Detecting Phishing Websites through Reinforcement Learning and NLP

Nikunj Rathod
M.Tech (ICT) ML
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, 382007
Mentor: Prof. Gopinath Panda, Prof. Abhishek Jindal
202211014@daiict.ac.in

***Abstract*—Phishing is a type of cybercrime in which people visits or clicks the website/URL that is phishing URL but it looks like normal URLS but it is actually harmful and steals all the sensitive information of the user within no time. This type of cyber-attack threats almost all internet users and institutions. To reduce the risk of revealing confidential information, there is a need of the system to successfully detect the phishing url before clicking it or opening it. Here in this paper, I have used NLP based word2vec technique to extract the features and then applied Reinforcement learning based deep q-network to train the data. Many test have been applied to the several architecture and it is seen that the best architecture among the tested ones gives the success rate of 89%. I have used dataset from phistank where phishing web-urls are being updated on daily basis.**

Keywords—Phishing, Legitimate, Reinforcement Learning, deep Q-Network, NLP, Word2Vec

Chapter I: Reinforcement Learning

Introduction to Reinforcement Learning

Reinforcement Learning is somewhat similar to Machine Learning but different from the supervised learning where the model is trained from the labelled dataset. Reinforcement Learning includes Agent, Action and Environment. Here, agent is the user in the Environment and taking some actions and based on the actions, agent gets a numerical reward. Based on the rewards it gets, the agent learns if the action was worth taking or not. The reward is the kind of feedback to the user (here agent). The reward can be positive or negative. If the reward is positive, it means it is beneficial to the user and the action it has taken in good to take in future. But if the user gets negative reward, it means the action taken was not so good and user will avoid taking that action in the future. The ultimate goal of the system is to maximize the future reward.

Here in Figure 1, we can see the basic Diagram of Reinforcement Learning where Agent takes an Action (A_t) in the Environment from the current State (S_t) and according to the action gets a Reward (R_t) and goes to the next state$(S_t + 1)$.

Let's understand each term of Reinforcement Learning.

<u>Environment</u>	: In which the agent operates and interacts with to learn and make decisions.
<u>Agent</u>	: An agent is an entity that interacts with the environment and learns to make decisions based on feedback in the form of rewards or penalties.
<u>State</u>	: Position of an Agent at a given time T.
<u>Action</u>	**:** Agent's choice of activity in state S.
<u>Reward</u>	**:** Price of taking right/wrong action.

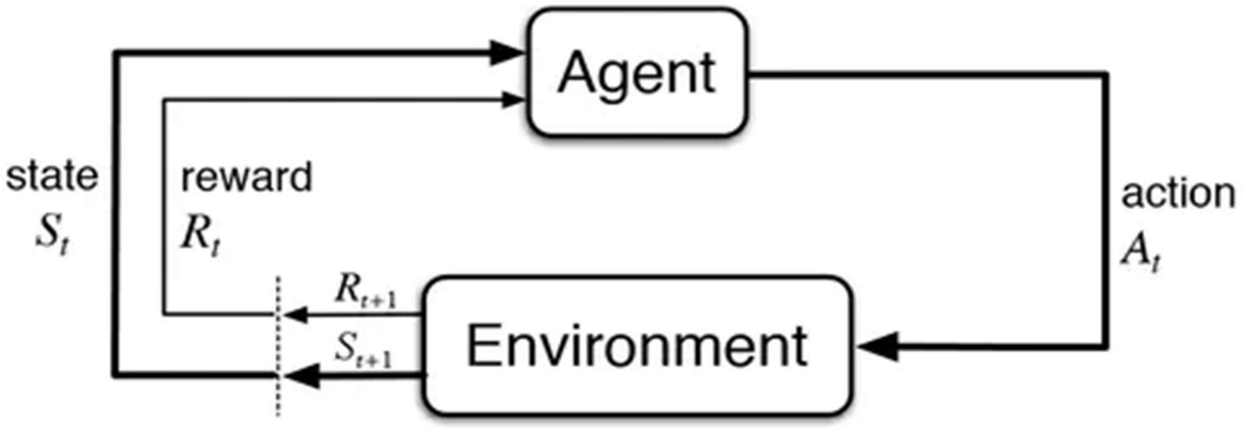


Figure 1. Basic Diagram of Reinforcement Learning[4].

Let's break down this diagram into steps.

1. At time **t**, the environment is in state S_t.
2. The agent observes the current state and selects action A_t.
3. The environment transitions to state $S_t + 1$ and grants the agent reward $R_t + 1$.
4. This process then starts over for the next time step, **t+1**.

- Note, **t+1** is no longer in the future, but is now the present. When we cross the dotted line on the bottom left, the diagram shows **t+1** transforming into the current time step **t** so that $S_t + 1$ and $R_t + 1$ are now S_t and R_t**.**

Elements to Reinforcement Learning

Beyond the above elements, there are other main sub elements of reinforcement learning system: A policy; A Reward function and Value function.

Policy defines the learning agent's way of behaving at a given time and state. Simply it is about how the agent will act in a given state at a given time. The action it takes is dependent on the policy it follows. Policy can be a simple function or table. In general, Policy may be stochastic.

Reward function is the function for calculating the reward that is received by the Agent. Basic goal of the Reward function is to maximize the total Reward it receives. Basically, the function assigns the numeric value (either positive or negative) to the state-value pair. This function defines what the good and bad events are for the agent. For example, If the agent gets a low or negative reward for taking an action following policy, then the policy may be changed to select some other action in the future in the same situation.

Value function is the term to specify what is good for the long run. It is total expected reward for the agent in the future, starting from the current state. We seek actions that bring out the highest value, not highest reward, because action obtains the greatest amount of reward for over the long run.

Reward Function

We saw that the agent's objective is to maximize the expected return of rewards it receives. If the sequence of rewards it receives after time step *t* is denoted by $r_{t+1},\ r_{t+2},\ r_{t+3},\ \dots\ ,\ r_T$ then, the return is the sum of the rewards:

$$\boldsymbol{R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T}$$

where, T is the final time step.

Another concept comes here is that of a discounting. The agent tries to select the actions so that the sum of the discounted rewards it receives over the future is maximized. It chooses A_t to maximize the expected discounted return:

$$\begin{aligned} \boldsymbol{G_T} &= \boldsymbol{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots} \\ &= \sum_{\boldsymbol{k=0}}^{\infty} \boldsymbol{\gamma^k r_{t+k+1}} \end{aligned}$$

where γ is the discounting factor having value $0 < \gamma < 1$.

The discounted rate determines the present value of the future rewards. A reward received k time steps in the future are worth only k-1 times what it would be worth if it were received immediately.

$$\begin{aligned} G_T &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots \\ G_T &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots) \\ G_T &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

If $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence $\{r_k\}$ is bounded. If $\gamma = 0$, the agent is being concerned only with maximizing the immediate reward, not the future rewards. As approaches 1, the objective takes future rewards into consideration more strongly: the agent become more farsighted.

$$\boldsymbol{G_T} = \sum_{\boldsymbol{k=0}}^{\infty} \boldsymbol{\gamma^k} = \frac{\boldsymbol{1}}{\boldsymbol{1-\gamma}}$$

Transition Function

By applying action $a \in A$ in a state $s \in S$, the system makes a transition from s to a new state $s' \in S$, based on a probability distribution over the set of possible transitions.

