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

Marlon B. Ramos[1], **Thiago P. Ribeiro** [1] **Rodrigo S. Miani** [1]

[1] Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU) – Uberlândia, MG – Brasil

{marlonbrendoramos, tpribeiro, miani}@ufu.br

Abstract. *The use of dictionaries in brute force attacks has shortcomings when addressing new passwords to be exploited, making the success of this type of attack impossible. In this scenario, this paper analyzes the use of recurrent neural networks to design new passwords based on the existing ones. The results show that the proposed model presents good hits, ensuring better chances of success in this type of attack.*

Resumo. *A utilização de dicionários em ataques de força bruta apresentam deficiências ao abordar novas senhas a serem exploradas, impossibilitando o sucesso desse tipo de ataque. Nesse cenário, o presente trabalho analisa a utilização das redes neurais recorrentes para concepção de novas senhas baseando-se previamente nas existentes. Os resultados demonstram que o modelo proposto apresenta bons acertos, garantindo melhores chances de sucesso nesse tipo de ataque.*

1. Introdução

Os mecanismos de quebra de senhas atualmente são extremamente visados em uma época onde o uso de criptografias seguras são cada vez mais intensificados. Nesse contexto, técnicas de força bruta, como a baseada em dicionários, surgem como um meio eficiente se comparado ao método tradicional [Narayanan and Shmatikov 2005]. No método tradicional, as senhas são geradas através de várias combinações e permutações de caracteres [Raza et al. 2012]. Todavia, o sucesso de ambos ataques se limitam as senhas contidas nos dicionários, impossibilitando que novas abordagens de senhas e possíveis relações semânticas entre as palavras presentes no dicionário possam ser criadas.

Dessa forma, o ataque de dicionário se fundamenta na obtenção de uma lista de possíveis senhas previamente construídas por meio de vazamentos ou de combinações de termos pessoais usando ferramentas como o *Common User Passwords Profile* (CUPP). Posteriormente, tais senhas são testadas uma a uma, visando descobrir a correta [Kessler and C. 1996]. A Figura 1 ilustra o ataque baseado em dicionário.

Por outro lado, por ser tratar de um método simples, senhas complexas ou variações das mesmas podem não estar contidas nos dicionários utilizados, inviabilizando o ataque. Desse modo, o presente trabalho explora as Redes Neurais Recorrentes, por meio da arquitetura *Long Short-Term Memory* (LSTM), em virtude da sua aplicabilidade em *Natural Language Processing* (NLP) e geração de textos, tomando uma sequência de palavras como entrada e tentando prever a possibilidade da próxima palavra ou caractere. E, dessa forma, através da construção de um modelo, obter padrões para prever e gerar

novas senhas e então analisar a viabilidade do seu uso no ataque de força bruta baseado em dicionário.

Ou seja, a ideia deste trabalho consiste em criar um modelo de Rede Neural Recorrente sobre a arquitetura LSMT capaz de obter padrões de escritas baseadas no treinamento de listas de senhas usados em ataques de força bruta baseado em dicionários. E, desse modo, realizar várias gerações de novas senhas e analisar as taxas de acertos para diversas bases de teste, buscando verificar se tais senhas geradas pelo modelo possuem boas taxas previsão e podem ser usadas ou agregradas em um novo ataque.

Trabalhos como [Trieu and Yang 2018] e [Hitaj et al. 2017] lidaram, de forma parcial com o problema de geração de senhas. O trabalho de [Trieu and Yang 2018] se apresenta de forma semelhante a este trabalho, propondo a utilização do algoritmo Torch-RNN para a geração de senhas. As comparações demonstraram uma alta taxa de acerto no ataque de Força Bruta baseado em AI se comparado ao método tradicional. Por outro lado, o trabalho de [Hitaj et al. 2017] introduziu a ferramenta *PassGAN* por meio das Redes Adversárias Generativas (GANs) para gerar senhas "inteligentes" e precisamente "parecidas" com senhas geradas por humanos. Diferentemente de tais trabalhos, a presente proposta foca nos melhores parâmetros para construção do modelo e diferentes abordagens e combinação para geração das senhas, além da utilização de diferentes dicionários para treinamento e validação do modelo. Permitindo assim, em virtude da aleatoriedade e combinação, inúmeros resultados relevantes para o contexto do problema.

