
Redes Neurais Recorrentes para Geração de Senhas em Ataques de Força Bruta baseado em Dicionário

Marlon Brendo Ramos



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Monte Carmelo - MG
2022

Marlon Brendo Ramos

**Redes Neurais Recorrentes para Geração de
Senhas em Ataques de Força Bruta baseado em
Dicionário**

Trabalho de Conclusão de Curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau de
Bacharel em Sistemas de Informação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Thiago Pirola Ribeiro

Monte Carmelo - MG

2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

R175 2022	Ramos, Marlon Brendo, 1998- Redes Neurais Recorrentes para Geração de Senhas em Ataques de Força Bruta baseado em Dicionário [recurso eletrônico] / Marlon Brendo Ramos. - 2022. Orientador: Thiago Pirola Ribeiro. Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Sistemas de Informação. Modo de acesso: Internet. Inclui bibliografia. Inclui ilustrações. 1. Tecnologia da informação - Administração. I. Ribeiro, Thiago Pirola, 1978-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Sistemas de Informação. III. Título. CDU: 658:681.3
--------------	---

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Aos meus pais, Cleudia e Vicente (in memoriam) e minha irmã Polyane

Agradecimentos

Ao professor Dr. Thiago Pirola Ribeiro, pela orientação, incentivo e paciência pelo período em que me orientou, no qual os seus precisos conselhos e o seu zelo pela escrita contribuíram muito para a conclusão deste trabalho.

Gostaria de expressar minha gratidão à Universidade Federal de Uberlândia (UFU), em especial aos professores da Faculdade de Computação (FACOM) do Campus Monte Carmelo. Agradeço pelo exímio ensino e suporte, tanto nos meus estudos quanto na minha vida pessoal. Obrigado por contribuírem na minha formação e no meu crescimento como indivíduo.

Agradeço aos meus pais, Cleudia e Vicente (*in memoriam*), pelo apoio, humildade, simplicidade e palavras de motivação nessa jornada tão difícil. Meus agradecimentos à minha vó Vicentina (*in memoriam*) e minha tia Tereza, pelas palavras, vivências e pela humildade e força adquirida. Agradeço também a minha irmã Polyane, pelas nossas conversas, dicas, pelo seu suporte e palavras de incentivo.

“Queremos buscar a verdade, não importa aonde ela nos leve. Mas para encontrá-la, precisaremos tanto de imaginação quanto de ceticismo. Não teremos medo de fazer especulações, mas teremos o cuidado de distinguir a especulação do fato.”

Carl Sagan

Resumo

O ciberataque é um dos grandes problemas enfrentados pela sociedade contemporânea e, notoriamente, impacta em diversos ramos empresariais e pessoais. Um dos incidentes mais reportados refere-se ao scan, no qual o atacante coleta várias informações de diversas portas e protocolos e viabiliza seu plano de ataque. Um dos tipos de ataques frequentemente usados que são decorrentes do scan, é o ataque de força bruta baseado em dicionários. Tradicionalmente, técnicas de força bruta usam dicionários vazados ou gerados através de combinações de caracteres ou de perfis pessoais. Desse modo, o sucesso desse tipo de ataque está ligado a qualidade e variedade das senhas obtidas desse meio. O modo de realizar esse tipo de ataque, essencialmente baseado nesses dicionários, inviabiliza que as técnicas tradicionais sejam capazes de capturar relações semânticas existentes entre os dados, como, por exemplo, a formação de padrões em escritas de senhas. Em contrapartida, o uso de redes neurais tem se mostrado como uma via vantajosa para reconhecimento de padrões complexos ou numerosos, se destacando através de métodos de natureza dinâmica com a possibilidade de realização de sistemas preditivos, mostrando capacidade contínua de melhoria e adaptação às mudanças de padrões. Deste modo, o objetivo principal deste trabalho é a aplicação das redes neurais recorrentes, por meio da arquitetura LSTM, para a geração e previsão de senhas com base em dicionários usados em ataques de força bruta e, dessa maneira, analisar a viabilidade e o equilíbrio do seu uso.

Palavras-chave: Redes neurais recorrentes, arquitetura LSTM, aprendizado de máquina, ataque de dicionário, geração de senhas.

Lista de ilustrações

Figura 1 – Incidentes Reportados ao CERT.br - Janeiro a Dezembro de 2020. Fonte: Extraído de (CERT.br, 2020a).	16
Figura 2 – 20 Portas TCP que mais sofreram varreduras no 1º Semestre de 2020. Fonte: Extraído de (CERT.br, 2020b)	17
Figura 3 – Funcionamento do modelo de SL (RASCHKA; MIRJALILI, 2019). . .	23
Figura 4 – Funcionamento do modelo de UL (Morphio, 2020).	23
Figura 5 – Funcionamento do modelo de aprendizagem por reforço (RASCHKA; MIRJALILI, 2019).	24
Figura 6 – Descrição básica dos componentes de um neurônio biológico (RAS- CHKA; MIRJALILI, 2019).	24
Figura 7 – Ilustração de um exemplo de dados de série temporal (RASCHKA; MIRJALILI, 2019)	27
Figura 8 – Ilustração da estrutura de um MLP e uma RNN (RASCHKA; MIR- JALILI, 2019).	28
Figura 9 – Arquitetura básica de um neurônio LSTM (RASCHKA; MIRJALILI, 2019).	30
Figura 10 – Representação do <i>key logger</i> na hierarquia de sistemas. Adaptado de (BAIG; MAHMOOD, 2007).	32
Figura 11 – Digrama do método de Força Bruta baseado em dicionário. Fonte: Autoria própria	34
Figura 12 – Exemplo das 30 primeiras e 30 últimas senhas do dicionário <i>Top2Billion- probable-v2</i> extraído do site (GitHub, 2019).	37
Figura 13 – Exemplo das 30 primeiras e 30 últimas senhas do dicionário <i>Top204Thousand- WPA-probable-v2</i> extraído do site (GitHub, 2018).	37
Figura 14 – Uma ilustração geral do ataque de força bruta em senhas baseado em AI. Fonte: Autoria própria	38
Figura 15 – Representação do modelo de Rede Neural. Adaptado de (BOSCO; PILATO; SCHICCHI, 2018).	39

Figura 16 – Diagrama de representação do modelo de RNN/LSTM para geração de senhas. Fonte: Autoria Própria.	40
Figura 17 – Erros obtidos para cada unidade de memória com Treino/Teste de 10/90. Fonte: Autoria Própria	41
Figura 18 – Erros em função de unidades de memória obtidos para 1 e 2 camadas do conjunto 10/90 e 30/70 para a base <i>Top204Thousand-WPA-probable-v2</i> . Fonte: Autoria Própria.	42
Figura 19 – Comparação de erros obtidos para tamanho de sequência de 25 e 100 para bases segmentadas em 10/90, 30/70 e 20/80. Fonte: Autoria Própria.	44
Figura 20 – Épocas obtidas para os 3 segmentos que obtiveram os menores erros para os hiperparâmetros das seções anteriores. Fonte: Autoria Própria.	45
Figura 21 – Épocas e Erros obtidos para a base <i>Top2Billion-probable-v2</i> . Fonte: Autoria própria.	46

Lista de tabelas

Tabela 1	– Comparação entre os resultados de um ataque de Força Bruta tradicional e o ataque utilizando as Redes Neurais Recorrentes (TRIEU; YANG, 2018).	35
Tabela 2	– Erros e Acurácias obtidos no segmento 10/90 e 30/70 para 1 e 2 camadas.	42
Tabela 3	– Melhores conjuntos de treino/teste e hiperparâmetros obtidos para a base <i>Top204Thousand-WPA-probable-v2</i> .	45
Tabela 4	– Taxas de acertos para 2° geração usando 5 sementes para base de teste extraído de <i>Top2Billion-probable-v2</i> .	48
Tabela 5	– Taxas de acertos para a 2° geração usando 15 sementes para a base de teste extraído de <i>Top2Billion-probable-v2</i> .	48
Tabela 6	– Hit Ratio (HR) unindo dicionários da 2° geração de senhas para a base de teste extraído de <i>Top2Billion-probable-v2</i> .	49
Tabela 7	– Hit Ratio (HR) unindo todos os dicionários da 2° geração para 4 diferentes dicionários de senhas	49
Tabela 8	– Hit Ratio (HR) unindo os melhores dicionários da 2° geração para 4 diferentes dicionários de senhas	50
Tabela 9	– Erro e acurácia de cada segmento para a base <i>Top204Thousand-WPA-probable-v2</i> variando o número de unidades de memória.	61
Tabela 10	– Erro e acurácia de cada segmento para a base <i>Top204Thousand-WPA-probable-v2</i> variando o número de células/camadas e utilizando 1024 unidades de memória obtido como melhor hiperparâmetro na seção A.1	62
Tabela 11	– Erro e acurácia de cada segmento para a base <i>Top204Thousand-WPA-probable-v2</i> variando o número de sequência e utilizando os melhores hiperparâmetros da seção A.1 e seção A.2.	63
Tabela 12	– Número de épocas, erro e acurácia de cada segmento para a base <i>Top204Thousand-WPA-probable-v2</i> utilizando os melhores hiperparâmetros da seção A.1, seção A.2 e seção A.3.	63

Tabela 13 – Taxa de acerto para os dicionários selecionados unindo todas as senhas geradas pela rede. No total foram geradas 31673 senhas para a 1ª geração, 29257 senhas para 2ª geração e 28368 senhas para a 3ª geração	64
Tabela 14 – Taxa de acerto para os dicionários selecionados utilizando os dicionários gerado pela rede que obtiveram maiores acertos. No conjunto, foram obtidas 3441 senhas para 1ª geração, 1943 para a 2ª geração e 1079 para 3ª geração	65

Lista de siglas

Adagrad Adaptive Gradient Algorithm - *Algoritmo de Gradiente Adaptativo*

ANN Artificial Neural Network - *Rede Neural Artificial*

Adam Adaptive Moment Estimation - *Estimativa de Momento Adaptativo*

Adadelta Adaptive Learning Rate Method - *Método de Taxa de Aprendizagem Adaptativa*

ASCII American Standard Code for Information Interchange - *Código Padrão Americano para o Intercâmbio de Informação*

CUPP Common User Passwords Profiler - *Criação de Perfis de Senhas de Usuários Comuns*

GANs Generative Adversarial Networks - *Redes Adversárias Generativas*

HR Hit Ratio - *Taxa de Acerto*

AI Artificial Intelligence - *Inteligência Artificial*

IDS Intrusion Detection System - *Sistema de Detecção de Intrusão*

IPS Intrusion Prevention Systems - *Sistema de Prevenção de Intrusão*

LSTM Long Short-Term Memory - *Memória de Curto Prazo Longo*

ML Machine Learning - *Aprendizado de Máquina*

MD5 Message Digest Algorithm - *Algoritmo de Resumo de Mensagem*

MLP Multi Layer Perceptron - *Perceptron de Multicamadas*

NAG Nesterov Accelerated Gradient - *Gradiente Acelerado de Nesterov*

Nadam Nesterov-accelerated Adaptive Moment Estimation - *Estimativa de Momento Adaptativo Acelerada por Nesterov*

POP3 Post Office Protocol - *Protocolo dos Correios*

PNL Natural Language Processing - *Processamento de Linguagem Natural*

RFID Radio Frequency Identification - *Identificação por Rádio Frequência*

RNN Recurrent Neural Network - *Rede Neural Recorrente*

RMSProp Root Mean Square Propagation - *Propagação do Quadrado Médio da Raiz*

SMTPS Simple Mail Transfer Protocol Secure - *Protocolo de Transferência de Correio Simples Seguro*

SL Supervised Learning - *Aprendizado Supervisionado*

SHA-0 Secure Hash Algorithms - *Algoritmo de Hash Seguro*

SHA-2 Secure Hash Algorithms version 2 - *Algoritmo de Hash Seguro versão 2*

SGD Stochastic Gradient Descent - *Gradiente Descendente Estocástico*

TANH Hyperbolic Tangent - *Tangente Hiperbólica*

TBPTT Truncated Backpropagation Through Time - *Retropropagação Truncada ao Longo do Tempo*

TCP Transmission Control Protocol - *Protocolo de Controle de Transmissão*

UL Unsupervised learning - *Aprendizado não Supervisionado*

Sumário

1	INTRODUÇÃO	16
1.1	Justificativa	19
1.2	Objetivo geral	20
1.3	Objetivos específicos	20
1.4	Resultados esperados	20
1.5	Organização do trabalho	21
2	FUNDAMENTAÇÃO TEÓRICA	22
2.1	Aprendizado de máquina	22
2.2	Redes Neurais	24
2.2.1	Definição básica sobre neurônio artificial	25
2.2.2	Função de perda	26
2.3	Redes Neurais Recorrentes	26
2.3.1	Representação Sequencial	27
2.3.2	Modelos de Sequências	27
2.3.3	Estrutura e Fluxo de uma RNN	28
2.3.4	Função de Perda	29
2.3.5	Explosão e Desaparecimento de Gradiente	29
2.4	Unidades de memória de longo e curto prazo	30
2.5	Mecanismos de Obtenção e Quebras de Senhas	31
2.5.1	Tipos de Ataques	32
2.5.2	Ataque de Força Bruta	33
2.6	Trabalhos Correlatos	34
3	EXPERIMENTOS E ANÁLISE DOS RESULTADOS	36
3.1	Base de Dados	36
3.2	Pré-Processamento dos Dados	38
3.3	Construção dos Modelos	38

3.4	Modelo de Rede Neural	39
3.5	Hiperparâmetros	40
3.5.1	Número de Unidades Memória	41
3.5.2	Número de Células	41
3.5.3	Tamanho do Lote	42
3.5.4	Otimizador	43
3.5.5	Tamanho de Sequência	43
3.5.6	Número de Épocas	44
3.6	Aplicação das Definições na Base <i>Top2Billion-probable-v2</i> . . .	45
3.7	Avaliação dos Resultados	46
3.7.1	Ataque de Força Bruta baseado em Dicionários	49
3.7.2	Dicionários	49
4	CONCLUSÃO	51
4.1	Principais Contribuições	52
4.2	Trabalhos Futuros	52
	REFERÊNCIAS	54

APÊNDICES 59

APÊNDICE A	– TABELAS COMPLEMENTARES	60
A.1	Unidades de Memória	60
A.2	Células	62
A.3	Tamanho de sequência	62
A.4	Número de épocas	63
A.5	Geração de Senhas	64

Introdução

Desde o recente aumento da utilização de tecnologias da informação voltadas para desenvolvimento e progressão de atividades pessoais e econômicas, desde pequenos e médios negócios à corporações e conglomerados empresariais, as principais preocupações estão voltadas as ameaças e riscos provenientes de dispositivos, redes, aplicativos e serviços. Em virtude disso, há uma grande necessidade de implementações e discussões acerca da segurança em uma série de áreas de aplicação como protocolos de internet, serviços web, Radio Frequency Identification (RFID), dentre outros (LEE, 2020).

Com o aumento recorrente dos números de incidentes de segurança da informação e suas variadas abordagens, surge a necessidade de novos estudos sobre cibersegurança e segurança da informação (PurpleSec, 2022). A Figura 1 ilustra de forma percentual os ataques mais utilizados no Brasil no ano de 2020.

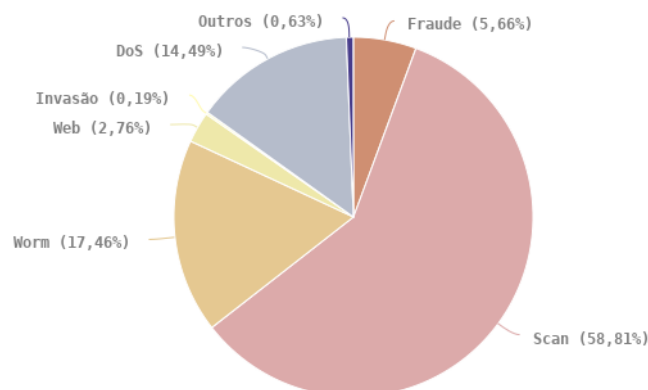


Figura 1 – Incidentes Reportados ao CERT.br - Janeiro a Dezembro de 2020. Fonte: Extraído de (CERT.br, 2020a).

De acordo com a Figura 1, cerca de 58,81% dos incidentes reportados refere-se a *scan*. O *scan* consiste na capacidade do atacante de coletar informações a respeito de um alvo, e esta coleta foca-se em várias portas abertas para obter dados mais específicos, como a versão do sistema operacional e os serviços que estão rodando. Com a obtenção dessas informações, o invasor pode sintetizar melhor o seu plano de ataque (MORENO, 2017). A

Figura 2 ilustra as 20 portas de Transmission Control Protocol (TCP) que mais receberam tráfego no 1º semestre de 2020 (CERT.br, 2020b). As portas 23, 22, 81, 5555, 8000 e 8080 sofreram um grande tráfego, e por isto, tais portas são alvos direto de tentativas de força bruta ou de exploração de vulnerabilidades. Outras portas como 110 e 465 empregando, respectivamente, o Post Office Protocol (POP3) e o (SMTPS) estão associadas a serviços e-mail e também tiveram um aumento no seu tráfego, o que pode estar associado aos ataques de força bruta para autenticações de logins.