The system being called is Markov if the result of an action does not depend on the previous actions and visited states (history), but only depends on the current (last) state, i.e.

$$\boldsymbol{P(s_t, a_t, s_{t-1}, a_{t-1}, \cdots) = P(s_{t+1} | s_t, a_t)}$$

$$\boldsymbol{= T(s_t, a_t, s_{t+1})}$$

The idea of MDP is that the current state s gives enough information to make an optimal decision; if is not important which states and actions preceded s. Another way of saying this, is that if you select an action a, the probability distribution over next states is the same as the last time you tried this action in the same state.

Policies

Policy is defined as how probable it is for an Agent to take a specific action from any given state. It is a function that maps probabilities of selecting each possible actions from a given state and it is denoted by π.

By saying that Agent is following the policy π, we elaborate that the probability of Agent taking an action A from the given state S at time t.

$\boldsymbol{\pi(a, s)}$, where $A_t = a$ and $S_t = s$

For each state $s \in S$, π is a probability distribution over $a \in A(s)$.

State Value Function

The State Value Function is defined as how good the current state is for Agent following the policy $\boldsymbol{\pi}$. It gives us the value of the state under $\boldsymbol{\pi}$. The value of state $\boldsymbol{s}$ under policy $\boldsymbol{\pi}$ is the expected return from starting from state $\boldsymbol{s}$ at time $\boldsymbol{t}$ and following policy $\boldsymbol{\pi}$ thereafter. Mathematically we define s as

$$\boldsymbol{\vartheta_\pi(s) = E_\pi(G_t | S_t = s)}$$

$$\boldsymbol{= E_\pi\left[\sum_{k=0}^{\infty} \gamma^r r_{t+k+1} | S_t = s\right]}$$

Action Value Function

The action-value function for policy π, denoted as q_π tells us how good it is for the agent to take any given action from a given state while following policy π. In other words, it gives us the value of an action under π. The value of action ***a*** in state ***s*** under policy π is the expected return from starting from state ***s*** at time ***t***, taking action ***a***, and following policy π thereafter. Mathematically, we define $q_\pi(s,a)$ as

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

$$= E_\pi\left[\sum_{k=0}^{\infty} \gamma^r r_{t+k+1} | S_t = s, A_t = a\right]$$

The above-mentioned function is called Q-function and the output of the function is called the Q-value. The letter "Q" represents the quality of taking an action in a given state.

Bellman Equation

The optimal action-value function q_* satisfies the below given equation. This is a fundamental property of the optimal action-value function.

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

Bellman Optimality Equation

This equation tells that, at time ***t,*** for any state-action pair ***(s, a)***, the expected return from starting state ***s***, taking action ***a,*** and with the optimal policy afterward will be equal to the expected reward ***Rt+1*** we can get by selecting action ***a*** in state ***s***, in addition with the maximum of "expected discounted return" that is achievable of any ***(s', a')*** where ***(s', a')*** is a potential next state-action pair.

Considering the agent following an optimal policy, the latter state ***s'*** will be the state from we can take the best possible next action ***a'*** at time ***t+1***. This seems somewhat difficult to understand.

The above property can be observed in the equation as we find **q*(s', a')** which denotes the expected return after choosing an action ***a*** in state ***s*** which is then maximized to gain the optimalQ-value.

$$q^{new}(s,a) = (1-\alpha)q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)$$

q-value update equation

Q-learning

Q-learning is a reinforcement learning technique used to make decisions in an environment with the goal of maximizing a cumulative reward. It is a model-free approach, meaning that it doesn't require a model of the environment and learns directly from interactions.

The Q-learning algorithm typically involves the following steps:

Initialize the Q-value: Initialize Q-values for all state-action pairs arbitrarily.

Exploration vs Exploitation: The agent selects an action based on its exploration-exploitation strategy. It can either explore new actions to learn more about the environment or exploit the current knowledge to choose the action with the highest Q-value.

Action Execution and Observation: The agent executes the selected action, observes the next state, and receives the immediate reward from the environment.

Update Q-Value: The Q-value of the taken action in the current state is updated using the Bellman equation:

Repeat: Steps 2-4 are repeated until the agent converges to an optimal policy.

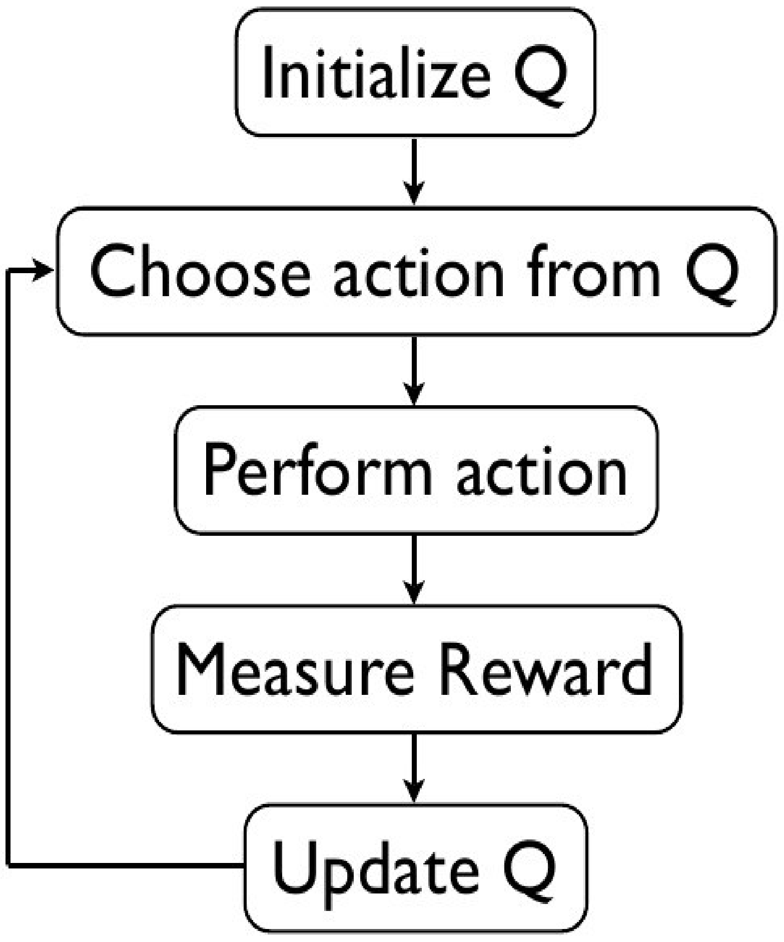


Figure 2: Steps in Q-updation[17]

Q-learning is particularly suitable for problems with discrete state and action spaces. It has been successfully applied in various domains, including robotics, game playing, and control systems. Extensions of Q-learning, such as Deep Q Networks (DQN), use neural networks to handle more complex environments with continuous state spaces or high-dimensional input spaces.

Deep Q-Network

Deep Q-Learning (DQN) is an extension of the traditional Q-learning algorithm that uses deep neural networks to approximate the Q-function. This technique is particularly useful when dealing with complex environments with high-dimensional state spaces, such as images in computer vision tasks or raw sensor data in robotics.

Here are the key components and features of Deep Q-Learning:

Neural Network Architecture (Deep Q-Network):

- In DQN, a deep neural network is used to approximate the Q-function Q(s,a).
- The neural network takes the state s as input and outputs Q-values for each possible action a.

- Commonly, convolutional neural networks (CNNs) are used when the input is an image or a high-dimensional state.

