0)
61  model.save_weights("saved_weights.h5")
62
63  model.save("trained_model_500.keras")
```

Listing 6.1 – Modelo completo do código saco de palavras de geometria

Para iniciarmos o processo de programação da rede neural, o script acima primeiro instala a biblioteca Keras por meio do gerenciador de pacotes Python, "Pip". Por meio de Awari (2023b), ressaltamos que a biblioteca Keras possui ferramentas de pré-processamento de dados em seu formato bruto, e conta com suporte de redimensionamento, normalização e transformação de dados.

A segunda linha trata do comando“from keras.preprocessing.text import Tokenizer”. O comando é responsável por importar a classe Tokenizer do módulo keras.preprocessing.text no ambiente de programação Python. Vale relembrar, que o tokenizer é o processo de divisão de um texto em unidades menores chamadas “tokens”, que em nosso caso serão palavras, e converte o texto em uma sequência de números, conforme explicado anteriormente.

Em seguida, temos a linha “from keras.preprocessing.sequence import pad_sequences”, que preenche as sequências com valores nulos no inicio ou fim, ou efetua a truncagem em sequências muito longas.

6.1.1 A entrada para o treinamento da rede neural

Prosseguindo com o nosso estudo pelo código, temos as linhas “keras.models import Sequential” e “from keras.layers import Dense, Embedding, Flatten, Conv1D, MaxPooling1D”. A primeira importa a classe Sequential do Keras, e a segunda importa as camadas específicas que serão utilizadas no modelo. De forma semelhante, “import numpy as np” importa a bibliotaca numpy para o código.

Em seguida, temos vinte linhas de texto, utilizadas para o treinamento da rede neural: "’Um triângulo tem 3 lados. ’, ’Um quadrilátero tem 4 lados. ’, ’Um triângulo pode ser escaleno, isósceles ou equilátero’, ’Um triângulo escaleno tem 0 lados iguais’, ’Um triângulo isósceles tem 2 lado iguais’, ’um triângulo equilátero tem 3 lados iguais’, ’Um quadrado é um quadrilátero’, ’Um quadrado é um losango’, ’Um quadrado é um retângulo’, ’Um retângulo tem 4 ângulos retos’, ’Um triângulo é um quadrilátero’, ’Um quadrilátero é um triângulo’, ’Um quadrado é um triângulo’, ’Um retângulo é um triângulo’, ’Um losango é um triângulo’, ’Um quadrado tem 0 lados iguais’, ’Um triângulo tem 2 ângulos retos’, ’Um quadrado tem 0 ângulos retos’, ’Um triângulo isósceles é escaleno’, ’Um triângulo isósceles é equilátero’]

Conforme mencionado anteriormente, aqui, cada frase passa a ser considerada um documento.

A seguir temos a linha "labels = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0]". Este comando, atribui um rotulo zero ou um para documentos que possuem mesmo contexto. Em nosso caso, como podemos notar no texto de treinamento, "1"para afirmações positivas e "0"para negativas.

Em seguida, na linha "labels = np.array(labels)"esta lista de rótulos python é convertida em um Array NumPy, oferecendo mais eficiencia computacional. É interessante ressaltar que o NumPy, oferece uma vasta gama de possibilidades eficientes de criação de matrizes e manipulação de dados numéricos. Segundo NumPy Developers (s.d), enquanto uma lista Python pode possuir tipos distintos de dados em uma lista, todos os elementos em uma matriz NumPy são homogêneos, o que deixa os array NumPy mais rápidos e compactos, além de reduzir o uso de memória.

6.1.2 Tokenização no código

A próxima linha, é responsável por criar um objeto Tokenizer. O processo de criação deste objeto se deu pelas linhas “tokenizer = Tokenizer()” e “tokenizer.fit_on_texts(docs)”. Conforme visto anteriormente, a primeira linha é utilizada para vetorizar o corpo do texto, convertendo cada token (palavra) em números. Já a segunda linha, fit_on_texts(docs) ajusta o Tokenizer aos textos fornecidos na lista docs. Assim, o Tokenizer examinará todos os textos contidos na lista e irá construir um vocabulário com base neles, atribuindo um índice único a cada palavra (ou token) no vocabulário. Dessa forma, o tokenizer aprende o mapeamento entre as palavras e números.

Ao executar o tokenizer em nosso texto de treinamento teremos um vocabulário do tipo:

> {’um’: 1, ’triângulo’: 2, ’é’: 3, ’tem’: 4, ’quadrado’: 5, ’lados’: 6, ’quadrilátero’: 7, ’isósceles’: 8, ’iguais’: 9, ’escaleno’: 10, ’equilátero’: 11, ’0’: 12, ’retângulo’: 13, ’ângulos’: 14, ’retos’: 15, ’3’: 16, ’4’: 17, ’2’: 18, ’losango’: 19, ’pode’: 20, ’ser’: 21, ’ou’: 22, ’lado’: 23}

Após esse processo, temos a aplicação da linha “sequences = tokenizer.texts_to_sequences(docs)” que utiliza o método “texts_to_sequences”, do objeto Tokenizer criado anteriormente, com a finalidade de converter os documentos de texto em sequências de números. Aqui, cada token é substituido pelo índice correspondente criado pelo Tokenizer anteriormente, obtendo uma sequência de números, como vimos em um tópico anterior. Assim, o primeiro documento, teremos algo como [1, 2, 4, 16, 6], representando o texto "Um triângulo tem 3 lados". Em suma, são vinte linhas de código, então temos 20 documentos. Note que cada frase (documento) se tornou um vetor, de tal forma que cada número é uma palavra especifica.

[1, 2, 4, 16, 6]	[1, 7, 4, 17, 6]	[1, 2, 20, 21, 10, 8, 22, 11]	[1, 2, 10, 4, 12, 6, 9]
[1, 2, 8, 4, 18, 23, 9]	[1, 2, 11, 4, 16, 6, 9]	[1, 5, 3, 1, 7]	[1, 5, 3, 1, 19]
[1, 5, 3, 1, 13]	[1, 13, 4, 17, 14, 15]	[1, 2, 3, 1, 7]	[1, 7, 3, 1, 2]
[1, 5, 3, 1, 2]	[1, 13, 3, 1, 2]	[1, 19, 3, 1, 2]	[1, 5, 4, 12, 6, 9]
[1, 2, 4, 18, 14, 15]	[1, 5, 4, 12, 14, 15]	[1, 2, 8, 3, 10]	[1, 2, 8, 3, 11]

Note que cada frase (documento) possui um número de palavras diferentes, e portanto, os vetores tem números de entradas distintos. No entanto, para o processo de aprendizagem, é necessário que todos os vetores possuam a mesma quantia de entradas. Para corrigir tal problema, utilizamos o comando "pad_sequences" por meio da linha "padded = pad_sequences(sequences, padding='post', maxlen=5)".

Conforme dito anteriormente, o pad_sequences preenche com as sequências com valores nulos no inicio ou fim, ou efetua a truncagem em sequências muito longas. Porém, nesse caso em específico, temos em nosso código "padding='post"', indicando que os valores nulos serão inseridos ao fim da sequência numérica, caso o número de entradas do vetor seja menor que o definido. Mais a frente, temos o termo "maxlen=5", que indica o número máximo de entradas (elementos da sequência) nos vetores. Dessa forma, as sequências são ajustadas para terem no máximo 5 elementos. Em suma, caso a sequência tenha menos que 5 elementos, será completada à direita por elementos nulos até atingir esse numero de termos, e caso o seu número de termos seja superior a 5, será truncada para esse tamanho. Após esse processo, obtemos algo do tipo:

[1, 2, 4, 16, 6]	[1, 7, 4, 17, 6]	[1, 2, 20, 21, 10]	[1, 2, 10, 4, 12]
[1, 2, 8, 4, 18]	[1, 2, 11, 4, 16]	[1, 5, 3, 1, 7]	[1, 5, 3, 1, 19]
[1, 5, 3, 1, 13]	[1, 13, 4, 17, 14]	[1, 2, 3, 1, 7]	[1, 7, 3, 1, 2]
[1, 5, 3, 1, 2]	[1, 13, 3, 1, 2]	[1, 19, 3, 1, 2]	[1, 5, 4, 12, 6]
[1, 2, 4, 18, 14]	[1, 5, 4, 12, 14]	[1, 2, 8, 3, 10]	[1, 2, 8, 3, 11]

Observe que após a aplicação do comando, todos os vetores passaram a ter a mesma quantia de entradas (totalizando cinco cada) favorecendo a aprendizagem da rede neural. É importante ressaltar que o comando pode ser vantajoso se as sequências forem suficientemente curtas e se o limite não eliminar informações essenciais. No entanto, se o comprimento limite cortar dados importantes ou alterar drasticamente o contexto, a aprendizagem será prejudicada. Dessa forma, O ideal é sempre ajustar o parâmetro maxlen de acordo com a distribuição de comprimento das sequências e a natureza da tarefa. Note que essas sequências, ou vetores, podem ser representados por meio de uma única matriz com 20 linhas, de forma que cada linha represente um documento.

A linha "vocab_size = len(tokenizer.word_index) + 1" determina o tamanho total do vocabulário, incluindo um token adicional (token de preenchimento). Ao passo que, a

linha “embed_dim = 8”, estabelece a dimensão dos vetores embedding a serem aprendidos pelo modelo durante o treinamento, neste caso, o embed_dim definido é de 8 dimensões, assim cada palavra ou documento será representado por um vetor de 8 dimensões após o treinamento do modelo. É interessante ressaltar, que a camada embeding transforma as sequências de números inteiros (índices das palavras ou documentos) em vetores densos que capturam propriedades semânticas das palavra ou documentos, podendo verificar quais possuem características semelhantes. Neste caso, cada documento passa a ser representado por um vetor com 8 elementos, ou dimensão 8.

6.1.3 As camadas no código

A próxima linha, “model = Sequential()” define que as camadas terão ordem sequencial, cada camada é adicionada uma após a outra. A primeira delas é a Embedding, adicionada por meio da linha “model.add(Embedding(vocab_size, embed_dim, input_length=5))”. Aqui essa camada é adicionada e transformará os índices de palavras em vetores de embedding densos, com base no tamanho do vocabulário, na dimensão dos vetores de embedding e no comprimento máximo das sequências de entrada, neste caso o comprimento das sequências será 5.

A saída dessa camada será conectada à próxima, a camada Conv1D, adicionada pela linha “model.add(Conv1D (filters=32, kernel_size=3, activation=’relu’))”. Essa camada de covolução unidimensional, que aprenderá 32 filtros de uma janela covolucional de tamanho 3 e aplicará a função de ativação ReLU aos resultados da convolução, para extrair características locais das sequências de palavras.

O resultado da convolução é chamado de mapa de características. Ele contém informações sobre quais características (ou padrões) foram detectadas nas janelas de palavras pela camada de convolução. Em resumo, a convolução extrai características da camada e diminui a amostragem da entrada. Redes neurais convolucionais apresentam um estagio de pooling, conforme dito mais a frente.

É interessante destacar que a função de ativação ReLU produz resultados entre os valores zero e infinito. De acordo com Santos (2020), Essa função retorna 0 para todo valor negativo e retorna o próprio valor para valores positivos. Segundo o autor, devido a tal condição, para valores negativos pode ocorrer uma especie de "morte"de neurônios, onde não ocorra aprendizagem. Porém, Santos (2020) destaca que mesmo com as limitações é uma das funções mais utilizadas no treinamento de RNAs, sendo representada pela função ReLU (x) = max (0,x).

Em seguida, os dados avançam para a próxima camada, MaxPooling1D. Essa camada é adicionada pelo comando “model.add(MaxPooling1D(pool_size=2))”. A operação de pooling, tem o intuito de reduzir a dimensionalidade dos mapas de características gerados pelas camadas convolucionais, melhorando a eficiência e reduzindo o número de parâmetros.

