CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

CHARLLEY JUNIOR JABBAR
PEDRO ANTÔNIO CORDEIRO SOUSA

CIENC.IA
BUSCA E PRODUÇÃO AUTOMÁTICA DE TEXTOS

ANÁPOLIS - GO

2020

CHARLLEY JUNIOR JABBAR
PEDRO ANTÔNIO CORDEIRO SOUSA

CIENC.IA
BUSCA PRODUÇÃO AUTOMÁTICA DE TEXTOS

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientadora: Prof. Ma. Natasha Sophie Pereira.

ANÁPOLIS - GO

2020

CHARLLEY JUNIOR JABBAR
PEDRO ANTÔNIO CORDEIRO SOUSA

CIENC.IA
BUSCA E PRODUÇÃO AUTOMÁTICA DE TEXTOS

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em ____ de _____ de 2020, composta por:

_____
Profa. Ma. Natasha Sophie Pereira
Orientador

_____
Prof. [nome do professor]
Examinador Interno

_____
Prof. [nome do professor]
Examinador Interno

Agradecimentos

Gostaria de agradecer à minha mãe Germana Inácio e avó Maria da Luz de Freitas, que esteve sempre do meu lado dando muito carinho e apoio;

Agradecer ao meu irmão, Rafael Jabbar que me deu palavras de força para continuar trilhando este caminho durante minhas crises e fases de entropia;

Sempre ao meu lado gostaria de estar agradecendo à minha namorada Laressa Rayane Rosa Lima, que me aturou por toda esta trilha, sendo uma grande parceira e pilar, apoiando as minhas decisões e sonhos;

Sem eles este trabalho de conclusão e o curso em si não seria possível de ser concluído. Agradeço aos meus Professores que dedicaram tempo e esforços lecionando diversos aprendizados além de lições, estas não só técnicas quanto de vida e carreira;

Agradecimento especial para o Professor Kleber Silvestre Diogo, que ao entrar no curso, não foi somente um grande professor, mas sim um amigo que me mostrou a maravilha do mundo da tecnologia de primeira mão, com diversas experiencias do mundo real;

Agradeço à Professora Ma. Natasha Sophie Pereira pelo carisma, excelência e paciência, sendo companheira de pesquisas e orientadora desta obra prima na qual chamamos de CIENC.IA (trabalho de conclusão de curso), mas além de tudo sempre trazendo aquele ar positivo de felicidade e alegria pro dia a dia do curso;

Somos gratos a todos os professores Adrielle Beze Peixoto, Aline Dayany de Lemos, Alexandre Moraes Tannús, Clarimar José Coelho, Kleber Silvestre Diogo, Luciana Nishi, Marcelo de Castro Cardoso, Marcio Mariano da Silva, Millys Fabrielle Araújo Carvalhaes, Natasha Sophie Pereira, Raissa dos Santos Vieira Renata Dutra Braga, Viviane Carla Batista Pocivi, Walquíria Fernandes Marins, William Pereira dos Santos Júnior, que são muito importantes na minha vida acadêmica, como um todo, palavras vindas de dois acadêmico que antes éramos somente acadêmicos perdidos e sem foco, porém agora palavras de futuros Bacharéis em Engenharia de Computação !

Agradecemos aos amigos e colegas pela força e torcida para que tudo desse certo.

“Se cheguei até aqui foi porque me apoiei no ombro dos gigantes .”

Isaac Newton

Resumo

Com o passar dos anos a inteligência artificial tem crescido exponencialmente em nossa sociedade e tomando responsabilidades ao redor do globo em prol do homem. Com apoio das tecnologias do Data Science Suit (*Machine Learning, Deep Learning, Neural Networks*), diversos sistemas e aplicações foram desenvolvidos visando solucionar problemas predefinidos, desde o reconhecimento de voz, processamento de imagens, *chatbots*, processamento de linguagem natural, entre outros. O desenvolvimento de textos possui diversas particularidades que dificultam a sua produção como tempo de escrita, plágios, pesquisas de fontes relevantes, dentre outros fatores. Observando tal situação o propósito deste trabalho é desenvolver um agente inteligente capaz de realizar buscas textuais e produzir textos seguindo determinados parâmetros de pesquisa. Para isso, foi desenvolvida uma Rede Neural Artificial capaz de produzir textos a partir de exemplos coletados da internet de acordo com parâmetros pré-definidos pelo usuário. Com o desenvolvimento deste sistema, pretende-se auxiliar pesquisadores em suas pesquisas de fontes relevantes, e fichamento destas fontes.

Palavras-chaves: Inteligência Artificial; Busca e Produção de texto.

Abstract

Over the years, artificial intelligence has grown exponentially in our society and taking responsibility around the globe for the benefit of man. With the support of Data Science Suit technologies (Machine Learning, Deep Learning, Neural Networks), several systems and applications were developed aiming to solve predefined problems, beginning from speech recognition, image processing, chatbots, natural language processing, among others. The development of texts has several particularities that may delay or hinder their production, such as writing time, plagiarism, research from relevant sources(quality), among other factors. Observing this situation, the purpose of this work is to develop an intelligent agent capable of performing textual searches and producing texts following certain research parameters. For this, an Artificial Neural Network, capable of producing texts based on examples collected from the internet, according to pre-defined parameters by the user. With the development of this system, it is intended to assist researchers in their research of relevant sources, and to file these sources.

KEYWORDS: Artificial Intelligence; Text Production and Search.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Network*
CNN	*Convolutional Neural Network*
DFF	*Deep Feedforward*
GRU	*Gated Recurrent Unit*
IA	Inteligência Artificial
K-NN	*K-Nearest Neighbors*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
NLP	*Natural Language processing*
RNN	*Recurrent Neural Network*
SVM	*Support Vector Machine*

SUMÁRIO

1. INTRODUÇÃO

Conforme os anos vêm se passando a sociedade está sempre em constante progresso, acompanhar essa evolução é parte essencial da tecnologia. Conforme Fava (2018) apresenta em sua obra, a Inteligência Artificial (IA) é o acelerador da humanidade, com isso, tudo que foi criado está sempre em processo de modificações, se tornando diferente, e por meio disso é preciso adaptar as invenções pela ideia de aceitação e adaptabilidade.

Durante a evolução da sociedade a busca por informação virou significado de poder. De acordo com os autores Maciel & Albagli (2011), uma pessoa durante a sua vida toda não era capaz de receber a quantidade de informação que nos deparamos hoje em dia. Fato este que se remete à veracidade da informação além dos tipos de dados, sendo de cunho comum o saber científico.

De acordo com Souza (2012), as pessoas devem seguir a um conjunto de normas preestabelecidas pela forma culta da língua, que se baseia na gramática normativa, sem considerar as demais variedades linguísticas, espera-se que as pessoas internalizem a coesão e a coerência da língua e seu conjunto de regras sem levar em conta a qualidade que deve ser escrita, para ter um texto no padrão que seja publicado em sites, revistas ou eventos acadêmicos. Outro fator que deve ser observado é a questão da administração do tempo e a forma na qual a pessoa acata prazos ao se deparar com a necessidade de desenvolver um trabalho científico, aponta Antunes et al. (2011):

> Com a dificuldade para administrar o tempo, constata-se que muitos dos alunos alegaram ter uma sobrecarga de trabalho (...) argumentando ser bastante difícil conciliar atividades profissionais e pessoais (casa, filhos, outros compromissos) e ainda conseguir atender os prazos de entrega das etapas parciais do projeto de trabalho de conclusão de curso (p.6).

A busca textual segue com a necessidade social da população, o tipo textual possui a especificidade de apontar o conjunto de características gerais de um grupo, ao passo que o gênero, apropriando-se dessas generalidades, adapta-as ao uso Cassetari (2012). Por outro lado, conforme verificado pelos cientistas Baldin *et al.* (2008), em um estudo sobre as vantagens provenientes da automatização de processos por meio de uma IA, “a uniformidade e maior velocidade na execução do processo, baixo índice de erros, menores custos, entre outros” são características intrínsecas de uma IA. Sendo assim, pode-se observar como a IA tem sido capaz de agilizar e aperfeiçoar processos, aprimorando sua qualidade de acordo com os parâmetros desejados além de tornar possível a economia de tempo gasto durante tarefas que demandam muito esforço (“tempo”) do ser humano.

Portanto a Inteligência Artificial (IA) é uma área de pesquisa da Ciência da Computação que tem como objetivo o desenvolvimento de métodos e metodologias utilizando o computador, voltada para projetar e desenvolver softwares nas suas mais diversificadas modalidades e aplicações, desenvolver maneiras eficientes de ser resolver problemas computacionais. Ela realiza ou reforça a capacidade humana de se comportar de forma inteligente, seja na resolução de problemas, aquisição e representação de conhecimento, reconhecimento de padrões, entre outros (LIMA; PINHEIRO; SANTOS, 2014). Em síntese da afirmação apresentada, a IA auxilia diversos setores da sociedade, desde o âmbito da saúde com resultado de exames, ao de atendimento de pessoas em e-commerce, uma vez que estas atividades são empenhadas pelo ser humano e podem ser substituídas por um método automatizado, complementa CHOLLET (2018), automatizar tarefas intelectuais normalmente realizadas por seres humanos é o papel da IA.

Um sistema inteligente que filtra e define um parâmetro de dados, ajuda à tornar as tarefas mais amenas, como uma tentativa de deixar as informações cedidas menos complexas para o usuário, que ao desenvolver um texto que envolva pesquisas em várias fontes poderá ser auxiliado na escolha da produção textual em que baseará sua escrita (MENDES, 1997).

Resultados provenientes de uma inteligência artificial capaz de escrever o seu próprio conto, como a obra poética “1 the Road” foi cenário de um experimento novelístico feito por Ross Goodwin o qual foi redigido por uma rede neural treinada com determinados parâmetros (GRAHAM, 2018).

As Redes Neurais Artificiais (ANN, do inglês Artificial Neural Network) têm um aprendizado complexo, fato este que ao se unir com as capacidades de aprendizagem e processamento da linguagem natural, resulta em um aprendizado que envolve parâmetros de texto e normas já predefinidas criando um modelo que visa seguir as características da linguagem humana. Tendo como exemplo de base algoritmos cujo aprendizado é supervisionado, que aprende a responder de modo similar ao que a pessoa deveria fazer quando redige um texto, com um conjunto de entradas e saídas esperadas e desejadas (BITTENCOURT; OSÓRIO, 2001). Levando em consideração os pontos levantados pelos autores, a IA é capaz de aderir em diversos setores sociais quando aplicada adequadamente.

A revolução tecnológica, mais conhecida como “revolução 4.0” (SANSON, 2017), é tida como um avanço para a ciência bem como para a sociedade, uma vez que a IA está

redefinindo o jeito humano de ser. Aplicações como o IBM Watson for oncologist[1], ROSS[2], HER[3], são projetos que atualmente estão solucionando obstáculos encontrados pelo ser humano de maneira automatizada e prática. De acordo com a empresa IBM, seu robô está diagnosticando exames de pacientes com câncer nos ossos, através da análise de exames apresentados por eles (ROSS; SWETLITZ, 2017). Por outro lado, já no setor da aquisição de informações, como relatado pelo autor Joshi (2017), a busca de dados e informações científicas é comumente o método utilizado para o desenvolvimento de chatbots, robôs inteligentes, softwares para reconhecimento facial, jogos, entre outros, que são pesquisas e estudos voltados para a sociedade.

Sendo o principal objetivo desenvolver um sistema baseado em uma Inteligência Artificial (IA), que realiza a busca e produção de textos. Para tanto, será necessário cumprir as seguintes etapas: Primeiramente definir os parâmetros de produção e busca de textos, logo em seguida definir as normas de desenvolvimento do texto em questão, por fim treinar os agentes inteligentes e analisar a acurácia dos mesmos.

Em síntese do contexto abordado a busca pela informação começa por uma necessidade, portanto como usar técnicas de inteligência artificial para desenvolver e buscar textos de forma automática?

[1] Adapt DevOps to cognitive and artificial intelligence systems – IBM Developer
[2] What is AI - ROSS Intelligence
[3] https://www.newworldai.com/her/

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Inteligência Natural

De acordo com Goedert (2017), o cérebro pode ser considerado uma versão de computador altamente complexo, que tem como capacidade interagir com seus componentes estruturais, que ele define como neurônios, e com isso ajuda no processamento e realização das informações de forma muito rápida, através de ligações que ele define como sinápticas.