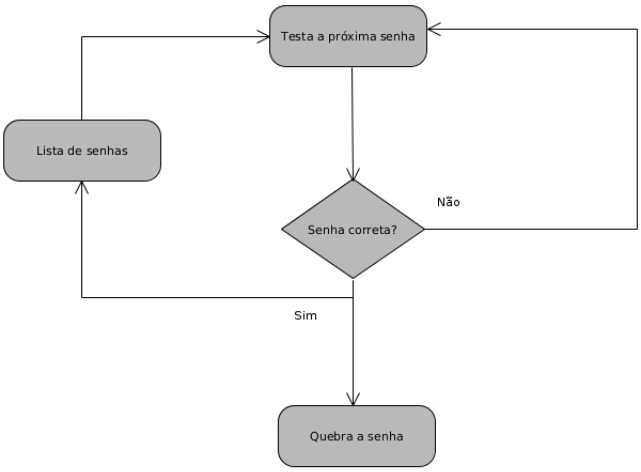

Figura 1. Digrama do método de Força Bruta baseado em dicionário. Fonte: Do Autor

2. Proposta

A modelo proposto neste estudo consiste em uma Rede Neural capaz de compreender as sentenças de entrada, ou seja, as senhas e, por meio do treinamento da rede, gerar uma saída que descreverá as probabilidades de cada caractere do vocabulário ser a próxima. Assim sendo, o modelo é chamado várias vezes para gerar uma sequência de caracteres, formando assim uma senha, como representado na Figura 2. Em suma, o objetivo principal deste trabalho é realizar uma análise comparativa das senhas geradas pela Rede Neural com alguns dos dicionários mais relevantes que são frequentemente utilizados um ataque

de força bruta baseado em dicionários. E por meio dos resultados sustentar a hipótese de que os dicionários de senhas geradas por aprendizado de máquina podem ser utilizadas propriamente ou agregadas no ataque de dicionários para a descoberta de senhas.

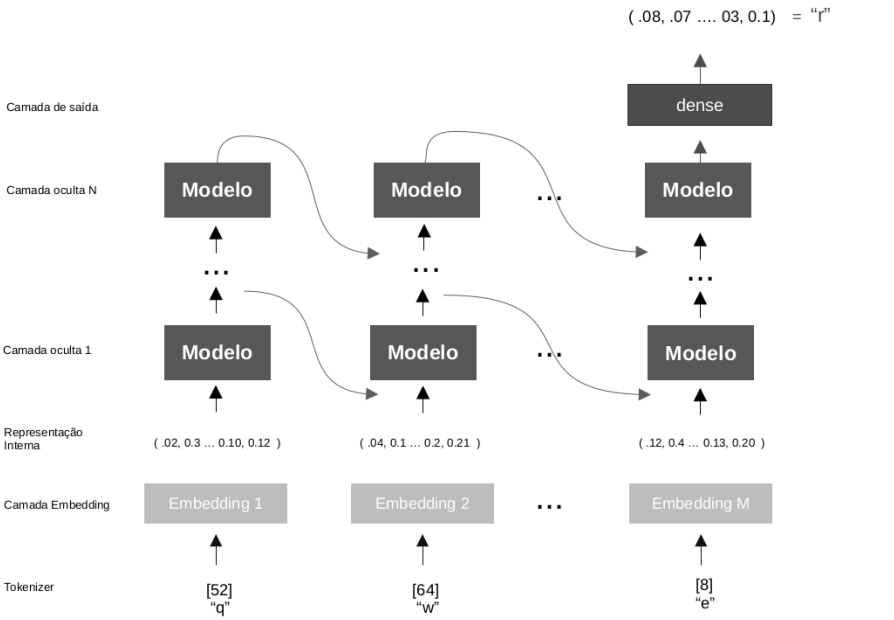

Figura 2. Diagrama de representação do modelo de RNN/LSTM para geração de senhas. Fonte: Do Autor

Em resumo, dada uma sequência de caracteres como entrada, o modelo realiza a tokenização, transformando as letras em números através da camada de entrada *Embedding*. Após essa etapa, os valores são transferidos para as camadas ocultas LSTM seguintes. E, por fim, através de uma camada densa de saída com o número de neurônios sendo o tamanho do vocabulário, ou seja, o número de caracteres únicos, foi aplicado a função de ativação Softmax, retornando as probabilidades de cada caractere no vocabulário.

3. Experimentos

As próximas subseções descrevem como foram determinados os hiperparâmetros para a construção e execução do modelo. Apresentam, também, os testes realizados de forma empírica, sendo que alguns dos hiperparâmetros foram escolhidos por meio de uma análise criteriosa da literatura.