Em suma, com base em Keras (Sem data), executa a operação de máxima de pooling em cada segmento do mapa de características definidas pelo parâmetro pool_size, que em nosso caso é 2. O que significa que a operação será aplicada em janelas de tamanho 2, reduzindo o comprimento do mapa de características.

A próxima camada adicionada em nosso modelo é a Flatten, por meio da linha "model.add(Flatten())" . Como já vimos anteriormente, essa camada, prepara os dados para a camada densa, com o papel de "achatar"a estrutura de dados multidimensional e convertê-las para uma única dimensão Viceri-Seidor (2020).

Após essa preparação nos dados, representados por um vetor unidimensional, fazemos a adição de uma camada densa, por meio do comando "model.add(Dense(10, activation='relu'))". O número 10 representa a quantidade de neurônios utilizada na camada, e "ReLU"a função de ativação utilizada.

Segundo Géron (2019), cada camada Densa faz o gerenciamento de sua própria matriz de pesos, que contém todos os pesos de conexão entre os neurônios e suas entradas, além de gerenciar um vetor de polarização (um para cada neurônio). De acordo com o autor, ao receber alguns dados de entrada, a camada realiza os cálculos por meio da expressão $h_{W,b}\ X = \varphi(XW + b)$, de forma que "$X$ representa a matriz de recursos de entrada (uma linha para cada instância e uma coluna para cada recurso), W é a matriz dos pesos de conexão (uma linha para cada entrada e uma coluna para cada neurônio), b o vetor de polarização, φ a função de ativação. O autor ressalta, que na ciência de dados, existe a "difusão", onde adicionar um vetor a uma matriz implica em adicioná-lo a cada linha da matriz. Ou seja, inicialmente, efetuaríamos o produto de X com W e depois, na matriz resultante, adicionaríamos o vetor b a cada linha dessa matriz.

A seguir, adicionamos uma segunda camada oculta densa, dessa vez com 1 neurônio e função de ativação sigmoid, como pode ver pela linha "model.add(Dense(1, activation='sigmoid'))" . É importante ressaltar que como essa é a última camada e nosso código foi elaborado para determinar se uma sentença é verdadeira ou falsa, obtemos como resultado uma saída binária, por isso, essa camada possui apenas um neurônio. Segundo Facure (2017), a função de ativação escolhida, também deve satisfazer essa condição, portanto a função escolhida foi a Sigmoid, uma vez que assume apenas valores entre 0 (não ativação) e 1 (ativação). De acordo com Reis (2016) a fórmula da função de ativação Sigmoid (utilizada neste código) é dada por:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

.

Quanto à linha "model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])" , ela configura o modelo para treinamento usando o otimizador 'Adam' (Adaptive Moment Estimation). De acordo com Vishwakarma (2024), o otimizador

Adam é um algoritmo de otimização iterativo usado para minimizar a função de perda durante o treinamento de redes neurais. Segundo o autor, ele auxilia no ajuste das configurações da rede (chamadas de parâmetros), o que facilita a compreensão de textos.

Já o comando loss='binary_crossentropy', segundo KERAS (s.d.) é responsável por calcular a perda de entropia cruzada entre rótulos verdadeiros e rótulos previstos. De acordo com Saxena (2023), 'Binary_crossentropy', também conhecida como entropia cruzada, é uma função de perda utilizada em problemas de classificação binária, utilizada no aprendizado de máquinas e aprendizado profundo, com o intuito de medir a diferença entre resultados binários previstos e rótulos binários reais. De acordo com o autor, a função quantifica a dissimilaridade entre as distribuições de probabilidade. Segundo Saxena (2023), a função compara cada probabilidade prevista com a saída real da classe, podendo assumir valores "0"ou "1", e em seguida efetua o calculo da pontuação que efetua a penalização com base no valor esperado. De acordo com Godoy (2018), uma boa função de custo retorna valores altos para previsões ruins e valores baixos para previsões boas. O autor afirma que essa função pode ser descrita como:

$$H_p(q) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i)) \tag{6.1}$$

De forma que y é o rótulo de uma classe real ou um valor real associado a uma determinada amostra, p(y) é a previsão da probabilidade e "n", o número total de amostras ou elementos.

Por fim, temos o "metrics=['accuracy']" , que de acordo com Keras (s.d.a), calcula com que frequência as previsões equivalem aos rótulos. Segundo o autor, essa métrica cria duas variáveis locais, total e contagem, que são utilizadas para calcular a frequência com que y_pred corresponde a y_true. Essa frequência retorna como precisão binária: uma operação idempotente que efetua a divisão do total pela contagem.

6.1.4 A obtenção do arquivo do modelo treinado

Nossa proxima linha, é representada por “model.fit(padded, labels, epochs=500, verbose=0)”. De acordo com Keras (s.d.b), o método fit é utilizado usado para treinar o modelo com os dados fornecidos, por meio de um número fixo de épocas. Durante o treinamento, o modelo ajusta seus parâmetros internos para fazer previsões mais precisas. Aqui, foram anexados o Padded, Labels, epochs e verbose. O “Padded” se trata de conjunto de dados de entrada preenchidos para garantir sequências de mesmo comprimento. O “labels”, são rótulos correspondentes aos dados de entrada padded. O “epochs=500” um parâmetro que define o número de épocas, ou seja, quantas vezes o algoritmo de treinamento passará por todo o conjunto de dados de treinamento, neste caso 500 vezes. Por fim, temos o “Verbose=0” que segundo Saturn Cloud (2023), controla a quantidade de informações (registros) que serão impressas durante o treinamento, neste caso, 0 indica que nenhuma saída será impressa durante o treinamento.

A seguir temos a linha, “model.save_weights("saved_weights.h5")”. TensorFlow (2024) afirma que o comando é responsável por salvar os pesos do modelo em um formato especifico, “Saved_weights.h5”. É importante ressaltar que os pesos representam os parâmetros internos que foram ajustados durante o treinamento para previsões mais precisas, e ao salvar apenas os pesos permite carrega-los em um modelo com mesma arquitetura.

Já a última linha de nosso modelo, “model.save("trained_model_500.keras")” é utilizada para salvar todo o modelo treinado, o que inclui sua arquitetura, pesos e configurações, para que assim, possamos carregá-lo posteriormente. Aqui nomeamos o modelo como “trained_model_500.keras”. É interessante notar, que em uma única aba do Google Colab, podemos ter diversos modelos, assim, a nomenclatura auxilia na hora de carregar o modelo correto posteriormente.

7 TESTANDO O CÓDIGO ANTERIOR

Agora que já explicamos o funcionamento do modelo linha por linha, podemos efetuar o teste do código. Mas antes, vamos tentar entender como o processo se dá linha por linha, afinal, para testar o modelo são necessários novos códigos. Veja o código por completo:

```
1  from keras.models import load_model
2
3  loaded_model = load_model("trained_model_500.keras")
4
5  # Preprocess the input text
6  input_text = "Um quadrilátero tem 4 lados."
7  sequences = tokenizer.texts_to_sequences([input_text])
8  padded_sequences = pad_sequences(sequences, maxlen=5)
9
10 # Make a prediction
11 prediction = loaded_model.predict(padded_sequences)
12
13 # Interpret the prediction
14 threshold = 0.5
15 if prediction >= threshold:
16     print('"', input_text, '",' é uma afirmação verdadeira: ',
          prediction)
17 else:
18     print('"', input_text, '",' é uma afirmação falsa: ',
          prediction)
```

Listing 7.1 – Teste do Código

7.1 Detalhes do código de utilização e teste do modelo

As linhas 1 e 2, importam e carregam o modelo já treinado. Note que à frente do termo “load_model” temos “trained_model_500.keras”, o modelo que salvamos anteriormente, que queremos importar.

Na sexta linha, fazemos o input de um texto que será processado por nossa IA. Nesse caso, queremos fornecer como entrada, a afirmação "Um quadrilátero tem 4 lados".

Note que na linha 7, temos “sequences = tokenizer.texts_to_sequences([input_text])”. Ou seja, estamos fazendo o mesmo que fizemos com o texto de treinamento na linha 42 do código principal, fazendo a vetorização do texto, ou seja, convertendo o documento

de entrada em uma sequência de números inteiros, e posteriormente na linha 8, garantimos que a sequência de números obtida pelo documento seja do mesmo tamanho das sequências gerados nos documentos presentes na fase de treinamento, ou seja, tamanho 5.

Na linha 11, carregamos o modelo “predict”, utilizado para detectar tendências e fazer previsões com base nos dados obtidos por meio do padded_sequences.

7.1.1 Testando o modelo obtido

A partir da linha 14, iniciamos o processo de interpretação dos dados da previsão. A variável threshold define um limiar para a interpretação das previsões. Neste caso, como temos uma IA de classificação binária (valor de saída é continuo e estará entre 0 e 1), utilizamos threshold= 0.5. Com esse limiar, estabelecemos que, para valores maiores ou iguais à 0,5, temos que a premissa é verdadeira, e abaixo deste valor, a expressão se torna falsa. Assim, quanto maior o valor obtido, mais certeza se tem da veracidade da resposta, e quanto menor o valor, mais confiança se tem de que a afirmação é falsa, conforme explicitado pelas linhas de 15 à 18:

```
15 if saida >= limite:
16     print('"', input_text, '"', " é uma afirmação verdadeira: ",
           saida)
17 else:
18     print('"', input_text, '"', " é uma afirmação falsa: ",
           saida)
```

Listing 7.2 – Seu código Python

Note que ativamos a função "print"para imprimir os dados obtidos na saída do código, assim ao executar o código obtemos:

```
1/1 [==============================] - 0s 76ms/step

" Um quadrilátero tem 4 lados. "  é uma afirmação verdadeira:  [[0.9998088]]
```

Acima temos o tempo de processamento (com 76 milissegundos) e a seguir a frase testada e a afirmação obtida na saída da IA, "É Uma afirmação verdadeira". Isso ocorre, porque ela obteve um limiar de 0,9998088 de certeza, ou seja quase 1, o que implica em quase 100 % de certeza.

É interessante ressaltar que nas condições descritas acima, obtivemos mais certeza efetuando um treinamento de 500 épocas do que com 50 épocas, porém, não resultou sempre em respostas corretas. A IA, qualificou por exemplo, as afirmações "Um quadrilátero

tem 5 lados "(0.9754121) e "Um quadrilátero tem 0 lados "(0.89693815) como verdadeiras. Sugerindo que a inteligência artificial, não está conseguindo quantificar e distinguir valores maiores de menores, errando todas as questões numéricas, o que sugere que o código e a aprendizagem ainda podem ser aprimorados.

8 A MATEMÁTICA DOS CHATBOOTS E A BNCC

No inicio do tópico acerca dos conceitos matemáticos, definimos o que veria a ser a Base Nacional Curricular (BNCC), assim como ao longo dessa pesquisa, foram discutidos e apresentados vários conceitos e relações matemáticas. Neste tópico, vamos buscar comparar o que foi visto com as habilidades da BNCC. Porém, é necessário que possamos compreender alguns termos utilizados neste documento, como a definição do que viria a ser habilidades e competências.

De acordo com Educação (2018), o termo "competência" pode ser definido "como a mobilização de conhecimentos (conceitos e procedimentos), habilidades (práticas, cognitivas e socioemocionais), atitudes e valores para resolver demandas complexas da vida cotidiana, do pleno exercício da cidadania e do mundo do trabalho." Assim, de acordo com o documento, as competências buscam a formação de valores e estímulos, com perspectiva de uma transformação e humanização da sociedade. Além disso, a BNCC é articulada por meio de dez competências gerais, ao qual visam o desenvolvimento escolar desde a creche até os anos finais da educação básica.