Experience Replay:

- DQN introduces the concept of experience replay to stabilize and improve learning.
- During the agent's interaction with the environment, experiences (tuples of state, action, reward, next state) are stored in a replay buffer.
- Instead of updating the Q-values based on the most recent experience, random batches of experiences are sampled from the replay buffer for training.

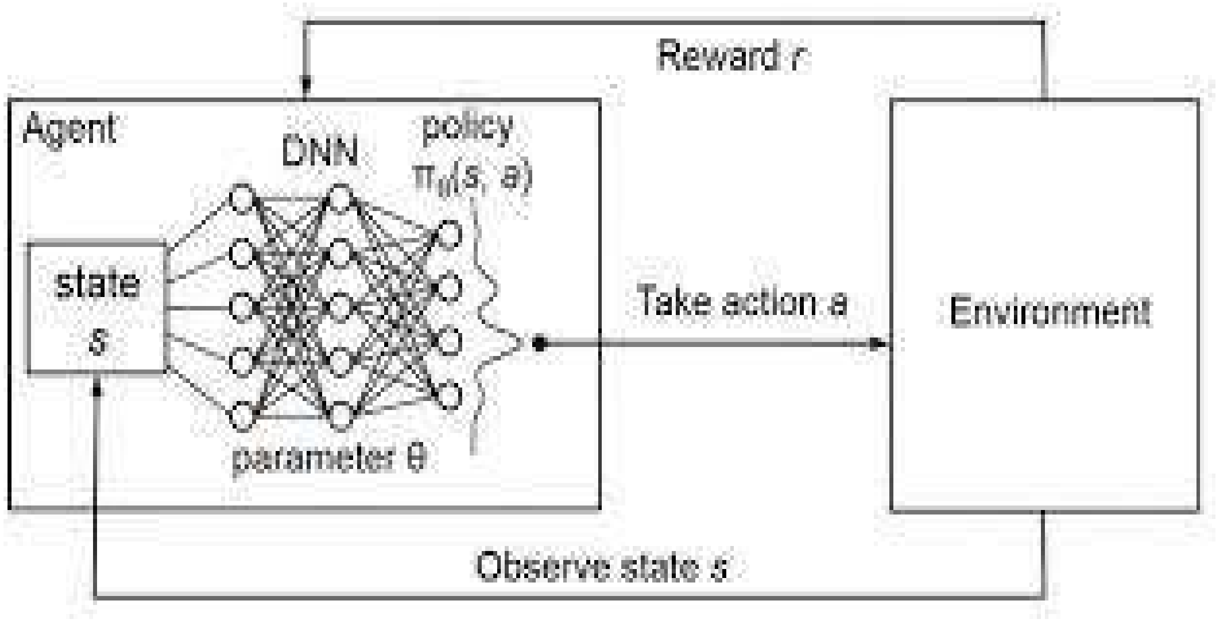


Figure 3: Deep Q-Network Diagram[18]

Target Network:

- To stabilize the learning process, DQN uses a separate target network.
- The target network is a copy of the main Q-network, but its parameters are updated less frequently.
- The target network is used to compute the target Q-values in the Bellman equation during the training process.

Loss Function:

- The loss function used for training the Q-network is a variant of the mean squared error.
- The goal is to minimize the difference between the predicted Q-values and the target Q-values.
- The target Q-values are computed using the Bellman equation with the target network.

Exploration-Exploitation:

- The agent still needs to balance exploration and exploitation. Commonly, an epsilon-greedy strategy is used, where the agent chooses the action with the highest Q-value with probability $1-\epsilon$ and explores a random action with probability ϵ.

Training Stability and Techniques:

- DQN incorporates several techniques to enhance training stability, such as gradient clipping and the use of a separate target network.

- Experience replay and target networks contribute to more stable and efficient learning.

Deep Q-Learning has been successfully applied to a variety of tasks, including playing Atari games, robotic control, and autonomous navigation. It allows agents to learn complex behaviors from raw sensory input, making it a powerful technique in the field of reinforcement learning.

Implementation of Gameplay using Q-learning

Reinforcement leaning is commonly used in gaming area. I learned a game named frozen-lake in which reinforcement learning is applied. I studied that game and implemented the code it in the python language [17].

The description of the game is *"Winter is here. You and your friends were tossing around a frisbee at the park when you made a wild throw that left the frisbee out in the middle of the lake. The water is mostly frozen, but there are a few holes where the ice has melted. If you step into one of those holes, you'll fall into the freezing water. At this time, there's an international frisbee shortage, so it's absolutely imperative that you navigate across the lake and retrieve the disc. However, the ice is slippery, so you won't always move in the direction you intend. The surface is described using a grid like the following:"*

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

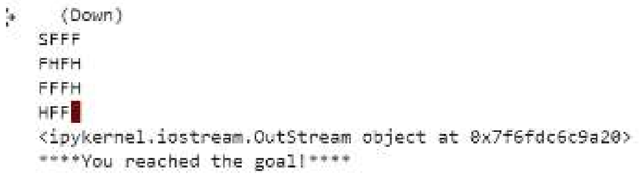
Table 1. 4x4 grid for gameplay.

Here, we have a 4x4 grid. Each grid is a state in terms of reinforcement learning. We have total of 16 states in our game.

Here, S is out agent's starting state and G is the destination, where F represents the frozen surface and it is safe to put leg on the F surfaces while H represents Hole in the ice and it is dangerous to the agent. The agent will receive the reward of +1 if it successfully reached to the G state without getting into the hole. The episode ends when the agent either reaches to the Goal or fails into the Hole and next episode starts.

For implementing this game, I have used OpenAI's gymnasium library, simply called as gym. Gym library provides wide range of gaming environments used by reinforcement learning from basic games to advanced level games.

Here is the snapshot of the output of the game played.



```
  (Down)
SFFF
FHFH
FFFH
HFFG
<ipykernel.iostream.OutStream object at 0x7f6fdc6c9a20>
****You reached the goal!****
```

Image1. Output of the frozen-lake game played[17].

After few experiments, I observed that the maximum rewards are being given with the following values of the hyper-parameters.

Hyper parameter	Value
num_episodes	10000
max_steps_per_episode	99
learning_rate	0.01
discount_rate	0.95
exploration_rate	1
Max_exploration_rate	1
Min_exploration_rate	0.001

Exploration_decay_rate	0.001

Table 2. Hyper-parameter values for maximum reward. [17]

Chapter II: Phishing

Introduction to Phishing

Phishing is a simple form of cyber-attack in which people receives email/text-message containing the url/link that seems to be true or came from legitimate source but actually has been created by attacker who wants to access user's personal data that may include username, password, financial IDs, credit card or debit card's transaction data, user's web activities, social activities, e-commerce, photos or other personal information as well. It is essential in today's era to have online presence all the time and therefor it has been easy for attackers to steal information through phishing urls. Phishers are discovering different techniques in creating websites to fool the users and tempting them. A new fraudulent website is expected to be generated every second.