Com base nisso Goedert (2017) cita o autor Haykin (2009), mostrando um exemplo no sistema de visão humana, capaz de reconhecer padrões e cita como modelo um rosto familiar em uma cena desconhecida e com isso mostra que a capacidade para reconhecer leva aproximadamente entre 100 e 200 ms e compara isso com um computador convencional, que a maiorias das pessoas usam, contatando que o computador levaria alguns dias para realizar tarefas menos complexas que essa.

2.2. Inteligência Artificial

De acordo com Russell e Norvig (2013), a inteligência artificial pode ser definida como um estudo de agentes inteligentes que recebem ações e executam percepções do ambiente que foram designados a estudar Com isso, cada agente tem uma função a ser implementada que mapeia as sequências de ações que serão executadas pela IA de maneiras diferentes, tais como sistemas de produção, agentes reativos, redes neurais, planejadores condicionais em tempo real e sistemas de teoria de decisão.

De acordo com Teixeira (2019), o conceito de inteligência artificial é como um *commodity* que envolve uma forma de fluxo de dados que serão fornecidos por algumas empresas, com isso ele cita como exemplo o que já ocorreu com a água, a energia elétrica e os serviços de telefonia.

2.3. Machine Learning

O aprendizado de máquina conhecido também como *Machine Learning* (ML) é um dos métodos mais populares da Inteligência Artificial dentro da sociedade. O *software* é projetado e desenvolvido para que seja capaz de aprender sobre os dados. Tendo como base estes modelos de aprendizagem, se torna possível a realização de predições em dados desconhecidos (JOSHI, 2017, p. 18).

O uso do aprendizado de máquina abrange não só o âmbito das predições, mas também problemas cognitivos e percepção. Segundo o autor Rusk (2016), *machine learning* é uma

forma que envolve o aprendizado de máquina que permite às Inteligências Artificiais resolverem problemas de percepção, como reconhecimento de imagem, fala e escrita. Lecun, Bengion e Hinton (2015) complementam que *machine learning* é uma tecnologia de aprendizado de máquina que abrange a sociedade moderna nas áreas de pesquisas e consulta na Web, filtragem de palavras em redes sociais, entre outros.

Segundo Vasilev *et al.* (2019)*, Machine Learning* é um termo que está bastante associado à um grande volume de dados (*big data*[4]) e também à inteligência artificial. ML é uma ferramenta para uso de processamento de dados em larga escala, juntamente com *deep learning*. O aprendizado de Máquina pode ser através da aprendizagem supervisionada, através da não supervisionada, ou por reforço.

- Aprendizagem supervisionada: Os algoritmos de uma aprendizagem supervisionada experienciam o *dataset* (conjunto de dados) contendo determinadas funcionalidades, porém cada exemplo é associado à uma regra ou um alvo a atingir (GOODFELLOW, I., BENGIO, Y., COURVILLE, A, 2016, p. 103). Em outras palavras, são apresentados a estes algoritmos dados previamente rotulados para que estes possam aprender suas funcionalidades, de modo que o mesmo algoritmo seja capaz de classificar dados não rotulados (VASILEV et al., 2019, p. 8).

- Aprendizagem não supervisionada: Os algoritmos neste caso não são apresentados a dados rotulados, portanto cabe a eles chegarem sozinhos a uma conclusão. Um bom exemplo é a técnica de *clustering* (separar os dados em sub conjuntos) (VASILEV et al., 2019, p. 16). Esta categoria de ML consiste em encontrar transformações do dado de entrada sem ajuda nenhuma, com o propósito de visualização dos dados, compressão de dados, eliminar ruídos ou até mesmo uma melhor compreensão sobre uma correlação de dados (CHOLLET, F. ,2018, p. 94);

- Aprendizagem por reforço: Consiste em desenvolver o agente para aprimorar a sua performance a cada interação com o ambiente. Este tipo de aprendizagem, é usado para jogos de xadrez (RASCHKA, S. & MIRJALILI,V. , 2017, p. 6), onde o agente realiza uma ação no ambiente alterando o estado do jogo, agindo

[4] "Big data é um termo usado para descrever gigantescos *datasets,* que são criados como resultados do crescimento de dados adquiridos e guardados. Por exemplo, requisições da bolsa de valores, redes sociais, motores de jatos dentre outros meios massivos de produção de dados" (VASILEV *et al.*, 2019).

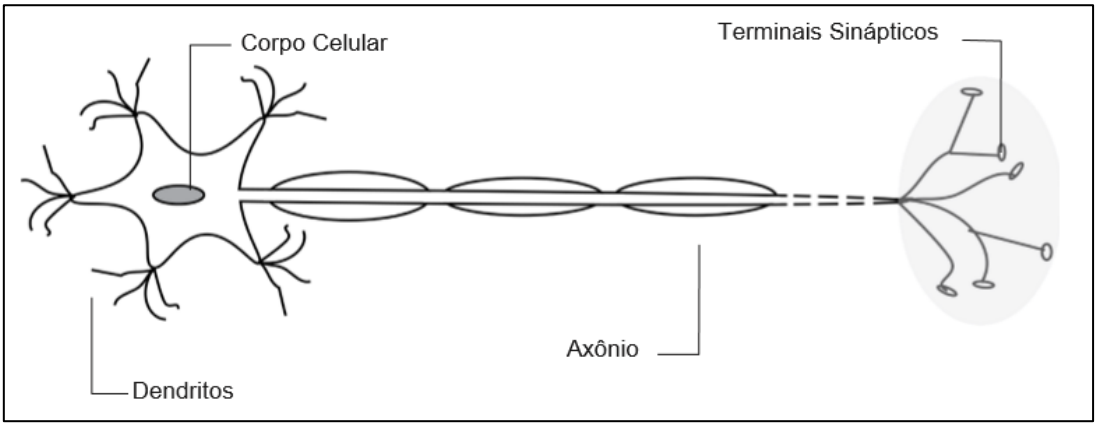
com base no novo estado e o seu resultado o agente procura determinar a sua nova decisão (VASILEV et al., 2019, p. 20).

2.4. **Redes Neurais Artificiais**

Com o decorrer da história o primeiro exemplo de rede neural, conhecido como *perceptron*, foi inventado por Frank Rosenblatt em 1957 (VASILEV *et al.*, 2019, p. 26). *Perceptron* é um algoritmo de classificação semelhante ao de "*logistic regression*" (VASILEV *et al.*, 2019, p. 27), também conhecido como rede neural "*feedforward*", que é um *perceptron* formado por uma simples camada. As redes neurais artificiais, identificadas como *Artificial Neural Network (ANN*), são inspiradas nas redes neurais biológicas e implementadas em programas de computador projetados para simular a forma com a qual o cérebro humano processa as informações (AL-SHAWWA; BERESFORD, 2000).

O neurônio humano recebe informações de outros neurônios por meio dos dendritos que transmitem a informação para ser processada nos axônios, após esta etapa ela é enviada para o próximo neurônio por meio dos terminais sinápticos (MOREIRA, 2013), como podemos observar na estrutura do neurônio biológico (Figura 1).

Figura 1 - Estrutura de um Neurônio Biológico.



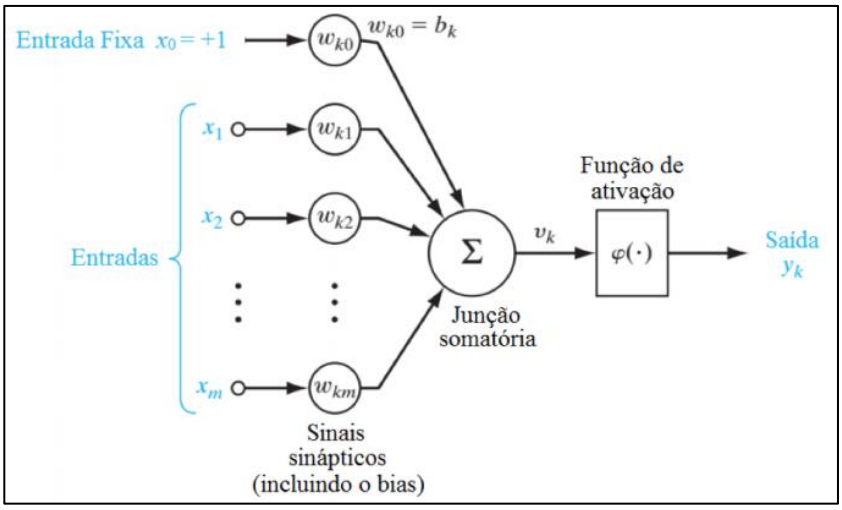
Fonte: Adaptado de Buduma, 2017.

A rede neural artificial é a aplicação da inteligência artificial, relatam os pesquisadores Al-Shawwa, et al. (2018). As Redes Neurais assumem a responsabilidade de adquirir conhecimento além de detectar os padrões e os relacionamentos perante os dados e a aprendizagem (AGATONOVIC-KUSTRIN& BERESFORD, 2000)

O funcionamento das redes neurais consiste em pesos e decisões cognitivas que consideram situações particulares em cada etapa. Joshi (2017, p. 400) complementa que os pesos e as decisões influenciadas são ajustados para processos de treinamentos de modo que

venha a ser um bom modelo de aprendizagem para determinada situação. Cada neurônio recebe um conjunto de entradas, as processa e então apresenta o valor de saída (resultado), conforme pode ser visto na Figura 2, que é a representação de um neurônio artificial baseado no modelo de McCulloch - Pitts[5], conhecido também como *Perceptron*.

Figura 2 - Perceptron.



Fonte: Adaptado de Haykin, 2009.

O neurônio então é definido matematicamente pela Equação. (1):

$$y = f\left(\sum_i x_m w_m + b\right) \tag{1}$$

cujas etapas são representadas por Vasilev *et al.* (2019, p. 37) da seguinte forma: Primeiramente é computado o valor da somatória das entradas de x_m com os pesos de w_m (conhecido também como valor de ativação). Neste caso x_m podem ser valores numéricos representantes tanto de dados de entrada, quanto de dados de saída do neurônio (caso este neurônio seja parte de uma rede neural com mais camadas). Sendo assim, os pesos de w_m, são valores numéricos que representam tanto força de entrada quanto a força das conexões entre os neurônios, e o peso de b é um valor especial chamado de "bias". O resultado da somatória dos pesos, como uma entrada de ativação da função $f(\varphi)$, que é conhecida também como função de transferência, resulta na operação desejada. No caso dos perceptrons estas funções são classificadas em não lineares.

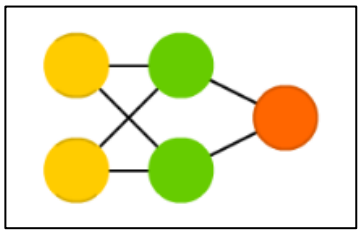
[5] Sendo um fisiologista e conhecendo as ondas de potencial de membrana, ele interpretou o funcionamento do neurônio como sendo um circuito binário. (MCCULLOCH, 1965).

Além do *perceptron* humano as redes neurais podem ser classificadas em diversos outros tipos de redes, cada uma com sua particularidade.

2.4.1. **Redes Neurais Feedforward**

A Rede Neural *Feedfoward* (Figura 3), é responsável por conectar diversos "neurônios" únicos à uma rede com mais de uma camada (multicamadas), sendo então conhecida também pelo nome de *Multilayer Perceptron* (MLP) (RASCHKA *et al*., 2017, p. 384).

Figura 3 - Rede Neural *Feedforward*.



Fonte: Adaptado de The Asimov Institute, 2016.

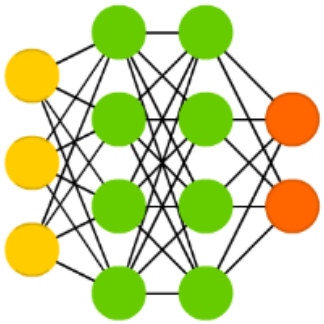
Conforme Raschka & Mirjalili (2017) e Vasilev et al. (2019) complementam em suas obras que a formação das redes neurais *Feedforward* é baseada em 3 etapas, sendo, pelo menos, uma camada de entrada (*in*), uma camada oculta (*hidden layer*) e por fim uma camada de saída (*out*). Camadas estas que seguem um padrão de ligação onde as camadas ocultas estão ligadas com todas as camadas de entradas e camadas de saída, propagando a informação adiante desde as camadas de entrada até as de saída, o que reflete o próprio nome da rede RASCHKA, S. & MIRJALILI,V. , 2017, p. 384; VASILEV *et al.*, 2019, p. 27).