Essas competências são trazidas por (Educação, 2018):

> 1. Valorizar e utilizar os conhecimentos historicamente construídos sobre o mundo físico, social, cultural e digital para entender e explicar a realidade, continuar aprendendo e colaborar para a construção de uma sociedade justa, democrática e inclusiva.
>
> 2. Exercitar a curiosidade intelectual e recorrer à abordagem própria das ciências, incluindo a investigação, a reflexão, a análise crítica, a imaginação e a criatividade, para investigar causas, elaborar e testar hipóteses, formular e resolver problemas e criar soluções (inclusive tecnológicas) com base nos conhecimentos das diferentes áreas.
>
> 3. Valorizar e fruir as diversas manifestações artísticas e culturais, das locais às mundiais, e também participar de práticas diversificadas da produção artístico-cultural.
>
> 4. Utilizar diferentes linguagens – verbal (oral ou visual-motora, como Libras, e escrita), corporal, visual, sonora e digital –, bem como conhecimentos das linguagens artística, matemática e científica, para se expressar e partilhar informações, experiências, ideias e sentimentos em diferentes contextos e produzir sentidos que levem ao entendimento mútuo.
>
> 5. Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver

> problemas e exercer protagonismo e autoria na vida pessoal e coletiva.
>
> 6. Valorizar a diversidade de saberes e vivências culturais e apropriar-se de conhecimentos e experiências que lhe possibilitem entender as relações próprias do mundo do trabalho e fazer escolhas alinhadas ao exercício da cidadania e ao seu projeto de vida, com liberdade, autonomia, consciência crítica e responsabilidade.
>
> 7. Argumentar com base em fatos, dados e informações confiáveis, para formular, negociar e defender ideias, pontos de vista e decisões comuns que respeitem e promovam os direitos humanos, a consciência socioambiental e o consumo responsável em âmbito local, regional e global, com posicionamento ético em relação ao cuidado de si mesmo, dos outros e do planeta.
>
> 8. Conhecer-se, apreciar-se e cuidar de sua saúde física e emocional, compreendendo-se na diversidade humana e reconhecendo suas emoções e as dos outros, com autocrítica e capacidade para lidar com elas.
>
> 9. Exercitar a empatia, o diálogo, a resolução de conflitos e a cooperação, fazendo-se respeitar e promovendo o respeito ao outro e aos direitos humanos, com acolhimento e valorização da diversidade de indivíduos e de grupos sociais, seus saberes, identidades, culturas e potencialidades, sem preconceitos de qualquer natureza.
>
> 10. Agir pessoal e coletivamente com autonomia, responsabilidade, flexibilidade, resiliência e determinação, tomando decisões com base em princípios éticos, democráticos, inclusivos, sustentáveis e solidários. (Educação, 2018)

É importante ressaltar, que além das competências gerais que guiam o currículo, existem ainda as competências especificas de cada área do conhecimento. Na área de matemática e suas tecnologias para o Ensino Médio, temos as seguintes:

> 1. Utilizar estratégias, conceitos e procedimentos matemáticos para interpretar situações em diversos contextos, sejam atividades cotidianas, sejam fatos das Ciências da Natureza e Humanas, ou ainda questões econômicas ou tecnológicas, divulgados por diferentes meios, de modo a consolidar uma formação científica geral.
>
> 2. Articular conhecimentos matemáticos ao propor e/ou participar de ações para investigar desafios do mundo contemporâneo e tomar decisões éticas e socialmente responsáveis, com base na análise de problemas de urgência social, como os voltados a situações de saúde, sustentabilidade, das implicações da tecnologia no mundo do trabalho, entre outros, recorrendo a conceitos, procedimentos e linguagens próprios da Matemática.
>
> 3. Utilizar estratégias, conceitos e procedimentos matemáticos, em seus campos – Aritmética, Álgebra, Grandezas e Medidas, Geometria, Probabilidade e Estatística –, para interpretar, construir modelos e resolver problemas em diversos contextos, analisando a

> plausibilidade dos resultados e a adequação das soluções propostas, de modo a construir argumentação consistente.
>
> 4. Compreender e utilizar, com flexibilidade e fluidez, diferentes registros de representação matemáticos (algébrico, geométrico, estatístico, computacional etc.), na busca de solução e comunicação de resultados de problemas, de modo a favorecer a construção e o desenvolvimento do raciocínio matemático.
>
> 5. Investigar e estabelecer conjecturas a respeito de diferentes conceitos e propriedades matemáticas, empregando recursos e estratégias como observação de padrões, experimentações e tecnologias digitais, identificando a necessidade, ou não, de uma demonstração cada vez mais formal na validação das referidas conjecturas.
>
> (Educação, 2018)

É interessante ressaltar, que para cada competência existem um conjunto de habilidades que necessitam ser desenvolvidas, e cabem as escolas trabalhar cada uma de acordo com suas especificidades. Segundo Educação (2018)bncc2018, as habilidades expressam as aprendizagens essenciais, e se relacionam a diferentes objetos do conhecimento (entendidos como conteúdos, conceitos e processos) organizados em unidades temáticas. Cada habilidade é identificada por um código alfanumérico, composta como na figura 5.

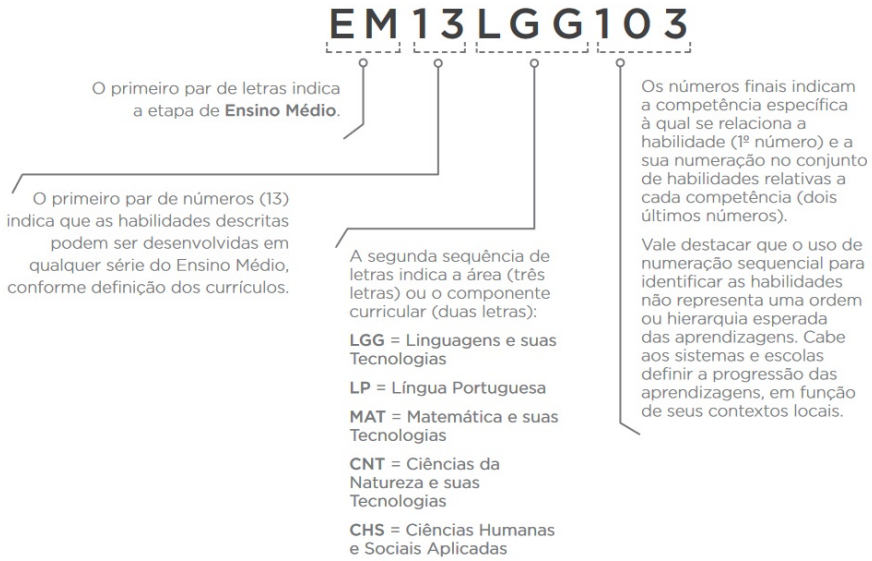

Figura 9 – Composição do código alfanumérico das habilidades de ensino

Fonte: (Educação, 2018)

Assim, faremos uso desta forma, na representação das habilidades a seguir.

8.1 Tópicos e habilidades compatíveis com o currículo do Ensino Médio

Conforme dito anteriormente, as competências são subdivididas em habilidades, porém teremos enfoque apenas naquelas que se envolvem diretamente com os tópicos matemáticas deste trabalho. Baseado nisso, faremos um filtro, ao qual descreveremos as habilidades possivelmente envolvidas e a relacionaremos com um tópico ou conteúdo abordado durante o levantamento matemático no funcionamento das IAs. É importante ressaltar que os textos de identificação das habilidades representadas no quadro abaixo foram retirados na integra da BNCC.

Competências	Habilidades
Competência 3	(EM13MAT302) Resolver e elaborar problemas cujos modelos são as funções polinomiais de 1º e 2º graus, em contextos diversos, incluindo ou não tecnologias digitais.
	(EM13MAT304) Resolver e elaborar problemas com funções exponenciais nos quais é necessário compreender e interpretar a variação das grandezas envolvidas, em contextos como o da Matemática Financeira e o do crescimento de seres vivos microscópicos, entre outros.
	(EM13MAT305) Resolver e elaborar problemas com funções logarítmicas nos quais é necessário compreender e interpretar a variação das grandezas envolvidas, em contextos como os de abalos sísmicos, pH, radioatividade, Matemática Financeira, entre outros.
	(EM13MAT316) Resolver e elaborar problemas, em diferentes contextos, que envolvem cálculo e interpretação das medidas de tendência central (média, moda, mediana) e das de dispersão (amplitude, variância e desvio padrão).
Competência 4	(EM13MAT403) Comparar e analisar as representações, em plano cartesiano, das funções exponencial e logarítmica para identificar as características fundamentais (domínio, imagem, crescimento) de cada uma, com ou sem apoio de tecnologias digitais, estabelecendo relações entre elas.
	(EM13MAT405) Reconhecer funções definidas por uma ou mais sentenças (como a tabela do Imposto de Renda, contas de luz, água, gás etc.), em suas representações algébrica e gráfica, convertendo essas representações de uma para outra e identificando domínios de validade, imagem, crescimento e decrescimento.
	(EM13MAT406) Utilizar os conceitos básicos de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

Tabela 1 – Competências e habilidades

Fonte: Elaborado pelo autor com informações de (Educação, 2018)

Dentre as quatro competências da BNCC para a área de Matemática e suas tecnologias, ao fazer uma análise, notamos que as competências 3 e 4 são as que mais se relacionam ao que foi apresentado neste trabalho.

Inicialmente, pela competência 3, temos a habilidade (EM13MAT302) que está diretamente relacionada ao estudo de funções de 1º e 2º graus. É interessante observar, que trabalhamos com uma função muito semelhante a estrutura de uma função de 1º grau, $h_{W,b}$ $X = \varphi(XW + b)$, porém tal função está diretamente relacionada com matrizes e vetores. Mas, é interessante ressaltar que funções de primeiro e segundo graus podem ser utilizadas como funções de ativação em uma Inteligência artificial simplificada, com o intuito de exemplificar o funcionamento das camadas. Outro exemplo abordado nessa pesquisa, o polinômio apresentado na figura 3, $p_1x_1 + p_2x_2 + p_3x_3 + b_1$ poderia ser escrita na forma de uma função linear, porém de múltiplas variáveis. Aqui, poderíamos utilizar o contexto de máximo e mínimo visto pelos estudantes em funções de 2° grau, para compreenderem um pouco do treinamento de RNAs, representada pela função ReLU (x) = max (0,x), e pelo exemplo contido na figura 3, como $max(p_1x_1 + p_2x_2 + p_3x_3 + b_1, 0)$.

Quanto a habilidade (EM13MAT316), trabalha o calculo e interpretação de medidas de tendência central e dispersões. Em nossa analise, abordamos uma função que mede o nível de precisão da rede neural e efetua previsões:

$$f(m, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Assim, após a apresentação das medidas de tendência e dispersões, tal função poderia ser apresentada e testada com valores fictícios para que os estudantes entendam a precisão de diferentes redes neurais.

Já a habilidade (EM13MAT304), faz referência a resolução e elaboração de problemas com funções exponenciais e suas aplicabilidades. Nessa pesquisa apresentamos a função de ativação Sigmoid, dada por:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

. Claramente, tal função possui aplicações exponenciais, assim poderia ser trabalhada no ensino médio por meio desta habilidade, além de envolver problemas que podem auxiliar os alunos a compreender intervalos, gráficos e valor numérico de funções, visto que essa função admite apenas valores entre 0 e 1. Neste contexto poderiam ser abordados o funcionamento de camadas densas, que utilizam a função.

A habilidade (EM13MAT305), trabalha resolução e elaboração de problemas envolvendo funções logarítmicas, além de compreender algumas de suas aplicabilidades. Neste

caso, temos a função de custo,

$$H_p(q) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i)) \tag{8.1}$$

Porém, é importante que os estudantes já conheçam as propriedades de logarítmicas. Aqui, os estudantes podem desenvolver a função com valores fictícios, entender que é uma função relacionada ao cálculo de probabilidade e compreender um pouco mais sobre o aprendizado de máquina.