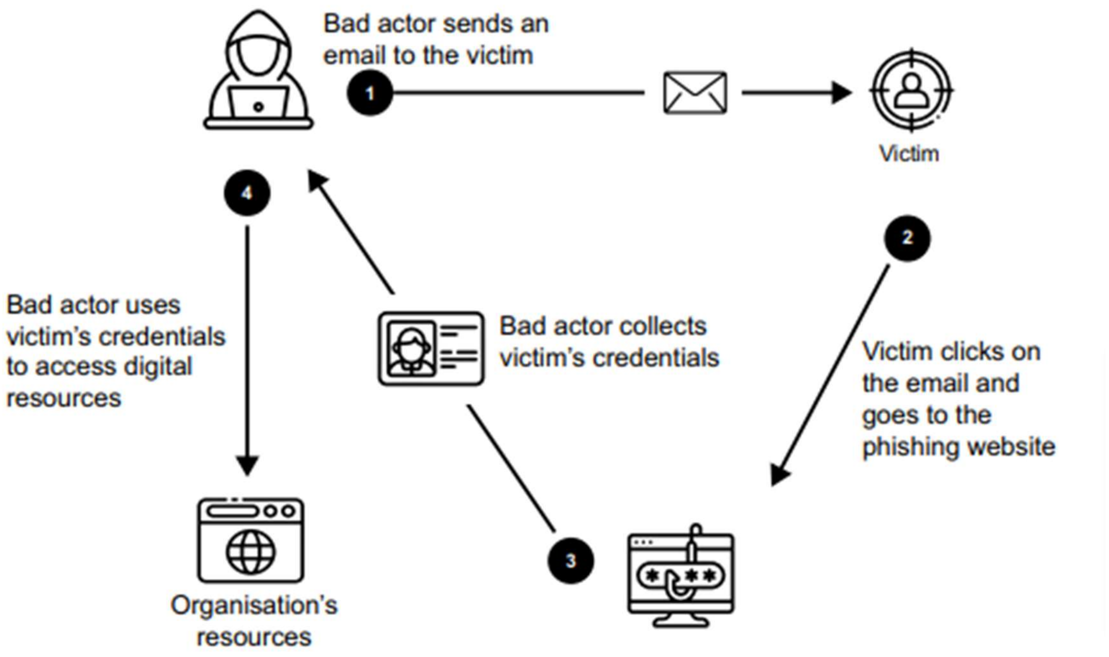

Bad actor sends an email to the victim
1
Victim
2
Victim clicks on the email and goes to the phishing website
3
Bad actor collects victim's credentials
4
Bad actor uses victim's credentials to access digital resources
Organisation's resources

Figure 4: Steps Involved in Phishing

Here are some common types of phishing:

1. **Email Phishing:**

- **Spear Phishing:** Targets a specific individual or organization, often using personalized information to make the fraudulent communication more convincing.
- **Whaling:** Targets high-profile individuals such as executives or CEOs within an organization.
- **Clone Phishing:** Involves creating a replica of a legitimate email or communication to trick the recipient into providing sensitive information.

2. **Smishing (SMS Phishing):**

Attackers use text messages to trick individuals into clicking on malicious links or providing sensitive information.

3. **Vishing (Voice Phishing):**

Attackers use phone calls to trick individuals into providing sensitive information or performing actions such as transferring money.

4. **Social Media Phishing:**

Attackers create fake social media profiles or pages to impersonate a legitimate entity and trick users into divulging personal information.

5. **Pharming:**

Involves redirecting website traffic to fraudulent websites, often through the manipulation of DNS (Domain Name System) settings.

6. **Man-in-the-Middle (MitM) Attacks:**

An attacker intercepts communication between two parties, gaining access to sensitive information.

7. **Search Engine Phishing:**

Attackers create fake websites that appear in search engine results, tricking users into entering sensitive information.

8. **Watering Hole Attacks:**

Attackers compromise websites that the target group is known to visit, infecting these sites with malware to capture sensitive information.

9. **Angler Phishing:**

Exploits the trust between a user and a legitimate service provider by posing as customer support and tricking users into providing login credentials.

10. **USB Phishing (Baiting):**

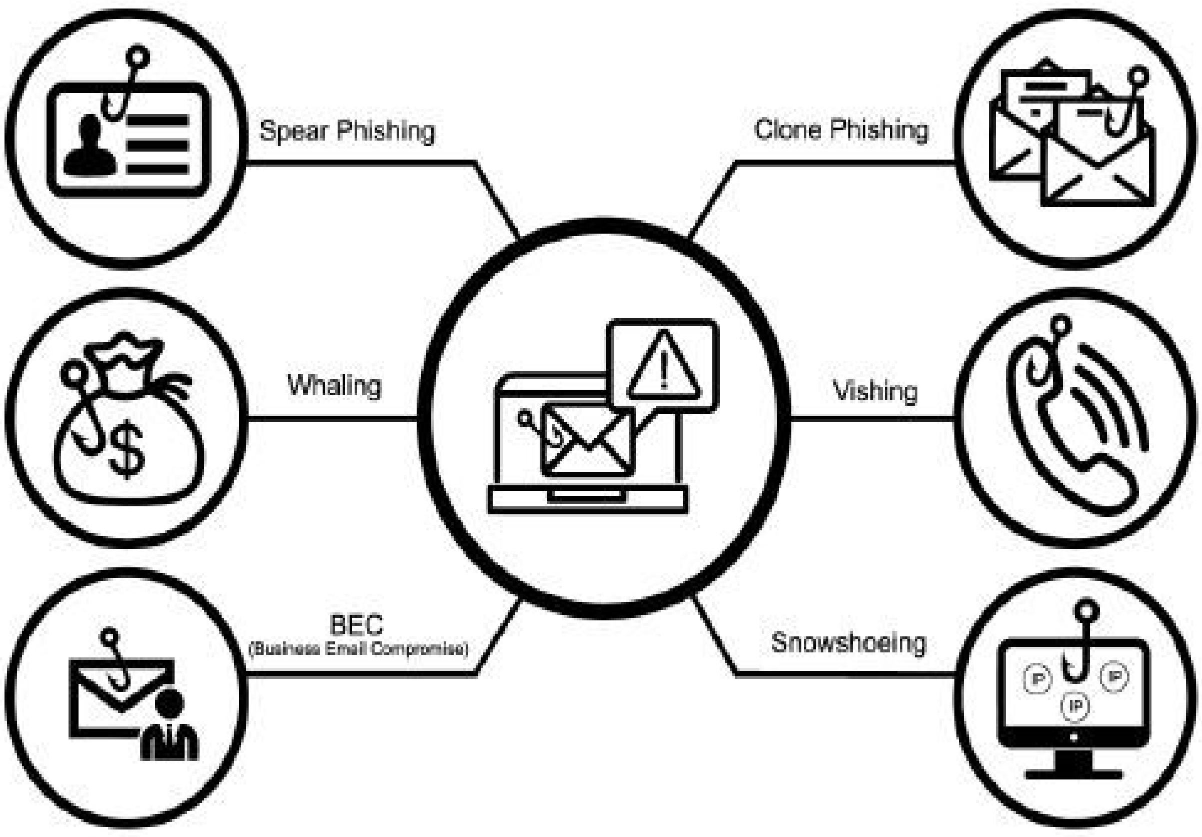


Figure 5: Types of Phishing Attacks

Attackers leave infected USB devices in public places, hoping someone will pick it up, connect it to their computer, and unknowingly install malware.

It's crucial for individuals and organizations to be aware of these phishing techniques and employ security measures such as email filters, multi-factor authentication, and cybersecurity training to mitigate the risks associated with phishing attacks.

Literature Survey

In recent years, there has been research on applying machine learning to classify the phishing problem. Furthermore, with the advancement of the computer power, various deep learning methods have been used in classification of phishing websites. Deep-learning based approaches are particularly effective when the number of features is larger in size.