O motivo da propagação das unidades entre as camadas pode variar de acordo com a condição pré-programada para a rede, de forma sucinta as mesmas se propagam para atingir a condição de ativação de determinada função dentro da camada de saída, como explica Vasilev *et al.* (2019, p. 27). Primeiramente a informação está presente na camada de entrada e em seguida é processada pelo computador, por fim adentrando às camadas ocultas, ou de ativação, (H_n, sendo n cada camada oculta). Desta forma propagando a informação para a camada de saída, tendo-a como entrada para a próxima camada até atingir o resultado da operação (VASILEV et al 2019, p. 27).

Já as Redes Neurais *Deep Feedforward* (Figura 4) são aquelas quando uma rede *feedforward* vem a possuir mais de uma camada oculta, também sendo considerada uma rede neural artificial profunda (RASCHKA, S. & MIRJALILI,V. , 2017, p. 384), os autores

Goodfellow, Bengio e Courville (2016, p.164) relatam a importância das redes do tipo *feedforward* no mundo real, visto que elas compõem a base de diversas aplicações comerciais, como por exemplo as redes neurais convolucionais (CNN, do inglês *Convolutional Neural Network*) para reconhecimento de objetos em imagens, que são tipos especializados de redes *feedforward*.

Figura 4 - Rede Neural *Deep Feedforward (DFF)*.



Fonte: Adaptado de The Asimov Institute, 2016.

O termo "*Deep*", de profundidade, está diretamente relacionado à capacidade da rede de possuir mais de uma camada, como explicam os pesquisadores Goodfellow, Bengio e Courville (2016), a estrutura encadeada de funções $f^n(x)$ é o que forma as camadas, conforme apresentado na Equação. (2):

$$f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right) \tag{2}$$

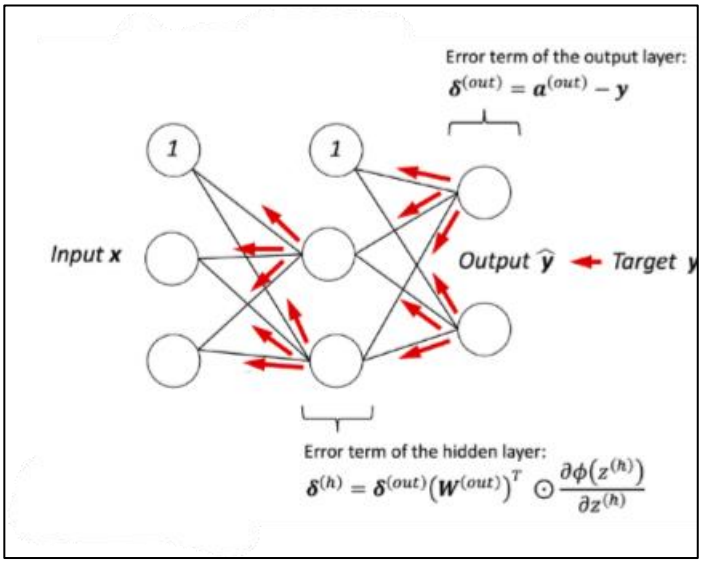
Estruturas estas comumente utilizadas nas redes neurais, no caso $f^{(1)}$ é a primeira camada da rede, $f^{(2)}$ a segunda camada da rede e assim continua. Por fim o comprimento destes encadeamentos resulta na profundidade (*Deep*) do respectivo modelo das *Deep feedforward*.

2.4.2. **Algoritmo de Backpropagation**

O algoritmo de *Backpropagation*, ou retroprogramação (Figura 5), procura atuar como agente de transporte, realizando ajustes de pesos em cada camada baseados em erros já obtidos nas camadas de saída da rede. O algoritmo vai caminhando até a última camada oculta, e então volta para a penúltima camada oculta e assim por diante até atingir a primeira. Goodfellow (2016, p. 401) comenta sobre o passo inverso pela rede explicando o algoritmo como "[...] o gradiente sobre $h^{(t+1)}$propagando um passo para trás, durante o *backpropagation*". Raschka & Mirjalili (2017) complementa que, "o *backpropagation* é um procedimento

computacionalmente bastante eficiente, para computar as derivativas parciais de uma função de custo complexo dentro de uma rede neural de multicamadas".

Figura 5 - Fluxo do algoritmo de *backpropagation* em uma rede neural.



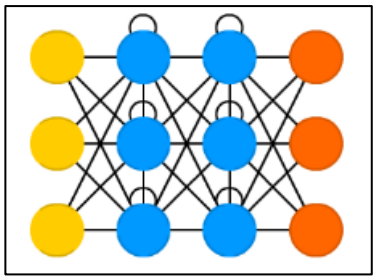
Fonte: Adaptação de Raschka & Mirjalili, 2017.

A figura 5 apresenta um exemplo do fluxo do algoritmo de backpropagation onde uma entrada X (input x) é propagada por todas as camadas da rede até atingir a camada de saída (output y) , em seguida o termo de erro calculado para a saída é retropropagado por todas as camadas ocultas (hidden layer) da rede.

2.5. **Recurrent Neural Network**

As *Recurrent Neural Network* (RNN), também conhecidas como redes neurais recorrentes(Figura 6), são redes que possuem um estado interno (ou memória), que é baseada em toda ou parte dos dados de entrada já alimentados na rede (VASILEV *et al*., 2019, p. 73).

Figura 6 - *Recurrent Neural Network* (RNN).

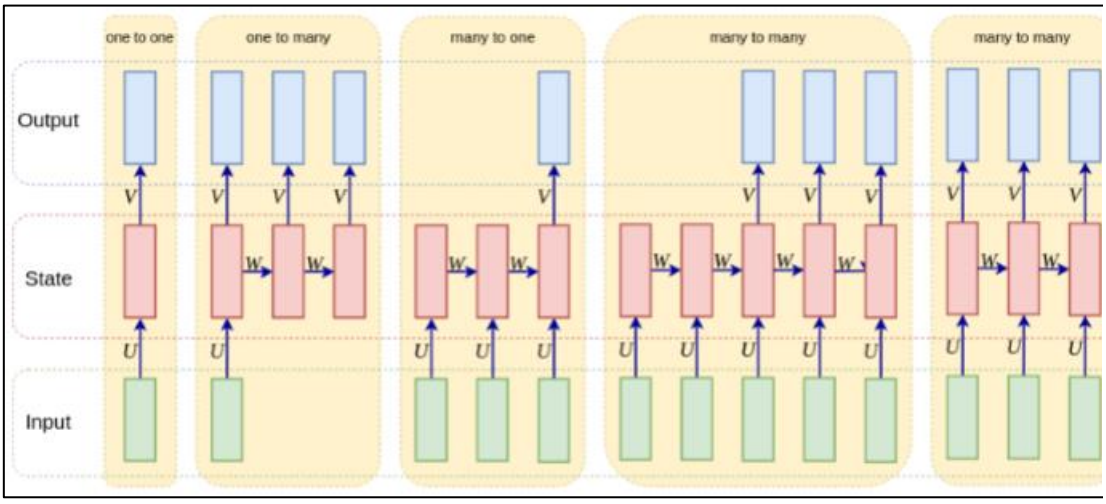


Fonte: Adaptado de The Asimov Institute, 2016.

Segundo Rumelhart *et al.* (1986 *apud* Goodfellow, Bengio e Courville, 2016), as redes neurais recorrentes fazem parte de uma família específica para processamento sequencial de dados, semelhante às redes convolucionais que são especializadas no processamento de uma grade de valores MxN, como é o caso de uma imagem. A respeito das RNNs, os autores Schäfer & Zimmermann (2006) afirmam que elas permitem a identificação de sistemas dinâmicos em forma de alta dimensão, como modelos não lineares de espaços nos estados, além de oferecer uma modelagem explícita de tempo e memória.

Pelo fato de as RNNs não serem limitadas ao processar dados fixos de entrada, elas possibilitam diversas combinações em suas sequências, conforme apresentado por RASCHKA & MIRJALILI. (2017) e Vasilev et al. (2019), e observado na Figura 7, que apresenta uma representação das possíveis categorias que as RNNs podem formar dependendo da configuração dos estados, entradas e saídas da rede.

Figura 7 - Possibilidades de configurações das RNNs.



Fonte: Adaptação de Vasilev, 2019.

A Figura 7 traz cinco possibilidades de configuração para uma RNN:

- *One to One* (Um para um): Este não é um processamento sequencial, sendo assim, as redes neurais *feedforward* e as redes neurais convolucionais se encaixam nesta categoria. Nota-se que não há muita diferença entre uma rede *feedforward* e a aplicação de uma RNN uma única vez. A Classificação de Imagens é um exemplo de sua aplicação (VASILEV *et. al.*, 2019).

- *One to Many* (Um para muitos): A entrada dos dados é pelo formato padrão, não sequencial, porém a saída de dados é sequencial, ou seja, este processamento é baseado em uma única entrada e resulta em uma saída sequencial. A

geração de legendas para imagens é exemplo para esta categoria (RASCHKA & MIRJALILI., 2017; VASILEV *et al.*, 2019, p. 200).

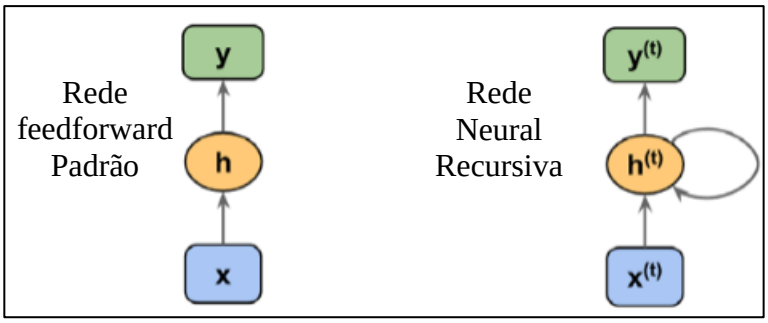
- *Many to One* (Muitos para um): A saída deste processamento gera apenas um único resultado, baseado na sequência de entradas, um dos exemplos de sua aplicação é a classificação de sentimentos advindos de textos. (VASILEV et al., 2019)

- *Many to Many* (Muito para muitos (Indireto)): A sequência é codificada em um vetor de estado, em seguida esse vetor de estado é decodificado em uma nova sequência. Como exemplo tem-se a tradução de uma linguagem (VASILEV *et al.*, 2019).

- *Many to Many* (Muito para muitos (Direto)): Ambas as entradas e saídas são sequenciais, atuando em sincronia na recepção de *frames* de dados (RASCHKA & MIRJALILI., 2017, pg.540). Como exemplo está o reconhecimento de voz (VASILEV *et al.*, 2019).

Os autores Rhaschka & Mirjalili (2017), procuram deixar claro a diferença entre uma rede neural *feedforward* e uma RNN, conforme representado na Figura 8.

Figura 8 - Rede neural *feedforward* em comparação com uma RNN.



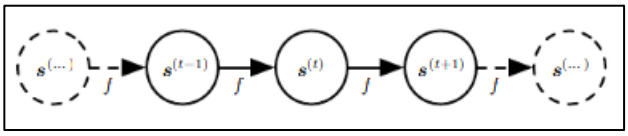
Fonte: Adaptação de Rhaschka, 2017.

Para uma explicação mais prática sobre o funcionamento das RNNs, Goodfellow, Bengio e Courville (2016, p. 369), mostram que elas operam em uma sequência que contém um vetor $x(t)$ no qual o index de passo do tempo t, se estende a um limite entre 1 à τ. Sendo assim, tem-se a forma clássica de um sistema dinâmico representado pela equação. (3):

$$s^{(t)} = f\left(s^{(t-1)}; \theta\right) \quad (3)$$

Na qual $s^{(t)}$ é conhecido como estado do sistema. A Eq. (3) é considerada recorrente devido à definição de s no momento t, levando em consideração a mesma definição de tempo de $t-1$. Conforme apresentado na Figura 9, que ilustra a Eq. (3) como um gráfico computacional não dobrado, onde cada nó representa o estado em algum tempo, e a função define o estado no tempo $t+1$. Os mesmos parâmetros (o mesmo valor de θ usado para parametrizar f) são usados para todas as etapas de tempo.

Figura 9 - O sistema dinâmico clássico descrito pela equação 3.



Fonte: Adaptação de Goodfellow, 2016.

Porém, o foco da pesquisa não será trabalhar todas as redes neurais existentes, apenas aquelas que são indicadas especificamente para o processamento de texto.