já na competência 4, temos a habilidade (EM13MAT403), que traz a comparação e analises de funções exponencia e logarítmica e suas características. Aqui os estudantes, por meio de softwares, poderiam desenvolver a parte gráfica das funções descrita nas habilidades (EM13MAT304) e (EM13MAT305), e compara-las, extraindo domínio, imagem e efetuando a analise de crescimento/decrescimento de cada uma.

Para a habilidade (EM13MAT405), na competência 4, temos

a Função de limiar:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \tag{8.2}$$

Função de limiar por partes:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0,5 \\ \frac{v}{2} & \text{se } \ v + 0,5 > v > -0,5 \\ 0 & \text{se } v \leq \ -0,5 \end{cases} \tag{8.3}$$

Note que ambas as função são definidas por varias sentenças, e portanto satisfazem a habilidade descrita, cabendo apenas, aplica-las as camadas de um modelo simples, e pedir para que os alunos façam aplicações com números, além de analisarem os gráficos, domínio, imagem, crescimento e decrescimento das funções.

Por fim, temos a habilidade (EM13MAT406), ao qual faz referência a linguagem de programação na implementação de algoritmos. Aqui, os estudantes poderiam conhecer um pouco sobre a linguagem Python, criar pequenos algoritmos, e em seguida, entender um pouco mais sobre como ocorre a programação de redes neurais, analisando as implicações de cada linha, além de efetuar testes e simulações.

9 CONCLUSÃO

Nesta seção vamos discorrer sobre algumas das conclusões que tivemos no decorrer do trabalho, em especial ao final dos trabalhos de pesquisa.

9.1 Sobre o desenvolvimento do trabalho

Durante esta pesquisa foi realizado um pré levantamento de quais tópicos de matemática seriam necessários para a compreensão do funcionamento de uma IA do tipo chatbot. Assim, de inicio, percebemos que a matemática que permeia essa área é bastante abrangente, então tivemos um enfoque maior na álgebra linear e em tópicos como funções.

No desenvolvimento, é perceptível o quanto o campo das IAs evoluiu ao longo do tempo, tal como suas aplicabilidades. É notável, que a matemática dos conceitos utilizados foi se tornando mais complexa e completa, com o intuito de aprimorar a eficiência das redes neurais. Assim, para uma melhor compreensão, optamos por construir uma rede neural em linguagem Python, cujo intuito era classificar afirmações matemáticas em verdadeiras ou falsas e a partir daí descrever os recursos computacionais necessários, e analisar seu funcionamento a cada linha.

A rede neural que vimos em mais detalhes possui seis camadas (Embedding, onvolucional 1D,Max Pooling 1D,Flatten, Densa - com ativação ReLU, Densa - com ativação Sigmoid) e uma única saída (verdadeira ou falsa). O modelo relativo a esta rede foi implementado e testado no Google Colab, o nível de acerto do modelo foi razoável, dentro do esperado para um trabalho inicial que não tem como objetivo a otimização do modelo. É importante observar que o modelo aqui estudado não foi otimizado para que tenha a maior quantidade possível de respostas corretas. Este trabalho se concentra em estudar os princípios matemáticos das redes neurais, em particular dos chatbots, e verificar quais dos conceitos matemáticos envolvidos tem relação com o ensino médio, segundo a BNCC.

Como essa pesquisa possuía o intuito de procurar por tópicos de matemática acessíveis aos estudantes de nível médio, fizemos um filtro, ao evitar aprofundar em tópicos muito complexos, porém, mostrando a sua finalidade. Ao fim, ao comparar os tópicos com a BNCC, tornou-se perceptível, que apesar da matemática dos Chatbots ser um tanto complexa, existem tópicos e formas de se aplicar vários dos seus conceitos no Ensino Médio. Em nossa pesquisa, demonstramos esta possibilidade por meio de tópicos de funções de 1° e segundo grau, funções logarítmicas e exponenciais, além de linguagem de programação e medidas de tendência central e dispersões.

Porém, para que seja possível o desenvolvimento destes conceitos no ensino básico,

é crucial que os estudantes já tenham certo domínio em alguns conceitos e propriedades dos tópicos listados, além de que a IA a ser trabalhada deve possui uma função de ativação simplificada, dentro da matriz curricular dos estudantes, como funções polinomiais simples.

9.2 Sobre aspectos da BNCC encontrados na pesquisa

Após este estudo, foi possível identificar competências e habilidades da BNCC diretamente envolvidas com conceitos matemáticos relevantes para o funcionamento das IAs, como vimos no capítulo **??**, resumido na lista a seguir:

Competência 3: Habilidades EM13MAT302 e EM13MAT316

Competência 4: Habilidades EM13MAT304, EM13MAT305, EM13MAT403, EM13MAT405 e EM13MAT406.

9.3 Sobre o que não coube no escopo da pesquisa

Durante a realização dessa pesquisa, tínhamos o objetivo de entender a matemática por trás do funcionamento de inteligências artificiais do tipo Chatbot. Porém, durante o levantamento bibliográfico foi possível perceber uma vasta quantidade de possibilidades de se formar uma rede a depender de qual seria a sua utilidade e quantidade de neurônios. Assim, tornou-se perceptível, que a matemática que rege essa área é muito densa, e pode atingir grande complexidade, o que impediu o aprofundamento nesta pesquisa. Procuramos então selecionar algumas das principais aplicabilidades matemáticas que mais se aproximassem do nível básico, com o intuito de simplificar ao máximo a compreensão dessa pesquisa ao leitor. Além disso, a parte educacional, poderia ser bastante ampla, em virtude da quantidade de aplicações matemáticas relacionadas a redes neurais artificiais, o que resultarias em muitas possibilidades para o ensino.

De início, havia uma expectativa de que tópicos como autovalores, autovetores e diagonalização de matrizes surgissem logo cedo de maneira explícita, uma vez que são de grande relevância para sistemas de informação e algoritmos em geral, porém não foi possível, neste trabalho, encontrar pontos de uso direto destes tópicos.

Além disso, a parte educacional poderia facilmente render vários outros trabalhos, haja visto que além de promover uma grande quantidade de aplicações matemáticas, existem outras questões que poderiam facilmente ser trabalhadas acerca da utilização de IA's, como ética, importância social , avaliações na era da IA, etc.

É importante ressaltar que este trabalho caminhou desde o inicio na direção de aproximar a matemática contida na sala de aula à redes neurais, além de procurar por pontos onde a matemática pode se apoiar para contextualização de situações de ensino. Evidentemente, podemos perceber que assim como ocorreu a popularização da internet nos

últimos 30 anos, a universalização da inteligencia artificial deve acarretar em mudanças em diversos aspectos da sociedade, em particular, no ensino, e de forma automática no currículo de matemática, que provavelmente irá se adaptar as novas condições, o que demonstra a importância cada vez maior neste tipo de estudo.

9.4 Sobre os próximos trabalhos

É interessante notar que, apesar de temos restringido a pesquisa, focando apenas em chatbots de saída binaria e na parte matemática mais relevante ao ensino médio, ainda é um tema bastante abrangente com varias possibilidades de abordagem. Assim, é interessante explorar essas possibilidades em trabalhos futuros, como aprofundar em tópicos mais complexos ou aprimorar o código.

Quanto ao código, nunca foi o foco dessa pesquisa, conforme dito anteriormente, o mesmo foi criado com o intuito de realizarmos algumas simulações e entender o processo de funcionamento e a matemática contida em cada passo, assim podemos entender o que é realizado em cada linha do código.

Todavia, o código possui falhas, haja visto que não consegue classificar as afirmações de forma eficiente. Assim, uma possibilidade para uma próxima pesquisa, seria o aprimoramento desse código, melhorando sua acurácia e desempenho. Ao realizar varias simulações e entender o comportamento de uma rede neural, entendemos algumas possibilidades para refinar o código. Uma possibilidade seria aumentar a dimensionalidade dos vetores embedding (em nosso trabalho dim = 8), que passariam a analisar uma quantidade maior de informações deixando o treinamento mais eficaz, embora resulte em um maior custo computacional. Porém é importante salientar que deve se analisar a dimensionalidade que melhor se adequaria, para não deturbar a aprendizagem e "decolar afirmações". Outra possibilidade, seria aumentar a quantidade de afirmações acerca de geometria, visto que quanto maior a quantidade de dados para treinamento, melhor será seu aprendizado e mais eficaz seu processo de classificação.

Além disso, em trabalhos futuros, seria interessante a aplicação da sequência didática, que embora tenha sido produzida baseada no que foi desenvolvido nesta pesquisa, não tivemos tempo hábil para aplica-la. Ressaltamos, que essa sequência didática da apendice A foi produzida como um produto educacional, que pode ser obtido após a constatação de que o tema da pesquisa pode ser desenvolvido dentro da matriz curricular de matemática, analisando a compatibilidade com tópicos do ensino médio. É interessante que a sequência produzida abrange inclusive outras áreas do conhecimento, pois o problema focal é um festival de pipas que considera velocidade do vento, probabilidade de chuva e temperatura, o que poderia levar a multidisciplinaridade com geografia e física por exemplo.

Essa sequência didática oferece as ferramentas para os estudantes verifiquem a viabilidade do festival de pipas por meio de uma rede neural simples com duas camadas,e saída binaria. Além disso, os estudantes podem testar o código tanto de forma manual quanto por meio do Google Colaboratory, tendo acesso mais próximo a linguagem de programação (também previsto pela BNCC), neste caso, Python. Dessa forma, em trabalhos futuros, podem ser abordados uma vasta gama de possibilidade de aplicações no enino médio.

REFERÊNCIAS

ALI, M. **Introdução às funções de ativação em redes neurais: Aprenda a navegar pelo cenário das funções de ativação comuns - desde a firme ReLU até a proeza probabilística da softmax**. 2024. DataCamp. Acesso em: 29 de setembro de 2024. Disponível em: https://www.datacamp.com/pt/tutorial/introduction-to-activation-functions-in-neural-networks.

ALVES, P. M. **Inteligência Artificial e Redes Neurais**. 2020. IPEA - Centro de Pesquisa em Ciência, Tecnologia e Sociedade. Acesso em 21 de abril de 2024. Disponível em: https://www.ipea.gov.br/cts/pt/central-de-conteudo/artigos/artigos/106-inteligencia-artificial-e-redes-neurais.

Amazon Web Services. **O que é processamento de linguagem natural (PLN)**. 2023. AWS. Acesso em 01 de maio de 2024. Disponível em: https://aws.amazon.com/pt/what-is/nlp/.

ARAGÃO, J. F. B. Dissertação (Mestrado em Informática), **Sistemas Especialistas como ferramenta auxiliar para o ensino da disciplina Bases da Técnica Cirúrgica**. 2002. Disponível em: http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2002/Dissertacao_JoaoFernandesBrittoAragao.pdf.

AWARI. **Camadas Keras no TensorFlow: Construindo redes neurais com facilidade**. 2023. Acesso em: 01 de outubro de 2024. Disponível em: https://awari.com.br/camadas-keras-no-tensorflow-construindo-redes-neurais-com-facilidade/.

AWARI. **Deep Learning Keras – Utilizando a biblioteca Keras para Deep Learning**. 2023. Acesso em: 03 de outubro de 2024. Disponível em: https://awari.com.br/deep-learning-keras-utilizando-a-biblioteca-keras-para-deep-learning/.

AWARI. **Python: Aprenda a utilizar a biblioteca NumPy para análise de dados**. 2023. Disponível em: https://awari.com.br/python-aprenda-a-utilizar-a-biblioteca-numpy-para-analise-de-dados/. Acesso em: 14 de janeiro de 2024.