20 Portas TCP que mais sofreram varreduras no 1º Semestre de 2020 Ordenadas por número de pacotes											
#	Porta TCP	Pacotes	Bytes	Flows	IPs	Top 5 Country Codes (número de IPs de origem do tráfego)					
01	23/tcp	30,1 G	1,5 TB	452,8 M	5.116.300	EG (944.597)	BR (760.754)	CN (612.790)	IR (358.378)	GR (217.961)	
02	22/tcp	1,2 G	99,0 GB	250,2 M	669.219	BR (275.764)	US (58.256)	VN (50.653)	CN (47.936)	CL (19.532)	
03	445/tcp	595,2 M	43,5 GB	362,3 M	3.463.191	VN (591.054)	ID (472.459)	IN (408.267)	TH (287.359)	RU (275.528)	
04	3389/tcp	574,4 M	28,0 GB	172,8 M	174.502	CN (130.704)	US (9.822)	BR (3.418)	RU (2.399)	IN (2.266)	
05	80/tcp	280,0 M	14,6 GB	200,6 M	2.778.570	BR (554.900)	IR (315.673)	CN (226.831)	US (172.539)	RU (116.532)	
06	1433/tcp	202,3 M	8,4 GB	174,0 M	333.041	CN (99.491)	VE (44.012)	BR (21.060)	VN (19.403)	IN (17.919)	
07	443/tcp	95,6 M	4,1 GB	74,8 M	238.137	US (101.242)	CN (24.168)	SA (23.873)	GB (7.285)	DE (6.397)	
08	8080/tcp	77,2 M	3,9 GB	54,1 M	2.390.586	BR (573.733)	IR (374.614)	CN (184.865)	RU (109.195)	IT (95.061)	
09	110/tcp	54,5 M	2,3 GB	12,0 M	6.726	US (3.334)	DE (1.159)	CN (455)	SG (353)	TR (256)	
10	81/tcp	47,1 M	1,8 GB	45,4 M	888.116	VN (119.125)	BR (96.192)	CN (85.376)	MX (57.112)	TW (48.188)	
11	5900/tcp	44,7 M	2,1 GB	17,2 M	205.761	IN (26.508)	KZ (16.700)	IR (16.565)	UZ (11.192)	RU (9.322)	
12	5555/tcp	41,8 M	1,6 GB	39,8 M	729.696	CN (177.824)	HK (100.588)	TW (69.841)	BR (58.501)	KR (40.323)	
13	8545/tcp	37,9 M	1,4 GB	37,7 M	1.560	US (412)	SG (352)	CN (336)	GB (118)	RU (89)	
14	8291/tcp	37,6 M	1,6 GB	36,5 M	99.606	VN (25.523)	IN (11.842)	ID (10.845)	TH (10.446)	BR (9.235)	
15	21/tcp	35,2 M	1,4 GB	12,5 M	102.041	IN (19.462)	US (18.754)	PK (10.061)	CN (9.159)	RU (6.620)	
16	8000/tcp	28,3 M	1,5 GB	17,8 M	401.673	VN (58.276)	MX (43.532)	MY (34.872)	TW (27.814)	CN (22.617)	
17	465/tcp	23,7 M	1018,1 MB	18,6 M	15.901	US (8.682)	CA (1.090)	FR (957)	GB (932)	DE (699)	
18	139/tcp	20,3 M	1,0 GB	10,5 M	29.712	CN (4.674)	US (3.701)	ID (2.923)	VN (2.639)	VE (2.415)	
19	5038/tcp	20,2 M	777,4 MB	13,8 M	1.994	PS (424)	SG (351)	US (344)	CN (137)	NL (109)	
20	3390/tcp	19,5 M	761,4 MB	14,6 M	2.729	US (551)	RU (421)	SG (353)	CN (316)	GB (149)	

Figura 2 – 20 Portas TCP que mais sofreram varreduras no 1º Semestre de 2020. Fonte: Extraído de (CERT.br, 2020b)

A fim de evitar problemas com relação aos ataques aos serviços e as autenticações em sistemas, o uso de senhas e criptografias seguras são cada vez mais requisitados por organizações para fornecer uma proteção ao seus usuários contra invasores maliciosos. Com a finalidade de lidar com a eficiência de métodos de quebra de senhas, como em um ataque de força bruta ou de dicionários, nas aplicações tradicionais, são geradas imensas quantidades de dados que representarão as variedades de combinações de senhas que posteriormente serão testadas uma a uma (RAZA et al., 2012). Além de ser um processo que pode se tornar extremamente inviável, as inúmeras combinações poderiam conter senhas que não representariam uma senha baseada em perfis pessoais ou um conjunto de caracteres estrategicamente escolhidos (KESSLER; C., 1996). Concatenado a estes fatos, a *Artificial Intelligence (AI)* propõe-se em automatizar algumas atividades que por sua vez, geram enormes quantidades de dados (Aishath Murshida A et al., 2019). Sendo assim, os métodos de AI simulam ações computacionais de forma inteligente, permitindo que sejam obtidos resultados inerente à base de dados, obtendo em muitos casos uma eficiência enorme quando comparada aos métodos tradicionais. Dessa forma, as aplicações do Machine Learning (ML) podem ser empregados no contexto de geração e adivinhação

de senhas baseado em um treinamento anterior, possibilitando maiores sucessos em um ataque de força bruta baseado em dicionário.

Por conseguinte, existe um grande número de aplicações, que vão desde organizações como monitoramento de performance para a construção de um perfil de confiança para clientes, áreas da saúde (Aishath Murshida A et al., 2019), classificação de tráfego de redes de computadores (OLIVEIRA; SÁ, 2018), reconhecimento de regiões promotoras em organismos na família *mycoplasmataceae* que são marcadores de sequência de genes em DNA (VALIATI, 2006) à detecção de anomalias em comportamento de abelhas (CARVALHO et al., 2018). A AI suscita variadas perspectivas de execução, fomentando sua aplicação em diversas áreas e extensões das ciências de base. Apesar do conceito de AI parecer singular, uma grande subárea pode ser abordada. No *Machine Learning (ML)*, o propósito é resolver tarefas por meio da análise dados e do “treinamento”, buscando padrões que serão usados futuramente em novos dados e, dessa forma, os algoritmos computacionais são capazes de extrair conhecimento e “aprender” para analisar dados desconhecidos (BOU-TABA et al., 2018). Dentro do ML, uma subárea bastante estudada são as redes neurais, que consiste em algoritmos bioinspirados no neurônio biológico e, consequentemente, são muito utilizadas em reconhecimento de padrões e Natural Language Processing (PNL) (GOODFELLOW; BENGIO; COURVILLE, 2017).

Ademais, os algoritmos de ML podem ser classificados de acordo com sua taxonomia, estando intimamente relacionados com os resultados desejados advindos do problema (ZHANG, 2010). Dessa forma, pode-se considerar dois dos principais tipos: o *Supervised Learning (SL)* e *Unsupervised learning (UL)*. No SL, o propósito é buscar um algoritmo que por meio de instâncias previamente conhecidas, haja uma predição de hipóteses futuras. Em contrapartida, no UL não há instâncias previamente conhecidas, sendo assim, o modelo visará o agrupamento dos dados associado a uma medida de similaridade conhecida (AKINSOLA, 2017).

Dado a imensa quantidade de dados gerados atualmente, informações sensíveis como dados pessoais, contas bancárias e senhas, são comumente trafegados ou usados para acessar diversos serviços digitais como redes sociais, e-mail ou bancos. Desse modo, vem se tornando obrigatório em qualquer organização moderna que preze pela segurança, a aplicação de criptografias ou funções de hash criptográfica. Embora as técnicas de criptografia estejam presente em sistemas modernos, o conceito tem sido usado por séculos para proteger segredo militares e políticos (KAHN, 1996). Na criptografia, as entidades comunicantes compartilham entre si uma chave, podendo ser pública ou privada, mas que é desconhecida por terceiros, a mensagem é criptografada através de um sistema matemático, impedindo a legibilidade de um intermediante que não possui a chave para a descryptografia (DIFFIE; HELLMAN, 1976). Por outro lado, a função de hash criptográfica tem por objetivo receber uma mensagem de tamanho indeterminado e gerar um valor hash de tamanho fixo através de uma função de hash considerada praticamente

impossível de inverter (PRENEEL et al, 1994). Alguns exemplos de algoritmos de hash criptográficos e suas especificidades incluem o Message Digest Algorithm (MD5) (XIE; LIU; FENG, 2013), Secure Hash Algorithms (SHA-0) (MANUEL; PEYRIN, 2008), Secure Hash Algorithms version 2 (SHA-2)(SANADHYA; SARKAR, 2008), entre outros.

Existem vários ataques em que usuários maliciosos possam usar para burlar a autenticação de um sistema. Na autenticação baseado em senha, a preservação de toda privacidade do usuário é diretamente correlacionada a capacidade do mesmo de apresentar uma conjunto de caracteres previamente estabelecido. Dessa forma, a identidade do usuário será totalmente comprometida caso a complexidade de sua senha seja semanticamente baseada em seu próprio perfil, isto é, números de telefone, em números de matrículas, datas de nascimento, número de carteira de estudante (SHARMA et al., 2010).

Nessa circunstância, o processo de descoberta de senhas baseadas em força bruta pode levar cerca de segundos, horas, dias ou até anos. Contudo, ainda pode ser considerado um tempo curto dependendo da complexidade da senha a ser quebrada. Nesse contexto, a problematização deste trabalho consiste, com base em pesquisas bibliográficas e experimentos, avaliar o método RNN, afim de construir um modelo que possa gerar senhas consideradas mais “humanas”, baseadas nas observações das últimas ocorrências presentes em um dicionário de senhas. Por conseguinte, tais senhas serão comparadas com outros dicionários preexistentes utilizados em ataques de dicionário. Sendo assim este estudo apresenta a seguinte questão de pesquisa: Como o aprendizado de máquina, nesse contexto, as redes neurais recorrentes, impactam na geração e adivinhação de senhas em dicionários utilizados em um ataque de força bruta ?

1.1 Justificativa

De acordo com o texto exposto, a justificativa para a pesquisa de RNN na aplicação de quebra de senhas por força bruta, está concernente as incomensuráveis formas de aplicabilidade do aprendizado de máquina, em especial as redes neurais, na cibersegurança e segurança da informação. Tradicionalmente, o ataque a ser abordado, isto é, o ataque força bruta baseado em dicionário, ainda é uma forma eficaz de burlar a autenticação de um sistema. No entanto, a eficiência desse método computacional dependerá intimamente da qualidade das senhas contidas no dicionário (JAVED; PAXSON, 2013)

A temática de redes neurais voltadas para cibersegurança e segurança da informação não é incomum. Pelo contrário, em um contexto onde há cada vez mais usuários conectados a dispositivos computacionais e à internet, é natural que haja componentes de redes, softwares, hardwares capazes de suportar a demanda. Dessa maneira, torna-se evidente a necessidade de sistemas de cibersegurança. Contudo, muitos sistemas de Intrusion Detection System (IDS) ou Intrusion Prenvent Systems (IPS) e outras ferramentas, podem gerar enormes quantidades de tráfegos para serem analisadas, e que poderão gerar falsos

positivos ou falhas na detecção de organizações de diferentes tipo de ataques não presente nas literaturas (BUCZAK; GUVEN, 2015).

A fim de contornar os problemas da grande quantidade de dados e a qualidade das senhas contidas em um dicionário, o presente trabalho apresentará a aplicação da RNN utilizando a arquitetura Long Short-Term Memory (LSTM) para processar dicionários e gerar novas senhas. Por conseguinte, supor a hipótese de que as senhas geradas por ML, especificamente pelas Redes Neurais Recorrentes, podem ser utilizadas ou agregadas em ataques de dicionários.

1.2 Objetivo geral

O principal objetivo desse trabalho é o desenvolvimento e a aplicação do método de Recurrent Neural Network (RNN) para a geração de senhas com base em um dicionário previamente estabelecido. A escolha do método de RNN está ligado ao fato da sua aplicabilidade em modelagem de linguagens e geração de textos, tomando uma sequência de palavras como entrada e tentando prever a possibilidade da próxima palavra ou caractere. As novas senhas geradas serão usadas para a comparação com dicionários comumente utilizadas nesse tipo de ataque e estipulado se seria eficiente a sua utilização. A expressão eficiência evidencia nesse contexto, o equilíbrio entre a capacidade de gerar senhas através do ML e a qualidade das senhas geradas.

1.3 Objetivos específicos

1. Construir e aperfeiçoar um bom método de RNN para a geração de senhas.
2. Demonstrar que os métodos de aprendizado de máquina podem apresentar bons resultados no quesito de criação e previsão de senhas contidas em dicionários.
3. Analisar o equilíbrio entre a capacidade de gerar senhas através do ML e a qualidade dos resultados obtidos.

1.4 Resultados esperados

Como resultado principal espera-se estabelecer um bom modelo de RNN para a geração de novas senhas, baseando-se em um dicionário de senha usado para treinamento e teste da rede. Apoiado nessa aplicação, as senhas geradas serão aferidas através de outros dicionários, permitindo verificar a qualidade do modelo construído e capacidade de previsão e geração de novas senhas.

Além disso, o trabalho também permitirá verificar a importância do envolvimento do aprendizado de máquina nos processos de quebra de senhas, fornecendo um levantamento

de dados que poderão indicar que as senhas geradas proporcionam uma alta taxa de acerto se comparado com os dicionários tradicionais.

1.5 Organização do trabalho

Os demais capítulos do presente trabalho estão organizados da seguinte forma. No Capítulo 2 são apresentados os conceitos básicos de aprendizado de máquina e a RNN com a arquitetura LSTM abordada no contexto deste trabalho, como também os trabalhos correlatos presentes na literatura sobre aplicações das redes neurais para adivinhação de senhas. No Capítulo 3, contempla-se todas as etapas de desenvolvimento e melhores abordagens para obtenção e avaliação dos resultados gerados pelas investigações. As conclusões sobre o trabalho e algumas sugestões e oportunidades para trabalhos futuros são apresentados no capítulo Capítulo 4

Fundamentação Teórica

Neste capítulo serão apresentados e discutidos os conceitos e técnicas sobre RNN e estudos sobre os tipos de obtenção e quebra de senhas. Além disso, serão abordados os principais trabalhos que foram utilizados como referência.

2.1 Aprendizado de máquina

Na contemporaneidade tecnológica, diversos modelos computacionais surgiram a fim de executar tarefas com maestria e agilidade sobre uma grande quantidade de dados. Dessa forma, em meados do século XX, o conceito de ML foi tomando forma, tornando-se um subcampo da AI, capaz de propor algoritmos que consigam “aprender” por meio da obtenção de conhecimento inerente às bases de dados. Tal conhecimento é usado para melhorar progressivamente os modelos preditivos promovendo melhores tomadas de decisões futuras. A AI está inserida cotidianamente em nossas rotinas, aplicações como: filtros de spam para e-mails, inteligências voltadas para jogos, carros autônomos, vem se tornando onipresentes nas nossas rotinas diárias (RASCHKA; MIRJALILI, 2019).

Segundo Raschka e Mirjalili (2019), a área de aprendizado de máquina pode ser segmentada em três diferentes tipos:

- ❑ **Supervised Learning (SL):** No SL, tem-se de antemão dados de treinamento rotulados, isto é, são demarcados por meio de rótulos que descreverão um conjunto de dados que serão usados para treinamento. Em outras palavras, os dados serão classificados, como por exemplo em um conjunto de e-mails, como spam ou não. O algoritmo então processa os dados e é capaz de fornecer uma predição para novas instâncias de teste. Todavia, os rótulos empregados em um corpus de dados não precisam ser estritamente binários, sendo assim, no aprendizado supervisionado poderão ser inseridos quaisquer quantidades de rótulos necessários associado a base de dados (RASCHKA; MIRJALILI, 2019). A Figura 3 ilustra o comportamento do SL.

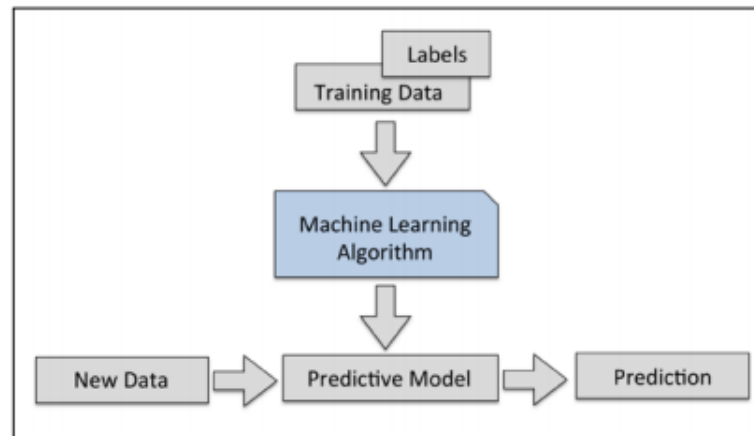


Figura 3 – Funcionamento do modelo de SL (RASCHKA; MIRJALILI, 2019).