2.5.1. **Long Short-Term Memory**

Às *Long Short-Term Memory* (LSTM) são um tipo especial de RNN, capaz de aprender dependências de longo prazo. Eles foram introduzidos por Hochreiter & Schmidhuber (1997), e foram refinados e popularizados por muitas pessoas no trabalho seguinte. Eles funcionam muito bem em uma grande variedade de problemas e agora são amplamente usados

Os LSTMs são projetados explicitamente para evitar o problema de dependência de longo prazo, também têm essa estrutura em cadeia, mas o módulo de repetição tem uma estrutura diferente. Em vez de ter uma única camada de rede neural, existem quatro, interagindo de uma maneira muito especial.

2.5.2. **GRU**

Uma variação um pouco mais dramática no LSTM é a *Gated Recurrent Unit*, ou GRU, introduzida por Cho, et al. (2014). Ele combina as portas de esquecer e de entrada em uma única "porta de atualização". Ele também mescla o estado da célula e o estado oculto e faz algumas outras alterações. O modelo resultante é mais simples do que os modelos LSTM padrão e tem se tornado cada vez mais popular.

2.5.3. **Deep Learning**

Foi observado nos tópicos anteriores os o funcionamento de uma rede neural e seus principais componentes, uma vez que uma rede neural é construída, ela é projetada para ser

capaz de atender as necessidades de cada situação. Segundo Joshi (2017), as redes neurais construídas com diversas camadas são denominadas "*deep neural network*", ou redes neurais profundas. A parte da inteligência artificial responsável em lidar com uma rede neural profunda pode ser referida como "*deep learning*".

O aprendizado profundo, ou *deep learning*, pode ser considerado aquele onde diversas camadas de decisões são programadas em uma rede neural com a finalidade de aperfeiçoar o aprendizado dessa rede (STERNE, 2017, p.85). Estas camadas seguem um conjunto de métodos e representações, fato este explicado pelos autores LeCun, Bengio e Hinton (2015):

> Os métodos de *Deep-learning* são métodos de "representação de ensinamentos", com diversos níveis de representação, obtidos através da composição de módulos simples, mas não lineares, que transformam a representação em um nível (começando com a entrada bruta) para uma representação em um nível mais alto e um pouco mais abstrato. (p. 436).

2.6. **Bibliotecas e Pacotes Populares Open-Source para Data Science**

Sem as devidas ferramentas, não se cria algo, como explica Vasilev (2019), existem diversas bibliotecas de fonte livre que permitem a criação de redes neurais profundas em linguagem de programação Python, sem a necessidade de se escrever o código do zero. Dentre as mais populares atualmente no mercado tem-se o TensorFlow[7], o Keras[8] e o PyTorch[9]:

- **TensorFlow (TF):** conhecida como a biblioteca mais popular de *deep learning*, é desenvolvida pela Google. Ela elimina a necessidade de se usar explicitamente a GPU, porém exige que o desenvolvedor determine quais unidades de processamento serão destinadas para o projeto. O TensorFlow possui uma curva de aprendizado bastante acentuada.

- **Keras:** Pode ser considerada uma rede neural de alto nível, sendo uma biblioteca do Python que roda sobre o TensorFlow, CNTK[10], ou Theano. Em comparação com o TF, Keras é mais rápido e prático de se utilizar, sendo mais fácil e ideal para experimentos rápidos.

[7] https://www.tensorflow.org
[8] http://keras.io
[9] https://pytorch.org/
[10] https://github.com/Microsoft/CNTK

- **PyTorch**, é uma biblioteca para *deep learning*, baseada em Torch e desenvolvida pelo Facebook. É considerada relativamente mais prática de se utilizar que as demais.

Cada uma das bibliotecas faz uso de pacotes de dados, que contém algoritmos fundamentais para a manipulação dos dados, afirma Raschka & Mirjalili (2017). Algumas das bibliotecas mais utilizados em *data science,* são a SciPy, a NumPy, a Scikit-learn, a Matplotlib e a Panda.

- **SciPy:** É uma biblioteca para rotinas numéricas desenvolvida na linguagem de programação Python, proporcionando blocos fundamentais de construção para modelagem e soluções de problemas científicos (Virtanen, P., Gommers, R., Oliphant, T.E. *et al*, 2020, p. 261).

- **NumPy:** É uma estrutura de dados conhecida como *N-dimensional array* que compreende a matriz NumPy, além do conjunto de funções matemáticas que lhe acompanha, o seu uso tem sido adotado em faculdades, laboratórios e indústrias, com aplicações de seu uso entre desenvolvimento de jogos à exploração do espaço sideral (VAN DER WALT, S., COLBERT, S. C., & VAROQUAUX, G., 2011, p. 22).

- **Scikit-learn:** Presente no ambiente da linguagem Python, é rico em fornecer implementações de ponta para muitos algoritmos de aprendizado de máquina conhecidos, mantendo uma interface fácil de usar e fortemente integrado com a linguagem Python (Pedregosa F., Varoquaux G., Gramfort A., et al., 2011, p. 2826).

- **Matplotlib:** Biblioteca de plotagem em linguagem Python focada no desenvolvimento de aplicações gráficas. Utilizada para plotar gráficos interativos nas dimensões 2D e 3D (Ranjani J., Sheela A., Meena K. P., 2019, p. 1), é muito versátil na customização de gráficos (RASCHKA, S. & MIRJALILI, V., 2017, p. 15).

- **Panda:** Biblioteca do Python que possui uma variedade de estruturas de dados e ferramentas de estatística que foi inicialmente desenvolvida para aplicações quantitativas de economia (McKinney ,W. ,2010, p. 56).

2.7. Processamento de Dados

Uma etapa crucial para o início das pesquisas e dos projetos, é o tratamento dos dados a serem utilizados, conforme apresenta Vasilev (2019):

> O processamento de dados é tipicamente um problema que pode ser solucionado por meio de técnicas clássicas de ML... O processamento de dados implica a limpeza dos dados, tais como remoção de redundância ou funcionalidades altamente correlacionadas, ou até mesmo a ausência de um dado e por fim compreendendo as funcionalidades determinadas como dados de teste.

Segundo Chollet (2018, p. 135), os dados devem ser formatados propriamente para serem inseridos em uma rede. O algoritmo se baseia em dados repletos de informações para reproduzir, de forma automatizada, os mesmos padrões definidos e utilizados como base de seu processamento (DONEDA, 2018). Sendo assim, de acordo com Doneda (2018), os dados aparentemente inofensivos também podem ser usados como base para a produção, em razão da grande capacidade atual de processamento e cruzamento de informações que podem ser usados para que ocorram as trocas de dados.

De acordo com Lemos (2017), a computação cognitiva tem possibilidades de processamento de grande volume de dados, com capacidade elevada de aprendizado e identificação de padrões que pode ser comparado ao de um ser humano e ainda permite que o ser humano possa interagir com o sistema que envolve a linguagem natural, aumentando a experiência.

2.8. **Processamento de Linguagem Natural**

O *Natural Language processing* (NLP), conhecido também em português como Processamento de Linguagem Natural, é a capacidade de um computador em compreender a linguagem dos seres humanos e reagir como tal. Conforme dito por Goyal, Pandey e Jain (2018, p. 16), “O processamento de linguagem Natural, de forma sucinta, é a habilidade que o computador/sistema tem de compreender a linguagem do homem e processá-la da mesma forma que o ser humano faz”.

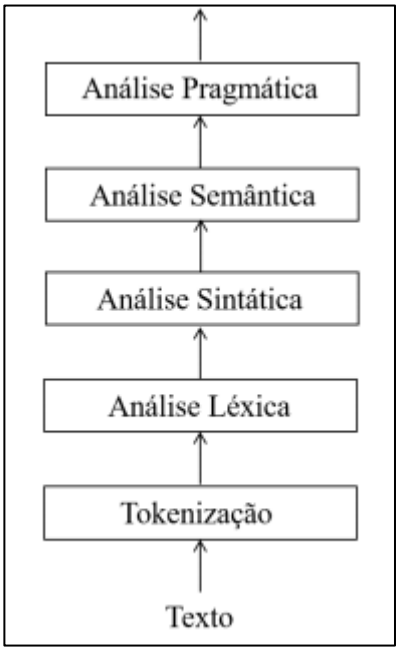
Segundo Indurkhya & Damerau (2010 *apud* Barbosa 2017, p. 336), NLP pode ser definido como uma área da computação que tem como objetivo retirar partes de representações, trazendo a elas significados.

De acordo com Indurkhya & Damerau (2010 *apud* Barbosa (2017, p. 337) o processamento de linguagem natural se utiliza de vários conceitos linguísticos em forma de classes de palavras, como por exemplos os substantivos, verbos, adjetivos, entre outros, que são chamadas de *Part-Of-Speech*, além de ter algumas estruturas gramaticais. Dentre as várias vertentes nas quais o NLP é capaz de apontar, alguns exemplos de aplicações que atendem as necessidades do senso comum utilizando NLP são a produção e resumos de textos, marcações

em textos, reconhecimento de entidades nomeadas, chatbots, reconhecimento de voz, entre outras (GOYAL, PANDEY E JAIN, 2018, p. 18).

Com base no que dizem Goyal, Pandey e Jain (2018), o NLP também pode lidar, através de várias representações de conhecimentos, com situações que envolvem mais complexidades, como uma anáfora e até ambiguidades, exemplos léxicos das palavras e até seus significados, propriedades deles e as regras que envolvem a gramatica da linguagem. Podemos notar os estágios da análise no NLP na figura 10, abaixo.

Figura 10 - Estágios da análise no NLP.



Fonte: Dale, 2010.

2.9. **Tokenização**

Segundo Palmer (2010 *apud* Barbosa, 2017, p. 338), em computação, as palavras são tidas como *tokens*. A tokenização ou segmentação das palavras, permite ao algoritmo a ação de quebra de uma sequência de caracteres que estão presente em um texto e localização do limite que cada palavra, ou seja, os devidos pontos onde aquela palavra termina e quando a outra começa.

Complementando a Ideia apresentada por Palmer (2010 *apud* Barbosa, 2017) com as ideias de Goyal, Pandey e Jain (2018) e fazendo uso da biblioteca NLTK pode-se criar um token simples e rápido, como o apresentado do código abaixo.

```
Import nltk
#Tokenização
sent_ = “ Estou quase no fim da linha”
tokens_ = nltk.word_tokenize(sent_)
tokens_
>>> [‘Estou’, ‘quase’, ‘no’, ‘fim’, ‘da’, ‘linha’]
```

2.10. **Análise Textual**

- Analise Léxica: De acordo com Barbosa (2017), a análise léxica pode ser definida como a atividade de relacionar as variantes morfológicas aos seus lemas, ou seja, a forma canônica das palavras ou como elas estão sendo definidas nos dicionários.

- Análise Sintática: Segundo Barbosa (2017), a análise sintática trabalha com o significado das frases, que podem ser expressas por uma proposição, uma ideia ou um pensamento, mas com isso a análise tem que extrair o significado e verificar se as estruturas estão corretas.

- Análise Semântica: Segundo Goddard & Schalley (2010 apud Barbosa, 2017, p. 342) a análise semântica pode ser entendida como uma análise do significado que as palavras retornam, expressões fixadas que ela define em texto, sentenças inteiras que são construídas a partir daquelas palavras, com isso na prática pode ser feita a tradução das expressões originais em um tipo de metalinguagem que pode ser definido.

- Análise Pragmática: Segundo Carbonell & Hayes (1987 apud BARBOSA, 2017, p. 343), Minker (1998 apud BARBOSA, 2017, p. 343) e Barker (1998 apud BARBOSA, 2017, p. 343) algumas estruturas mais utilizadas na análise pragmática são as gramáticas baseadas em casos, que podem ser definidas como gramáticas semânticas não-terminais para formar padrões, com isso, caso uma frase expressa encaixe nos padrão definidos nas outras análises, ela poderá ser reconhecida e ser selecionada para o contexto que está sendo analisado.

2.11. **Bibliotecas para Natural Language Processing**

Para um melhor desenvolvimento do assunto vale ressaltar o uso de bibliotecas especiais e específicas para NLP em Python, como é o caso da NLTK[11], TextBlob[12], SpaCy[13], Gensim[14], Pattern[15]>.

- **NLTK:** É um dos pacotes mais comuns e utilizados, trabalha desde a corpora, categorização de textos, estrutura de análise linguísticas e outras (GOYAL; PANDEY; JAIN, 2018, p. 20)

- **TextBlob:**, é uma biblioteca do Python utilizada para o processamento textual dos dados. Outro fator desta biblioteca é que ela oferece uma API bem simples, para o processamento profundo das operações do NLP (GOYAL; PANDEY; JAIN, 2018, p. 22).