BARONI, I.; PADILHA, E. J. **A Matemática por Trás das Redes Neurais**. 2021. Centro Universitário Internacional UNINTER. Acesso em 3 de março de 2024. Disponível em: https://repositorio.uninter.com/bitstream/handle/1/1023/ITANIM%c3%81%20BARONI%20-%20RU%202607819.pdf?sequence=1&isAllowed=y.

BARRETO, J. M. **Introdução às Redes Neurais Artificiais**. 2002. Disponível em: http://www.inf.ufsc.br/~j.barreto/tutoriais/Survey.pdf.

BELL, E. Generative ai: How it works, history, and pros and cons. **Investopedia**, maio 2023. Bussiness: Products and services. Disponível em: https://www.investopedia.com/generative-ai-7497939.

BOLDRINE, J. L.; OUTROS. **Álgebra Linear**. [*S.l.: s.n.*]: Harper e Row do Brasil, 1980.

BORGES, L. E. **Python para Desenvolvedores**. São Paulo - SP: Novatec Editora, 2014. Disponível em: https://awari.com.br/camadas-keras-no-tensorflow-construindo-redes-neurais-com-facilidade/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Camadas%20Keras%20no%20TensorFlow:%20Construindo%20redes%20neurais%20com%20facilidade.

BROWNLEE, J. **How to Prepare Text Data for Deep Learning with Keras**. 2019. Deep Learning for Natural Language Processing. Disponível em: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/.

CAO, W. *et al.* A comprehensive survey on geometric deep learning. **IEEE Access**, v. 8, p. 35929–35949, 2020. Disponível em: https://ieeexplore.ieee.org/abstract/document/9003285/authors#authors.

CARDOSO, C. **AAF-GAN: Aplicação de funções de ativação hiperbólicas adaptativas em redes neurais adversárias generativas**. 2021. Tese (Doutorado) — UFRJ/COPPE, Rio de Janeiro, 2021. Disponível em: https://www.cos.ufrj.br/uploadfile/publicacao/2992.pdf.

CARNEIRO, A. L. C. **Redes Neurais Convolucionais para Processamento de Linguagem Natural**. 2020. Data Hackers. Acesso em: 27 de janeiro de 2024. Disponível em: https://medium.com/data-hackers/redes-neurais-convolucionais-para-processamento-de-linguagem-natural-935488d6901b.

CHEN, A. C.-H. **Deep Learning: A Simple Example**. 2023. National Taiwan Normal University. Acesso em: 07 de janeiro de 2024. Disponível em: https://alvinntnu.github.io/NTNU_ENC2045_LECTURES/nlp/ml-simple-case.html.

COZMAN, F. G.; PLONSKI, G. A.; NERI, H. (ed.). **Inteligência Artificial: Avanços e Tendências**. São Paulo: IEA e C4AI-USP, 2021. Disponível em: https://www.livrosabertos.sibi.usp.br/portaldelivrosUSP/catalog/download/650/579/2181?inline=1.

DARA, S.; AL. et. Role of mathematics in machine learning. Indore, Madhya Pradesh - Índia, v. 4, 2022. Disponível em: https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/21435/final/fin_irjmets1652378206.pdf.

DASARADH, S. **A Gentle Introduction To Math Behind Neural Networks**. Chengalpattu, Tamil Nadu-India, 2020. Disponível em: https://towardsdatascience.com/introduction-to-math-behind-neural-networks-e8b60dbbdeba.

Data Science Academy. **Deep Learning Book**. 2022. Website. Disponível em: https://www.deeplearningbook.com.br/funcao-de-ativacao/.

DÖRR, J. B.; AYLON, L. B. R. Um estudo sobre técnicas utilizadas para o reconhecimento de sons com o uso de inteligência artificial e python. *In*: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Congresso Latino-Americano de Software Livre e Tecnologias Abertas (LATINOWARE), 19.** Porto Alegre, 2022. p. 103–112. Disponível em: https://sol.sbc.org.br/index.php/latinoware/article/view/22974.

EDUCAÇÃO, B. M. da. **Base Nacional Comum Curricular**. 2018. Brasília: MEC. Acesso em 6 de abril de 2024. Disponível em: http://fila.mec.gov.br/manutgeral.htm.

FACURE, M. **Funções de Ativação: Entendendo a importância da ativação correta nas redes neurais**. 2017. Matheus Facure. Acesso em: 18 de fevereiro de 2024. Disponível em: https://matheusfacure.github.io/2017/07/12/activ-func/.

FELISBINO, R. **Inteligência Artificial e Redes Neurais: Conceitos e Aplicações**. 2012. 47 p. Monografia, Assis - SP, 2012. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/0911270619.pdf.

FERREIRA, H. H. **Processamento de Linguagem Natural e Classificação de textos em Sistemas Modulares**. 2019. Dissertação (Mestrado) — Universidade de Brasília- UnB, Instituto de Ciências Exatas, Departamento de Ciência da Computação, Brasília, 2019. Disponível em: https://bdm.unb.br/bitstream/10483/25114/1/2019_HugoHondaFerreira_tcc.pdf.

FONSECA, C. **Word Embedding: fazendo o computador entender o significado das palavras**. 2021. Disponível em: https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057.

FREITAS, A. T. **Neurónio - Elemento de Processamento**. 2020. Instituto Superior Técnico, Universidade de Lisboa. Acesso em 03 de março de 2024. Disponível em: http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexb3a3.html?id=110.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition**. 2019. O'Reilly Media, Inc. Acesso em: 18 de fevereiro de 2024. Disponível em: https://powerunit-ju.com/wp-content/uploads/2021/04/Aurelien-Geron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-OReilly-Media-2019.pdf.

GHAHRAMANI, Z. Probabilistic machine learning and artificial intelligence. **Nature**, Cambridge - Reino Unido, v. 521, p. 452–459, maio 2015. Disponível em: https://www.nature.com/articles/nature14541.

GODOY, D. **Uma explicação visual para função de custo "binary cross-entropy" ou "log loss"**. 2018. Ensina.AI. Acesso em 24 de fevereiro de 2024. Disponível em: https://medium.com/ensina-ai/uma-explica%C3%A7%C3%A3o-visual-para-fun%C3%A7%C3%A3o-de-custo-binary-cross-entropy-ou-log-loss-eaee662c396c.

GOOGLE. **word2vec**. 2013. Texto retirado do Google Code Archive. Disponível em: https://code.google.com/archive/p/word2vec/.

GOOGLE. **O que é o Colab?** s.d. Disponível em: https://colab.research.google.com/?utm_source=scs-index#scrollTo=Nma_JWh-W-IF.

GOOGLE. **Primeiros passos: treinamento e previsão com Keras**. s.d. Disponível em: https://cloud.google.com/ai-platform/docs/getting-started-keras?hl=pt-br.

Google Developers. **Loss Functions**. 2022. Website. Disponível em: https://developers.google.com/machine-learning/gan/loss.

GÜNTZEL, J. L.; NASCIMENTO, F. A. **Introdução aos Sistemas Digitais**. Santa Catarina, 2001. Disponível em: http://www.inf.ufsc.br/~guntzel/isd/isd.html.

HEFEZ, A.; FERNANDEZ, C. S. **Introdução à Álgebra Linear**. [*S.l.: s.n.*]: SBM, 2016.

KATHURIA, R. S. *et al.* Real time sentiment analysis on twitter data using deep learning(keras). *In*: **International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)**. [*S.l.: s.n.*], 2019. Disponível em: https://ieeexplore.ieee.org/document/8974557.

KAZMI, H. **What are the different types of Keras models?** 2023. Disponível em: https://www.educative.io/answers/what-are-the-different-types-of-keras-models. Acesso em: 16 de janeiro de 2024.

KERAS. s.d. Disponível em: https://keras.io/.

Keras. **Accuracy metrics**. s.d. Keras. Acesso em 24 de fevereiro de 2024. Disponível em: https://keras-io.translate.goog/api/metrics/accuracy_metrics/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#binaryaccuracy-class.

Keras. **Model training APIs**. s.d. Keras. Acesso em 4 de fevereiro de 2024. Disponível em: https://keras-io.translate.goog/api/models/model_training_apis/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc.

KERAS. **Probabilistic losses**. s.d. Keras. Acesso em 24 de fevereiro de 2024. Disponível em: https://keras-io.translate.goog/api/losses/probabilistic_losses/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#binarycrossentropy-class.

KERAS. **Camada MaxPooling1D**. Sem data. Disponível em: https://keras-io.translate.goog/api/layers/pooling_layers/max_pooling1d/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc. Acesso em: 18 de fevereiro de 2024.

KHAKUREL, J. *et al.* The rise of artificial intelligence under the lens of sustainability. Basel, Switzerland, v. 6, n. 100, 2018. Disponível em: https://www.mdpi.com/2227-7080/6/4/100.

LA, F. Como as redes neurais aprendem? **MSDN Magazine Issues**, v. 34, n. 4, 2019. Acesso em: 21 de janeiro de 2024.

LEON, S. J. **Álgebra Linear com Aplicações**. 8. ed. LTC, 2010. Acesso em 29 de setembro de 2023. Disponível em: https://docs.ufpr.br/~higidio/Ensino/Books/Leon%28Algebra%20linear%29.pdf.

LUIZ, M.; ASCHER, D. **Aprendendo Python**. 2. ed. Porto Alegre: Bookman, 2007.

LUZ, F.; SALVATORE, F.; FINGER, M. **Processamento de Linguagem Natural por meio de Redes Neurais Profundas: Teoria e Aplicações**. Porto Alegre: [*S.l.: s.n.*], 2018. Universidade Federal do Rio Grande do Sul (UFRGS). Acesso em 16 de março de 2024. Disponível em: https://www.inf.ufrgs.br/propor-2018/wp-content/uploads/2017/09/Propor2018LuzSalvatoreFinger.pdf.

MADELEINE, S. **Apresentação da Rede Neural Convolucional Aplicadas a Imagens Médicas**. 2021. IMAIOS. Acesso em 17 de março de 2024. Disponível em: https://www.imaios.com/br/recursos/blog/classificacao-de-imagens-medicas-entendendo-a-rede-neural-convolucional-cnn.

MARQUES, A. **O uso da inteligência artificial no ensino da matemática**. 2020. 32 p. Dissertação, Macapá - AP, 2020. Disponível em: https://sca.profmat-sbm.org.br/profmat_tcc.php?id1=6076&id2=171053929.

MASSONI, G. **Análise de textos por meio de processos estocásticos na representação word2vec**. 2021. Mestrado, 2021. Disponível em: https://repositorio.ufscar.br/bitstream/handle/ufscar/14241/%5BUFSCAR%5D%20Gabriela%20Massoni%20-%20Disserta%C3%A7%C3%A3o_Vers%C3%A3oFinal.pdf?sequence=1&isAllowed=y.

MELO, C. **Redes Neurais Multicamadas com Python e Keras**. 2019. Disponível em: https://sigmoidal.ai/redes-neurais-python-keras-2/.

MIKOLOV, T. *et al.* **Efficient Estimation of Word Representations in Vector Space**. 2013. ArXivLabs: experimental projects with community collaborators - Computer Science. Acesso em: outubro de 2024. Disponível em: https://arxiv-org.translate.goog/abs/1301.3781?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc.

MIRANDA, L.; MATTAR, M. **Informática Básica**. Recife-PE: IFPE, 2014. Disponível em: https://www.ufsm.br/app/uploads/sites/413/2018/12/arte_informatica_basica.pdf.

MULINARI, B. **Numpy Python: O que é, vantagens e tutorial inicial**. 2021. Disponível em: https://harve.com.br/blog/programacao-python-blog/numpy-python-o-que-e-vantagens-e-tutorial-inicial/. Acesso em: 14 de janeiro de 2024.