- ❑ **Unsupervised learning (UL):** No UL, não há existência de rótulos aliado a base de dados, tratando-se então de dados não rotulados ou estruturas de dados desconhecidas. Dessa maneira, por meio da análise exploratória da base de dados e uma medida de similaridade, criam-se grupos de objetos que compartilham entre si um grau de correspondência (RASCHKA; MIRJALILI, 2019). A Figura 4 ilustra o funcionamento do UL.

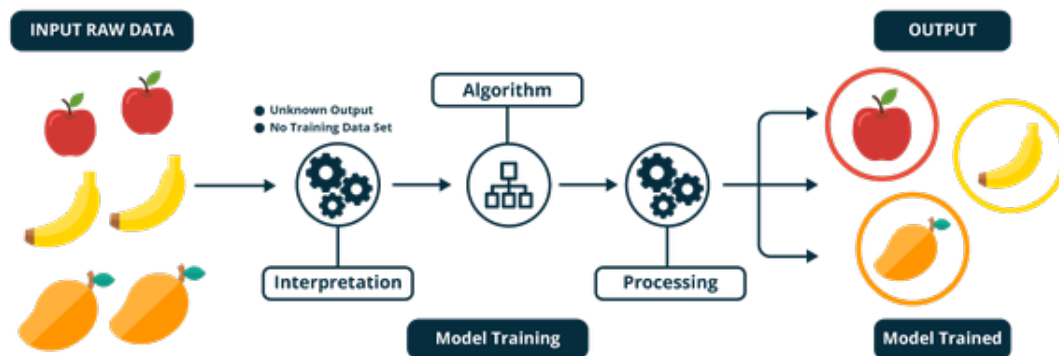


Figura 4 – Funcionamento do modelo de UL (Morphio, 2020).

- ❑ **Aprendizado por Reforço:** Já no aprendizado por reforço, o que predomina é a interação com o ambiente, ou seja, não há rótulos ou medidas de similaridade mas sim um parâmetro de quão bem a ação foi medida por uma função de recompensa. Assim sendo, decisões corretas são recompensadas e as incorretas são penalizadas por um agente observador. Um exemplo habitual consiste em um jogo de xadrez, em que o agente modela o comportamento ideal do algoritmo no jogo por meio de uma

série de movimentos que vão depender do estado do tabuleiro, em outras palavras, o ambiente (RASCHKA; MIRJALILI, 2019). A Figura 5 ilustra o funcionamento da aprendizagem por reforço.

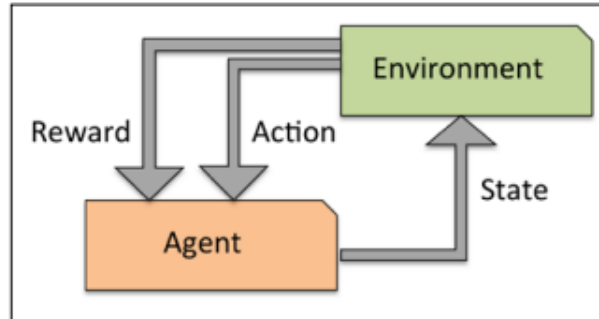


Figura 5 – Funcionamento do modelo de aprendizagem por reforço (RASCHKA; MIRJALILI, 2019).

2.2 Redes Neurais

Dentro da área de AI existem diversas subáreas e tipos de algoritmos, dentre eles as Artificial Neural Network (ANN) se destacam por serem muito empregadas no ML e em reconhecimento de padrões. Em geral, as ANN representam um conjunto de algoritmos bioinspirados nos neurônios biológicos e seu funcionamento. De maneira concisa, os neurônios agem como interruptores, recebendo, processando e propagando um sinal para outros neurônios, formando uma malha de conexões (MARCHI; MITCHELL, 2019). Adicionalmente, os neurônios são descritos como células nervosas presentes no cérebro que agem no processamento e transmissão por meio de pulsos eletroquímicos e neurotransmissores (Figura 6).

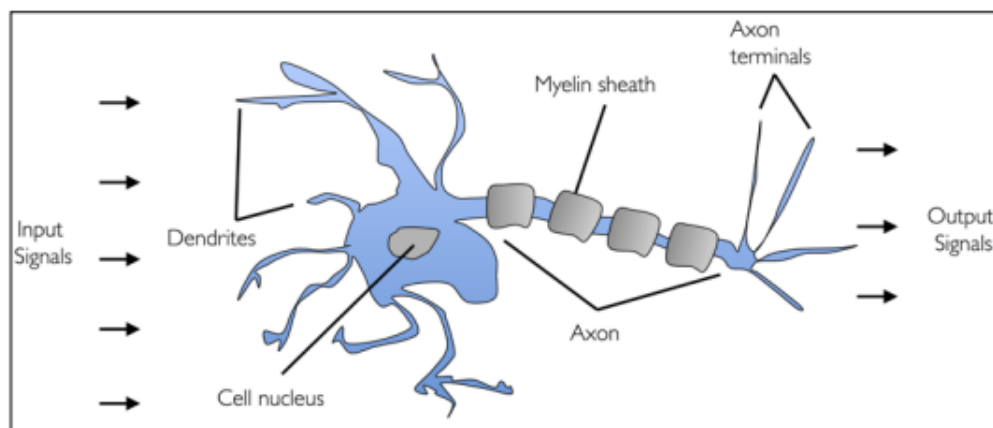


Figura 6 – Descrição básica dos componentes de um neurônio biológico (RASCHKA; MIRJALILI, 2019).

Originalmente, tais células são descritas como portas simples que geram binários como saídas. Posto isso, vários sinais chegam nas projeções dos neurônios, isto é, nos dendritos, e são incorporados à célula, se o sinal acumulado exceder um certo limite, um sinal de saída é propagado pelo axônio (MCCULLOCH; PITTS, 1943).

2.2.1 Definição básica sobre neurônio artificial

O neurônio artificial é uma unitária representação computacional para o processamento de informações que compõe uma Rede Neural. De maneira geral, é possível identificar três elementos básicos na definição de um neurônio artificial (HAYKIN, 2009):

1. A utilização de pesos definidos para as sinapses de conexão. Para cada conjunto de sinais x_j recebidos de entrada para um determinado neurônio k , tem-se um conjunto de pesos w_k , podendo ser positivos ou negativos. A multiplicação das múltiplas entradas pelo conjunto de pesos indicará a sua influência na saída da unidade;
2. A realização de uma soma ponderada das conexões sinápticas consideradas fortes, ou seja, sinais que produziram um certo nível de atividade;
3. A aplicação de uma função de ativação que permitirá estabelecer qual será a saída do neurônio. Além disso, a função de ativação vai fazer com que as sucintas variações no conjunto de pesos w_k de entrada implique em mudanças nos valores e na faixa de amplitude permissível do sinal de saída.

Formalmente o funcionamento dos neurônios artificiais pode ser exemplificado utilizando um contexto de uma simples classificação binária, utilizando-se duas classes, uma positiva e outra negativa. Desta forma, é definido o vetor de pesos w e o vetor de entradas x , criando-se a relação entre eles obtemos a seguinte combinação linear $z = w_1x_1 + \dots + w_mx_m$ (RASCHKA; MIRJALILI, 2019), podendo ser representado na forma vetorial por:

$$z = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} w_1x_1 \\ w_2x_2 \\ \vdots \\ w_mx_m \end{bmatrix}$$

Nesse contexto, é estabelecido a função de ativação $\theta(z)$ que definirá a saída do neurônio por meio de um limite na função θ . O limite é definido de acordo com cada de tipo de função de ativação, no exemplo mais reducionista de uma Rede Neural, tem-se o perceptron, que tenta imitar como um único neurônio do cérebro funciona. Assim sendo, a sua função de ativação é baseado em disparar ou não, logo, sendo realizado por meio de um único passo (1) (RASCHKA; MIRJALILI, 2019):

$$\theta(z) = \begin{cases} 1 & , \text{ se } z \geq \theta \\ -1 & , \text{ caso contrário.} \end{cases} \quad (1)$$

A fim de simplificar as expressões, a soma da combinação linear do vetor de pesos w e as entradas x pode ser descrita como um produto escalar vetorial denotada por:

$$z = w_1x_1 + \dots + w_mx_m = \sum_{j=0,}^m x_jw_j = w^T x$$

Em resumo, este é o modelo matemático que representa um neurônio artificial simples, representado como uma soma explícita e como uma operação de matrizes. O termo $w^T x$ é a representação vetorial da fórmula, em que w^t é a matriz transposta dos pesos que é multiplicada pelo vetor de entradas x .

2.2.2 Função de perda

No domínio de um algoritmo de otimização, a função usada para avaliar uma solução candidata, isto é, um conjunto de pesos, é referida como função objetivo. Nesse sentido, a função pode ser minimizada ou maximizada, sendo que na minimização, essa tarefa pode ser chamada de função de erro, função de custo ou propriamente função de perda. A responsabilidade dessa função gira em torno de um único número, de modo que as soluções candidatas possam ser comparadas em prol de que as melhorias nesse número sejam um indício de um modelo melhor (GOODFELLOW; BENGIO; COURVILLE, 2017).

Um grande desafio é a escolha correta da função, visto que, a função objetivo deverá capturar as propriedades do problema a ser tratado e representar fielmente os objetivos a serem alcançados. Caso a função escolhida não seja apropriada ao contexto do problema, é totalmente previsível a obtenção de resultados insatisfatórios (REED; MARKSII, 1999).

2.3 Redes Neurais Recorrentes

Recurrent Neural Network (RNN) operam na resolução de problemas nos quais os dados apresentam informações de sequência, que é de suma relevância conhecer as ordens dos eventos. Devido a sua natureza sequencial, suas aplicações abrangem uma ampla variedade de campos, os quais incluem reconhecimento de fala, previsão de sequência de vídeos, tradução automática de idiomas, composição musical, reconhecimento de caligrafia, aprendizagem de gramática, previsão de série temporais para antever dados econômicos, clima, preço das ações (MARCHI; MITCHELL, 2019). Como seu aspecto consiste na ordem dos eventos, as RNN possuem diferentes categorias de modelagem de sequências que permitirão descrever as ordens de entradas e saídas da rede.

2.3.1 Representação Sequencial

Do ponto de vista da representação sequencial, uma certa informação atual $x^{(2)}$ é explicitamente influenciada por meio do estado anterior $x^{(1)}$ de uma informação ao decorrer do tempo. De certa forma, as redes recorrentes modelam os sistemas dinâmicos, visto que, precisamos de um modelo matemático para descrever a dinâmica de um sistema não linear, pois como característica típica da arquitetura RNN se tem uma conexão cíclica, a qual permite que a rede possua a capacidade de atualizar o estado atual com base em estados anteriores juntamente com os dados de entrada atuais (RASCHKA; MIRJALILI, 2019). A Figura 7 elucida as sequências de dados $x^{(1)}, x^{(2)}, \dots, x^{(T)}$ não identicamente distribuídos:

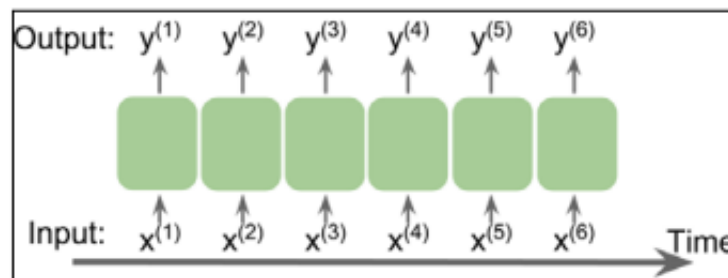


Figura 7 – Ilustração de um exemplo de dados de série temporal (RASCHKA; MIRJALILI, 2019)

2.3.2 Modelos de Sequências

Um dos principais motivos pelo qual as RNNs são interessantes é que ela nos fornece a capacidade de trabalhar com sequências de vetores: sequências na entrada, na saída ou, no caso mais geral, ambas. Devido essas formas de sequências, as seguintes categorizações são descritas (RASCHKA; MIRJALILI, 2019):

- **Many-to-One:** Essa categoria é utilizada quando a entrada é uma sequência, mas a saída não. Normalmente a saída é um escalar ou simplesmente um vetor. Um exemplo comum desse tipo de categoria é na análise de sentimentos, em que se tem como entrada um texto, isto é, uma sequência de palavras, e a saída apenas um rótulo, um binário ou um termo que determinará que sentimento esse texto expressa.
- **One-to-Many:** Nesse tipo de categoria, os dados de entrada não remetem a uma sequência, mas uma entrada padrão e a saída é uma sequência. Na literatura a descrição de imagem se adequa a esta categoria, na qual se tem como entrada uma imagem, em outras palavras, uma matriz de valores e como saída uma sequência de palavras que descreverá o seu contexto geral.

□ **Many-to-Many:** Já nessa categoria, a operação com as sequências de vetores é mais geral, dessa forma, tanto a entrada quanto a saída correspondem a sequências. Um exemplo é a descrição de vídeos, na qual se obtém como entrada uma sequência de imagens, dado que, um vídeo é composto por sucessivas imagens e uma saída que é uma sequência de palavras que descrevem o vídeo. Uma outra abordagem do Many-to-Many é por meio de um *delay*, em que a entrada de dados é apresentada em um instante, e após o término dessa entrada é que a rede começa a gerar uma sequência de resposta. Uma utilização dessa categoria é o Google Tradutor, que recebe como entrada uma sequência de palavras, realiza uma análise semântica e após isso, é efetuada a tradução para um determinado idioma.

2.3.3 Estrutura e Fluxo de uma RNN

A compreensão da estrutura e o fluxo de uma RNN pode ser realizada por meio de uma comparação simples com um Multi Layer Perceptron (MLP). A Figura 8 ilustra uma visão sobre ambas as redes.

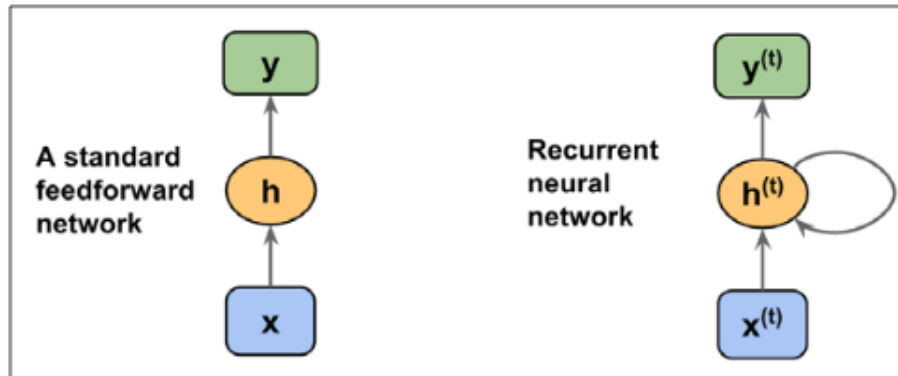


Figura 8 – Ilustração da estrutura de um MLP e uma RNN (RASCHKA; MIRJALILI, 2019).

Dessa perspectiva de observação, é retratado de forma simplificada uma MLP, na qual está presente a camada de entrada x , o neurônio oculto h e a camada de saída y . A modificação que determina o surgimento da RNN é a adição de uma estrutura de um laço recorrente no neurônio $h^{(t)}$. Tal estrutura implica dizer que a saída do neurônio oculto $h^{(t)}$, que foi calculada pela função de ativação, vai retornar ao respectivo neurônio como informação. Pela mesma razão, na camada de entrada $x^{(t)}$ e saída $y^{(t)}$, como também no neurônio oculto $h^{(t)}$, a adição do índice t revelará um vetor de características inseridas em um específico tempo.