- **SpaCy:** Proporciona uma rápida análise sintática precisa (a biblioteca mais veloz em relação as demais), possui entidades de reconhecimento de entidades nomeadas além de pronto acesso a vetores de palavras (GOYAL; PANDEY; JAIN, 2018, p. 25)

- **Gensim** : Outra biblioteca crucial, que é utilizada primariamente para a modelagem de tópicos. (GOYAL; PANDEY; JAIN, 2018, p. 27)

- **Pattern** : Utilizada em uma variedade de tarefas do NLP, reconhecimento de voz, *n-grams search*, análise de sentimentos e WordNet e *machine learning*, tais como espaço vetorial, modelagem, *k-means clustering*, *K-nearest neighbors* (K-NN) e máquinas classificadoras de suporte à vetores (SVM) (GOYAL; PANDEY; JAIN, 2018, p. 29).

[11] www.nltk.org/
[12] http://textblob.readthedocs.io/en/dev/index.html
[13] https://spacy.io/
[14] https://radimrehurek.com/gensim/
[15] https://pypi.python.org/pypi/Pattern

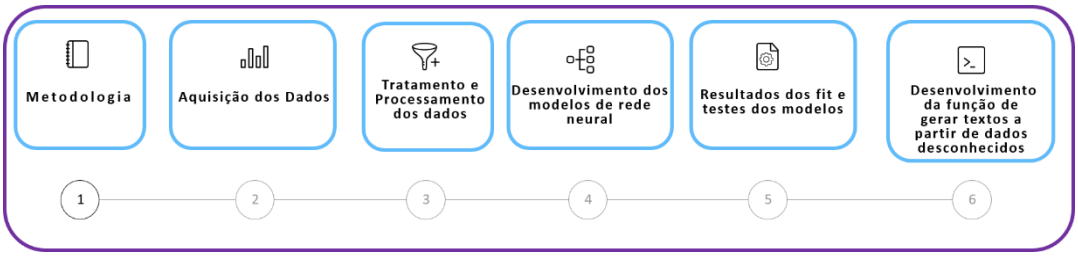
3. METODOLOGIA

A definição de um método é crucial em qualquer proposta de projeto, ainda mais para a construção de uma inteligência artificial, pois nessa seção que se explicita como ela será desenvolvida e quais ferramentas utilizadas para atingir os objetivos.

Primeiramente foi desenvolvido o levantamento teórico em relação a buscas e produções, abrangendo os principais conceitos necessários para o desenvolvimento deste trabalho de conclusão de curso, sendo os mesmos: Inteligência Artificial, *Machine Learning*, Redes Neurais, *Deep Learning* e por fim *Natural Language Processing*. Foi observado as características que definem cada conceito como o próprio além de levar em consideração o papel e impacto de cada um na proposta do projeto.

O desenvolvimento do trabalho foi dividido em 6 etapas, como é demonstrado na figura 11:

Figura 11 - Diagrama de Blocos do Desenvolvimento da CIENC.IA.



Fonte: Autores.

Metodologia: Como foi dito anteriormente a Metodologia será o alicerce deste projeto, com conceitos concisos e claros, evitando ambiguidade dos dados. A metodologia procura a definir etapas necessárias e condições específicas para que seja possível caminhar para a próxima fase do projeto. Condições definidas como a linguagem, modelos neurais a serem utilizados, que o projeto CIENC.IA.

Aquisição dos Dados: Momento no qual será realizado a aquisição dos *datasets* para alimentar a Inteligência artificial, momento caracterizado pela aquisição dos dados, estes que serão materiais de estudo e tratamento nos processos seguintes.

Tratamento e Processamento dos dados: Uma vez que o *dataset* foi adquirido, ocorrerá o tratamento e o processamento do mesmo, etapa está especializada em retirar as impurezas de formatações do *datasets* e caracteres especiais que a própria inteligência artificial não fará uso. Também nesta fase a mesma é responsável em transformar os dados em uma estrutura de dados

específica para que seja possível a inteligência artificial processar a linguagem natural do ser humano, ou seja transformar os dados em que um certo momento são palavras e textos para uma sequência de textos.

Desenvolvimento dos Modelos de Rede Neural: Etapa focada em fazer a separação das *features* e *labels* que serão utilizadas como material de treinamento do modelo neural. Em seguida será realizada a criação do modelo de rede neural utilizando uma determinada combinação de redes neurais disponíveis nas bibliotecas do Keras, além da definição de parâmetros para cada camada.

Resultados dos Fit e testes dos Modelos: Representação dos Resultados do model.fit de cada modelo desenvolvido para ser testado em conjunto com os devidos apêndices representando cada resultado dos treinamentos.

Desenvolvimento da Função de gerar textos a partir de dados desconhecidos: Etapa onde será desenvolvido o código em si que agrupa os resultados de todas as etapas anteriores e rodar em um script que será responsável em gerar textos a partir de um “input” do usuário. Tendo em nota as delimitações e condições necessárias para as entradas.

No capítulo seguinte será dado início ao desenvolvimento do agente inteligente CIENC.IA, conforme foi apresentado as etapas metodológicas estabelecidas neste capítulo.

4. DESENVOLVIMENTO

O processo de desenvolvimento e o treinamento de um agente inteligente, não consiste apenas em alimentar o algoritmo da Inteligência artificial e abandonar o mesmo. Para desenvolver um agente inteligente capaz de gerar textos de acordo com parâmetros desejados pelo usuário, requisitou uma compreensão intermediária das fundamentações teóricas entorno de todos os conceitos que englobam o assunto de IA, portanto, a etapa de fundamentação teórica constitui um peso perante o desenvolvimento e os resultados do "Projeto CIENC.IA".

O desenvolvimento deste módulo "Produção automática de textos" do "Projeto CIENC.IA" foi dividido em 6 etapas podendo as mesmas possuir sub etapas ou não para um melhor entendimento do processo como um todo, etapas estas que são classificadas em:

- Metodologia
- Aquisição dos dados.
- Tratamento e Processamento dos dados
- Desenvolvimento dos modelos da rede neural
- Resultados do fit e testes dos modelos.
- Desenvolvimento da função de gerar textos a partir de dados desconhecidos.

Primeiramente de forma intrínseca para o desenvolvimento de uma inteligência artificial, foi realizado um levantamento teórico sobre os principais conceitos específicos do ramo, como *machine learning, natural language processing, artificial neural network, deep learning* e suas diversas tendências, a fim de alcançar as metas em relação ao problema levantado.

Após ter sido realizado o levantamento teórico dentre asas técnicas e conceitos levantados para desenvolver uma IA, responsável pela produção de textos automáticos tendo em vista que a especialidade e funcionalidade dos métodos apresentados deverá ser capaz de atender ao propósito do sistema, verificação esta que será baseada na necessidade do uso para prosseguir com o desenvolvimento do agente inteligente que terá o propósito produzir textos.

A priori a busca dos dados científica foi feita na internet levando em consideração os parâmetros predefinidos. Em sequência os agentes inteligentes serão desenvolvidos na linguagem de programação Python. Já a produção de textos será baseada nos estudos de processamento de linguagens naturais (NLP) concomitantemente com aprendizagem de máquina dando apoio aos objetivos de desenvolvimento do texto. A Cienc.ia (agente

inteligente) será treinada a ponto de atingir uma destreza em seus resultados, resultados estes com o propósito de obter a melhor precisão da rede neural e consigo a produção de textos coerentes.

Em seguida, o desenvolvimento do agente inteligente responsável em gerar textos de forma automática, será desenvolvido em um ambiente simulado com um conjunto de dados selecionados sobre um referente tema, sendo este em questão *Deep Learning* de forma manual, por questões didáticas e para um desenvolvimento modular do projeto. Para atingir tal feito, as tecnologias selecionadas para desenvolver o módulo de produção de textos da agente inteligente CIENC.IA, são as seguintes:

- Python será a linguagem de programação para o projeto devido à experiencia dos membros com a mesma além de sua vasta quantidade de bibliotecas, facilitadores e pesquisas científicas para servir de material de apoio para o projeto. Em contraposição a linguagem R (Linguagem concorrente) preocupar majoritariamente com questões estatísticas e numéricas a mesma não será utilizada neste módulo.

- Para realizar os processamentos e tratamentos dos dados, foi selecionada as bibliotecas Spacy por realizar o NLP, contendo em seu "kit", métodos de tratamento de informação, limpeza e vocabulários, Numpy para uma melhor organização dos dados em uma matriz e para facilitar o "*hot-encoding*" da informação facilitando o processo de predição em *machine learning*, Keras atuará no pre-processamento dos dados atuando na "tokenização" das strings e elaboração das "features[16]" e "*labels*[17]".

- Para o desenvolvimento do modelo de rede neural, também será utilizado **Keras** para realizar tal tarefa, uma vez que a mesma tecnologia possui implementada em sua biblioteca diversas ferramentas, sendo algumas delas como métodos de criação de modelos, formulas de ativação de camadas

[16] *Features*: O termo *feature* que do inglês significa funcionalidade, porém para IA o mesmo significa o conjunto de dados identificados como objeto que será fornecido para IA "estudar"e em seguida realizar futuras predições sobre os mesmos.As features são geralmente identificadas como "X" por questões de boas práticas de programação.

[17] *Label*: O termo *label* significa rótulo, rótulo este que estará indicando o resultado esperado de determinada predição de um conjunto de testes sob uma determinada *feature*. Por outro lado os *labels* são comumente identificados como "Y" por questões de boas práticas de programação.

ocultas, todas estas pré-construídas para uma melhor experiência de programação, consequentemente será o “construtor das redes neurais”.

A Cienc.ia (agente inteligente) será treinada a ponto de atingir uma destreza em seus resultados, resultados estes com o propósito de obter a melhor precisão da rede neural dentre os modelos treinados e consigo a produção de uma asserção com sentido.

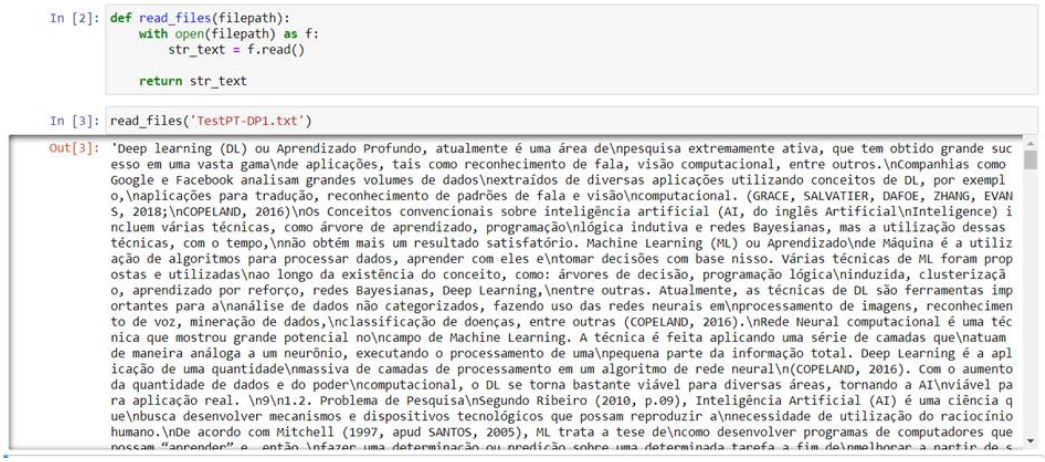
4.1. **Aquisição dos dados**

O agente inteligente CIENC.IA irá contar com um script de busca automática de textos na internet, que será de acordo com a escolha do usuário. Porém para a etapa de desenvolvimento do módulo de produção de textos, o mesmo atua de forma independente do script de busca textual, seguindo uma arquitetura modular.

Os dados selecionados para alimentar a Inteligência Artificial é um compilado de artigos científicos, referentes ao tema de *Deep Learning*, como pode ser visto no ApêndiceA.

Portanto como foi explicado anteriormente na metodologia, o processo será simulado de forma manual no qual o arquivo contendo os textos já tenha sido enviado para o módulo de Produção de textos. Como podemos ver no método “*read_files( )*” desenvolvido a seguir na figura 12:

Figura 12 - Método de leitura do arquivo.



Fonte: Autores.