MUZAFFAR, I. **What are Keras layers?** 2023. Disponível em: https://www.educative.io/answers/what-are-keras-layers. Acesso em: 21 de janeiro de 2024.

NELSON, D. Generative vs. discriminative machine learning models. **Unite.AI**, janeiro 2021. IA101. Disponível em: https://www.unite.ai/generative-vs-discriminative-machine-learning-models/.

NETO, A. P.; NETO, A. C. M. **O Conceito de Vetor**. 2023. Apostila do Terceiro Ano do Ensino Médio, Módulo: Vetores em R2 e R3, Portal da OBMEP. Disponível em: https://portaldaobmep.impa.br/index.php/modulo/ver?modulo=65&tipo=7.

NumPy Developers. **NumPy: the absolute basics for beginners**. s.d. Disponível em: https://numpy.org/doc/stable/user/absolute_beginners.html. Acesso em: 17 de fevereiro de 2024.

PAIVA, E. S. d.; PEREIRA, F. S. **Capítulo 2: Deep Learning para Processamento de Linguagem Natural**. 2022. Livro: Tópicos Especiais em Sistemas de Informação. Minicursos do XVIII Simpósio Brasileiro em Sistemas de Informação (SBSI). Acesso em 17 de março de 2024. Disponível em: https://sol.sbc.org.br/livros/index.php/sbc/catalog/view/86/379/654.

PAIVA, F. A. P. d. *et al.* **Introdução a Python com Aplicações de Sistemas Operacionais**. Natal - RN: Editora IFRN, 2020. Disponível em: https://memoria.ifrn.edu.br/handle/1044/2090.

PARANHOS, P. **O que é Word2Vec**. 2023. Disponível em: https://edrone.me/pt/blog/o-que-e-word2vec.

PINHEIRO, A. J.; SILVA, P. C. L. d. **Introdução à Álgebra Linear**. EdUFERSA, 2016. Acesso em 22 de setembro de 2023. Disponível em: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://educapes.capes.gov.br/bitstream/capes/204022/2/INTRODU%25C3%2587%25C3%2583O%2520A%2520ALGEBRA%2520LINEAR.pdf&ved=2ahUKEwiug7uNiu6HAxXaqZUCHX2VBrMQFnoECBIQAQ&usg=AOvVaw3XOe-2cDj7UU7Blwl53ChD.

REIS, B. **Redes neurais – Funções de ativação**. 2016. Laboratório Imobilis de Computação - UFOP. Acesso em: 18 de fevereiro de 2024. Disponível em: https://www2.decom.ufop.br/imobilis/redes-neurais-funcoes-de-ativacao/.

REIS, G. L.; SILVA, V. V. da. **Geometria Analítica**. Rio de Janeiro, RJ: LTC - Livros Técnicos e Científicos Editora S.A., 1996.

ROBIN, F. de Almeida Barros e J. **Processamento de Linguagem Natural**. Recife, PE, s.d. Documento. Disponível em: https://www.cin.ufpe.br/~fab/cursos/jai96/ProcessamentoDeLinguagemNatural.pdf.

RODRIGUES, D. A. **Deep Learning e Redes Neurais Convolucionais: Reconhecimento Automático de Caracteres em Placas de Licenciamento Automotivo**. João Pessoa: [*S.l.: s.n.*], 2018. Centro de Informática, Universidade Federal da Paraíba. Acesso em 16 de março de 2024. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/15606/1/DAR20052019.pdf.

RODRIGUES, D. A. **Deep Learning e Redes Neurais Convolucionais: Reconhecimento Automático de Caracteres em Placas de Licenciamento Automotivo**. 2018. Dissertação (Mestrado) — Centro de Informática, Universidade Federal da Paraíba (UFPB/CI), 2018. Acesso em: 27 de janeiro de 2018.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ: Prentice Hall, 2010. Disponível em: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf.

SANTOS, L. F. d. A. **Classificador baseado em Redes Neurais Convolucionais para algoritmos RMLSA em EONs**. Brasília: [*S.l.: s.n.*], 2020. Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Ciência da Computação. Acesso em: 18 de fevereiro de 2024. Disponível em: https://bdm.unb.br/bitstream/10483/27263/1/2020_LuizFilipeDeAndradeSantos_tcc.pdf.

SANTOS, R. J. **Matrizes, Vetores e Geometria Analítica**. Imprensa Universitária da UFMG, 2010. Acesso em 25 de agosto de 2023. Disponível em: https://www.ime.unicamp.br/~deleo/MA141/ld01a.pdf.

Saturn Cloud. **Understanding the Use of Verbose in Keras Model Validation**. 2023. S.d. Acesso em 24 de fevereiro de 2024. Disponível em: https://saturncloud-io.translate.goog/blog/understanding-the-use-of-verbose-in-keras-model-validation/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc.

SAXENA, S. **Binary Cross Entropy/Log Loss for Binary Classification**. 2023. Analytics Vidhya. Acesso em 24 de fevereiro de 2024. Disponível em: https://www-analyticsvidhya-com.translate.goog/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc.

SCHALKOFF, R. J. (ed.). **Artificial Intelligence: An Engineering Approach**. New York, NY: McGraw-Hill, 1990.

SHARMA, P. **Keras Tokenizer Tutorial with Examples for Beginners**. 2021. Disponível em: https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit_on_texts-texts_to_sequences-texts_to_matrix-sequences_to_matrix/.

SICHMAN, J. Inteligência artificial e sociedade: avanços e riscos. **Estudos Avançados**, v. 35, n. 101, p. 37–50, abril 2021. Disponível em: https://www.revistas.usp.br/eav/article/view/185024.

SILVA, M. P. e. **Aplicação de Redes Neurais Artificiais no Diagnóstico de Falhas de Turbinas a Gás**. 2010. Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio. Acesso em 10 de março de 2024. Disponível em: https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=16580@1.

SOUSA, J. R. d. *et al.* Python e predição de dados usando redes neurais multicamadas. **Brazilian Journal of Development**, Curitiba, v. 6, n. 7, p. 54181–54185, jul 2020. Disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/14311/11924.

STEINBRUCH, A.; WINTERLE, P. **Álgebra Linear**. 1ª edição. ed. [*S.l.: s.n.*]: Pearson Universidades, 1995.

TATIBANA, C. Y.; KAETSU, D. Y. Redes neurais. 2023. Apud (Felisbino, 2012).

TATIBANA, C. Y.; KAETSU, D. Y. **Redes Neurais**. s.d.

TENSORFLOW. **Keras: The high-level API for TensorFlow**. 2023. Disponível em: https://www.tensorflow.org/guide/keras. Acesso em: 16 de janeiro de 2024.

TENSORFLOW. **Module: `tf.keras.preprocessing.sequence`**. 2023. Disponível em: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence. Acesso em: 14 de janeiro de 2024.

TENSORFLOW. **Word embeddings**. 2023. Disponível em: https://www.tensorflow.org/text/guide/word_embeddings. Acesso em: 21 de janeiro de 2024.

TensorFlow. **Save and load models**. 2024. TensorFlow Core. Acesso em 25 de fevereiro de 2024. Disponível em: https://www.tensorflow.org/tutorials/keras/save_and_load.

VICERI-SEIDOR. **Arquiteturas de Redes Neurais Convolucionais para reconhecimento de imagens**. 2020. Disponível em: https://viceri.com.br/insights/arquiteturas-de-redes-neurais-convolucionais-para-reconhecimento-de-imagens/. Acesso em: 27 de janeiro de 2024.

VIDAL, M. **O que são GANs?** 2020. Website. Disponível em: https://www.venturus.org.br/blog/o-que-sao-gans/.

VISHWAKARMA, N. **What is Adam Optimizer?** 2024. Analytics Vidhya. Acesso em: 14 de junho de 2024. Disponível em: https://www-analyticsvidhya-com.translate.goog/blog/2023/09/what-is-adam-optimizer/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc.

ZANETTE, A. **Framework x Biblioteca x API. Entenda as diferenças!** 2017. Disponível em: https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas/#comments.

ZHU, S.; CHOLLET, F. **Understanding masking & padding**. 2023. Disponível em: https://www.tensorflow.org/guide/keras/understanding_masking_and_padding. Acesso em: 14 de janeiro de 2024.

APÊNDICES

APÊNDICE A – SEQUÊNCIA DIDÁTICA

Como realizamos um levantamento acerca dos tópicos e habilidades matemáticas contidas na BNCC, achamos que seria interessante apresentar uma sequência didática, de forma que qualquer professor possa utilizar em sala de aula, caso ache relevante para a aprendizagem dos seus estudantes. Assim como qualquer plano de aula, nos atentamos às informações necessárias para a correta aplicação do tema "A Matemática por trás de Inteligências Artificiais do Tipo ChatBot, tema que explora Redes Neurais Artificiais.

Objetivos Gerais: O objetivo dessa sequência didática consiste em mostrar aos estudantes a presença da matemática em inteligências artificiais simples, uma vez que tal aparato já faz parte de nosso cotidiano, e ao mesmo tempo trabalhar habilidades contidas na BNCC, acerca de tópicos relevantes como funções exponenciais, valor numérico de uma função e funções definidas por mais de uma sentença.

Objetivos específicos:

- Entender o funcionamento básico de inteligências artificiais do tipo Chatbot/Redes Neurais Artificiais a nível do ensino básico.
- Compreender um pouco da matemática por trás de IAs por meio de um exemplo básico.
- Ampliar o conhecimento em tópicos como funções (Lineares, Exponenciais, definidas por várias sentenças e valor numérico).
- Conhecer o código Python de forma superficial, e realizar testes por tentativa e erro.

Conteúdo:

Nesta sequência estarão presentes os conteúdos de função exponencial, funções definidas por várias sentenças, valor numérico de uma função e linguagem de programação (de forma superficial).

Procedimentos metodológicos:

Para a aplicação desta sequência, faremos uma separação em em seis momentos. Sendo eles:

1. Neste primeiro momento é interessante que o professor retome de forma breve as definições de funções (Exponencial, definidas por várias sentenças e valor numérico) para que os estudantes relembrem esses conteúdos, visto que faz-se a necessidade

que já tenham sido estudados. Para que assim possam entender as funções a serem trabalhadas nessa sequência didática:

Função ReLU (Rectified Linear Unit): f(x) = max (0,x)

$$f(x) = \begin{cases} 0 & \text{se } x = 0 \\ x & \text{se } x > 0 \end{cases} \tag{A.1}$$

Função Sigmoide: Explique como a função sigmoide é usada como função de ativação.

$$\varphi(y) = \frac{1}{1 + e^{-y}}$$

Podemos notar a presença da exponencial, valor numérico (uma vez que os estudantes irão atribuir valores para estas funções) e funções definidas por várias sentenças, como é o caso da função ReLu e de alguns limitantes em nosso código que será devidamente explicado mais à frente.