Na recorrência desse modelo, além do vetor de pesos w_{hx} para as entradas, há a adição de um vetor de pesos extras w_{hh} no neurônio oculto para a recorrência (RASCHKA; MIRJALILI, 2019). Sendo assim, a definição da entrada pode ser dada por:

$$z_h^{(t)} = W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + b_h$$

Desse modo, é dado o vetor das entradas $x^{(t)}$ em um determinado instante de tempo t , ponderado pelos pesos W_{hx} , as saídas dos neurônios ocultos $h^{(t-1)}$ em um instante anterior, ponderadas pelos pesos extras W_{hh} da recorrência, e por fim, o bias b_h . Dessa maneira, a saída $z_h^{(t)}$ será inserido na função de ativação, resultando na saída do neurônio $h^{(t)}$:

$$h^{(t)} = \theta h(z_h^{(t)})$$

Da mesma forma, a saída $y^{(t)}$ do neurônio é definida por:

$$z_y^{(t)} = W_{yh}h^{(t)} + b_y$$

$$y^{(t)} = \theta y(z_y^{(t)})$$

2.3.4 Função de Perda

No âmbito das RNNs, a função de perda deverá ser calculada de acordo com o tempo t , como a cada recorrência haverá n saídas, a função de perda deverá ser computada para cada saída $y^{(t)}$. Portanto, a perda é calculada como a soma de todas as perdas $L^{(T)}$ de cada passo ao longo do tempo, denotada por:

$$L = \sum_{t=1}^T L^{(T)}$$

2.3.5 Explosão e Desaparecimento de Gradiente

Um grande desafio em Redes Neurais é lidar com o problema de explosão e desaparecimento de gradientes. No problema de explosão de gradiente, durante a fase de treinamento da rede, grandes gradientes de erros se acumulam e resultam em enormes atualizações para os pesos. Em redes profundas ou RNNs, o acúmulo desses gradientes de erro, por sua vez, pode tornar uma rede completamente instável. Os valores dos pesos podem se tornar tão grandes que são capazes de gerar um *overflow* de valores na memória (GOODFELLOW; BENGIO; COURVILLE, 2017). Em RNN, a instabilidade gerada na rede pelo gradiente de erro, faz com que a rede não consiga aprender com os dados durante a fase de treinamento e, dessa forma, a rede não consegue aprender sobre as longas sequências de dados na entrada (PASCANU; MIKOLOV; BENGIO, 2013). Por outro lado, pode haver uma perda da informação do gradiente devido as várias camadas de uma rede ou devido a recorrência da mesma. Como nas RNNs os gradientes locais são

propagados e combinados nos neurônios, quanto mais se afasta da camada de saída, mais a informação do gradiente diminui (GOODFELLOW; BENGIO; COURVILLE, 2017).

Algumas soluções surgiram a fim de resolver tais problemas. Uma das primeiras soluções propostas é a de recorte de gradiente, a qual se resolve inserindo um valor máximo que um gradiente pode atingir. Embora seja uma solução disponível, não funciona efetivamente em casos de desaparecimento de gradientes. Uma outra proposta idealizada é a Truncated Backpropagation Through Time (TBPTT), em que simplesmente consiste em limitar o número de recorrências e, partir disso, a propagação será condizente ao longo do tempo. Outra tarefa bastante conhecida é a adição de uma célula de Long Short-Term Memory (LSTM), que essencialmente representa a camada oculta que realiza atualizações aditivas, o que torna as memórias mais estáveis. Assim, o gradiente "flui" e dificulta o desaparecimento.

2.4 Unidades de memória de longo e curto prazo

A LSTM é um tipo de Rede Neural Recorrente que é usada em diversos cenários de processamento de linguagem natural, sendo capazes de classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. A ideia de inserir recorrências para produzir caminhos nos quais o gradiente possa fluir por longos períodos e não desaparecer muito rapidamente, é uma contribuição desse modelo (MARCHI; MITCHELL, 2019). Um neurônio LSTM, representado na (Figura 9), contém um fluxo de controle similar a uma RNN.

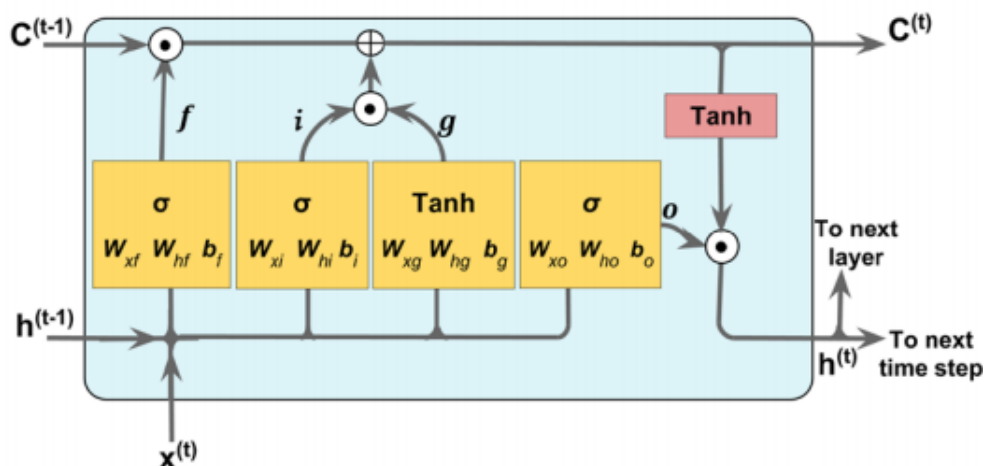


Figura 9 – Arquitetura básica de um neurônio LSTM (RASCHKA; MIRJALILI, 2019).

Nesse panorama de observação, o fluxo das informações não passam pelas mesmas etapas de uma RNN simples. A distinção consiste nas estruturas chamadas de células, já que internamente, em uma LSTM há quatro camadas de interações denominadas de portões (*gates*), em que cada uma possui uma matriz de pesos W associado a si. Além

disso, há também a presença de uma matriz de estado de célula $C^{(t-1)}$ e $C^{(t)}$ que tem como função principal de "lembrar" dos valores em determinados estados de tempos.

Como a informação é mantida nas células e as manipulações de memória são feitas pelos portões, destaca-se três portões existentes (RASCHKA; MIRJALILI, 2019):

- **Portão de esquecimento:** Esse tipo de portão controlará se a informação do estado da célula anterior $C^{(t-1)}$ deverá ser "lembrada" ou não. Dessa maneira, as entradas $x^{(t)}$ e $h^{(t-1)}$, correspondem as entradas no momento t e a saída da célula anterior. As entradas são multiplicadas por matrizes de pesos e adicionadas ao bias. Dado isso, é aplicado a função de ativação, gerando uma saída binária como resultado. Caso a saída da célula for 0, a informação é 'esquecida' e caso a saída for 1, a informação é guardada para as próximas iterações;
- **Portão de entrada:** O portão de entrada definirá quais são as informações novas que entrarão na rede e quais não entrarão. Assim, duas células vão operar em conjunto para definir a adição de informações úteis ao estado da célula. Em síntese, dado o estado da célula inicial, ao ser propagado pelo portão de esquecimento, seu conteúdo será esquecido ou não, em seguida, as informações que não foram esquecidas serão adicionadas no portão de entrada.
- **Portão de saída:** O portão de saída representa basicamente a saída de uma Rede Neural tradicional, porém nesse portão será decidido como os valores das unidades ocultas serão regulados por meio de uma função de Hyperbolic Tangent (TANH) e multiplicados para serem enviados como uma saída para a próxima célula.

2.5 Mecanismos de Obtenção e Quebras de Senhas

As técnicas para garantir acesso a sistemas e obter arquivos de senhas, como também reverter os processos de encriptação ou valores de hash podem incluir diferentes métodos que vem se aperfeiçoando a longo do tempo. Embora outras formas de autenticações existentes sejam mais eficazes, sistemas baseados em senhas ainda são a forma mais utilizada para proteger informações sensíveis, desde usuários comuns à corporativos. Essa massiva predominância vem do fato de que sua simplicidade, praticidade, custo de infraestrutura e treinamento é baixo, se comparado com outros meios de autenticações, como a biometria, que requer maiores investimentos e treinamento pessoal para o uso (CHOU et al., 2013). Em virtude disso, as senhas são comumente abordadas em ações de hackers, uma vez que, podem ser mais vulneráveis aos vários tipos de ataques e, garantindo o acesso, o atacante tem total domínio do sistema. As próximas subseções abordam alguns principais tipos de ataques de obtenção e quebra de senhas.

2.5.1 Tipos de Ataques

Os mecanismos para obtenção e quebra de senhas podem ser alcançadas por meio de inúmeras formas, desde técnicas psicológicas e comportamentais, como a engenharia social, a utilização de falsas aplicações, softwares destinados a captura de teclas, até outras formas que se aprofundam mais em sistemas computacionais, a fim de criar algoritmos e estruturas que possam quebrar senhas de forma eficiente.

2.5.1.1 *Phishing*

Nesse tipo de ataque, um atacante enfatiza os ambientes voltados para web, por conseguinte, o invasor redireciona o usuário, em vários casos, por meio de links enviados por e-mails, para um site que mantém de forma idêntica e quase indistinguível uma cópia de outro site. Os invasores podem persuadir as vítimas por meio de engenharia social e fazer com que façam login no site falso. Dessa maneira, as informações são recuperadas pelo invasor. O invasor então redireciona o usuário ao site original e faz o login do usuário a fim de mascarar o ataque (KIRDA; KRUEGEL, 2005).

2.5.1.2 *Key Loggers*

Os *key loggers* são programas que agem entre a camada de hardware do teclado e o sistema operacional. O software utiliza recursos das próprias funcionalidades do sistema operacional para que sempre que uma tecla é pressionada, uma interrupção de hardware específica é enviada para o software de espionagem, essa tecla é mapeada por meio do conceito de chave-valor e armazenada em uma fila de mensagens. A Figura 10 exemplifica o funcionamento de um *key logger* (BAIG; MAHMOOD, 2007).

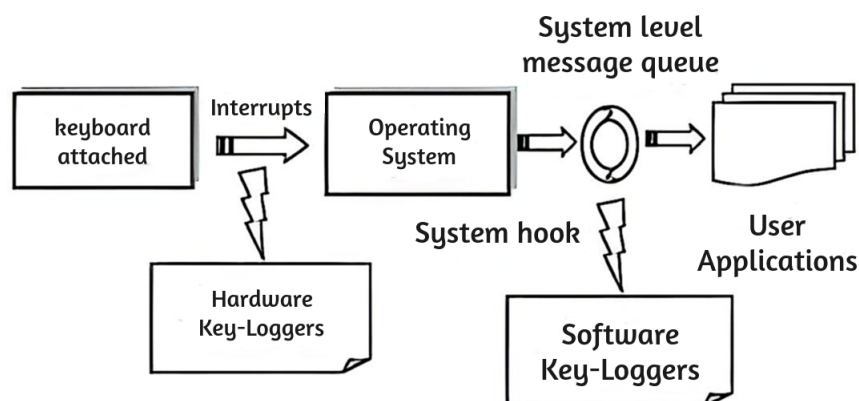


Figura 10 – Representação do *key logger* na hierarquia de sistemas. Adaptado de (BAIG; MAHMOOD, 2007).

2.5.1.3 *Keystroke Dynamics*

Nessa técnica, o que se predomina não é a obtenção direta da senha e sim como o usuário a digita. Assim, alguns padrões são guardados para que futuramente possa haver uma predição da senha, padrões como: tempo entre a tecla pressionada e a liberação, tempo entre as duas teclas pressionadas, o nome da tecla pressionada, são usados para que possa ser definido uma biometria e personificação de digitação do usuário por meio da análise dos padrões de pressionamento de tecla. Alguns métodos para predição da senha usam lógica fuzzy, Redes Neurais, técnicas estatísticas e a combinação dessas abordagens para aprender o comportamento de digitação de um usuário (HAIDER; ABBAS; ZAIDI, 2000).

2.5.1.4 *Ataque de Repetição*

O Ataque de Repetição é comumente usado para obter acesso a informações armazenadas em uma rede protegida usando credenciais supostamente válidas. O ataque constitui quando um invasor intercepta e repete uma transmissão de dados válido que trafega por uma rede. O atacante captura dados trafegados por usuários autorizados e retransmitem como se fosse uma transmissão de dados normal. Como há somente uma retransmissão do tráfego, o invasor não precisa se preocupar com os protocolos de criptografia dos pacotes, porém o invasor não pode alterar os dados do pacotes que estão sendo trafegados (SYVERSON, 1994).

2.5.2 *Ataque de Força Bruta*

Neste tipo de ataque, o propósito compõe-se em obter um conjunto de todas os possíveis combinações de senhas que serão aplicadas para quebrar a senha criptografada. O alvo mais abordado são os arquivos de senhas de texto puro que são encriptados, como por exemplo, o arquivo de senhas dos usuários do sistema operacional Linux (RAZA et al., 2012). Apesar do método ser bastante simplório, consistindo apenas em realizar inúmeras tentativas usando as variadas combinações de senhas até encontrar a correta, o custo computacional e o tempo podem se tornar inviáveis de acordo com o aumento proporcional do número de caracteres na *string*.

Como exemplo, considerando uma senha formada pelos caracteres “computer123!”, só pela presença dos caracteres numéricos ‘123’ e o caractere especial ‘!’, o número de possibilidades para a quebra aumenta de 26^8 para 68^{12} . Além dessa desvantagem relacionado ao tempo e ao custo computacional, algumas funções hash e de criptografias são combinadas com uma função aleatória, que faz com que as mesmas senhas produzam diferentes valores de saída (CHOU et al., 2013). Por outro lado, o método é considerado eficaz se utilizado senhas curtas, se for inserido uma senha de apenas 8 caracteres minúsculos sem números ou caracteres especiais, a combinação será de

$26^8 = 208827064576$, supondo que uma máquina suporte 1000 senhas por segundo, tem-se então $208827064576/1000 = 208827064.576$ horas (RAZA et al., 2012)

2.5.2.1 Ataque de Dicionário

O ataque baseado em dicionário é relativamente mais eficiente que o ataque de Força Bruta (NARAYANAN; SHMATIKOV, 2005). Neste ataque, não são realizadas as possibilidades de combinações de caracteres, mas é fundamentado em dicionários de senhas que já foram previamente construídos. Os dicionários são criados explorando a fraqueza humana de associar termos pessoais para a criação de senhas por meio de termos utilizados diariamente na vida da vítima ou gostos pessoais, como: músicas favoritas, cantores favoritos, filmes, números de telefones, nomes de filhos, esposas, animais de estimação ou datas comemorativas. Estes, são analisados e coletados para poderem ser adicionados nos dicionários ou inseridos em ferramentas de Common User Passwords Profiler (CUPP), que geram várias combinações a partir destes dados (KESSLER; C., 1996). Além disso, os dicionários também podem ser obtidos a partir de senhas que foram vazadas das diversas organizações ao longo do tempo. As senhas contidas podem conter variadas codificações de caracteres, tamanhos e linguagens. A Figura 11 ilustra o ataque baseado em dicionário.

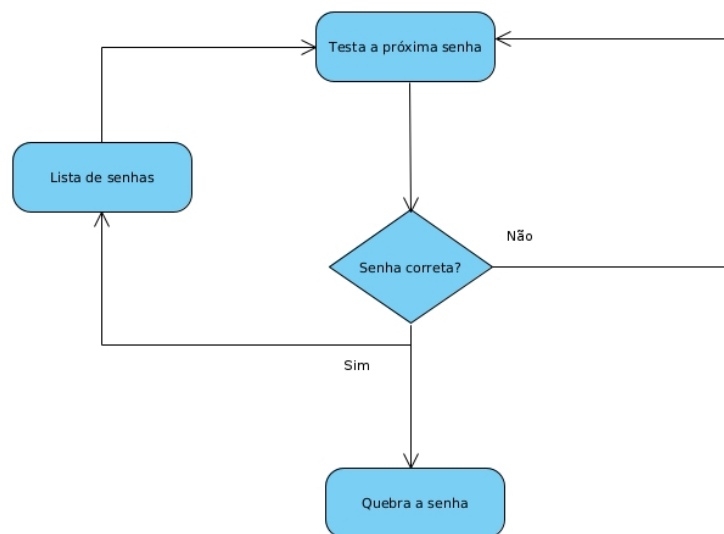


Figura 11 – Digrama do método de Força Bruta baseado em dicionário. Fonte: Autoria própria

2.6 Trabalhos Correlatos

Trieu e Yang (2018) propuseram a utilização do algoritmo Torch-RNN para gerar novas senhas após o treinamento da rede com uma base de dados de senhas. As novas senhas geradas pelo algoritmo foram adicionadas em um dicionário e posteriormente usadas nos testes de ataque de Força Bruta em tempo real. Em seu desenvolvimento, o treinamento da rede criou vários arquivos de ponto de checagem para certificar que o

modelo “aprendeu” as informações corretas por meio de senhas anteriores e foi capaz de gerar novas senhas consideradas precisas e inteligentes. A Tabela 1 demonstra a comparação entre os resultados de um ataque de Força Bruta tradicional usando dicionários de senhas comumente utilizados e um ataque resultante da geração de senhas usando Redes Neurais Recorrentes, ambos para os seguintes diferentes comprimentos de dicionário, ou seja, número de senhas: 50, 100, 250, 500 e 750.

Tabela 1 – Comparação entre os resultados de um ataque de Força Bruta tradicional e o ataque utilizando as Redes Neurais Recorrentes (TRIEU; YANG, 2018).