Como pode ser visto na figura 12, a aquisição dos dados foi realizada de forma simples e direta para o algoritmo que será responsável em estudar este texto e “aprender” sobre o mesmo. Ficou claro que o arquivo TestPT-DP1.txt possui algumas formatações dos arquivos de texto da fonte retirada, como quebras de linhas, negritos, sinais e pontuações não desejadas em seu conteúdo, fatores estes que será realizado o tratamento na etapa de tratamento e processamento de dados, a seguir.

4.2. **Tratamento e Processamento dos dados.**

Nesta etapa, para o desenvolvimento do algoritmo de produzir textos, foi realizada as seguintes tarefas para tornar os dados adquiridos legíveis para a IA:

- Tratamento do Texto, Processamento do Texto e Tokenização;
- Sequenciamento dos textos com Keras.
- Pré-processamento dos tokens com Keras.

4.3. **Tratamento do Texto, Processamento dos dados e Tokenização**

Observando anteriormente na figura 12. Foi possível notar “impurezas” nos dados coletados, fator este que necessita ser tratado para propósitos de treinamento da IA, para que não influencie os resultados finais.

Lembrando da consideração de Vasilev que os *datasets* somente incluir um valor finito e limitado de números de variáveis além de haver diversas outras variáveis fora de nosso controle (Vasilev et al, 2019).

Primeiramente ocorrerá uma importação das bibliotecas de NLP, no caso a utilizada é a Spacy e seu conjunto “Portuguese” (PT), para ser possível o tratamento e a tokenização do texto como um todo, sendo este processo representado na figura 13:

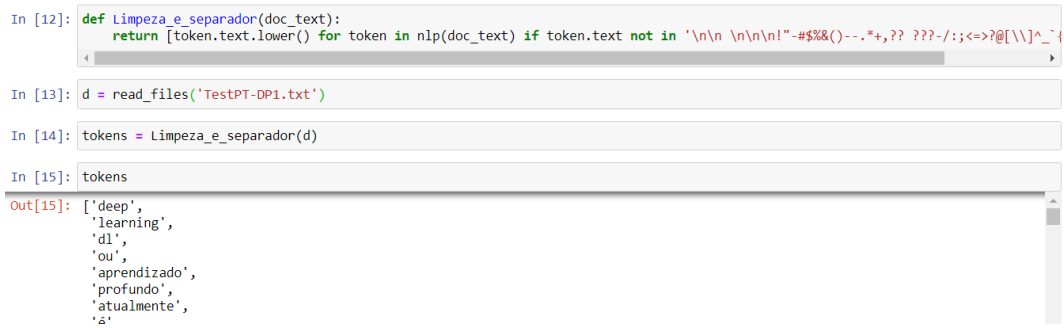
Figura 13 - Importação da Biblioteca Spacy para NLP.

```
In [4]: #Processo de Limpeza e preparo dos dados para tokenização
In [5]: import spacy
In [6]: nlp = spacy.load('pt_core_news_lg',disable=['parser','tagger','ner'])
In [7]: #nlp = spacy.load('pt',disable=['parser','tagger','ner'])
In [8]: nlp.max_legth = 1198623 # A quantidade aqui deve ser = ou > com a quantidade de caracter no .txt lido
```

Fonte: Autores.

Em seguida foi necessário a definição de um método responsável pela limpeza das “impurezas” do *dataset*, semelhante à um *regex* em expressões regulares, fator este representado nesta linha de código a seguir na figura 14.

Figura 14 - Métodos de limpeza e separação do texto em tokens.



Fonte: Autores.

O método em questão realiza as seguintes operações com o texto que será utilizado como argumento:

- Diminui as letras para caixa baixa;
- Realizará uma iteração por todo o documento e transformará cada palavra em token se o mesmo estiver dentro da condição de não ser o seguinte "*regex*": '\n\n \n\n\n!"-#$%&()--.*+,?? ???-/:;<=>?@[\\]^_`{|}~\t\n - - '

Em seguida na figura 14, realiza a leitura do *dataset* e aciona o método Limpeza_e_separador(d), transformando todos os caracteres do *dataset* e toda a informação como tokens. Também é possível observar que os comandos de formatação do texto não se encontram mais no conjunto.

4.4. **Sequenciamento dos textos com Keras**

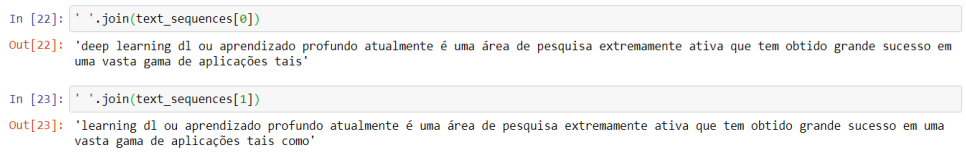
Como o objetivo é realizar uma IA capaz de gerar textos então temos que preparar o método que a mesma estará aprendendo tal tarefa, com apoia da biblioteca Keras iremos criar um modelo de *machine learning* juntamente com as redes neurais capaz de predizer a próxima palavra em uma asserção. Portanto foi feita a decisão de um valor de máximo de tokens que a IA irá abstrair por vez, caso contrário poderá ocorrer um *overfitting*, segundo Vasilev a importância de selecionar *features* válidas e evitar que o aprendizado da máquina memorize os dados, este termo é referenciado como *overfitting (2019*),complementando Raschka & Mirjalili, que o ato de um modelo sofrer de *overfitting* o mesmo será muito bom em predizer os dados de treinamento que ao ser colocado em prática com dados não vistos terá um desempenho inferior do que o esperado.

Desta forma foi considerado um valor de 25 palavras que serão passados para a rede neural ler e a mesma terá como objetivo predizer a vigésima sexta palavra, garantindo que não seja pouca palavra para a rede ler, mas também para que ela tenha noção do conteúdo e de como é construída uma frase no geral.

Para o sequenciamento dos tokens e o processo de delimitação do mesmo para um limite de 25 palavras, que ao modificar o index da lista teremos as mesmas 25 palavras, porém 1 token a frente, ou seja, se uma frase começar com “Deep Learning significa” a próxima lista terá o seguinte resultado “Learning significa em”, sendo assim material para ser utilizado como treinamento da IA durante a etapa de desenvolvimento do modelo de rede neural.

Foi optado uma lista para desenvolver estas *features*, com o conjunto de tokens que será tratada e organizada com algumas formatações de texto no final teremos o seguinte resultado. Como podemos observar na figura 15. A lista de palavras da primeira e segunda sequência, a seguir:

Figura 15 - A lista de palavras da primeira e segunda sequência.

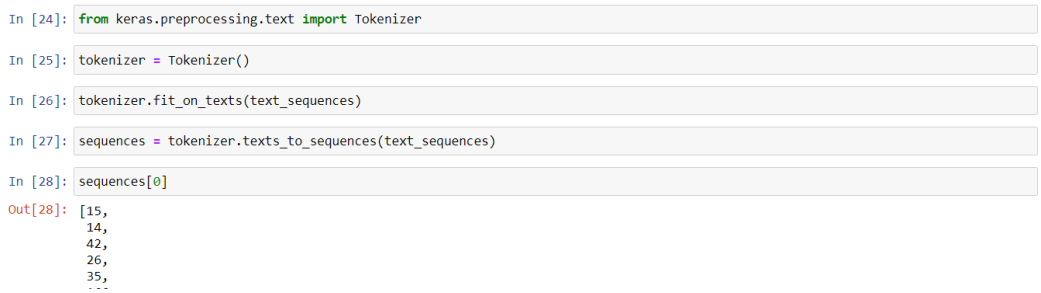


Fonte: Autores.

4.5. **Preprocessamento dos tokens com Keras**

Nesta etapa teremos que formatar os tokens dos resultados representados na figura 15, para um sistema numérico para que o “tokenizador” do Keras seja capaz de operar. Fato este que a própria biblioteca do Keras cobre com a importação na figura 16.

Figura 16 - Tokenizer.



Fonte: Autores.

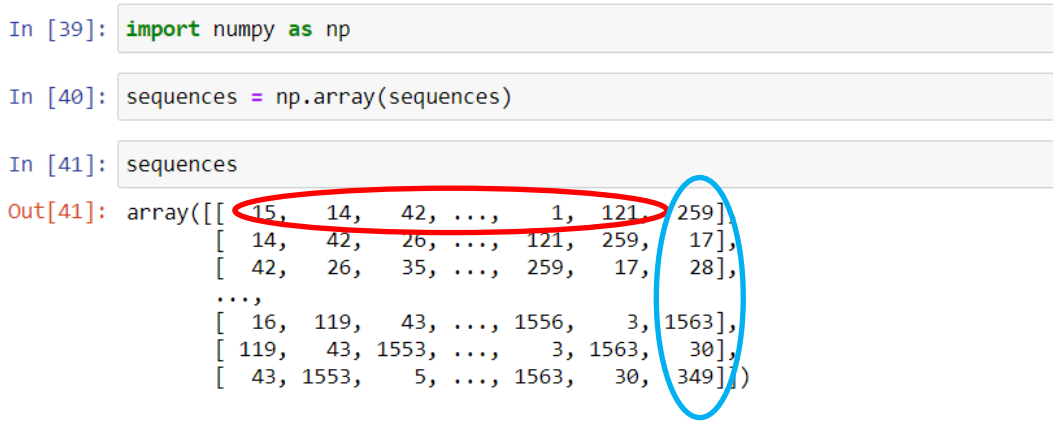
Podemos observar que na linha de entrada 24, foi realizada a chamada do "Tokenizador", para realizar a transformação dos tokens de textos em tokens numéricos, facilitando a manipulação do mesmo dentro do conjunto de ferramentas de manipulação de dados do Keras.

Em seguida foi realizada a definição de algumas variáveis que serão de grande importância para o resto do algoritmo, sendo elas a lista "sequences" na linha de entrada 27 e o inteiro tam_vocabulario, que recebe a seguinte definição "tam_vocabulario = len(tokenizer.word_counts) ". Também vale notar que cada palavra que fazia parte da sequência agora se tornou um número, este que é um ID único para cada palavra.

Caso esteja curioso em saber a quantidade de vezes que uma determinada palavra se repete em seu *dataset* basta realizar o comando tokenizer.word_counts **,** no conjunto deste *dataset* temos as palavras *deep* repetindo 1197 vezes e *learning* um total de 1224.

Em seguida os dados da sequência serão convertidos em uma matriz com apoio da biblioteca Numpy, para facilitar a definição das *features* e *labels*, representado na figura 17 Conversão para matriz:

Figura 17 - Conversão para matriz, features e labels.



Fonte: Autores.

Conforme representado na figura 17, a matriz resultante é a frase que estará sendo alimentada para o modelo, o conjunto circulado de vermelho representa a frase de treinamento as *features* X e o conjunto circulado de azul os *labels* Y.

4.6. **Desenvolvimento dos modelos da rede neural**

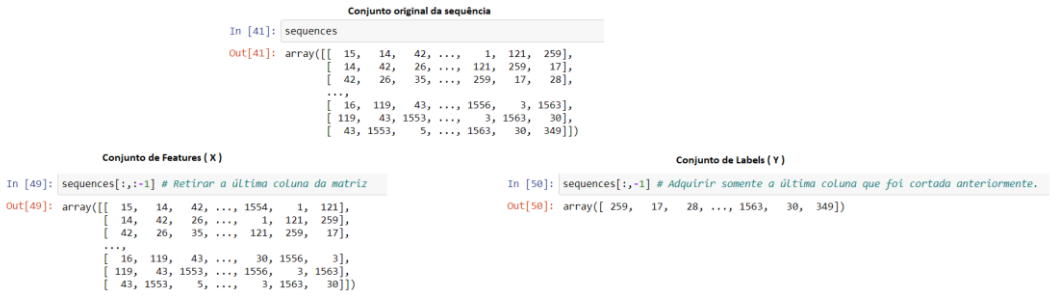
Para a etapa de desenvolvimento dos modelos de rede neural estaremos fazendo uso do Keras este que por sua vez foi desenvolvido de tal forma que torna a construção de modelos de deep learning ainda mais convenientes relata Rashcka e Mirjalili (2017, pg.418). Além disso foi considerado fatores como o ato da inteligência artificial esquecer o que lhe foi ensinado, para tal tarefa estaremos desenvolvendo não só um modelo de rede neural recursiva, mas também uma LSTM (Long-short term memory) que é uma rede neural que simula as nossas células de memória de longo e curto prazo de armazenamento de memória. Como objetivos desta etapa de desenvolvimento temos as seguintes etapas:

- Separar os dados entre features e labels, sendo X as *features* (as primeiras 25 palavras da sequência) e Y as *labels* (Próxima palavra da sequência)
- Criar um modelo de rede neural baseado nas LSTM's
- Por fim o model fit, no qual é a análise dos resultados da rede em seu desenvolvimento.