2. Em um segundo momento, o professor deverá desenvolver uma breve noção de uma Inteligência Artificial e Redes Neurais Artificiais, tal como sua evolução no decorrer dos anos, os impacto que elas deixam e podem deixar no futuro e funcionamento, correlacionando as funções descritas com o resumo:

   A inteligência artificial é uma área relativamente jovem, e possivelmente teria surgido em 1956, na Dartmouth College Conference. Porém, embora seja uma área promissora, passou por momentos de dúvida denominados “Invernos da IA” onde a Inteligência artificial sofreu diversas críticas em relação a alta expectativa da tecnologia e seu baixo retorno, implicando na retirada do suporte financeiro, em especial nas décadas de 1970 e 1990. Todavia, durante a década de 1990, muitas das técnicas computacionais foram desenvolvidas, e com o avanço tecnológico das últimas décadas, foi possível o desenvolvimento de redes mais complexas. Atualmente, atravessamos um novo momento de entusiasmo com relação aos potenciais benefícios que a inteligência artificial pode promover, com inúmeras aplicabilidades nas mais vastas áreas do conhecimento humano. Embora haja inúmeros avanços, o termo “inteligência artificial” não é bem definido, pois além do fato de que o conceito de inteligência é subjetivo, se trata de uma área muito vasta, com diversas interpretações e conceitos que nem sempre estão alinhados. É interessante ressaltar que a técnica de Redes Neurais Artificiais (RNAs), surgiu com o intuito de fazer uma analogia entre neurônios biológicos e circuitos eletrônicos, de modo a criar uma representação que simulasse as conexões sinápticas. Essas redes, podem aprender por meio de exemplos com diferentes técnicas de aprendizagem, e posteriormente fazer interpolações do que foi assimilado. Redes Neurais são compostas por uma camada de entrada, uma camada de saída e pelo menos uma camada intermediária chamada de camada oculta. A

camada oculta processa as informações de entrada e as transforma em um formato útil para a camada de saída. Cada uma dessas camadas é formada por neurônios artificiais que realizam cálculos para processar os dados. Na figura abaixo temos um paralelo entre um neurônio artificial e um neurônio biológico.

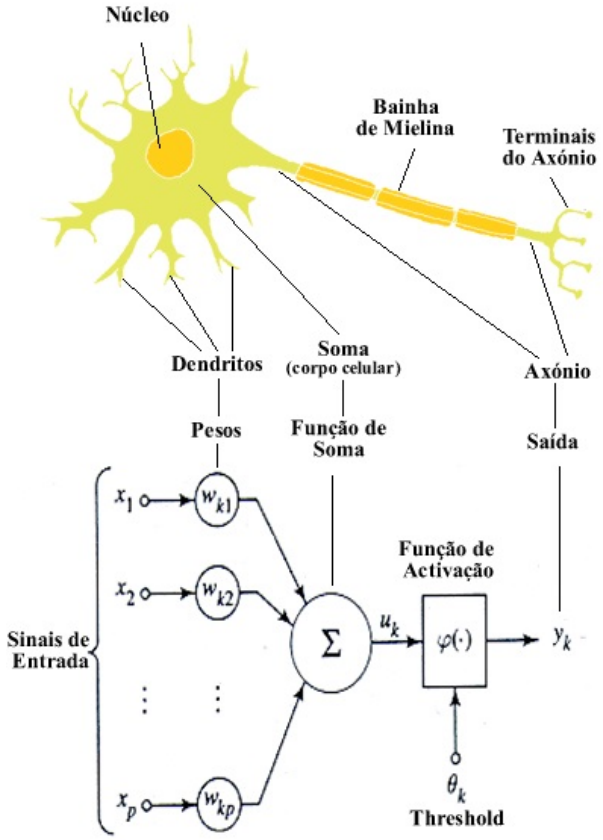

Figura 10 – Neurônio Biológico (acima) Vs Neurônio Artificial (abaixo)

Fonte: (Freitas, 2020)

De acordo com Baroni e Padilha (2021), o cérebro humano necessita de treinamento para cada nova tarefa, o neurônio biológico recebe sinais por meio dos diversos dendritos, que serão analisados e encaminhados ao próximo neurônio, ou não (Threshold). A partir daí, são definidos os pesos, ao qual chamamos de memorização. Já as Redes Neurais Artificiais (RNAs), segundo os autores, são estruturas compostas por camadas de nós, sendo a primeira a camada de entrada, seguida por camadas ocultas e a camada de saída. Os neurônios em uma RNA recebem sinais de entrada, que podem vir dos dados brutos ou de neurônios em camadas anteriores, realizam cálculos e enviam sinais de saída para neurônios mais profundos na rede. Segundo Baroni e Padilha (2021), a (2021), os neurônios podem ter sinapses conectadas a mais de um neurônio na camada anterior, e a partir daí efetuam cálculos e enviam sinais de saída para neurônios mais profundos por meio da sinapse. Cada sinapse tem um peso associado, determinando a importância do neurônio anterior na rede. Assim,

o ajuste dos pesos é crucial no treinamento de modelos de aprendizado profundo. Ou seja, após receberem as entradas, os neurônios somam os sinais multiplicados pelos pesos correspondentes e os passam por uma função de ativação, que efetua o cálculo do valor de saída do neurônio, valor que é repassado adiante para a próxima camada da rede por meio de outra sinapse.

3. Neste momento, o professor apresenta o problema "Festival de Pipas" aos estudantes, explorando e interligando a matemática e redes neurais artificiais. É interessante que os estudantes participem da leitura e de uma interpretação inicial do problema. O professor deve apresentar as variáveis a serem consideradas (velocidade do vento, precipitação de chuva e temperatura) e a função de ativação.

O problema: Festival de Pipas

Considere que a comunidade local de certa cidade quer realizar um festival de pipas. Todavia, necessitam averiguar as condições climáticas para determinar a viabilidade do evento em determinada data. Assim, os organizadores decidiram utilizar uma rede neural simples organizada da seguinte forma: 3 neurônios na camada de entrada (representando probabilidade de chuva, velocidade do vento e temperatura); 3 neurônios na camada oculta (com pesos e bias associados a cada neurônio); função de ativação ReLU $max(tx_m + vx_n + px_z + b, 0)$; um neurônio na camada de saída (output) e função de ativação Sigmoide,

$$\varphi(y) = \frac{1}{1 + e^{-y}}$$

Será levado em consideração três variáveis principais: velocidade do vento, temperatura e probabilidade de chuva. Suponha que os resultados de entrada na camada oculta, sejam obtidos por meio das expressões $max(tx_1 + vx_2 + px_3 + b_1, 0)$, $max(tx_4 + vx_5 + px_6 + b_2, 0)$ e $max(tx_7 + vx_8 + px_9 + b_3, 0)$ de tal forma que t, v e p sejam a temperatura (em graus Celsius), velocidade do vento (km/h) e a probabilidade de chuva. Considere ainda que b_1, b_2 e b_3 sejam valores constantes de bias, e x_1 , x_2, x_3, x_4, x_5, x_6, x_7, x_8 e x_9, os respectivos pesos. Suponha ainda, que os resultados de saída dessa camada, sejam obtidos pela função ReLU descrita acima, de tal forma que y seja o maior valor obtido dentre a saída das três funções de entrada. Efetue tentativas atribuindo valores fictícios a cada variável, e escolha pesos de acordo com sua necessidade para cada valor de x e b. Considere:

$output >= 0,5$ - O Festival de Pipas é viável

$output < 0,5$ - O Festival de Pipas é inviável

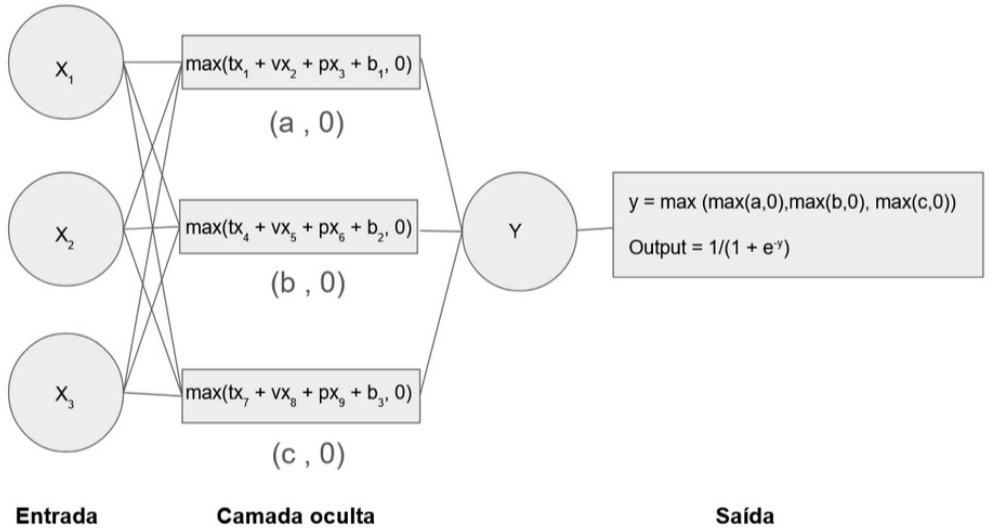

Figura 11 – Modelo de Rede Neural a ser utilizado

Fonte: Elaborado pelo autor

4. Neste passo teremos a resolução do problema. Assim, os estudantes devem por meio de cálculos de tentativa e erro identificar quando será possível realizar o festival considerando temperatura, probabilidade de chuva e velocidade de vento. É interessante que o professor divida a sala em pequenos grupos, para que possam discutir e testar algumas das possibilidades. Abaixo, temos um exemplo de atribuição de pesos, onde a chuva apresenta valor negativo, uma vez que impede a realização do evento.

   Temperatura: 22.0 °C, Vento = 8.0 km/h e Chuva=0.2. Considerando

   $X_1 = 0,05, x_2 = 0,06, x_3 = -3,1$ e $b_1 = 0$ $x_4 = 0,04, x_5 = 0,07, x_6 = -2,8$ e $b_2 = 0$ $x_7 = 0,03, x_8 = 0,05, x_9 = -2,9$ e $b_3 = 0$

   Para $max(tx_1 + vx_2 + px_3 + b_1, 0)$:

   $max(22\ .\ 0,05 + 8\ .\ 0,06 + 0,2\ .\ (-3,1) + 0\ ;\ 0)$

   $max(1,1 + 0,48 - 0,62 + 0\ ;\ 0)$

   $max(0,96\ ;\ 0) = 0,96$

   Para $max(tx_4 + vx_5 + px_6 + b_2\ ;\ 0)$

   $max(22\ .\ 0,04 + 8\ .\ 0,07 + 0,2\ .\ (-2,8) + 0\ ;\ 0)$

   $max(0,88 + 0,56 - 0,56 + 0\ ;\ 0)$

   $max(0,88\ ;\ 0)$

   Para $max(tx_7 + vx_8 + px_9 + b_3; 0)$

   $max(22\ .\ 0,03 + 8\ .\ 0,05 + 0,2\ .\ (-2,9) + 0\ ;\ 0)$

$max(0,66 + 0,4 - 0,58 + 0\ ;\ 0)$

$max(0,48\ ;\ 0) = 0,48$

Assim,

$max((max(0,96\ ;\ 0); max(0,88\ ;\ 0); max(0,48\ \ 0)) = 0,96$

Para a função de ativação temos:

$output = \frac{1}{1+e^{-0,96}}$

$output = 0,7231$

5. Nesta etapa, os alunos devem apresentar alguns dos resultados que encontraram e verificar se os dados encontrados são ou não factíveis com a realização de um festival de pipas. Por exemplo, no caso testado anteriormente tivemos $output = 0,7231$. Ou seja, as condições são ideais para a realização do festival de pipas, pois $0,7231 > 0,5$.

6. Por fim, o professor pode apresentar aos estudantes um exemplo de rede neural em código Python, onde os estudantes podem realizar testes e ajustar os pesos e bias. Deve ficar claro que se trata apenas de um código inicial e que uma rede neural precisa realizar treinamento com uma quantidade significativa de dados para se obter um resultado satisfatório. O código está disponível para acesso em: (https://colab.research.google.com/drive/1YkXcOmXw6wQeYwhWAEgenuKnmFRuJc89?usp=sharing).

Basicamente, o professor deve fornecer o link do Google Colab aos estudantes, ao qual deverá acessar utilizando um computador. Nas linhas de 15 à 17, estão as matrizes referentes aos pesos e bias, conforme a figura 3, sendo a probabilidade de chuva, velocidade do vento e temperatura respectivamente. Cada linha representa um neurônio, assim podem conter valores diferentes para os pesos. Já na linha 18, seguem as constantes de cada bia. Em nosso exemplo essa linha é representada por valores nulos, porém podem ser modificados de acordo com as necessidades do usuário.