#of Words in the Dictionary	Success Rates of AI-Based Algorithm	Success Rates of Traditional Algorithm
50	24%	2%
100	38%	3%
250	35%	13%
500	44%	12%
750	56%	16%
1000	57%	24%

A comparação demonstra uma alta taxa de acerto no ataque de Força Bruta baseado em AI se comparado ao método tradicional. A abordagem é bem semelhante ao propósito desse trabalho, mas este estudo é mais simplificado no quesito do tamanho do dicionário e das métricas para construção e validação do modelo.

No contexto de geração de senhas “inteligentes” e precisas, Hitaj et al. (2017) introduziram a ferramenta *PassGAN*, que propõe uma nova abordagem utilizando as Generative Adversarial Networks (GANs), que são consideradas o estado da arte para gerar regras. As senhas geradas não correspondiam ao conjunto da base de teste e “pareciam” senhas geradas por humanos e, portanto, poderiam corresponder a contas de usuários reais. Uma das abordagens é a utilização da base de dados para treinamento da rede, na qual foram utilizadas um conjunto de senhas extraídas do vazamentos de dados do *LinkedIn*¹ e de um conjunto de dados denominado *RockYou*. Independentemente das abordagens serem totalmente voltadas para conjunto de senhas da língua inglesa, algumas concepções podem ser interessantes para a realização deste trabalho.

A utilização do Aprendizado de Máquina para a geração de senhas é uma área recente quando comparado aos métodos tradicionais. Existem várias ferramentas baseadas em regras, que estão disponíveis para abordagens de obtenção e quebra de senhas, como o *HashCat*(HashCat, 2021) e *JohnTheRipper*(OpenWall, 2021), que geram possíveis candidatos a senhas por meio do ataque de Força Bruta ou de dicionário, nos quais um dicionário de palavras ou de senhas previamente conhecidas é aumentado por um conjunto de regras, escritas à mão ou geradas por meio de um software de automatização.

¹ <<http://www.linkedin.com>>

Experimentos e Análise dos Resultados

Este capítulo tem o objetivo de descrever como os hiperparâmetros foram determinados para a construção e execução do protótipo. Para validar e medir a performance do modelo de Rede Neural Recorrente LSTM para a geração de senhas, foi implementado um modelo utilizando a linguagem de programação Python, com o framework Keras, que opera como uma interface para a biblioteca de código aberto Tensorflow. Todas as implementações para construção deste trabalho estão disponíveis em (Marlon Brendo Ramos, 2022b) e (Marlon Brendo Ramos, 2022a)

3.1 Base de Dados

O conjunto de senhas utilizados para o treinamento do modelo foram obtidas das listas de palavras *Top2Billion-probable-v2* (GitHub, 2019) e *Top204Thousand-WPA-probable-v2* (GitHub, 2018) no formato texto puro. As listas de palavras representam as senhas mais vazadas ao longo dos últimos anos das mais diversas organizações. Dessa forma, as bases contém senhas em American Standard Code for Information Interchange (ASCII) de variados tamanhos com letras maiúsculas e minúsculas, números, acentos e caracteres especiais. Para este trabalho, foram selecionados aleatoriamente do primeiro dicionário cerca de 100 milhões para treino e teste. Já para o segundo, foram utilizadas todas as senhas contidas no dicionário. As Figuras 12 e 13 apresentam as 30 primeiras e 30 últimas senhas obtidas das listas de senhas dos respectivos arquivos.

30 first 100 million passwords	30 last 100 million passwords
123456	...
password	fay
123456789	fay
12345678	faxzax
12345	faxyou22
qwerty	faxymalim
123123	faxyfaxy
111111	faxy1983
abc123	faxxxfilm
1234567	faxxx19
dragon	faxxq
1q2w3e4r	faxxioux
sunshine	faxxidous
654321	faxx7676
master	faxx361
1234	faxx3007
football	faxx2006
1234567890	faxww
000000	faxworthy
computer	faxworksing
666666	faxworkses
superman	faxworksed
michael	faxworks9
internet	faxworks8
iloveyou	faxworks7
daniel	faxworks6
1qaz2wsx	faxworks5
monkey	faxworks4
shadow	faxworks3
jessica	faxworks2
...	faxworks0

Figura 12 – Exemplo das 30 primeiras e 30 últimas senhas do dicionário *Top2Billion-probable-v2* extraído do site (GitHub, 2019).

30 first 204 thousand passwords	30 last 204 thousand passwords
password	...
123456789	02011973
12345678	02010201
1q2w3e4r	01210121
sunshine	01121978
football	01111978
1234567890	01102002
computer	01101986
superman	01101975
internet	01091983
iloveyou	01091981
1qaz2wsx	01091977
baseball	01081973
whatever	01081972
princess	01071997
abcd1234	01071979
starwars	01061975
trustno1	01061971
password1	01051998
jennifer	01051981
michelle	01051972
mercedes	01041996
benjamin	01041981
11111111	01021980
samantha	010203010203
victoria	01011962
alexander	0099887766
987654321	00880088
asdf1234	00220022
1234qwer	000999888
...	*123456*

Figura 13 – Exemplo das 30 primeiras e 30 últimas senhas do dicionário *Top204Thousand-WPA-probable-v2* extraído do site (GitHub, 2018).

3.2 Pré-Processamento dos Dados

Antes de usar as listas de senhas para treinar os modelos, é necessário realizar um pré-processamento nos dados com a finalidade de garantir que a base de dados não possua campos que possa prejudicar a performance e o aprendizado da RNN-LSTM, assim como também, executar tarefas adicionais que possam impactar diretamente no aprendizado do modelo. A princípio, é realizado somente uma varredura na base de dados para se verificar e garantir que não haja valores nulos (*null*) ou vazios, removendo dessa maneira, as linhas que se encontrarem com esses padrões.

3.3 Construção dos Modelos

Com os dados previamente tratados, foram construídos um total de 20 modelos LSTM para as duas bases de dados selecionadas. A princípio, a base *Top204Thousand-WPA-probable-v2* (GitHub, 2018) foi utilizada para a definição dos melhores hiperparâmetros considerados ótimos, que serão descritos detalhadamente nas seções subsequentes. Dessa forma, os modelos treinados foram gerados sobre as seguintes variações de treino e teste: 10/90, 90/10, 20/80, 80/20, 30/70, 70/30. Com base nos melhores hiperparâmetros obtidos, foi realizado o treinamento e teste sobre a base *Top2Billion-probable-v2* (GitHub, 2019). A Figura 14 ilustra as atividades relacionadas a AI e o ataque de força bruta baseado em dicionário.

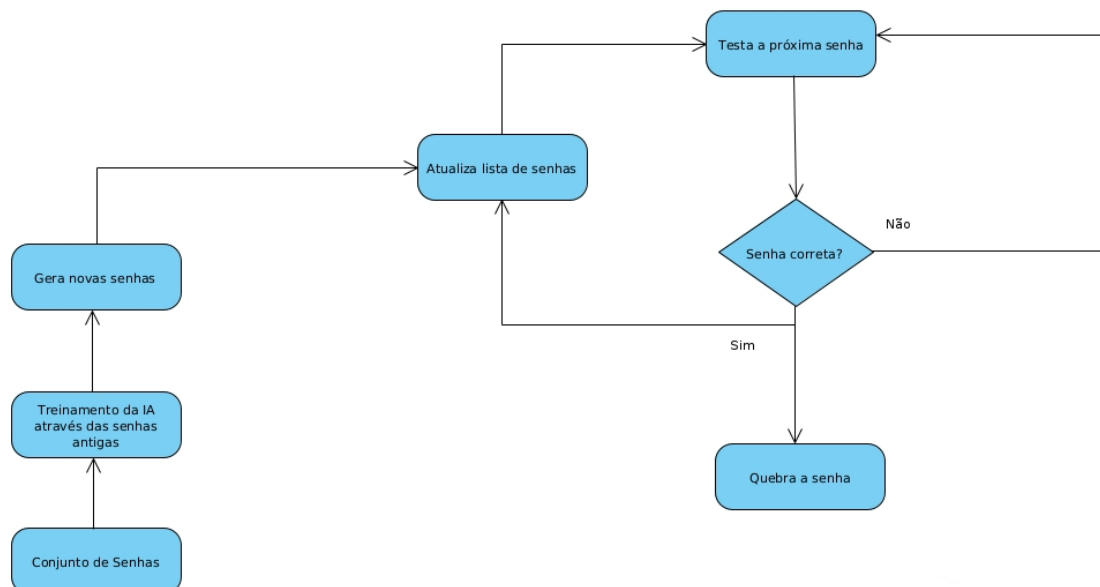


Figura 14 – Uma ilustração geral do ataque de força bruta em senhas baseado em AI.
Fonte: Autoria própria

3.4 Modelo de Rede Neural

Para construir o modelo de Rede Neural capaz de compreender as sentenças de entrada, é necessário que os dados sejam primeiramente ajustados, visto que, os algoritmos de Redes Neurais não são capazes de interpretar e processar textos puros. Dessa maneira, antes de iniciar o processamento, realiza-se uma vetorização textual, convertendo o texto puro em vetores numéricos. Duas das maneiras mais convencionais para se fazer isso são a representação *Word Embeddings* e *One-hot Encoding*. Para este trabalho foi escolhida a representação *Word Embeddings*, pois a representação *One-hot Encoding* é uma abordagem de incorporação inviável nesse contexto por exigir grande espaço de armazenamento para os vetores de palavras, reduzindo a eficiência do modelo. Por outro lado, *Word Embeddings* é feita por vetores de ponto flutuante de baixa dimensionalidade, que são aprendidos a partir do próprio texto. Desse modo, a codificação das palavras são realizada de modo que as palavras mais próximas no vetor possuam significados semelhantes. Com a utilização dessa camada, é necessário definir a dimensão do vetor na qual as palavras serão mapeadas, de forma heurística é comum escolher um valor potência de 2 para acelerar o tempo de treinamento, comumente são utilizados dimensões de 128 ou 256. Portanto, para este trabalho foi definido arbitrariamente a dimensão da camada como sendo 256. Além da camada de *Embedding*, o modelo base foi construído sobre 1 e 2 camadas ocultas, que serão melhores descritos na seção 3.5, onde serão obtidos os melhores hiperparâmetros para o modelo. E, por fim, uma camada densa de saída com o tamanho do vocabulário de caracteres usados. A Figura 15 ilustra o modelo de Rede Neural.

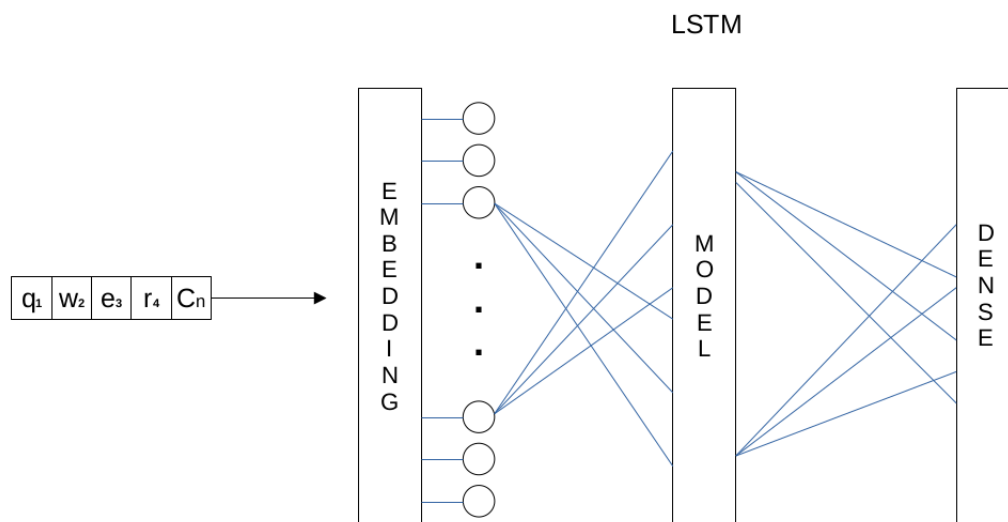


Figura 15 – Representação do modelo de Rede Neural. Adaptado de (BOSCO; PILATO; SCHICCHI, 2018).

Dessa maneira, o texto inserido é representado como caracteres e, dessa forma, dada

uma sequência de entrada, o modelo é treinado para prever o próximo caractere, gerando um de cada vez. Assim sendo, o modelo é chamado várias vezes para gerar uma sequência de caracteres, formando assim uma senha. A Figura 16 ilustra o modelo de RNN/LSTM para geração de senhas.

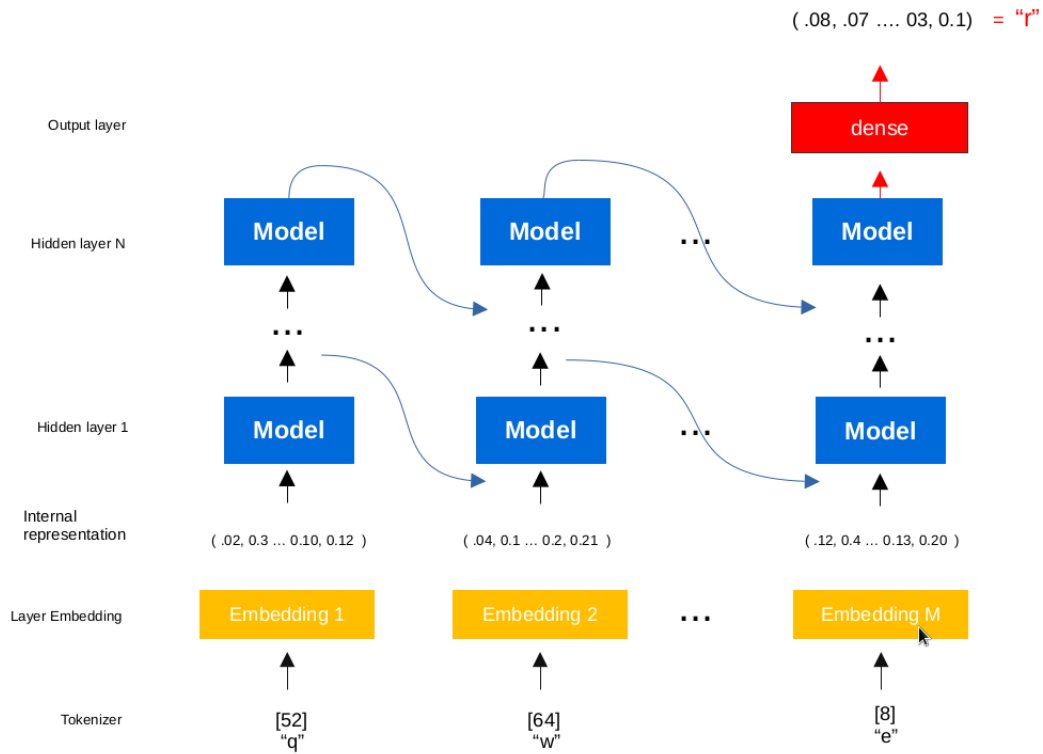


Figura 16 – Diagrama de representação do modelo de RNN/LSTM para geração de senhas. Fonte: Autoria Própria.

A geração do texto a nível de caracteres se inicia ao ser definido uma sequência de caracteres iniciais, comumente chamado de *seeds*, que servirão como ponto de partida para a geração. Após essa etapa, será previsto as probabilidades de cada caractere do vocabulário ser a próxima na senha, e então escolhido a com maior probabilidade.

3.5 Hiperparâmetros

Em Redes Neurais Artificiais, hiperparâmetros são parâmetros que os valores não são estabelecidos em tempo de execução ou em ajustes ao longo do treinamento, são definidos manualmente antes da etapa de treinamento da Rede Neural. Os hiperparâmetros fornecidos para a rede impactarão profundamente no aprendizado do modelo criado. Dessa forma, para determinar quais os melhores parâmetros para o modelo, foi realizada uma pesquisa minuciosa em outros trabalhos semelhantes, afim de verificar os melhores valores para o tamanho do lote e função de ativação. Para os parâmetros comprimento de

sequência, número de unidades de memória e número de células, foram realizados vários testes com diferentes valores.

A partir do modelo com o menor erro dentre os treinados, foi determinado como sendo o melhor hiperparâmetro para a rede. O erro indica a distância entre o que o modelo acredita que a distribuição de saída deveria ser e o que a distribuição original realmente é.

3.5.1 Número de Unidades Memória

Para cada célula da Rede neural Recorrente LSTM há um número fixo de unidades memória. Para determinar o melhor valor para esse parâmetro, foram realizados testes com o número de unidades variados: 32, 64, 128, 256, 512 e 1024.

Na Figura 17 é possível observar os valores de custo da função de entropia cruzada e acurácia para cada variação de unidades e Treino/Teste. Dentre os segmentos de base treinados, o menor erro foi obtido na segmentação 10/90 com valor de 1024 de unidades de memória. Dessa maneira, esse valor representa o melhor custo, portanto foi adotado como sendo o melhor parâmetro para o modelo. Todos os testes realizados variando os hiperparâmetros para cada segmento de base podem ser observados no Apêndice A (seção A.1).