There are mainly two approaches for detecting phishing websites [13]. (1) Blacklisting. In this method, we compare the given website from the previously detected phishing url list and if we found it in the list, we mark them as a phishing url and legitimate otherwise. In this method the chances of classifying the phishing url correctly is very less because it is not mandatory that the given url is previously seen as tons of new phishing urls are being generated every day. (2) Second method is heuristic based. Extraction of various features from the given url to predict whether the given url is phishing or legitimate. This method is also applicable on the newly created phishing url as it extracts the features in real-time.

Features assessment technique used in this paper[12] for automatic phishing detection is of four types addressbar based features, abnormal based features, HTML and Javascript based features and Domain related features and they successfully abled to collect and analyze 17 different features from above mentioned categories.

In this paper, I have used word2vec technique of NLP followed by q-learning concept of Reinforcement Learning for training and testing of the data.

DATASET

The dataset was created with the representation of the URL into different features. Given here is the URL structure.

I have used dataset provided by the Phishtank website[16], where there is 86760 listed phishing URLs, and for benign urls there were over 35300 listed URLs. I have randomly picked 35000 phishing urls for my training to overcome the imbalance data issue. The dataset is discrete and contains deterministic classes. The proposed model can make binary predictions of the test data.

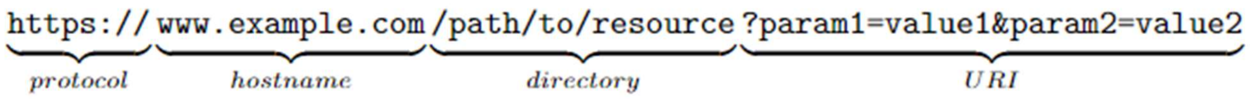
Source of dataset	No. of good urls	No. of bad urls	Total No. of urls
PhishTank	35378	86760	122138

Table 2: Dataset Information

A typical URL has two parts (1) Protocol: Specifies the protocol to be used for communication between user and web server. (2) Resource identifier: indicating the IP address or the domain space where the resource is located. A colon and two forward slashes separate the protocol from the resource identifier, as shown in the Image.

Total no. of samples	No. of training samples	No. of testing samples
10000	5000	5000

Table 3: Dataset Split Details



Img 2. URL Structure[13]

The extracted features are categorised into

- ***Addressbar based features***
- ***HTML and JavaScript based features***

1. ***Addressbar based Features***

Many features can be extracted that can be considered as address-bar based features. Out of them. Below mentioned features are used in this project. I have used Python for extracting all the features and used 'urllib', 'ipaddress' and 'regular expression' library for extracting the address-bar based features.

I. IP Address in URL

I check for the presence of IP address in the url, if it found in the url, there is a big chance that someone is trying your personal information using this. 1 will be assigned to this feather if found else 0.

II. "@" Symbol in URL

I check for the '@' symbol in the url, '@' symbol makes browser to ignore whatever written before it. And only the content written after the symbol will be valid and processed. 1 will be assigned if it found else 0.

III. Length of URL

Phisher use long urls to hide the harmful content between it, so if the length of the url exceeds the 54 characters then this feature is set to 1 or it is set to 0.

IV. Depth of URL

This computes the depth of the URL. This feature calculates the number of subpages in the given url based on the '/'.

V. Redirection "//" in URL

Checks if '//' found in the url other than once in the beginning. If found feature is set to 1 or else 0.

VI. "http/https" in Domain name

The phishers can use the protocols in the domain names to trick the user and steal the information, so if this found in the domain name of the url, then the feature is set to 1 or else it is set to 0.

VII. Using URL Shortening Services "TinyURL"

URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL.

VIII. Prefix or Suffix "-" in Domain

The dash symbol is mostly used in phishing urls to make then look like legitimate so this feature is set to 1 if dash (-) is found or else 0.

1. ***HTML and JavaScript based features***

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project. In these types of features, first the response text is extracted from the url using the urllib and requests library of python and then processing is done on that response text.

I. IFrame Redirection

IFrame tag is use to show a webpage in another webpage using frame, phishers can use this frame and can make it invisible to fool the users. I found for this in the response and set this feature to 1 if found, or 0.

II. Status bar Customization

Phishers may use JavaScript to display a fake URL in the status bar to the users. To extract this feature, we must explore the webpage source code particularly the 'onmouseover' event and check if it makes any changes to the status bar.

III. Disabling Right Click

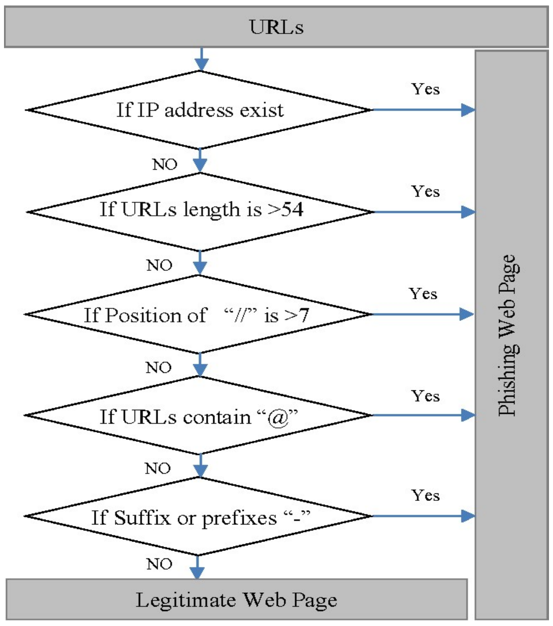
Phishers use JavaScript to disable the right-click function, so that the users cannot view and save the source code. This feature is treated exactly as 'Using onMouseOver to hide the Link'. However, for this feature, we will search for event 'event.button = = 2' in the source code and check if right click is disabled. 1 is set if found else 0.

IV. Website Forwarding

Website forwarding found on websites are liable to be exploited by phishers to create a link to their site. This feature is rarely used in legitimate websites. Thus, if the redirection number is less than 2 then we will assign '0', otherwise we will assign '1'.

After extracting all the above mentioned features from all the urls I finally had the dataset for performing the Reinforcement task. The shape of the dataset is 70000x13 which means 70000 urls with each of 12 features and one column for true label either 0 (legitimate) or 1 (for phishing). All the values of the columns have been made numerical so that neural network can be applied on the same.

The deep Network Model contains one input layer, two hidden layers with ReLU activation function and an output layer with 'adam' optimizer and 'mse' loss function. The data set is not yet prepared as of now so that I cannot show the actual results here in figures.



Img 3. Phishing URL Checking Algorithm[6].

Chapter III: NLP

INTRODUCTION TO NLP

NLP stands for Natural Language Processing.

NLP is a subfield of artificial intelligence (AI) that focuses on the interaction between computers and humans through natural language. The ultimate objective of NLP is to enable computers to understand, interpret, and generate human language in a way that is both valuable and meaningful.