3.1. Base de dados

O conjunto de senhas utilizados para o treinamento foram obtidas das listas *Top2Billion-probable-v2* [GitHub 2019] e *Top204Thousand-WPA-probable-v2* [GitHub 2018] na codificação ASCII de variados tamanhos com letras maiúsculas e minúsculas, números, acentos e caracteres especiais. Para este trabalho, foram segmentados 100 milhões de senhas do primeiro dicionário, em virtude da ausência de poder computacional. Já para o segundo, foram utilizados todas as senhas contidas no mesmo. A princípio, para a obtenção dos melhores hiperparâmetros, foi utilizado a base *Top204Thousand-WPA-probable-v2* por conter um menor número de senhas, levando assim, menos tempo para o modelo ser treinado. Após essa etapa, o modelo então foi treinado com a base *Top2Billion-probable-v2* com os melhores hiperparâmetros obtidos.

3.2. Arquitura do modelo

Antes de iniciar o treinamento da rede, foram aplicada uma camada de vetorização textual, para converter texto puro em vetores numéricos por meio da representação *Word Embeddings*. Com a adição dessa camada, foi necessário definir a dimensão do vetor na qual as palavras serão mapeadas. De forma arbitrária, para acelerar o treinamento, foi escolhido uma dimensão potência de 2. Nesse caso, foi definido o valor da dimensão como sendo 256. Após a vetorização textual, o modelo foi construído sobre camadas ocultas e uma camada densa de saída com o tamanho do vocabulário de caracteres usado.

3.3. Hiperparâmetros

Para este trabalho, foram abordados os seguintes hiperparâmetros: tamanho de sequência, número de unidades de memória, número de células, tamanho do lote, otimizador e função de ativação. Sendo o tamanho do lote e função de ativação obtidos na literatura. Os demais foram obtidos por meio do treinamento de vários modelos e, o valor ótimo foi obtido pela análise do menor erro dentre os modelos treinados. Desta 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.

Para o número de unidades de memória ou neurônios, foram realizados treinamentos com os seguintes valores: 32, 64, 128, 256, 512 e 1024. Dentre os segmentos de base treinados, o menor erro foi obtido na segmentação 10/90 com valor de 1024 de unidades de memória como observado na Figura 3.

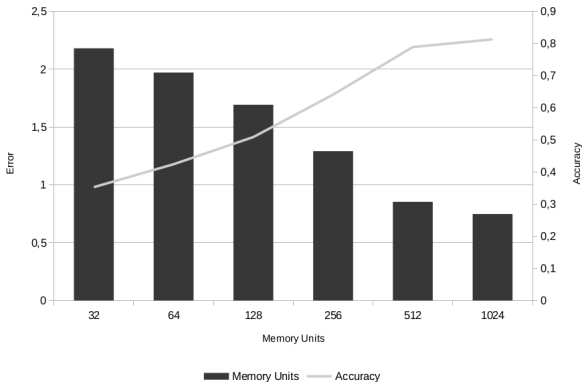

Figura 3. Erros obtidos para cada unidade de memória com Treino/Teste de 10/90. Fonte: Do Autor

Para o número de células ou camadas do modelo, foram conduzidos 2 treinamentos com 1 e 2 camadas, respectivamente. Os menores erros foram obtidos no segmento 30/70 e 10/90 com 1024 unidades de memória para cada camada do modelo, como demonstrado na Tabela 1

Tabela 1. 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%

Em relação o tamanho do lote ou *Batch Size*, para este trabalho, foi adotado o valor ótimo como sendo 32. Pois de forma empírica, tamanhos de lotes menores geram treinamentos mais rápidos além de apresentarem maiores generalizações para a base de dados [Bengio 2012].

Quanto ao otimizador, em Redes Neurais Recorrentes, os otimizadores Adam, SGD e o RMSProp são frequentemente utilizados, dentre eles, o Adam apresenta melhores resultados se comparado com os outros algoritmos estocásticos [Kingma and Ba 2014]. Em virtude disso, o otimizador Adam foi escolhido para os testes realizados neste trabalho.

Para o tamanho de sequência, foram abordados dois valores baseando-se em dois trabalhos de processamento de linguagem natural. Sendo eles, o tamanho de sequência de 25 obtido do trabalho [Greg 2018] e o tamanho de 100 do trabalho de [Jason Brownlee 2018]. Para os conjuntos de segmentos de base treinados, constatou-se que o melhor valor foi obtido foi de 100 no segmento 30/70 (Figura 4).