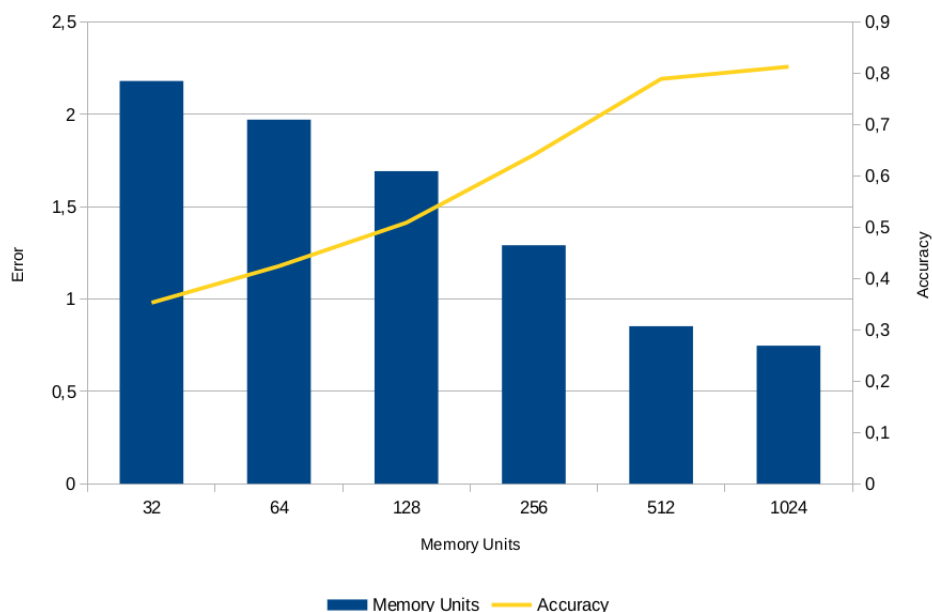


Figura 17 – Erros obtidos para cada unidade de memória com Treino/Teste de 10/90.
Fonte: Autoria Própria

3.5.2 Número de Células

Ao construir a Rede Neural é necessário determinar o número de camadas que serão utilizadas. O número ideal de camadas ou de neurônios não é uma propriedade comple-

tamente fixa e exata. Para determinar o número ideal de camadas que o modelo deve ter para obter a melhor performance possível, foram realizados testes para 1 e 2 camadas de cada conjunto de treino/teste até se encontrar o melhor valor para o erro. Todos os resultados obtidos podem ser visualizados no Apêndice A (seção A.2). A Figura 18 demonstram os menores erros obtidos variando os valores de células. Os menores erros foram obtidos no segmento 30/70 e 10/90 com 1024 unidades de memória para cada camada do modelo, na qual foram obtidos na seção anterior como melhor hiperparâmetro.

Tabela 2 – Erros e Acurácias obtidos no segmento 10/90 e 30/70 para 1 e 2 camadas.

Treino/Teste	Camadas	Erro	Acurácia(%)
10/90	1	0,7472	81%
	2	0,7612	79%
30/70	1	1,1893	65%
	2	0,4712	87%

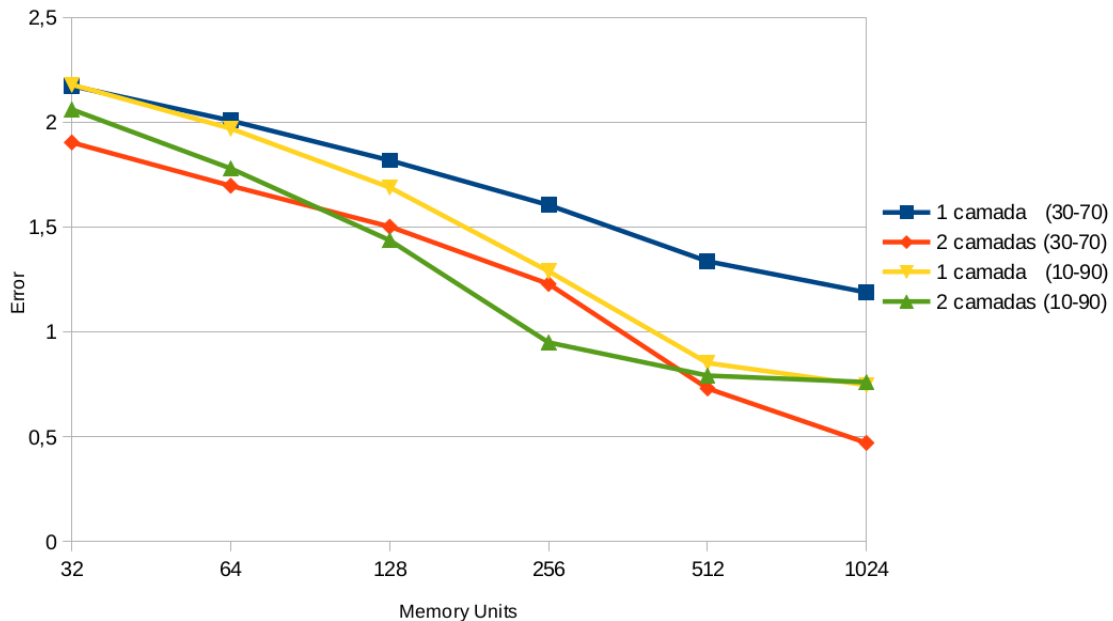


Figura 18 – Erros em função de unidades de memória obtidos para 1 e 2 camadas do conjunto 10/90 e 30/70 para a base *Top204Thousand-WPA-probable-v2*. Fonte: Autoria Própria.

3.5.3 Tamanho do Lote

Em aprendizado de máquina, o *Batch Size* ou tamanho do lote, refere-se a quantidade de amostras que serão inseridas para treinamento no modelo a cada iteração. À vista disso, seja um lote de tamanho 16, corresponde dizer que os 16 primeiros dados da base de treinamento são inseridas na rede e treinadas, e assim sucessivamente, até o final da base de treinamento. Dentre os tipos de tamanho de lote existentes (*Batch mode*, *Mini-*

batch mode, *Stochastic mode*), foi determinado o *Mini-batch mode* como sendo o ideal para o treinamento do modelo. A escolha do tipo do lote foi realizada de acordo com a sua definição, isto é, o tamanho do lote é maior que um, mas menor que o tamanho total do conjunto de dados. A definição exata do valor do tamanho do lote foi baseada no grau de generalização de conjuntos de dados que implementaram tamanho de lotes grandes e pequenos. Dessa maneira, foi observado empiricamente que tamanhos de lotes menores além de possuírem um processo de treinamento mais rápido, apresentam também uma maior generalização para a base de dados. Portanto, em virtude do contexto, foi selecionado o tamanho do lote como sendo 32, pois representa um bom valor padrão para o tipo *Mini-batch mode* (BENGIO, 2012).

3.5.4 Otimizador

Otimizadores são algoritmos capazes de realizar ajustes de maximização e minimização nos erros obtidos através de uma função de custo. Dentre os vários parâmetros utilizados para o treinamento, o otimizador é um dos que mais impactam diretamente no desempenho geral da Rede Neural. Dessa forma, internamente, o otimizador atuará por meio de uma função matemática nos parâmetros que são ajustados e atualizados em direção a solução ótima, por exemplo, os pesos e bias da Rede Neural. Em suma, quanto menor o erro obtido pela rede, maior a acurácia dos resultados e consequentemente melhor o desempenho da Rede Neural. Dentre os otimizadores disponíveis, os mais comuns são o *Momentum*, Nesterov Accelerated Gradient (NAG), Adaptive Gradient Algorithm (Adagrad), Adaptive Learning Rate Method (Adadelata), Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSProp), Nesterov-accelerated Adaptive Moment Estimation (Nadam), Stochastic Gradient Descent (SGD) e AMSGrad (SRIVASTAVA et al., 2018). No entanto, para Redes Neurais Recorrentes, os otimizadores Adam, o SGD e o RMSProp são mais frequentemente utilizados, dentre eles, o Adam apresenta melhores resultados se comparado com os outros algoritmos estocásticos (KINGMA; BA, 2014). Em virtude disso, o otimizador Adam foi escolhido para os testes realizados neste trabalho.

3.5.5 Tamanho de Sequência

Tamanho de sequência em Redes Neurais define o comprimento das sequências de dados de entrada no modelo. As RNN "aprendem" o conjunto sequencial de dados que foram inseridos. Dessa maneira, a rede consegue lembrar das sequências que foram inseridas. No entanto, essa arquitetura pode se tornar difícil de lidar com grandes sequências de entradas, implicando diretamente em tempos de treinamento muito longos. Além de tudo, tal fator pode resultar no problema de desaparecimento de gradiente, impedindo o aprendizado da rede. Para este trabalho foram selecionados dois valores de tamanho de

sequência, ou seja, 25 e 100, respectivamente. O tamanho de 25 foi obtido do trabalho de Greg (2018) e o tamanho de 100 do trabalho de Jason Brownlee (2018). No Apêndice A (seção A.3) estão todos os testes realizados para obtenção dos melhores resultados. O gráfico da Figura 19 demonstra os melhores erros para tamanho de sequência de 25 e 100 obtidos para 3 segmentos de bases diferentes, baseando-se nos melhores hiperparâmetros obtidos nas seções anteriores. Analisando os resultados obtidos, constata-se que as bases que obtiveram melhores resultados foram de tamanho de sequência de 100. Dentre elas, o segmento 30/70 obteve o menor erro se comparado aos demais. Desse modo, o tamanho 100 foi definido como sendo o hiperparâmetro ótimo para o modelo.

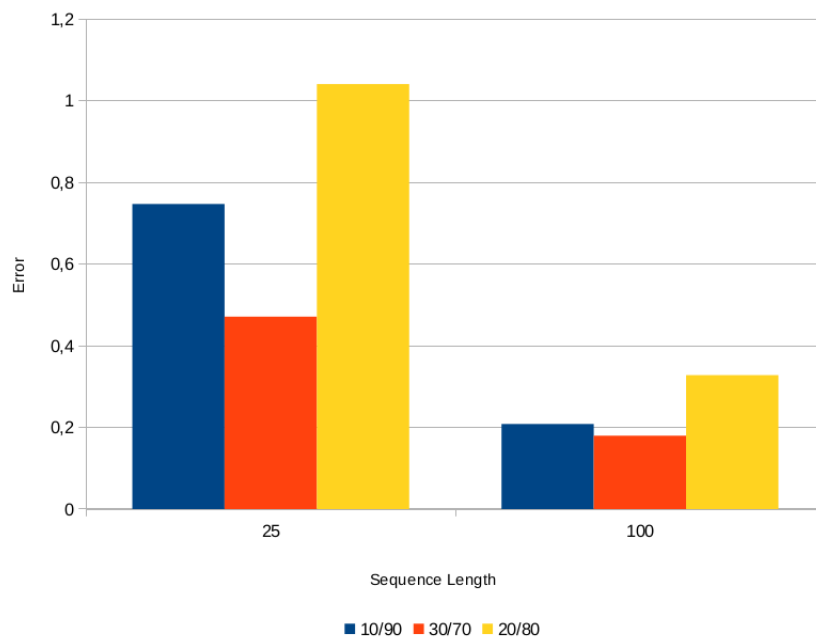


Figura 19 – Comparação de erros obtidos para tamanho de sequência de 25 e 100 para bases segmentadas em 10/90, 30/70 e 20/80. Fonte: Autoria Própria.

3.5.6 Número de Épocas

Uma época consiste em um ciclo completo da Rede Neural pela base de dados de treinamento. Assim sendo, definir o número exato de iterações do modelo na base requer uma análise indispensável, visto que, quanto maior o número de épocas, mais o modelo se adapta ao conjunto de dados, porém um número alto de épocas pode levar a um *Overfitting*, isto é, o modelo se ajusta bem, mas se torna ineficaz em predição de novos dados. Por outro lado, usarmos poucas épocas, poderemos ter problemas de *Underfitting*, em outras palavras, o modelo não aprende tudo o que poderia com os dados de treinamento. Para definir o número exato de épocas, foi utilizado a técnica de regularização *Early Stopping*, comumente chamada de parada precoce. No contexto do trabalho, a técnica analisou o erro obtido a cada fim de época, ou seja, se a cada n épocas o valor não diminuir, o treinamento é finalizado. A fim de não estipular um valor n de forma severa, e gerar uma

alta elevação no tempo de processamento, o parâmetro foi definido como sendo 5. Os resultados gerados para cada segmento de base e o número de épocas necessárias podem ser visualizados na seção A.4 do Apêndice A. Observando a Figura 20, tem-se as épocas necessárias para o conjunto de treino/teste de 10/90 com erro de 0,20 e acurácia de 95%, para o conjunto 30/70 com erro de 0,44 e acurácia de 88% e por fim para o conjunto 20/80 com erro de 0,32 e acurácia de 92%.

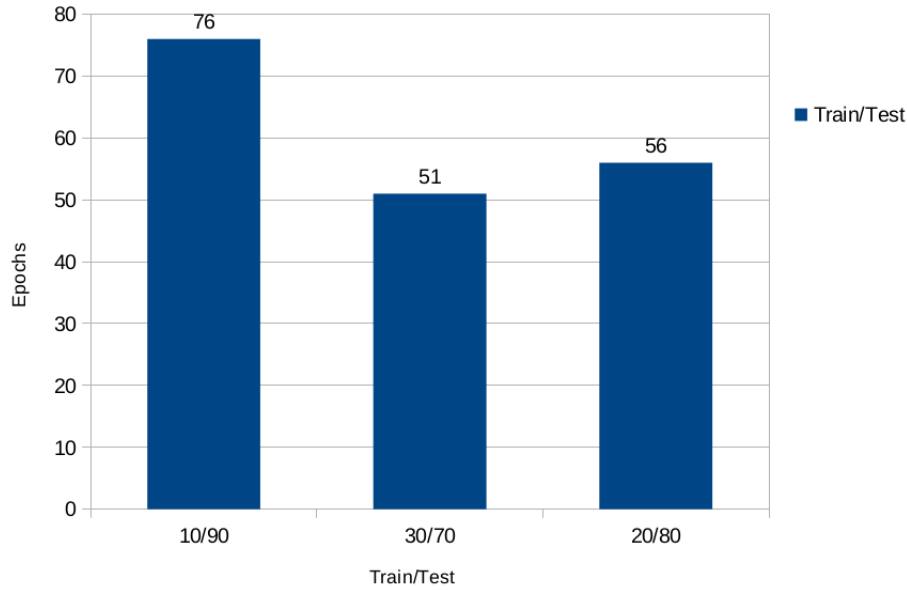


Figura 20 – Épocas obtidas para os 3 segmentos que obtiveram os menores erros para os hiperparâmetros das seções anteriores. Fonte: Autoria Própria.

3.6 Aplicação das Definições na Base *Top2Billion-probable-v2*

Com os melhores valores selecionados nas subseções da seção 3.5, foram obtidos dois perfis de melhores hiperparâmetros, visualizados na Tabela 3.

Tabela 3 – Melhores conjuntos de treino/teste e hiperparâmetros obtidos para a base *Top204Thousand-WPA-probable-v2*.

Treino/Teste	Camadas	Unidades de Memória	Tamanho de sequência
10/90	1	1024	100
30/70	2	1024	100

À vista disso, foi treinado outro modelo de RNN sobre a base *Top2Billion-probable-v2* (GitHub, 2019), baseando-se nos melhores valores obtidos. A princípio, por limitações de processamento de hardware foi utilizado somente o primeiro perfil de valores:

- ❑ 10/90 para Treino/Teste;
- ❑ 1 células;
- ❑ 1024 unidades de memória;
- ❑ tamanho de sequência de 100;
- ❑ otimizador Adam e
- ❑ tamanho do lote de 32.

A Figura 21 ilustra os erros obtidos em função das épocas necessárias para o treinamento do segmento de 100 milhões extraídos da base original. Assim sendo, foram necessárias 14 épocas, das quais apenas 9 foram suficientes para encontrar o melhor erro, em virtude da técnica de parada precoce. Dessa maneira, o menor erro encontrado foi de 0,98 com acurácia de 68%.