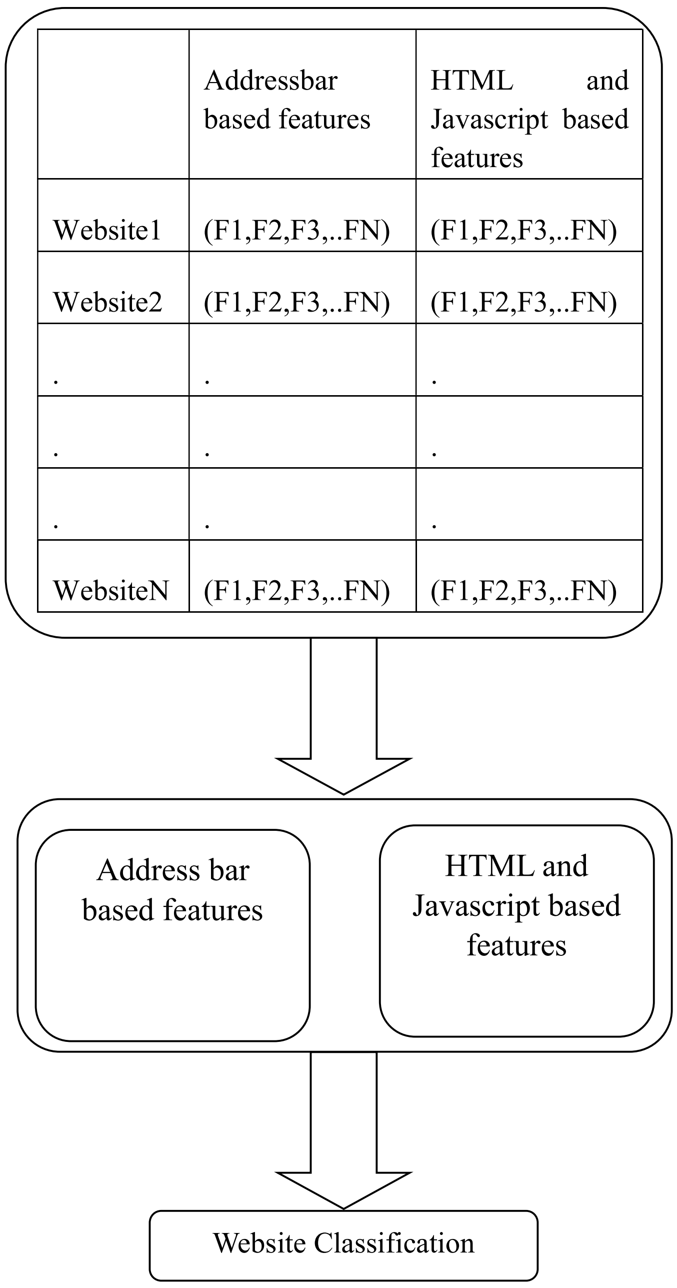
NLP involves several challenges, including:

1. **Tokenization:** Breaking down text into individual words or phrases (tokens).

2. **Syntax and Parsing:** Understanding the grammatical structure and relationships between words in a sentence.

3. Semantics: Extracting meaning from text, considering the context and the relationships between words.

4. Named Entity Recognition (NER): Identifying entities such as names of people, organizations, and locations in text.



	Addressbar based features	HTML and Javascript based features
Website1	(F1,F2,F3,..FN)	(F1,F2,F3,..FN)
Website2	(F1,F2,F3,..FN)	(F1,F2,F3,..FN)
.	.	.
.	.	.
.	.	.
WebsiteN	(F1,F2,F3,..FN)	(F1,F2,F3,..FN)

Address bar based features
HTML and Javascript based features
Website Classification

Figure 2. Proposed phishing prediction hierarchical model[12].

5. Coreference Resolution: Determining when different words or phrases in a text refer to the same entity.

6. Sentiment Analysis: Determining the sentiment or emotion expressed in a piece of text.

7. Machine Translation: Translating text from one language to another automatically.

NLP is widely used in various applications, including virtual assistants (like Siri or Alexa), chatbots, language translation services, sentiment analysis, and more. It plays a crucial role in making human-computer interactions more natural and intuitive.

INTRODUCTION TO WORD2VEC

"Word2Vec" is a popular technique in natural language processing (NLP) that is used to represent words as vectors in a continuous vector space. It was introduced by researchers at Google, including Tomas Mikolov, in a series of papers published in 2013.

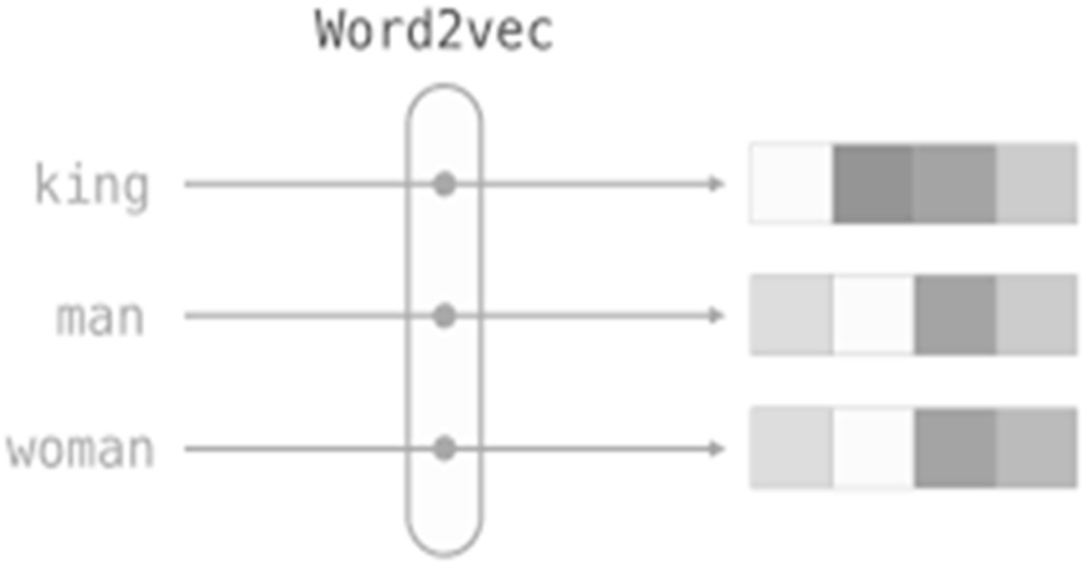
The basic idea behind Word2Vec is to capture the semantic relationships between words by representing them as dense vectors, where similar words are mapped to nearby points in the vector space. This is in contrast to traditional methods that often represent words as discrete symbols or one-hot vectors, which don't capture the relationships between words.