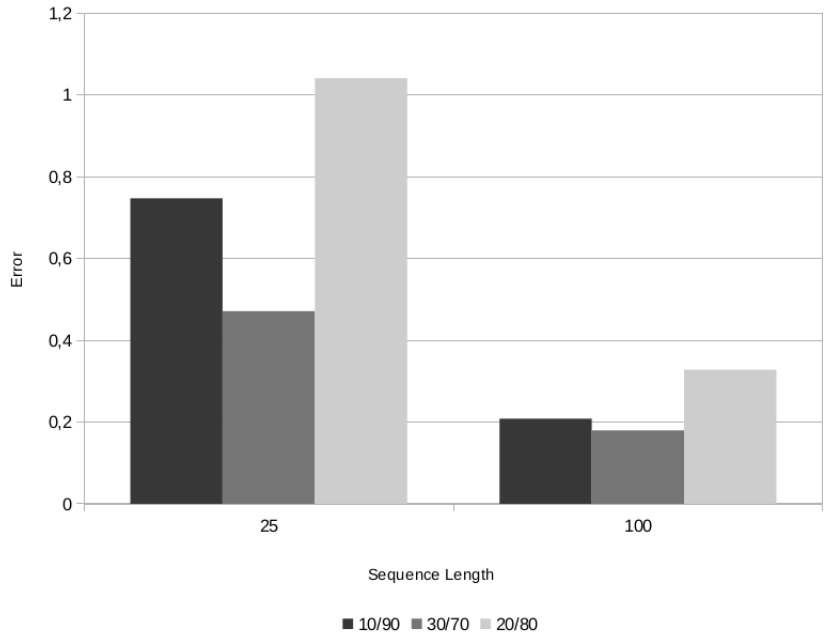

Figura 4. Comparação de erros obtidos para tamanho de sequência de 25 e 100 para bases segmentadas em 10/90, 30/70 e 20/80.Fonte: Do Autor

Por fim, com o objetivo de não gerar *Overfitting* e *Underfitting* no modelo vigente, foi utilizada a técnica *Early Stopping* para obtenção do número ótimo de épocas para este trabalho. Dessa maneira, de modo arbitrário, se a cada 5 épocas o erro do modelo não sofrer uma diminuição, o treinamento é parado.

3.4. Aplicação das Definições na Base Top2Billionprobable-v2

Após a obtenção dos melhores hiperparâmetros, foram obtidos dois perfis de valores. Desse modo, devido a ausência de poder computacional, foi utilizado somente o primeiro perfil de valores para o treinamento da segunda base que contém um número maior de senhas.

Tabela 2. 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

Com isso, foi treinado outro modelo sobre a base *Top2Billion-probable-v2* com os hiperparâmetros: 10/90 para Treino/Teste, 1 célula, 1024 unidades de memória, tamanho de sequência de 100, otimizador Adam e tamanho do lote de 32. Portanto, foram necessárias 14 épocas, das quais apenas 9 foram suficientes para encontrar o menor erro, em virtude da técnica de parada precoce. Dessa maneira, o menor erro encontrado foi de 0,98 com acurácia de 68%.

4. Avaliação dos Resultados

Para a geração das senhas, foram realizadas combinações do parâmetro de temperatura que controla a distribuição de probabilidade para a previsão do próximo caractere e a quantidade de caracteres que vão ser gerados. Para a temperatura, foram utilizados valores de 0,1 á 1,5 com variação de 0,1 e para o número de caracteres, foi estipulado o início em 500 até 5000 com variação de 500. Ademais, outra abordagem utilizada foi a utilização de sementes, que permitirão gerar um ponto de partida para o modelo gerar as próximas senhas. Nesse cenário, as sementes são senhas que foram obtidas da base de teste de forma aleatória.

Sendo assim, em virtude da aleatoriedade das sementes e dos algoritmos de aprendizado de máquina em geral não serem determinísticos, foram realizadas 3 gerações de senhas utilizando 5 e 15 sementes, respectivamente. Após a geração das senhas, notou-se que usando 15 sementes obtiveram mais acertos do que 5 sementes. A taxa de acerto nesse contexto foi realizada usando a métrica *Hit Ratio* (HR), que provê a razão do número de senhas corretas geradas em relação a base de teste sobre o número total de senhas geradas. Portanto, a princípio, as próximas gerações realizadas para avaliação dos resultados foram utilizando 15 sementes.

Para a interpretação dos dados, os resultados foram unidos em dois dicionários de senhas. Na Tabela 3, para o item `Todos`, foram unidos todos os dicionários da 2° geração que obteve melhor resultado dentre as que foram geradas 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.