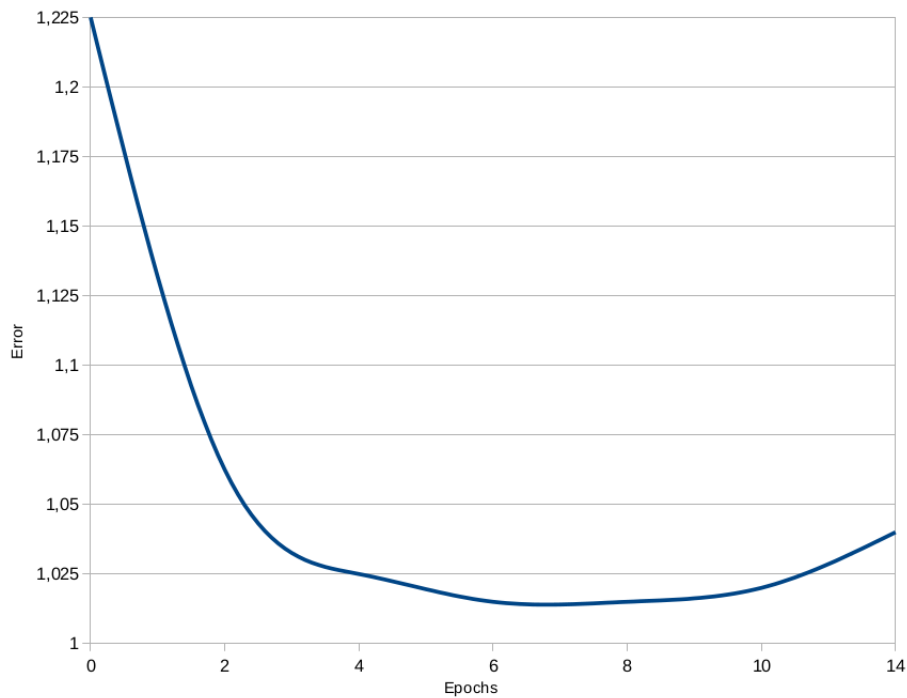


Figura 21 – Épocas e Erros obtidos para a base *Top2Billion-probable-v2*. Fonte: Autoria própria.

3.7 Avaliação dos Resultados

Para validar a qualidade do modelo e consequentemente das senhas geradas, foram utilizado a métrica Hit Ratio (HR) para aferir a taxa de acerto das senhas.

$$HR = \frac{|U_{hits}^l|}{|U_{all}|} \quad (1)$$

Na equação (1) o numerador $|U_{hits}^l|$ representa o número de senhas corretas geradas em relação a base de teste e $|U_{all}|$ representa o número total de senhas geradas.

Para obter os melhores resultados e verificar a melhor taxa de acerto, foi construído um *script* para realizar todas as combinações possíveis entre os hiperparâmetros de temperatura e número de caracteres gerados no momento de criação das senhas pelo modelo de Rede Neural. Nesse contexto, a temperatura é um hiperparâmetro que é usado para controlar a aleatoriedade das previsões. Dessa forma, quando aplicado um valor > 1 , maior será a distribuição de probabilidade para a previsão do próximo caractere, resultando em uma maior diversidade das senhas e consequentemente maiores erros. Por outro lado, valores entre o intervalo $]0, 1[$ corresponderão a geração de senhas menos aleatórias e mais próximas as senhas da base de treino. Já o número de caracteres, refere-se apenas a quantidade de caracteres que serão gerados como resultado. Para esse trabalho, foram realizadas todas as combinação de temperatura entre 0,1 e 1,5 com variação de 0,1 e o número de caracteres iniciados em 500 até 5000 com variação de 500.

Outra abordagem realizada para a geração de senhas, foi a geração de um conjunto de valores fixo extraído de uma distribuição aleatória, comumente chamada de sementes. As sementes funcionaram como um ponto de partida para o modelo gerar as próximas senhas. Em virtude disso, foram testados dois diferentes tamanhos de sementes. As Tabelas 4 e 5 demonstram as melhores taxas de acerto para 5 e 15 sementes, respectivamente. Assim sendo, para cada variação, foi gerado um arquivo de senhas na qual foram ordenadas e removidas todas as senhas repetidas, para que não houvesse uma seleção de observação dos resultados obtidos. Em virtude da aleatoriedade das sementes e dos algoritmos de aprendizado de máquina em geral não serem determinísticos, foram realizadas 3 gerações de senhas e obtidos os melhores resultados. Observando-se as tabelas 4 e 5, nota-se que a 2ª geração usando 15 sementes obtiveram mais acertos em vez de 5 sementes. Portanto, a princípio, as próximas gerações realizadas para avaliação dos resultados, foram utilizando 15 sementes.

Tabela 4 – Taxas de acertos para 2° geração usando 5 sementes para base de teste extraído de *Top2Billion-probable-v2*.

Quantidade de caracteres	Temperatura	Hit Ratio (HR)
500	0,3	53%
1000	0,6	30%
1500	0,6	54%
2000	0,1	57%
2500	0,1	50%
3000	0,1	50%
3500	0,4	28%
4000	0,2	50%
4500	0,2	37%
5000	0,1	50%

Tabela 5 – Taxas de acertos para a 2° geração usando 15 sementes para a base de teste extraído de *Top2Billion-probable-v2*.

Quantidade de caracteres	Temperatura	Hit Ratio (HR)
500	0,3	63%
1000	0,1	68%
1500	0,3	31%
2000	0,6	46%
2500	0,5	50%
3000	0,4	39%
3500	0,8	38%
4000	0,2	58%
4500	0,6	47%
5000	0,4	56%

Outras perspectivas adotadas na interpretação dos dados obtidos foi unir todos os resultados em um dicionário único, e também unir somente os melhores resultados. Para a junção dos melhores resultados, foram selecionados os dicionários que obtiveram maiores valores de acertos de senhas e não de Hit Ratio (HR), pois tal métrica interfere nos casos em que poucas senhas foram geradas, alcançando assim, altas porcentagem de acertos. A Tabela 6 apresenta a geração que obteve o melhor resultado, sendo que, foram separados os dados em dois grupos. No item *Todos*, foram unidos todos os dicionários da 2° geração de senhas para a base de teste e, no item *Melhores* foram unidos os melhores dicionários baseado na quantidade de acerto para cada combinação de quantidade de caracteres e temperatura obtidos da 2° geração para a base de teste. Ademais, a seção A.5 do Apêndice A demonstra todas as 3 gerações de senhas realizadas e a taxa de acerto para a base de teste e outros dicionários que foram selecionados na subseção 3.7.1.

Tabela 6 – Hit Ratio (HR) unindo dicionários da 2ª geração de senhas para a base de teste extraído de *Top2Billion-probable-v2*.

	Quantidade de senhas	Quantidade de acertos	Hit Ratio (HR)
Todos	29257	6578	22%
Melhores	1943	1079	55%

3.7.1 Ataque de Força Bruta baseado em Dicionários

Essa subseção tem como propósito realizar uma análise comparativa das senhas geradas pela Rede Neural na etapa anterior com alguns dos dicionários mais relevantes que são frequentemente utilizados nesse tipo de ataque. E, dessa maneira, por meio dos resultados corroborar com a hipótese de que os dicionários de senhas geradas pelo ML podem ser utilizadas propriamente ou agregadas no ataque para a quebra de senhas.

3.7.2 Dicionários

Os dicionários selecionados representam um conjunto de senhas vazadas ou roubadas de diversos sites ao longo do tempo. As Tabelas 7 e 8 exibem as melhores taxas de acerto obtidas para as bases (Crackstation-Human-Only, 2019), (RockYou, 2022), (Crackstation, 2019) e (Top85MillionWPA, 2018) em relação aos dicionários gerados pela rede. Examinando os resultados obtidos, observa-se que ao unir todos os dicionários, foram geradas 29257 senhas, na qual foi obtida a melhor taxa de acerto sobre a base *Crackstation*. Por outro lado, unindo os melhores resultados de cada etapa, foram geradas 1943 senhas, a qual para o mesmo dicionário abordado na união de todos os dicionários, foram obtidos uma taxa de acerto de 99%. Ao passo que, a taxa de acerto decaiu consideravelmente para a base *Top85MillionWPA*. A razão pela qual o dicionário *Crackstation* obteve maiores taxas de acertos está relacionado ao seu tamanho, isto é, o dicionário contém maiores quantidades de senhas para serem comparadas, garantindo maiores probabilidades de acertos.

Tabela 7 – Hit Ratio (HR) unindo todos os dicionários da 2ª geração para 4 diferentes dicionários de senhas

Dicionário	Acertos	Hit Ratio (HR)
Crackstation-Human-Only	4489	15%
RockYou	3382	11%
Top85MillionWPA	1888	6%
Crackstation	14046	48%

Tabela 8 – Hit Ratio (HR) unindo os melhores dicionários da 2ª geração para 4 diferentes dicionários de senhas

Dicionário	Acertos	Hit Ratio (HR)
Crackstation-Human-Only	656	33%
RockYou	536	27%
Top85MillionWPA	19	1 %
Crackstation	1940	99%

Todavia, as senhas geradas e comparadas foram obtidas a partir de sementes aleatórias da base de teste. Em virtude disso, as senhas selecionados em cada geração podem não conter os respectivos “padrões” de escrita utilizados nas senhas de um dicionário específico, podendo gerar baixas taxas de acerto como no caso do dicionário *Top85MillionWPA* para a 2ª segunda geração. No entanto, observa-se nas Tabelas 13 e 14 da seção A.5 do Apêndice A, que para a 3ª geração de senhas o dicionário *Top85MillionWPA* obteve maiores taxas de acerto, em virtude da aleatoriedade e “qualidade” das sementes escolhidas.

Considerações Finais

Neste trabalho foi proposto um estudo a cerca da técnica de Força Bruta utilizando dicionários para a geração de senhas, baseando-se na utilização da RNN com foco na arquitetura LSTM para a descoberta de dicionários de senhas. A técnica proposta é fundamentada na capacidade das redes LSTM criarem vínculo e dependências temporais da sua arquitetura, permitindo que as informações sejam lembradas ao longo do tempo. Desse modo, neste estudo foi investigado a capacidade das Redes LSTM gerarem e preverem senhas baseadas na aprendizagem de dicionários previamente conhecidos. E, por meio dos resultados obtidos, deduzir que utilizar os dicionários gerados por ML para ataques de dicionário podem ser promissores para a utilização nesse tipo de ataque. Além de testar a aplicabilidade do método em um conjunto de dados mais próximo do real, foi testada também a previsibilidades da geração de senhas para vários outros grandes dicionários de senhas.

A partir das métricas obtidas ao longo dos testes realizados, observou-se que, o modelo definido resultou em boas acurácias e taxas de acerto para os dicionários selecionados, isto é, a RNN obteve uma boa capacidade de generalização e foi capaz de abstrair os padrões de escrita, gerar e prever senhas mais “humanas”. Desse modo, o presente estudo conseguiu concluir o objetivo específico relacionado ao aperfeiçoamento do método de RNN para a geração de senhas e demonstrou que o ML pode apresentar bons resultados em relação a previsão de senhas e a criação de dicionários. Além disso, a partir da rede treinada e pesos adequados ao problema, é possível gerar sem muita complexidade computacional diversas quantidades de senhas variando randomicamente as sementes e garantindo a criação de diversas senhas não descritas e previstas nos dicionários utilizados para comparação. Garantindo desse modo, o terceiro objetivo específico deste trabalho, ou seja, há pouca complexidade computacional envolvida e boas taxas de acertos nas senhas geradas, mantendo assim o equilíbrio proposto. Logo, o ataque de dicionário pode ser complementado utilizando as senhas geradas e agregadas com os respectivos dicionários comumente utilizados, garantindo maiores possibilidades de sucesso no ataque.

O presente trabalho gerou um resumo que foi aceito para publicação na IX FACOM

TechWeek e XVI WTDCC (Workshop de Teses e Dissertações em Ciência da Computação) realizado pela Universidade Federal de Uberlândia (UFU) e será publicado nos anais do evento.

4.1 Principais Contribuições

A contribuição efetiva deste trabalho está no delineamento e aperfeiçoamento de um modelo de RNN com foco na arquitetura LSTM aplicado a um problema de Natural Language Processing (PNL). Em outras palavras, o presente trabalho abordou a utilização do ML para a abstração de padrões inerentes de escritas de senhas em dicionários e, por conseguinte, a capacidade do modelo de generalização para a previsão e geração de novas senhas. Em suma, a experimentação empírica que pode ser conduzida, permitiu testar várias condições e variações que permitiram encontrar diversos parâmetros de treinamento e geração de resultados. Além disso, um estudo abrangente foi realizado acerca dos hiperparâmetros que não foram testados, mas foram considerados ótimos para o modelo, o qual se mostrou ser totalmente coerente com os resultados obtidos para este contexto.

Diversos cenários distintos de treinamento e teste do modelo foram montados com o intuito de verificar sua qualidade e, por conseguinte, a generalidade e aplicabilidade do modelo no quesito de previsão e geração de novas senhas. Como por exemplo, a utilização de um dicionário menor para o treinamento e teste, para que assim, o tempo de treinamento fosse menor, e mais variações de testes fossem abordadas, como as segmentações de treino e teste e os hiperparâmetros: Unidades de Memória, Células, Tamanho de Sequência e Épocas.

Do mesmo modo, os resultados dos experimentos realizados com a implementação dos cenários contidos no Capítulo 3, também contribuíram para as diferentes perspectivas dos resultados, ou melhor, garantiram que a geração das senhas fossem obtidas através de inúmeras combinações de distribuição de probabilidade, quantidade de caracteres gerados e sementes. Permitindo assim, em virtude da aleatoriedade e combinação, inúmeros resultados relevantes para o contexto do problema.

4.2 Trabalhos Futuros

Em função da ausência de capacidade computacional e do tempo para a conclusão desta dissertação, recomenda-se para trabalhos futuros o treinamento da Rede Neural utilizando o segundo perfil obtido a partir dos experimentos realizados no Capítulo 3, ou seja, utilizar o segmento de 30 milhões para treino e 70 milhões para teste, 2 camadas ou células, 1024 unidades de memória e tamanho de sequência de 100. Da mesma forma, a experimentação de outros hiperparâmetros e parâmetros que não foram abordados neste trabalho em virtude do tempo. Como por exemplo, a alteração padrão da Taxa de Apre-

dizado da Rede, pois a mesma tem grande influência durante o processo de treinamento da Rede Neural. Uma taxa de aprendizado muito baixa torna o aprendizado da rede muito lento, à proporção que uma taxa de aprendizado muito alta provoca oscilações no treinamento e impede a convergência do processo de aprendizado. Assim como também a modificação do parâmetro *Momentum*, que tem como propósito aumentar a velocidade de treinamento da Rede Neural e reduzir o perigo de instabilidade. Além de outras variações a cerca do tamanho da dimensão do vetor utilizada na camada de vetorização textual ou valores para o tamanho de sequência. Pequenas diferenças nestes parâmetros podem levar a grandes diferenças tanto no tempo de treinamento como na generalização da rede.

Por fim, sugere-se também a incorporação de testes das senhas geradas pelo modelo utilizando “tabelas arco-íris” (RainbowCrack, 2020), Possibilitando quebrar senhas ou chaves-secretas baseadas em valores HASH. Sendo assim, as senhas contidas nos dicionários gerados seriam inseridas em vários algoritmos de mão única como MD5, SHA1, SHA2, LM e NTLM. Por fim, através da tabela pré-computada, reverter as funções hash criptográficas nas senhas de origem.

Pretende-se, também, submeter os resultados obtidos como trabalho completo para eventos ou revistas qualificadas da área de Computação.

Referências

- Aishath Murshida A et al. Survey on artificial intelligence. **International Journal of Computer Sciences and Engineering**, IJCSE, Indore, INDIA, v. 7, p. 1778–1790, 5 2019. ISSN 2347-2693. Citado 2 vezes nas páginas 17 e 18.
- AKINSOLA, J. E. T. Supervised machine learning algorithms: Classification and comparison. **International Journal of Computer Trends and Technology (IJCTT)**, v. 48, p. 128 – 138, 06 2017. Citado na página 18.
- BAIG, M. M.; MAHMOOD, W. **A Robust Technique of Anti Key-Logging using Key-Logging Mechanism**. [S.l.: s.n.], 2007. 314-318 p. Citado 2 vezes nas páginas 7 e 32.
- BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. **CoRR**, abs/1206.5533, 2012. Disponível em: <<http://arxiv.org/abs/1206.5533>>. Citado na página 43.
- BOSCO, G. L.; PILATO, G.; SCHICCHI, D. A neural network model for the evaluation of text complexity in italian language: a representation point of view. **Procedia Computer Science**, v. 145, p. 464–470, 2018. ISSN 1877-0509. Postproceedings of the 9th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2018 (Ninth Annual Meeting of the BICA Society), held August 22-24, 2018 in Prague, Czech Republic. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050918323962>>. Citado 2 vezes nas páginas 7 e 39.
- BOUTABA, R. et al. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. **Journal of Internet Services and Applications**, v. 9, 05 2018. Citado na página 18.
- BUCZAK, A.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. **IEEE Communications Surveys & Tutorials**, v. 18, p. 1–1, 01 2015. Citado na página 20.
- CARVALHO, H. V. de et al. Detecção de anomalias em comportamento de abelhas utilizando redes neurais recorrentes. In: SBC. **Anais do IX Workshop de Computação Aplicada a Gestão do Meio Ambiente e Recursos Naturais**. [S.l.], 2018. Citado na página 18.