```python
# Classe da rede neural ajustada
class SimpleNeuralNetworkReLU:
    def __init__(self):
        # Ajustando os pesos para dar menos importância à chuva e temperatura
        self.weights_hidden = np.array([[-3.1, 0.06, 0.05],
                                        [-2.8, 0.07, 0.04],
                                        [-2.9, 0.05, 0.03]])
        self.bias_hidden = np.array([0.0, 0.0, 0.0])
```

Figura 12 – Ajuste de pesos no ambiente Google Colab

Fonte: Elaborado pelo autor

Nas linhas 40 à 61, estão algumas funções (sentenças) para determinar se as condições são ou não ideais para alguns casos, como apresentado na figura 4. Com estas linhas podemos restringir algumas situações indesejáveis como ventos muito fortes, muito fracos, temperaturas muito altas ou baixas, além de limitar a probabilidade máxima para a chuva. Essas condições foram inseridas apenas como sugestões de possibilidades de análise, para que os estudantes e o professor tenham noção de quais condições podem observar. Além disso, é um código bem simples para dar uma previsão exata, sem contar que a rede necessitaria de um treinamento considerável. Porém, essas linhas podem ser excluídas caso o professor ou os estudantes queiram maximizar o uso do código, visto que assim podem verificar mais possibilidades, além do resultado de cada cálculo. Tais linhas podem ser observadas na imagem a seguir e no código mais abaixo.

```python
# Função para determinar se as condições são ideais para soltar pipa
def verificar_condicoes(chuva, vento, temperatura):
    # Verificar se o vento é superior a 35 km/h
    if vento > 35:
        print(f"Chuva: {chuva} probabilidade, Vento: {vento} km/h, Temperatura: {temperatura} °C -> Não ideal para soltar pipas (vento forte)")
        return
    if vento < 8:
        print(f"Chuva: {chuva} probabilidade, Vento: {vento} km/h, Temperatura: {temperatura} °C -> Não ideal para soltar pipas (vento fraco)")
        return
    # Verificar se a probabilidade de chuva é superior a 0.5
    if chuva > 0.5:
        print(f"Chuva: {chuva} probabilidade, Vento: {vento} km/h, Temperatura: {temperatura} °C -> Não ideal para soltar pipas (alta probabilidade de chuva)")
        return

    # Verificar se a temperatura é abaixo de 15°C ou acima de 30°C
    if temperatura < 15:
        print(f"Chuva: {chuva} probabilidade, Vento: {vento} km/h, Temperatura: {temperatura} °C -> Não ideal para soltar pipas (temperatura muito baixa)")
        return

    if temperatura > 30:
        print(f"Chuva: {chuva} probabilidade, Vento: {vento} km/h, Temperatura: {temperatura} °C -> Não ideal para soltar pipas (temperatura muito alta)")
        return
```

Figura 13 – Ajuste de condições ideais no ambiente Google Colab

Fonte: Elaborado pelo autor

Já nas linhas 71 à 75 (figura 5), a sentença que determina a tomada de decisão da rede, onde $output \geq 5$ equivale a condições ideais para a realização do evento e $output < 5$ não indica condições favoráveis à realização do festival de pipas.

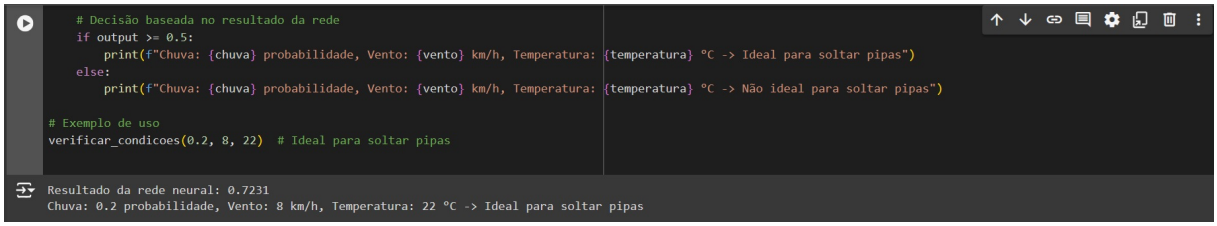

Figura 14 – Condições de tomada de decisão da rede no ambiente Google Colab

Fonte: Elaborado pelo autor

A linha 78 trás um exemplo de uso, onde tanto os estudante quanto o professor podem realizar testes atribuindo os valores referentes a probabilidade de chuva, velocidade do vento (em km/h) e temperatura (em °C) respectivamente. Note que o exemplo exibido na imagem é o mesmo utilizado anteriormente nos cálculos manuais (0.2 , 8 , 22). Ao substituir esses valores, o usuário deverá clicar em "executar a célula" (simbolizado por um círculo branco com um triângulo interno) localizado na coluna à esquerda do código, ou clicar em alguma área da célula e utilizar o comando Ctrl + Enter. Com essa ação o código será executado e irá determinar as condições para o evento e o resultado numérico obtido pela rede neural. Uma sugestão é que os alunos possam além de testar condições diversas, possam verificar os cálculos manuais por meio do código da rede atribuindo os mesmos pesos e bias utilizados por eles. Segue abaixo o código Python da rede neural a ser utilizada na integra.

```
1  import numpy as np
2  import numpy as np
3
4  # Função de ativação ReLU
5  def relu(x):
6      return np.maximum(0, x)
7
8  # Função de ativação Sigmoide
9  def sigmoid(x):
10     return 1 / (1 + np.exp(-x))
11
12 # Classe da rede neural ajustada
13 class SimpleNeuralNetworkReLU:
14     def __init__(self):
15         # Ajustando os pesos para dar menos importância à chuva
               e temperatura
16         self.weights_hidden = np.array([[-3.1, 0.06, 0.05],
17                                         [-2.8, 0.07, 0.04],
```

```
18                                          [-2.9, 0.05, 0.03]])
19          self.bias_hidden = np.array([0.0, 0.0, 0.0])
20
21      def forward(self, inputs):
22          # Camada oculta
23          net_hidden = np.dot(self.weights_hidden, inputs) +
                self.bias_hidden
24          output_hidden = relu(net_hidden)
25
26          # Pegando o valor máximo da camada oculta
27          max_hidden = np.max(output_hidden)
28
29          # Aplicando a função sigmoide no valor máximo da camada
                oculta
30          output_final = sigmoid(max_hidden)
31
32          return output_final
33
34  # Função para realizar simulações
35  def simulate(chuva, vento, temperatura):
36      nn = SimpleNeuralNetworkReLU()
37      inputs = np.array([chuva, vento, temperatura])
38      output = nn.forward(inputs)
39      return output
40
41  # Função para determinar se as condições são ideais para soltar
        pipa
42  def verificar_condicoes(chuva, vento, temperatura):
43      # Verificar se o vento é superior a 35 km/h
44      if vento > 35:
45          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
                km/h, Temperatura: {temperatura} C -> Não ideal para
                soltar pipas (vento forte)")
46          return
47      if vento < 8:
48          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
                km/h, Temperatura: {temperatura} C -> Não ideal para
                soltar pipas (vento fraco)")
49          return
50      # Verificar se a probabilidade de chuva é superior a 0.5
51      if chuva > 0.5:
52          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
```

```
            km/h, Temperatura: {temperatura}  C  -> Não ideal para
            soltar pipas (alta probabilidade de chuva)")
53          return
54
55      # Verificar se a temperatura é abaixo de 15 C  ou acima de
            30 C
56      if temperatura < 15:
57          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
            km/h, Temperatura: {temperatura}  C  -> Não ideal para
            soltar pipas (temperatura muito baixa)")
58          return
59
60      if temperatura > 30:
61          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
            km/h, Temperatura: {temperatura}  C  -> Não ideal para
            soltar pipas (temperatura muito alta)")
62          return
63
64      # Executa a rede neural para calcular a probabilidade
65      nn = SimpleNeuralNetworkReLU()
66      inputs = np.array([chuva, vento, temperatura])
67      output = nn.forward(inputs)
68
69      # Exibe o resultado da rede neural
70      print(f"Resultado da rede neural: {output:.4f}")
71
72      # Decisão baseada no resultado da rede
73      if output >= 0.5:
74          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
            km/h, Temperatura: {temperatura}  C  -> Ideal para
            soltar pipas")
75      else:
76          print(f"Chuva: {chuva} probabilidade, Vento: {vento}
            km/h, Temperatura: {temperatura}  C  -> Não ideal para
            soltar pipas")
77
78  # Exemplo de uso
79  verificar_condicoes(0.2, 8, 22)  # Ideal para soltar pipas
```

Listing A.1 – Código da Rede Neural - Festival de Pipas

Recursos didáticos: Código Python (Google Colab), lousa, televisão, notebook e

folhas de rascunho.

Avaliação: A avaliação pode ser dada de forma contínua e por meio de atividades práticas e participativas, envolvendo os cálculos e simulações que os estudantes irão realizar de forma manual e virtual.

ANEXOS

ANEXO A – CÓDIGO ABERTO NO GITHUB

Este capítulo conta com o texto disponível na página https://github.com/3sdras/sacodepalavras, que disponibiliza o código do chatbot criado nesta pesquisa de maneira aberta para qualquer pessoa interessada.

A.1 Saco de Palavras de Geometria

Este repositório apresenta uma implementação simples do modelo "saco de palavras" de redes neurais, usado exemplo durante os trabalhos de escrita da dissertação do acadêmico Lucas Meireles Pereira no PROFMAT, Programa de Mestrado Profissional em Rede Nacional, na UFJ, em Jataí-GO.

O PROFMAT é um Programa de Mestrado em Matemática, então o foco tanto da dissertação quanto deste trabalho é diferente do usual, sendo mais voltado para a comunidade matemática, em especial para professores da Educação Básica com interesse em trazer o tema da Inteligência Artificial para suas aulas de maneira simples.

Caso este seja seu primeiro contato com o Github, a Linguagem Python e as redes neurais, fica a recomendação da leitura da dissertação, voltada para a comunidade matemática, disponível no portal do profmat, seção "dissertações".

O script mostra como usar python para fazer um chatbot no estilo "Saco de Palavras" que consegue classificar frases de Geometria Plana como verdadeiras ou falsas. Este modelo de chatbot é ideal para classificações por ser bastante simples e ignorar diversas variáveis que fogem do escopo deste trabalho.

O computador é uma máquina capaz de realizar apenas dois tipos de operações: lógicas e matemáticas. Para que o mesmo consiga "entender"um texto, as frases são "quebradas" em palavras (chamadas tokens) e estes são depois transformados em vetores.

A máquina então é treinada para reconhecer padrões, como proximidade, entre estes vetores. A partir destes padrões entre vetores que correspondem a tokens/palavras, a máquina "entende" o significado das frases.

Uma introdução à Matemática por trás deste processo pode ser lida na dissertação, que recomendamos novamente a leitura.

A máquina então processa textos para que possa ser "treinada"para reconhecer padrões e alimentar um modelo de inteligência artificial baseado em redes neurais, que, neste caso, é o modelo "Saco de Palavras", que é especialmente simples por não levar em consideração diversos aspectos da linguagem.

Como você pode usar este script: O formato de arquivo deste script é .ipynb, um

formato que permite tanto blocos de texto, quanto blocos executáveis de programação na linguagem Python.

Os blocos de texto são breves e explicam o que você pode mudar, como executar e o que esperar de cada bloco de código.

O arquivo está pronto para ser usado no Google Colab e tem todas as informações de uso descritas passo a passo. Mais informações e detalhes podem ser encontrados na dissertação no link
https://colab.research.google.com/drive/1qQS6cytCKhdiLOXRoPFTwDMtpMK3QV5n?usp=sharing