4.7. **Features & Labels**

Para realizar a aquisição das features e labels que foram expostas na figura 16, foi realizado um corte da matriz fazendo uso da notação do numpy para matrizes, ou seja as primeiras colunas até a última estará sendo preservada e a última coluna será cortada do conjunto para X e o inverso será aplicado para o label, sendo assim então armazenada somente a última coluna para Y. Como podemos observar nas seguintes matrizes que serão atribuídas para as suas respectivas variáveis na figura 18 abaixo:

Figura 18 - Original X *Features* X *Labels*



Fonte: Autores.

Com o uso da utilidade que está presente na biblioteca Keras (“keras.utils import to_categorical”), somos capazes de realizar a categorização de das labels de forma adequada em conjunto com todo o vocabulário presente no texto fornecido inicialmente. Por outro lado foi observado após utilizar o comando X.shape que o conjunto de features(sequências para testes) foram 5636, sequencias de 25 palavras.

4.8. **Criação do Modelo Baseado em LSTM**

O desenvolvimento de um modelo de rede neural, consiste nas decisões de necessidade do projeto, para realizar uma mesma etapa repetidas vezes que é o caso das sequencias de palavras, pré estabelecidas de tamanho 25 tokens, fica evidente que o conceito de recursividade é crucial para atender o proposito do mesmo, outro fator crucial na questão do desenvolvimento de redes neurais são as camadas dado este que Chollet (2018) reforça:

> Diferentes composições de camadas são apropriadas para diferentes formatos de tensores de dados para serem processados. Tomamos por instância um simples vetor de dados num formato 2d de dados (amostra, features), tensor este que é geralmente processado por camadas de densidade [...] Dados sequenciais, tensores 3d de forma, como (amostra, tempo e features), são tipicamente processados por camadas recorrentes como as camadas LSTM. Chollet (2018, pg58)

Para o desenvolvimento das camadas o Keras possui algumas das mesmas pré moldadas para fácil implementação, como podemos observar na figura 19 a seguir a criação das camadas do modelo sendo elas camadas de densidade, LSTM e *Embedding* além dos parâmetros aderidos para a mesma além do modelo de compilação e método de optimização.

Figura 19 - Criando Modelo e aderindo camadas e parâmetros

Fonte: Autores.

A camada de *Embedding*, geralmente é utilizada como a primeira camada para construção de redes neurais está sendo responsável em adquirir a matriz de sequencias e colocar as mesmas dentro de um vetor denso (limitador) de valores entre 0 e 1.

Em seguida temos as camadas de LSTM, com os seus “neurônios” de memória, realizando o armazenamento da informação caso haja a necessidade da mesma.

Adiante há uma camada de densidade com a função de Ativação *Rectified Linear Unit* (ReLU), que é uma função relativamente simples que procura passar a equação z, pela função MAX (0,z), ou seja o *max* de z verifica se o valor de Z é positivo ou negativo , se o valor de z é negativo a função sempre retornará 0, complementa Chollet, ReLU é uma função que tem como seu significado zerar os valores negativos (2018, pg71), se for o contrário ele será

exatamente o valor de z que é diferente de 0. Tende a ser a função com as melhores performances em diversas situações.

Por fim será implementada uma camada de densidade para realizar a tradução dos números para uma palavra do vocabulário de output da informação tendo como função de ativação *softmax*, além dos modelos de compilação do mesmo e seus parâmetros de perda que no caso é a categorização de entropia cruzada que trata cada categoria como única para cada palavra, otimizado 'Adam' e por fim as métricas finais filtradas pela acurácia.

Como resultado podemos observar na figura 20, 21 e 22 a seguir o sumário de nosso primeiro modelo desenvolvido, além de outros dois que seguiram os seguintes nomes: Model1, Model2, Model3.

Figura 20 - Sumário Modelo 1

```
In [58]: model1 = create_model(tam_vocabulario+1,seq_len)

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 25, 25)            39100
_________________________________________________________________
lstm (LSTM)                  (None, 25, 50)            15200
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 50)                2550
_________________________________________________________________
dense_1 (Dense)              (None, 1564)              79764
=================================================================
Total params: 156,814
Trainable params: 156,814
Non-trainable params: 0
_________________________________________________________________
```

Fonte: Autores.

Figura 21 - Sumário Modelo 2

```
In [89]: model2 = create_model(tam_vocabulario+1,seq_len)

Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 25, 25)            39100
_________________________________________________________________
lstm_10 (LSTM)               (None, 25, 75)            30300
_________________________________________________________________
lstm_11 (LSTM)               (None, 75)                45300
_________________________________________________________________
dense_10 (Dense)             (None, 75)                5700
_________________________________________________________________
dense_11 (Dense)             (None, 1564)              118864
=================================================================
Total params: 239,264
Trainable params: 239,264
Non-trainable params: 0
_________________________________________________________________
```

Fonte: Autores.

Figura 22 - Sumário Modelo 3

```
In [108]: model3 = create_model(tam_vocabulario+1,seq_len)

Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, 25, 25)            39100
_________________________________________________________________
lstm_22 (LSTM)               (None, 25, 100)           50400
_________________________________________________________________
lstm_23 (LSTM)               (None, 100)               80400
_________________________________________________________________
dense_22 (Dense)             (None, 100)               10100
_________________________________________________________________
dense_23 (Dense)             (None, 1564)              157964
=================================================================
Total params: 337,964
Trainable params: 337,964
Non-trainable params: 0
_________________________________________________________________
```

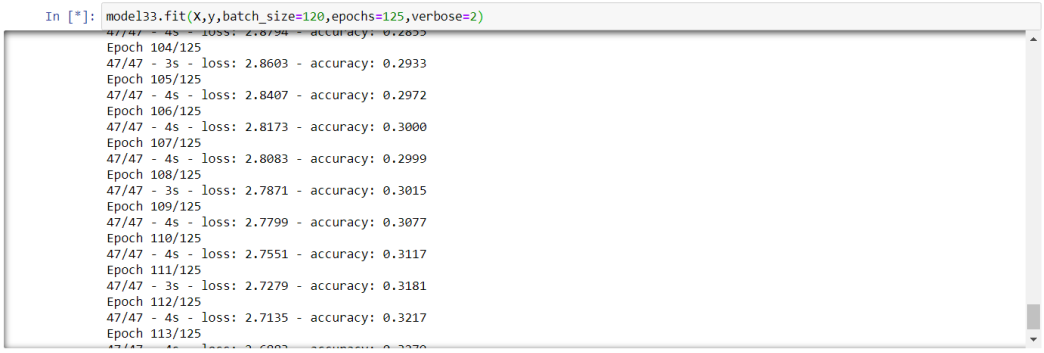
Fonte: Autores.

4.9. **Model Fit**

Model Fit é a etapa de treinamento do modelo, sendo transferido um determinado valor de dados para o processamento de cada iteração, no qual o mesmo realiza uma sequência de "*epochs*" termo que se refere à quantidade de vezes que o mesmo deverá realizar os testes, ou seja a dita a quantidade de repetições de um modelo.

Dentre os modelos apresentados os treinos foram realizados entre 25 a 125 *epochs*, todos seguindo do mesmo conteúdo. Na figura 23 podemos observar um exemplo deste processo:

Figura 23 - Model Fitting



Fonte: Autores.

4.10. **Resultados do model fit e dos testes**

Os resultados é também uma etapa de análise crucial para o desenvolvimento de um agente inteligente, como proposto pela metodologia, foi observado a necessidade do estudo de ante mão dos resultados da rede neural.

Resultados estes que podem ser observados como inferentes aos modelos propostos para treinamento, sendo os mesmos Model1 para o "ApêndiceB", Model2 para o "ApêndiceC" e por fim Model3 para o "ApêndiceD"

Após terem sido observado os resultados quanto às perdas e acurácia da rede podemos verificar se é necessário ou não uma repetição do processo de treinamento da rede, ou a utilização de um novo conjunto de datasets, para explorar os limites da mesma. Na próxima etapa do desenvolvimento será representado o método responsável em gerar textos por meio de entradas desconhecidas digitadas pelo usuário.

4.11. **Desenvolvimento do método de gerar textos automáticos**

O desenvolvimento de textos automáticos consiste em um método que adquiri os resultados de saída da rede neural em conjunto com o vocabulário do *dataset*, e em seguida o mesmo é argumentado com um novo dado de entrada desconhecido para o conjunto, sendo este dado uma pergunta “O que é *deep learning* para o ser humano no século XXI ? ”.

O método faz uso de um utilitário da biblioteca do Keras para auxiliar a indexação das sequencias pela importação pad_sequences. O output do gerador de texto irá girar em torno de uma lista de palavras que são consideradas semente e em seguida será realizada uma integração dos dados de saída da IA, juntamente com os tokens pré estabelecidos anteriormente. Como parâmetros destes métodos temos o modelo treinado, o tokenizador, o tamanho da sequência, dado de entrada e por fim o número de palavras a serem geradas. Como demonstrado na figura 24:

Figura 24 - Método de Gerar texto

```
In [ ]: def generate_text(model,tokenizer,seq_len,seed_text,num_gen_words):
            output_text = []

            # 25 palavras
            input_text = seed_text

            for i in range(num_gen_words):

                encoded_text = tokenizer.texts_to_sequences([input_text])[0]

                pad_encoded = pad_sequences([encoded_text],maxlen=seq_len,truncating='pre')

                pred_word_ind = model.predict_classes(pad_encoded,verbose=0)[0]

                pred_word = tokenizer.index_word[pred_word_ind]

                input_text += ' '+pred_word

                output_text.append(pred_word)

            return ' '.join(output_text)
```

Fonte: Autores.

Na linha onde está falando sobre o encoded_text, será tratado o texto inserido para acompanhar a sequência dos textos anteriores, em seguida o pad_encoded, onde irá ocorrer o tratamento da palavra em casos da palavra de entrada não for do tamanho correto, passar o limite nesta etapa ocorrerá o corte, ou seja, o pad_encoded está sendo responsável em garantir que a nossa entrada combine com o padrão dos nossos dados de entrada. Adiante ocorre a predição da próxima palavra onde é analisado no banco dos resultados da predição qual seria a % de determinada palavra se a seguinte no modelo depois desta predição seria passada o index do mesmo para pred_word, que iria fazer a transcrição do token da palavra. Então o dado de

entrada estaria acompanhado de um espaço em branco + a palavra da predição, tendo a mesma anexada como um apêndice ao vocabulário final.

Portanto levando em consideração os vários meios que pode ser realizado o input_text que será o seed_text para a função de gerar texto, podemos utilizar o exemplo da figura 25 como referência:

Figura 25 - Chamando a função

```
seed_text = "What is Deep Learning?"
generate_text(model3,tokenizer,seq_len,seed_text=seed_text,num_gen_words=20)
generate_text(model2,tokenizer,seq_len,seed_text=seed_text,num_gen_words=20)
generate_text(model3,tokenizer,seq_len,seed_text=seed_text,num_gen_words=20)
```

Fonte: Autores.