Tabela 3. HR unindo dicionários da 2° geração de senhas para a base de teste extraído de *Top2Billion-probable-v2*.

	Quantidade de senhas	Quantidade de acertos	HR
Todos	29257	6578	22%
Melhores	1943	1079	55%

5. Comparação de dicionários

Com o objetivo de verificar a generalização do modelo e a capacidade do mesmo de prever e gerar senhas, foram obtidos 4 grandes dicionários utilizados em ataques de força bruta para as devidas comparações. Assim sendo, foram gerados novos dicionários combinando os mesmos parâmetros de temperatura e quantidade de caracteres. Por outro lado, as sementes foram obtidas aleatoriamente do dicionário a ser comparado. Dessa maneira, as Tabelas 4 e 5 apresentam as taxas de acertos e acertos para os dicionários [Crackstation-Human-Only 2019], [RockYou 2022], [Crackstation 2019] e [Top85MillionWPA 2018]. Nesse contexto, a taxa de acerto (`HR`) representa a razão da quantidade de senhas corretas em relação a base comparada sobre o número de senhas gerados pelo modelo. Já os `Acertos`, representam apenas quantidade de senhas corretas comparadas com a base em questão. Do mesmo modo descrito na Seção 4, os resultados foram apresentados por meio dos itens `Todos` e `Melhores`.

Tabela 4. HR unindo todos os dicionários da 2° geração para 4 diferentes dicionários de senhas

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

Tabela 5. HR unindo os melhores dicionários da 2° geração para 4 diferentes dicionários de senhas

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

6. Conclusão

Neste trabalho as Redes Neurais Recorrentes sobre a arquitetura LSTM foram exploradas para a construção de um modelo de Processamento de Linguagem Natural capaz de abstrair padrões de escritas de senhas e, dessa maneira, gerar e prever senhas que poderão ser utilizadas em um ataque de dicionário. A experimentação empírica que pode ser conduzida neste trabalho demonstrou resultados satisfatórios na previsão de senhas, garantindo que as senhas geradas possam ser utilizadas ou agregadas com dicionários comumente utilizados, garantindo maiores possibilidades de sucesso nesse tipo de ataque. Como trabalhos futuros, pretende-se: (I) Utilizar o segundo perfil obtido a partir dos experimentos realizados na seção 3.4, (II) Alterar a Taxa de aprendizagem e *Momentum* e (III) Verificar reversões de Hash por meio de tabelas arco-íris.

Referências

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533.

Crackstation (2019). crackstation. Disponível em: https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm. Acessado em 23/08/2022.

Crackstation-Human-Only (2019). crackstation-human-only. Disponível em: https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm. Acessado em 13/08/2022.

GitHub (2018). Probable-wordlists. Disponível em: https://github.com/berzerk0/Probable-Wordlists/blob/master/Real-Passwords/WPA-Length/Top204Thousand-WPA-probable-v2.txt. Acessado em 02/07/2022.

GitHub (2019). Probable-wordlists. Disponível em: https://github.com/berzerk0/Probable-Wordlists/tree/master/Real-Passwords. Acessado em 5/09/2021.

Greg (2018). Password cracker. Disponível em: https://github.com/gsurma/password_cracker. Acessado em 31/08/2022.

Hitaj, B., Gasti, P., Ateniese, G., and Pérez-Cruz, F. (2017). Passgan: A deep learning approach for password guessing. *CoRR*, abs/1709.00440.

Jason Brownlee (2018). Text generation with lstm recurrent neural networks in python with keras. Disponível em: https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/. Acessado em 31/08/2022.

Kessler and C., G. (1996). Passwords - strengths and weaknesses.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Narayanan, A. and Shmatikov, V. (2005). Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, page 364–372, New York, NY, USA. Association for Computing Machinery.

Raza, M., Iqbal, M., Sharif, M., and Haider, W. (2012). A survey of password attacks and comparative analysis on methods for secure authentication. *World Applied Sciences Journal*, 19:439–444.

RockYou (2022). Rockyou. Disponível em: https://weakpass.com/wordlist/90. Acessado em 13/08/2022.

Top85MillionWPA (2018). Top85millionwpa. Disponível em: https://github.com/berzerk0/Probable-Wordlists/tree/master/Real-Passwords/WPA-Length. Acessado em 13/08/2022.

Trieu, K. and Yang, Y. (2018). Artificial intelligence-based password brute force attacks.