There are two main architectures for implementing Word2Vec:

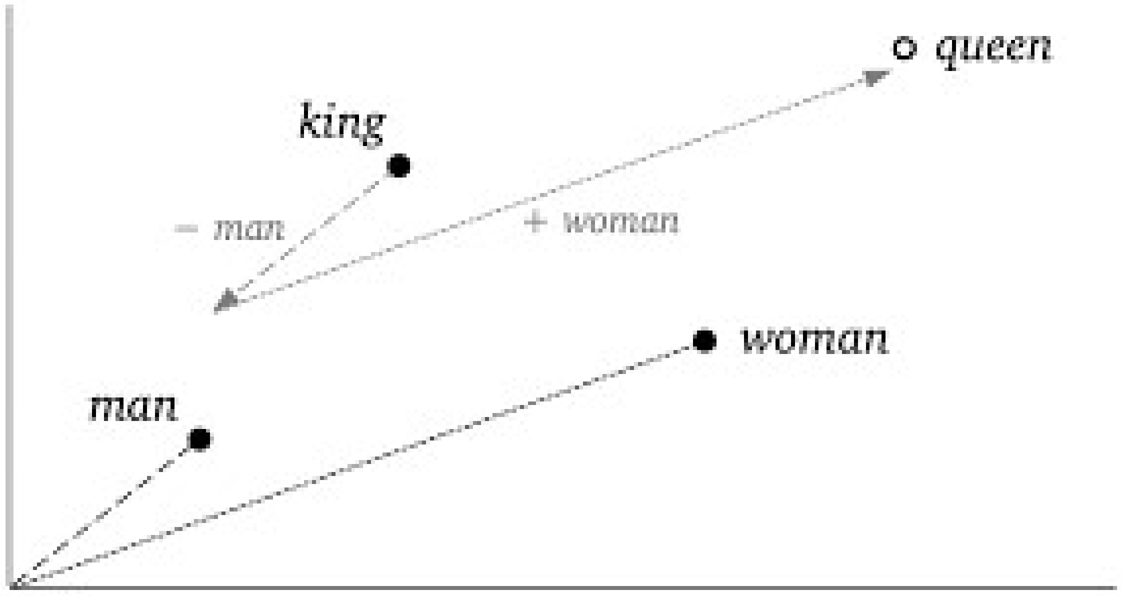
- **Continuous Bag of Words (CBOW):** This model predicts the target word based on its context, i.e., the surrounding words. It takes a context (a set of words) as input and predicts the target word.
- **Skip-Gram:** In this model, the target word is used to predict the context words. Given a target word, the model tries to predict the words that are likely to appear around it.

Both CBOW and Skip-Gram use a shallow neural network to learn the word embeddings. The resulting word vectors are capable of capturing semantic relationships between words. Words with similar meanings or contexts are represented as vectors close to each other in the high-dimensional space.

Word2Vec embeddings have been widely used in various natural language processing tasks, such as machine translation, sentiment analysis, and document clustering, due to their ability to capture semantic information and relationships between words.



Img 4: Word to Vector [8]



Img 5. Word Vector Mathematics Concept [7].

Chapter III: Proposed Model

Methodology

1. Dataset Loading and Sampling:

Two datasets, one for phishing URLs and one for legitimate URLs , are loaded from CSV files. Each dataset contains a list of URLs.

To manage computational resources and create a balanced dataset, a random sample of 5000 URLs is taken from each dataset.

2. URL Preprocessing:

The URLs are pre-processed by splitting them into individual words. This is done by the **preprocess_url** function, which splits a URL using the forward slash ('/') as a delimiter.

The result is a list of pre-processed URLs, where each URL is represented as a list of its constituent words.

3. Word2Vec Model Training:

All the pre-processed URLs, both phishing and legitimate, are combined into a single list (**all_urls**).

A Word2Vec model is trained on this combined list. The Word2Vec model learns to represent words as dense vectors in a continuous vector space. Key parameters include:

vector_size: The dimensionality of the word vectors.

window: The maximum distance between the current and predicted word within a sentence.

min_count: Ignores all words with a total frequency lower than this.

vector_size	window	min_count	sg
11	5	1	0

Table 1. Hyperparameters of Word2vec

4. Word Embeddings Generation:

The trained Word2Vec model is used to generate embeddings for each word in the pre-processed URLs.

The **url_to_word_embeddings** function converts a URL into a vector representation by taking the mean of the word vectors for each word in the URL. If a word doesn't have an embedding, a zero vector is used.

The result is a set of vector representations for each URL, capturing semantic relationships between words.

5. Labels Preparation:

Labels are assigned to each URL. Phishing URLs are labelled as **1**, and legitimate URLs are labelled as **0**.

The length of the labels corresponds to the total number of preprocessed URLs (phishing + legitimate).

6. Data Export:

The generated features and labels are organized into Pandas Data Frames.

The labels are exported to a CSV file, which can be used as ground truth for training a classification model.

Important Considerations:

The Word2Vec model is trained to understand the semantic relationships between words in the URLs. Adjusting parameters such as **vector_size** and **window** can impact the quality of learned embeddings.

The URLs are tokenized into words for training the Word2Vec model. This choice of tokenization can affect the quality of learned representations.

The resulting embeddings serve as features for a machine learning model. Commonly, a classification model is trained to predict whether a URL is phishing or legitimate based on these embeddings.

Ensure the datasets contain relevant information and are representative of the task at hand.

This methodology is a simplified representation of preparing data for a URL classification task using Word2Vec embeddings. Depending on specific requirements and characteristics of the data, additional steps or adjustments may be necessary.

7. Training Process:

Data Preparation:

The code loads training and testing data from CSV files, representing input features and labels.

The data is split into training and testing sets using the **train_test_split** function.

Q-Learning Agent:

The Q-learning agent is defined with methods to select actions (**select_action**), train the model (**train**), and decay the exploration rate (**epsilon**).

The agent uses a DQN model, implemented with TensorFlow/Keras, to approximate the Q-function.

8. Training Loop:

Episodes and States:

The training loop iterates through episodes, where each episode corresponds to a row in the training dataset.

The state of the environment is set to the input features of the current episode.

Action Selection:

The agent selects actions based on its policy, balancing exploration (random actions) and exploitation (actions based on learned Q-values).

Training the Model:

The Q-learning agent updates the Q-values based on the observed rewards and the difference between predicted and target Q-values.

The model is trained to minimize the mean squared error between predicted and target Q-values.

Exploration and Exploitation:

The exploration rate (**epsilon**) is decayed over time to shift from exploration to exploitation as the agent learns more about the environment.

9. Evaluation:

Testing the Model:

Total no. of samples	No. of training samples	No. of testing samples
10000	5000	5000

Table 3: Dataset Split Details

After training, the model is evaluated on a separate testing dataset.

The **evaluate_agent** function calculates the accuracy by comparing predicted actions with true labels.

Performance Metrics:

The code uses a simple reward scheme (1 for correct action, -1 for incorrect) to guide the learning process.

The evaluation provides insights into the model's ability to generalize to new, unseen data.

10. Output:

Printed Information:

During the training loop, information such as selected actions and total rewards for each episode are printed.

The final test accuracy is printed after the evaluation.

Important Considerations:

Data Representation:

The input data is assumed to be in a tabular format with rows representing different instances and columns representing features.

Parameter Tuning:

Model hyperparameters such as the number of hidden layers, nodes per layer, and learning rates may need to be tuned based on the specific characteristics of the dataset.

Dataset Specifics:

Dummy data is used in the code for illustration. Replace this with your actual data and adjust parameters accordingly for meaningful results.

Understanding and adjusting these aspects will help in adapting the code to specific use cases and datasets.