Durante a execução do trabalho, foram concluídas a etapa do referencial teórico acerca do assunto de Inteligência Artificial, desde máquinas de aprendizagem a aprendizagem profunda. Em seguida foi realizado o desenvolvimento dos scripts para a produção dos textos de forma automática, na qual serão gerados textos a partir de uma entrada de dados desconhecida pelo modelo

Referente ao modelo de produção de texto automática foram executados testes entre 3 modelos em português e 1 modelo em inglês, porém com um *dataset* diferente e parâmetros diferentes, estaremos utilizando o mesmo somente para representar a possibilidade da utilização do código em idiomas diferentes. Os Modelos em português possuem cada um deles parâmetros diferentes entre as camadas da rede neural e a quantidade de vezes que o mesmo modelo necessita iterar sobre o seu conjunto de testes, para atingir uma determinada acurácia, dados estes representados na tabela 1 abaixo:

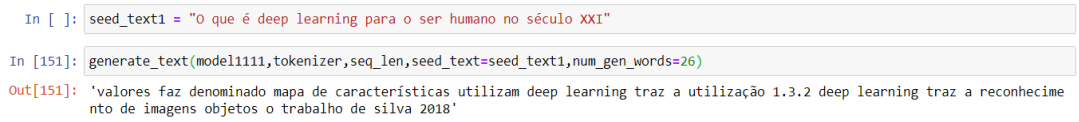
Tabela 1 - Testes de Acurácia

Modelos	Parâmetros das camadas	Epochs = 25	Epochs = 125	Epochs = 250	Epochs = 500
1	50	11,09%	39,78%	74,56%	98,46%
2	75	10,63%	34,26%	69,52%	98,67%
3	100	11,48%	35,84%	73,58%	99,98%

Fonte: Autores

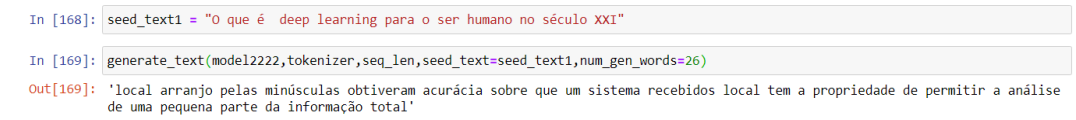
Podemos observar que o modelo 3, com 500 *epochs*, atingiu os melhores resultados em questão de acurácia. Os mesmos modelos foram analisados para gerar textos de forma automática a partir de textos de entradas que possuíam como temática o termo: "*Deep Learning"*. A frase dos respectivos de melhor acurácia dos modelos 1, 2 e 3 podem ser observados nas respectivas imagens 26, 27 e 28:

Figura 26 - Resultados Modelo 1 – 98,46%- 26 palavras



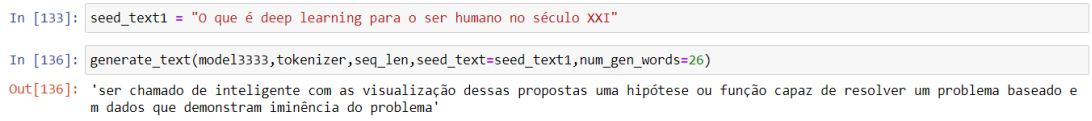
Fonte: Autores.

Figura 27 - Resultados Modelo 2 – 98,67% - 26 palavras



Fonte: Autores.

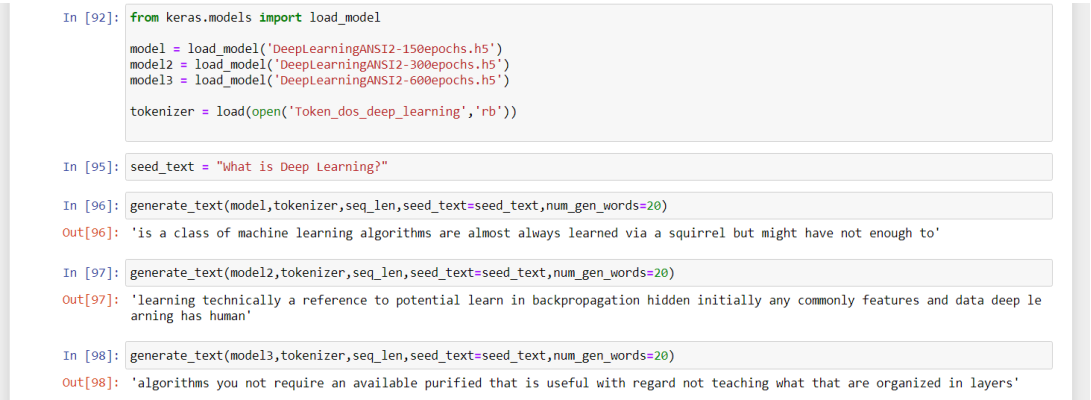
Figura 28 - Resultados Modelo 3 – 99,98%- Gerando 26 palavras



Fonte: Autores.

Os testes entre os modelos 1,2 e 3 também atingiram resultados dentro das expectativas, produzindo um texto coerente e com poucos erros de sintaxe e semântica como podemos observar no modelo 3. Já o modelo 1 apresentou algumas falhas gramaticais e repetições, sem contar a impureza da enumeração da pesquisa além de uma referência avulsa no texto.

Figura 29 - Resultados Modelos em Inglês – 43%, 84%,100%- Gerando 20 palavras



Fonte: Autores.

Dentre os resultados apresentados a semântica americana na figura 29, atingiu resultados satisfatórios mesmo sob um *dataset* limitado com valores abaixo de 90%, os textos produzidos anexam palavras chaves quanto ao tema questionado.

5. CONSIDERAÇÕES FINAIS/CONCLUSSÃO

O projeto CIENC.IA como um todo possui uma proposta bastante visionária e ambiciosa, querendo não só mudar o jeito de estudar e sim amplificar a capacidade humana de observar novas situações por diferentes perspectivas. O desenvolvimento do algoritmo de produção automática, nos mostrou um mundo desconhecido e único que é o da ciência de dados cheio de novidades além de soluções inovadoras não só para o mercado e sim para a vida. Pretende-se com o desenvolvimento deste sistema não só auxiliar pesquisadores, pelo contrário o mesmo trabalhará agregando qualidade. Consequentemente se espera que as novas pesquisas que ao utilizar os textos coletados ou fichados (pelo sistema), sejam capazes em aderir para ampliação das produções de cunho científico para a sociedade brasileira.

Como trabalhos futuros, foi observado alguns pontos não foram explanados durante o desenvolvimento deste trabalho de conclusão de curso, portanto vale ressaltar o quão grande é o ambiente de estudo da inteligência artificial. Levando em consideração o trabalho como um todo, ficou evidente que o projeto necessita de uma Integração do módulo de busca automática de textos com o módulo de produção automática de textos do Projeto CIENC.IA. Por outro lado, é interessante o desenvolvimento da interface de usuário para o Projeto CIENC.IA partindo das tecnologias disponíveis do mercado que envolve python, como tkinter, django dentre outras pensando na usabilidade e na experiencia de usuário.

A otimização e a regularização das camadas de redes neurais recursivas com uma seleção de outras camadas desenvolvidas por uma IA que adeque melhor que as propostas pelo projeto, também é uma vertente a se pensar. Por fim o que temos de ter em mente seria: “A que ponto ético e moral, os textos gerados por uma inteligência artificial podem chegar a infringir a lei de propriedade intelectual?”

6. REFERÊNCIAS

AGATONOVIC-KUSTRIN, S., & BERESFORD, R. **Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research.** Journal of Pharmaceutical and Biomedical Analysis, 22(5), 717–727. 2000.

AL-SHAWWA, M., AL-ABSI, A., ABU HASSANEIN, S., ABU BARAKA, K., & ABU-NASER, S. S. **Predicting Temperature and Humidity in the Surrounding Environment Using Artificial Neural Network.** International Journal of Academic Pedagogical Research (IJAPR), 2(9), 1-6. (2018).

ANTUNES, Elaine D.; BRUNETTA, Nádia; DEMARCO, Diogo J; PINHEIRO, Ivan A. **Desafios na Construção do Trabalho de Conclusão do Curso de Especialização em Negociação Coletiva / Modalidade a Distância.** UFRGS – Universidade Federal do Rio Grande do Sul. 2011.

BALDIN C. P, SCHAMBECK M. M., MATOS S. D., CRESCENCIO W., **A Inteligência Artificial na Automatização de Processos** ,2008

BITTENCOURT, João; OSÓRIO, FERNANDO. **Annef – Artificial Neural Networks Framework:** uma solução software livre para o desenvolvimento, ensino e pesquisa de aplicações de inteligência artificial multiplataforma. Anais do II Workshop sobre Software Livre, 2001

BUDUMA, N. **Fundamentals of deep learning**: designing next-generation machine intelligence algorithms. Sebastopol, CA: O'Reilly Media, 2017.

CASSETARI, Leila. **Metodos e Tecnicas de Pesquisa em Psicologia.** Editora EDICON**.** 2012

CHO, Kyunghyun. **Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.** 2014

CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 2018.

DONEDA, D. **Considerações iniciais sobre inteligência artificial, ética e autonomia pessoal**. Fortaleza, 2018.

FAVA, Rui. **Trabalho, Educação e Inteligência Artificial:** A Era do Indivíduo Versátil. Porto Alegre: Penso,232 p. 2018.

GOEDERT, Matheus L; PAULA FILHO, Pedro L; BLANCO, Daniel R. **Computação natural: conceitos e aplicações da computação inspirada na natureza**. Vol. 38 (Nº 34). 2017

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning.** [s.l.] MIT Press. In D. E.

GOYAL, P., PANDEY, S. & JAIN, K., **Deep Learning for Natural Language Processing**. Bangalore, India, Apress. 2018.

GRAHAM, Thomas. **Art made by AI is selling for thousands:** is it any good? [S. l.]: Thomas. 2018

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Upper Saddle River: Pearson, 2009.

HOCHREITER, Sepp & SCHMIDHUBER, Jürgen. **Long Short-Term Memory.** Neural Computation. 1997.

JOSHI, P. **Artificial Intelligence with Python:** Build Real-World Artificial Intelligence Applications with Python to Intelligently Interact with the World Around You. Birmingham, UK: Packt Publishing, 2017.

LECUN, Y., Bengio, Y. & Hinton, G. **Deep learning**. Nature 521, 436–444 (2015).

LIMA, I; PINHEIRO, C. A. M; SANTOS, F. A. O. **Inteligência Artificial.** Rio de Janeiro: Elsevier, 2014.

LEMOS, L. **Computação cognitiva e a humanização das máquinas**. Minas Gerais, 2017.

MACIEL, Maria; ALBAGLI, Sarita. **Informação, conhecimento e poder mudança tecnológica e inovação social**. Ed. Rio de Janeiro: Garamond, 327p. 2011.

MCCULLOCH, W.S. "**Embodiments of Mind"**, Cambridge, Massachusetts The MIT press, 1965.

MCKINNEY, W. **Data Structures for Statistical Computing in Python**, Proceedings of the 9th Python in Science Conference, p.51-56. 2010

MENDES, Raquel. **Inteligência artificial:** sistemas especialistas no gerenciamento da informação. Ci. Inf: Brasília, vol. 26, 1997.

MOREIRA, C**. Revista de Ciência Elementar**, 1(01):0006. 2013

PEDREGOSA F., VAROQUAUX G., GRAMFORT A., et al. **Scikit-learn:** Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830, 2011.

RASCHKA, S., MIRJALILI, V. **Python Machine Learning** Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2°Ed Packt BIRMINGHAM – MUMBAI, 2017.

RANJANI J., SHEELA A., MEENA K. P., **"Combination of NumPy, SciPy and Matplotlib/Pylab -a good alternative methodology to MATLAB - A Comparative analysis,"** 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), CHENNAI, India, pp. 1-5. 2019.

RUMELHART and J. L. MCCLELLAND, editors, **Parallel Distributed Processing**, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 2016.

ROSS, C; SWETLITZ, I. **IBM pitched its Watson supercomputer as a revolution in cancer care. It's nowhere close.** 2017**.**

RUSK, N. **Deep learning.** Nat Methods .2016.

RUSSEL, Stuart J.; NORVIG, Peter; **Inteligência Artificial:** Uma Abordagem moderna. Ed. Elsevier, 2013.

SANSON, Carlos. **Revolução 4.0 e a lição de Marx**. 2017.

SCHÄFER, A. M., & ZIMMERMANN, H. G. **Recurrent Neural Networks Are Universal Approximators.** Lecture Notes in Computer Science 632-640. 2006.

SOUZA, Silva. **Dificuldades encontradas na produção de texto em sala de aula**. Paraná, 2012.

STERNE, J. **Artificial Intelligence for Marketing:** Practical Applications, 2017.

TEIXEIRA, João. **O que é inteligência Artificial**. Ed. E-Galáxia, 2019

THE ASIMOV INSTITUTE. **The neural network zoo.** [S. l.]: FJODOR VAN VEEN 14 sep 2016 (Edited 2019) Disponível em:<https://www.asimovinstitute.org/neural-network-zoo/ >. Acesso em: 09 de abril de 2020.

VASILEV. Ivan, SLATER. Daniel, SPACAGNA. Gianmario, ROELANTS. Peter, ZOCCA. Valentino, **Python Deep Learning Exploring:** deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2°Ed Packt BIRMINGHAM – MUMBAI, 2019.

VAN DER WALT, S., COLBERT, S. C., & VAROQUAUX, G. ***The NumPy Array:*** *A* Structure for Efficient Numerical Computation. Computing in Science & Engineering, 13(2), 2011.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T.E. *et al.* **SciPy 1.0: fundamental algorithms for scientific computing in Python**. *Nat Methods* **17,** 261–272, 2020.