CERT.br. **Incidentes Reportados ao CERT.br – Janeiro a Dezembro de 2020**. 2020. Disponível em: <<https://www.cert.br/stats/incidentes/2020-jan-jun/tipos-ataque.html>>. Acessado em 28/05/2021. Citado 2 vezes nas páginas 7 e 16.

_____. **Incidentes Reportados ao CERT.br – Janeiro a Dezembro de 2020**. 2020. Disponível em: <<https://www.cert.br/stats/incidentes/2020-jan-jun/scan-portas.html>>. Acessado em 28/05/2021. Citado 2 vezes nas páginas 7 e 17.

CHOU, H.-C. et al. Password cracking based on learned patterns from disclosed passwords. **International Journal of Innovative Computing, Information and Control**, v. 9, p. 821–839, 01 2013. Citado 2 vezes nas páginas 31 e 33.

Crackstation. **crackstation**. 2019. Disponível em: <<https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm>>. Acessado em 23/08/2022. Citado na página 49.

Crackstation-Human-Only. **crackstation-human-only**. 2019. Disponível em: <<https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm>>. Acessado em 13/08/2022. Citado na página 49.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE Transactions on Information Theory**, v. 22, n. 6, p. 644–654, 1976. Citado na página 18.

GitHub. **Probable-Wordlists**. 2018. Disponível em: <<https://github.com/berzerk0/Probable-Wordlists/blob/master/Real-Passwords/WPA-Length/Top204Thousand-WPA-probable-v2.txt>>. Acessado em 02/07/2022. Citado 4 vezes nas páginas 7, 36, 37 e 38.

_____. **Probable-Wordlists**. 2019. Disponível em: <<https://github.com/berzerk0/Probable-Wordlists/tree/master/Real-Passwords>>. Acessado em 5/09/2021. Citado 5 vezes nas páginas 7, 36, 37, 38 e 45.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning (adaptive computation and machine learning series). **Cambridge Massachusetts**, p. 321–359, 2017. Citado 4 vezes nas páginas 18, 26, 29 e 30.

Greg. **Password Cracker**. 2018. Disponível em: <https://github.com/gsurma/password_cracker>. Acessado em 31/08/2022. Citado na página 44.

HAIDER, S.; ABBAS, A.; ZAIDI, A. A multi-technique approach for user identification through keystroke dynamics. In: **Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0. [S.l.: s.n.], 2000. v. 2, p. 1336–1341 vol.2**. Citado na página 33.

HashCat. **HashCat -advanced password recovery**. 2021. Disponível em: <<https://hashcat.net/hashcat/>>. Acessado em 28/05/2021. Citado na página 35.

HAYKIN, S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009. Citado na página 25.

HITAJ, B. et al. Passgan: A deep learning approach for password guessing. **CoRR**, abs/1709.00440, 2017. Disponível em: <<http://arxiv.org/abs/1709.00440>>. Citado na página 35.

Jason Brownlee. **Text Generation With LSTM Recurrent Neural Networks in Python with Keras**. 2018. Disponível em: <<https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/>>. Acessado em 31/08/2022. Citado na página 44.

JAVED, M.; PAXSON, V. Detecting stealthy, distributed ssh brute-forcing. In: . New York, NY, USA: Association for Computing Machinery, 2013. (CCS '13), p. 85–96. ISBN 9781450324779. Disponível em: <<https://doi.org/10.1145/2508859.2516719>>. Citado na página 19.

KAHN, D. **The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet**. [S.l.]: Scribner, 1996. ISBN 9781439103555. Citado na página 18.

KESSLER, C., G. Passwords - strengths and weaknesses. In: . [S.l.: s.n.], 1996. Citado 2 vezes nas páginas 17 e 34.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014. Citado na página 43.

KIRDA, E.; KRUEGEL, C. Protecting users against phishing attacks with antiphish. In: **29th Annual International Computer Software and Applications Conference (COMPSAC'05)**. [S.l.: s.n.], 2005. v. 1, p. 517–524 Vol. 2. Citado na página 32.

LEE, C. Security in telecommunications and information technology (7th edition) an overview of issues and the deployment of existing itu-t recommendations for secure telecommunications. In: . [S.l.: s.n.], 2020. Citado na página 16.

MANUEL, S.; PEYRIN, T. **Collisions on SHA-0 in one hour**. 2008. Citado na página 19.

MARCHI, L. D.; MITCHELL, L. **Hands-On Neural Networks: Learn how to build and train your first neural network model using Python**. [S.l.]: Packt Publishing Ltd, 2019. Citado 3 vezes nas páginas 24, 26 e 30.

Marlon Brendo Ramos. **Grifo**. 2022. Disponível em: <<https://github.com/MarlonBrendonx/Grifo>>. Acessado em 27/12/2022. Citado na página 36.

_____. **Passwords-Cracker**. 2022. Disponível em: <<https://github.com/MarlonBrendonx/Passwords-Cracker>>. Acessado em 27/12/2022. Citado na página 36.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 25.

MORENO, D. **Pentest em aplicações web**. [S.l.]: NOVATEC, 2017. ISBN 9788575226131. Citado na página 16.

Morphio. **Supervised vs Unsupervised Machine Learning – Guide For Growth Marketers**. 2020. Disponível em: <<https://morphio.ai/blog/supervised-vs-unsupervised-machine-learning/>>. Acessado em 28/05/2021. Citado 2 vezes nas páginas 7 e 23.

- NARAYANAN, A.; SHMATIKOV, V. Fast dictionary attacks on passwords using time-space tradeoff. In: **Proceedings of the 12th ACM Conference on Computer and Communications Security**. New York, NY, USA: Association for Computing Machinery, 2005. (CCS '05), p. 364–372. ISBN 1595932267. Disponível em: <<https://doi.org/10.1145/1102120.1102168>>. Citado na página 34.
- OLIVEIRA, D. J.; SÁ, A. A. R. d. Redes neurais aplicadas à classificação de tráfego de redes de computadores utilizando os protocolos tcp e udp. **Revista de Sistemas e Computação-RSC**, v. 8, n. 1, 2018. Citado na página 18.
- OpenWall. **OpenWall - bringing security into open environments**. 2021. Disponível em: <<https://www.openwall.com/john/>>. Acessado em 28/05/2021. Citado na página 35.
- PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: PMLR. **International conference on machine learning**. [S.l.], 2013. p. 1310–1318. Citado na página 29.
- PurpleSec. **Cyber Security Statistics The Ultimate List Of Stats Data, & Trends For 2022**. 2022. Disponível em: <<https://purplesec.us/resources/cyber-security-statistics/#Recent>>. Acessado em 08/12/2022. Citado na página 16.
- RainbowCrack. **List of Raiwbow Tables**. 2020. Disponível em: <<http://project-rainbowcrack.com/table.htm>>. Acessado em 17/09/2022. Citado na página 53.
- RASCHKA, S.; MIRJALILI, V. **Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2, 3rd Edition**. [S.l.]: Packt Publishing, 2019. (Expert insight). ISBN 9781789955750. Citado 9 vezes nas páginas 7, 22, 23, 24, 25, 27, 28, 30 e 31.
- RAZA, M. et al. A survey of password attacks and comparative analysis on methods for secure authentication. **World Applied Sciences Journal**, v. 19, p. 439–444, 01 2012. Citado 3 vezes nas páginas 17, 33 e 34.
- REED, R.; MARKSII, R. J. **Neural smithing: supervised learning in feedforward artificial neural networks**. [S.l.]: Mit Press, 1999. Citado na página 26.
- RockYou. **RockYou**. 2022. Disponível em: <<https://weakpass.com/wordlist/90>>. Acessado em 13/08/2022. Citado na página 49.
- SANADHYA, S. K.; SARKAR, P. **New Collision attacks Against Up To 24-step SHA-2**. 2008. Cryptology ePrint Archive, Report 2008/270. <<https://eprint.iacr.org/2008/270>>. Citado na página 19.
- SHARMA, A. et al. Password based authentication: Philosophical survey. In: **2010 IEEE International Conference on Intelligent Computing and Intelligent Systems**. [S.l.: s.n.], 2010. v. 3, p. 619–622. Citado na página 19.
- SRIVASTAVA, M. et al. Comparison of optimizers implemented in generative adversarial network (gan). **International Journal of Pure and Applied Mathematics**, v. 119, n. 12, p. 16831–16835, 2018. Citado na página 43.

- SYVERSON, P. A taxonomy of replay attacks [cryptographic protocols]. In: **Proceedings The Computer Security Foundations Workshop VII**. [S.l.: s.n.], 1994. p. 187–191. Citado na página 33.
- Top85MillionWPA. **Top85MillionWPA**. 2018. Disponível em: <<https://github.com/berzerk0/Probable-Wordlists/tree/master/Real-Passwords/WPA-Length>>. Acessado em 13/08/2022. Citado na página 49.
- TRIEU, K.; YANG, Y. Artificial intelligence-based password brute force attacks. In: . [S.l.: s.n.], 2018. Citado 3 vezes nas páginas 9, 34 e 35.
- VALIATI, J. F. Redes neurais aplicadas ao reconhecimento de regiões promotoras na família mycoplasmataceae. 2006. Citado na página 18.
- XIE, T.; LIU, F.; FENG, D. **Fast Collision Attack on MD5**. 2013. Cryptology ePrint Archive, Report 2013/170. <<https://eprint.iacr.org/2013/170>>. Citado na página 19.
- ZHANG, Y. **New Advances in Machine Learning**. [S.l.]: IntechOpen, 2010. ISBN 9789533070346. Citado na página 18.

Apêndices

Tabelas Complementares

O apêndice vigente apresenta um compilado de vários testes que foram relevantes para a obtenção dos resultados de cada etapa na construção dos experimentos. Desse modo, as seções subsequentes apresentam os treinamentos realizados para a obtenção dos melhores valores de hiperparâmetros para as unidades de memória ou neurônios, células ou camadas e épocas. E por fim, os parâmetros para a geração dos dicionários de senhas.

A.1 Unidades de Memória

Para os testes realizados sobre o hiperparâmetro de unidade de memória, foram realizados vários treinamentos para cada segmento de treino/teste e os seguintes valores de unidades de memória: 32, 64, 128, 256, 512, 1024. Analisando a tabela 9, verifica-se que a melhor acurácia e erro obtida foram para o conjunto de treino/teste de 10/90 e 1024 unidades de memória.

Tabela 9 – Erro e acurácia de cada segmento para a base Top204Thousand-WPA-probable-v2 variando o número de unidades de memória.

Treino/Teste	Unidades de memória	Erro	Acurácia(%)
10/90	32	2,1802	35%
	64	1,9711	42%
	128	1,6920	50%
	256	1,2910	64%
	512	0,8527	78%
	1024	0,7472	81%
90/10	32	2.2045	33%
	64	1,9812	41%
	128	1,8144	46%
	256	1,7032	49%
	512	1,6432	51%
	1024	1,4961	54%
20/80	32	2.2134	34%
	64	2,0316	40%
	128	1,7971	47%
	256	1,5336	55%
	512	1,1871	66%
	1024	1,0406	70%
80/20	32	2.1519	36%
	64	2.0157	40%
	128	1.8328	46%
	256	1.6610	50%
	512	1.4955	55%
	1024	1.4770	55%
30/70	32	2.1746	36%
	64	2.0092	41%
	128	1.8193	47%
	256	1.6062	52%
	512	1.3377	60%
	1024	1.1893	65%
70/30	32	2.1943	34%
	64	1.9967	41%
	128	1.8163	46%
	256	1.6543	51%
	512	1.5004	55%
	1024	1.4416	56%

A.2 Células

Em relação a quantidade de camadas ou células usadas no modelo, foram treinados vários modelos sobre as variações de segmento de treino/teste, com o melhor hiperâmetro de unidades de memória obtido na seção anterior e, 1 e 2 células, respectivamente. Observando a tabela 10, é possível identificar que foram obtidos dois ótimos perfis de resultados, sendo eles:

Perfil 1

- ❑ 10/90 para Treino/Teste;
- ❑ 1 célula;
- ❑ 1024 unidades de memória;

Perfil 2

- ❑ 30/70 para Treino/Teste;
- ❑ 2 células;
- ❑ 1024 unidades de memória;

Tabela 10 – Erro e acurácia de cada segmento para a base Top204Thousand-WPA-probable-v2 variando o número de células/camadas e utilizando 1024 unidades de memória obtido como melhor hiperparâmetro na seção A.1

Treino/Teste	Camadas	Erro	Acurácia(%)
10/90	1	0,7472	81%
	2	0,7612	79%
90/10	1	1,4961	54%
	2	1,4847	54%
20/80	1	1,0406	70%
	2	1,0708	68%
80/20	1	1,4770	55%
	2	1,4537	55%
30/70	1	1,1893	65%
	2	0,4712	87%
70/30	1	1,4416	56%
	2	1,4484	55%

A.3 Tamanho de sequência

Para o hiperparâmetro de tamanho de sequência, foram adotados os melhores valores obtidos nas seções anteriores e tamanhos de sequência de 25 e 100 para cada treinamento.

A tabela 11 demonstra que os melhores valores de erros e acurácias está concernente ao tamanho de sequência de 100. E, da mesma maneira, foram obtidos dois perfis ótimos, sendo os mesmos encontrados em seção A.2, e ambos com tamanho de sequência de 100.

Tabela 11 – Erro e acurácia de cada segmento para a base Top204Thousand-WPA-probable-v2 variando o número de sequência e utilizando os melhores hiperparâmetros da seção A.1 e seção A.2.

Treino/Teste	Tamanho de sequência	Erro	Acurácia(%)
10/90	25	0,7472	81%
	100	0,2083	95%
90/10	25	1,4961	54%
	100	0,8169	74%
20/80	25	1,0406	70%
	100	0,3280	92%
80/20	25	1,4770	55%
	100	0,7666	76%
30/70	25	1,1893	65%
	100	0,1798	96%
70/30	25	1,4416	56%
	100	0,7824	76%

A.4 Número de épocas

Do mesmo modo, o número ideal de épocas foram obtidos através da técnica de *Early Stopping* juntamente com os melhores hiperparâmetros obtidos anteriormente. Dessa maneira, a tabela 12 apresenta 3 segmentos de base onde foram obtidos os melhores resultados. Sendo eles: 10/90, 20/80 e 30/70.

Tabela 12 – Número de épocas, erro e acurácia de cada segmento para a base Top204Thousand-WPA-probable-v2 utilizando os melhores hiperparâmetros da seção A.1, seção A.2 e seção A.3.

Treino/Teste	Épocas	Erro	Acurácia(%)
10/90	76	0,2083	95%
90/10	38	0,8169	74%
20/80	56	0,3280	92%
80/20	36	0,7666	76%
30/70	51	0,4459	88%
70/30	32	0,7824	76%

A.5 Geração de Senhas

No contexto da geração das senhas, foram adotados a união de todos os dicionários gerados, como também a junção dos melhores resultados gerados. Ambos em função das combinações dos parâmetros de quantidade de caracteres e temperatura. As tabelas 13 e 14 apresentam os dicionários utilizados para a comparação, a quantidade de acerto e a taxa de acerto para cada geração realizada. Observa-se que para as duas análises realizadas, a 2ª geração obteve maiores acertos.

Tabela 13 – Taxa de acerto para os dicionários selecionados unindo todas as senhas geradas pela rede. No total foram geradas 31673 senhas para a 1ª geração, 29257 senhas para 2ª geração e 28368 senhas para a 3ª geração

Geração	Dicionário	Quantidade de acertos	Taxa de acerto(%)
1	Test	5333	16%
	Crackstation-Human-Only	3447	10%
	RockYou	2450	7%
	Top85MillionWPA	2974	9%
	Crackstation	11434	36%
2	Test	6578	22%
	Crackstation-Human-Only	4489	15%
	RockYou	3382	11%
	Top85MillionWPA	1888	6%
	Crackstation	14046	48%
3	Test	3177	11%
	Crackstation-Human-Only	2193	7%
	RockYou	1229	4%
	Top85MillionWPA	3466	12%
	Crackstation	6795	23%

Tabela 14 – Taxa de acerto para os dicionários selecionados utilizando os dicionários gerado pela rede que obtiveram maiores acertos. No conjunto, foram obtidas 3441 senhas para 1ª geração, 1943 para a 2ª geração e 1079 para 3ª geração

Geração	Dicionário	Quantidade de acertos	Taxa de acerto(%)
1	Test	1670	48%
	Crackstation-Human-Only	927	27%
	RockYou	753	21%
	Top85MillionWPA	445	12%
	Crackstation	3142	91%
2	Test	1079	55%
	Crackstation-Human-Only	656	33%
	RockYou	536	27%
	Top85MillionWPA	19	1%
	Crackstation	1940	99%
3	Test	229	21%
	Crackstation-Human-Only	107	9%
	RockYou	57	5%
	Top85MillionWPA	354	32%
	Crackstation	507	47%