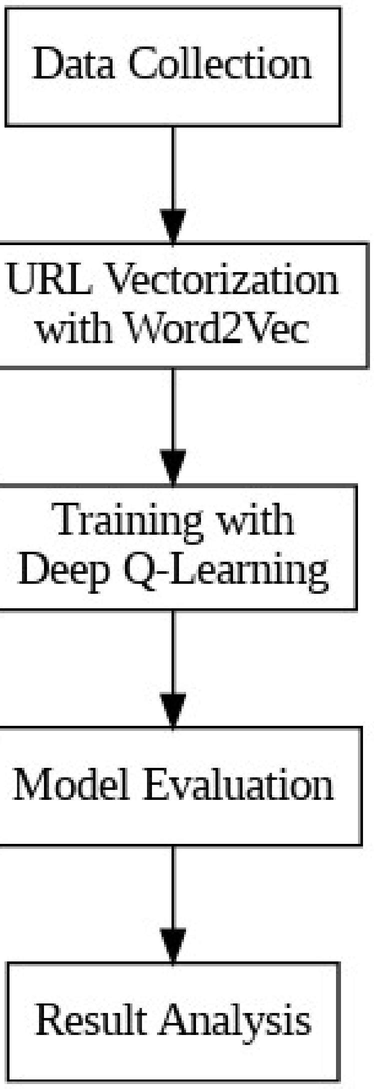

Data Collection
URL Vectorization with Word2Vec
Training with Deep Q-Learning
Model Evaluation
Result Analysis

Figure 2. Dataflow Diagram

Network Architecture

The Q-network architecture includes input layer, four hidden layer with 128,256,128,64 nodes respectively and relu as activation function and output layer with 2 nodes as it will give binary output. Adam optimizer and Mean_Square_Error used as a loss function while creating the architecture of the model.

Conclusion

This project set out to harness the capabilities of reinforcement learning, particularly Deep Q-Learning, for the critical task of URL classification—discerning between phishing and legitimate URLs. The journey unfolded through meticulous steps, each contributing to the overarching goal of enhancing cybersecurity measures.

Beginning with the foundation of data preprocessing, the project strategically sampled datasets containing both phishing and legitimate URLs. This thoughtful approach ensured a well-balanced and representative dataset for subsequent training and evaluation stages.

The adoption of Word2Vec embeddings proved instrumental in transforming textual information into numerical vectors. This embedding technique, coupled with the power of deep neural networks, laid the groundwork for the development of a robust URL classification model.

At the heart of the project's methodology was the implementation of Deep Q-Learning. The neural network, acting as a powerful approximator of the Q-function, underwent iterative updates based on interactions with the environment. This dynamic learning process enabled the model to discern patterns and make informed decisions regarding the nature of URLs.

Training and evaluation phases were pivotal, revealing insights into the model's performance and generalization capabilities. The delicate balance between exploration and exploitation, coupled with the continuous refinement of Q-values, contributed to the project's success in achieving a certain level of accuracy in URL classification.

While acknowledging the accomplishments, it's important to recognize the challenges encountered along the way. Issues such as data quality, hyperparameter tuning, and potential overfitting underscore the complexities of the cybersecurity landscape. This awareness opens the door to future improvements and refinements, suggesting avenues for further research.

In essence, this project stands as a testament to the potential of reinforcement learning in bolstering cybersecurity defenses. The combination of innovative techniques, rigorous experimentation, and a commitment to understanding the intricacies of URL classification has laid a foundation for future advancements in the field. As we navigate the evolving landscape of cyber threats, the insights gained from this project contribute to the ongoing dialogue surrounding the role of machine learning in threat detection.

Acknowledgment

I would like to express my heartfelt gratitude to the individuals and organizations whose unwavering support and contributions have played a pivotal role in the successful completion of this project.

I extend my sincere thanks to my project supervisor Prof. Abhishek Jindal sir and Prof. Gopinath Panda sir, for their invaluable guidance, mentorship, and expertise. Their insightful feedback and constructive criticism were instrumental in refining the project's direction and ensuring its overall quality.

I am deeply thankful to the for fostering an environment of academic excellence. The access to resources, facilities, and the encouragement received from the institution significantly contributed to the project's success.

My appreciation goes out to my classmates and colleagues who provided not only academic support but also a collaborative spirit that made this project an enriching experience. The exchange of ideas and shared enthusiasm for the subject matter greatly enhanced the project's depth and scope.

To my family, whose unwavering support and understanding have been my pillars of strength throughout this journey. Their encouragement and belief in my abilities motivated me to overcome challenges and persist in the pursuit of excellence.

I want to express my gratitude to my friends for their camaraderie, encouragement, and the countless brainstorming sessions. Their positive influence and shared passion for learning made the project more enjoyable and meaningful.

In conclusion, I am truly fortunate to have been surrounded by a network of supportive individuals and organizations. Each one of you has left an indelible mark on this project, and I am profoundly grateful for your contributions.

Thank you for being an integral part of this academic project.

REFERENCES

1. Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto [A Bradford book; Cambridge, Massachusetts London, England]

2. Deeplizard Reinforcement Learning Series[Online] Available at: https://deeplizard.com/learn/video/nyjbcRQ-uQ8

3. Deepmind Reinforcement Learning material [Online] Available at: https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021

4. P. Kaushik, "Title of the Blog Post," Published: Sep 20, 2021. [Online]. Available: [https://www.mcg.ai/post/reinforcement-learning]. Accessed: August 10, 2023.

5. L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996. [https://www.jair.org/index.php/jair/article/view/10166/24110]

6. A. A. Ahmed and N. A. Abdullah, "Real time detection of phishing websites," 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2016, pp. 1-6, doi: 10.1109/IEMCON.2016.7746247.

7. J. Gimpel, "A PRIMER ON WORD EMBEDDINGS: What's Behind Word2vec," Towards Data Science, Jul. 26, 2022. [Online]. Available: https://towardsdatascience.com/whats-behind-word2vec-95e3326a833a Accessed: October 12,2023.

8. J. Alammar, "The Illustrated Word2vec," Visualizing machine learning one concept at a time, Available: https://jalammar.github.io/illustrated-word2vec/ , Accessed: October 4,2023.

9. H. Dave, "Understanding the Bellman Optimality Equation in Reinforcement Learning," Data Science Blogathon, Updated On February 15th, 2021. [Online]. Available: [https://www.analyticsvidhya.com/blog/2021/02/understanding-the-bellman-optimality-equation-in-reinforcement-learning/]. Accessed: October 12, 2023.

10. A. Lakshmanarao, M. R. Babu and M. M. Bala Krishna, "Malicious URL Detection using NLP, Machine Learning and FLASK," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2021, pp. 1-4, doi: 10.1109/ICSES52305.2021.9633889.

11. H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," arXiv:1509.06461, [Online]. Available: https://arxiv.org/abs/1509.06461

12. R. M. Mohammad, F. Thabtah and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," 2012 International Conference for Internet Technology and Secured Transactions, London, UK, 2012, pp. 492-497.

13. M. Chatterjee and A. -S. Namin, "Detecting Phishing Websites through Deep Reinforcement Learning," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 227-232, doi: 10.1109/COMPSAC.2019.10211.

14. A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in Proceedings of the ACM Conference on Computer and Communications Security, 2010, pp. 54-60. doi: 10.1145/1866423.1866434.

15. A Maci, A. Santorsola, A. Coscia, and A. Iannacone, "Unbalanced Web Phishing Classification through Deep Reinforcement Learning," *Computers*, vol. 12, no. 6, p. 118, Jun. 2023, doi: 10.3390/computers12060118. [Online]. Available: http://dx.doi.org/10.3390/computers12060118

16. PhishTank | Join the fight against phishing. (n.d.). Retrived July 3,2023. From https://www.phishtank.com/

17. J. Shaikh, "Simple Beginner's guide to Reinforcement Learning & its implementation," *Shaikh's Data Science Insights*, Updated On April 5th, 2023. [Online]. Available: [https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/]. Accessed: November 12, 2023.

18. H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning," *IEEE Transactions on Networking* [Online] Available [http://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf]

19. My github repository [https://github.com/NikunjRathod200/reinforcementLearning]