Information Security Project

IE3092

Assignment

Information Security Project report

J.A.D.S.K Nanayakkara

IT21119194

Contents

Table of Figure

Abstract: *In the era of escalating cyber threats, robust solutions are imperative to detect malicious URLs promptly. This project presents a Deep Learning Malicious URL Detection system, leveraging advanced techniques in natural language processing and deep learning. With an emphasis on accuracy and efficiency, the system employs bidirectional LSTM networks and real-time processing, enhancing cybersecurity measures and ensuring swift identification of malicious URLs.*

Acknowledgement

The Deep Learning Malicious URL Detection project extends its sincere gratitude to the invaluable contributors who made this endeavor possible. The project acknowledges the diligent efforts of developers, data annotators, and testers whose expertise shaped the system. Special recognition is extended to the TensorFlow and Flask communities, whose open-source contributions significantly influenced the project's success.

1. Introduction

1.1. Overview

The Deep Learning Malicious URL Checker represents a cutting-edge solution at the intersection of artificial intelligence and cybersecurity. It serves as a sophisticated tool designed to combat the escalating threats posed by malicious URLs in today's digital landscape. At its core, this innovative product leverages advanced deep learning techniques, notably Bidirectional Long Short-Term Memory (Bi-LSTM) networks, to meticulously analyze and classify URLs in real-time. Unlike traditional methods, which often rely on static patterns, the Deep Learning Malicious URL Checker dynamically learns complex patterns within URLs, enabling it to identify potential threats with unprecedented accuracy.

The main goal of this product is to protect users from various online dangers like fake emails, harmful software, and tricks that try to trick you into giving out personal information. All these dangers usually start from website links. By using deep learning, the Malicious URL Checker can find and sort harmful website addresses. This makes a difference clients know what kind of threat they might experience. The easy-to-use interface permits everybody, counting people, businesses, and endeavors, to associated with it easily.

What sets this item separated is its capacity to ceaselessly advance its discovery capabilities. Through machine learning algorithms, it refines its understanding of emerging threats, staying one step ahead of cybercriminals. By emphasizing the innovative integration of deep learning algorithms within the realm of cybersecurity, the Deep Learning Malicious URL Checker redefines the standards of URL analysis, safeguarding users from an ever-expanding array of online threats.

1.2. Background and Motivation

The improvement of the Profound Learning Malevolent URL Checker was propelled by the disturbing rise in cyber dangers related with malevolent URLs. With the multiplication of online exercises, dangers like phishing assaults, malware dispersion, and social building strategies have ended up more modern and far reaching. Insights uncover a exasperating slant: a noteworthy increment in phishing endeavors focusing on both people and organizations. Cybercriminals

abuse tricky URLs to trap clients into unveiling delicate data, driving to money related misfortunes, personality robbery, and compromised security foundations. Real-world incidents underscore the urgency for advanced URL detection methods; traditional security measures often struggle to keep pace with the evolving tactics of malicious actors. Consequently, there is an imperative need for innovative solutions powered by deep learning technologies to combat these threats effectively. The Deep Learning Malicious URL Checker emerged as a response to this critical cybersecurity challenge, aiming to provide a proactive and robust defense against the ever-growing landscape of malicious URLs.

1.3. Objectives and Goals

The primary objectives of our Deep Learning Malicious URL Checker were meticulously designed to address the growing challenges in cybersecurity. Our foremost goal was to achieve an unparalleled accuracy rate in malicious URL detection, surpassing industry standards. We aimed to develop a cutting-edge solution capable of identifying a diverse range of malicious URLs, including sophisticated phishing attempts and malware distribution links. Real-time analysis stood as another pivotal objective, ensuring immediate threat identification and response, thus enhancing overall online safety. In parallel, we focused on creating an intuitive and user-friendly interface, catering to users with varying technical backgrounds. Ease of use was paramount to encourage widespread adoption and empower users to navigate the tool effortlessly. Throughout the development process, our team set specific metrics, such as achieving over 95% accuracy in classification, ensuring response times within milliseconds for real-time analysis, and garnering positive user feedback on interface usability. These benchmarks were pivotal in guiding our progress, ensuring the product met stringent quality standards, and ultimately providing a robust, reliable, and user-friendly solution in the realm of malicious URL detection.

1.4. Key Features and Functionalities

The Deep Learning Malicious URL Checker boasts a robust set of key features and functionalities designed to enhance online security. It provides real-time URL analysis, ensuring immediate detection and classification of malicious links, thereby preventing potential cyber threats. The user-friendly interface ensures effortless navigation, allowing both individuals and organizations to utilize the tool seamlessly. The system supports a wide array of URL formats,

accommodating diverse online platforms and communication channels. Moreover, it offers detailed reporting, providing users with comprehensive insights into detected threats. Additionally, the Deep Learning Malicious URL Checker is equipped with integration capabilities, allowing seamless collaboration with other cybersecurity tools and systems. This integration ensures a cohesive approach to cybersecurity, enhancing the overall effectiveness of threat detection and mitigation efforts.

1.5. Importance of Malicious URL Detection in Cybersecurity

Malicious URL detection is paramount in cybersecurity due to the escalating risks associated with cyber threats. Interacting with malicious URLs can lead to devastating consequences, including:

- Data Breaches: Malware-laden URLs can infiltrate systems, stealing sensitive data such as personal information, passwords, and financial records.
- Identity Theft: Cybercriminals exploit deceptive URLs to trick users into divulging personal details, leading to identity theft and impersonation.
- Financial Loss: Phishing URLs target financial accounts, leading to unauthorized transactions and monetary losses for individuals and organizations.
- Compromised Security Systems: Malicious URLs often serve as entry points for malware, compromising entire networks and security infrastructures.

Our Deep Learning Malicious URL Checker addresses these risks by leveraging advanced algorithms to swiftly identify and block malicious URLs in real time. By proactively detecting harmful links, our product fortifies online safety, ensuring users can navigate the internet securely, free from the threats posed by malicious URLs.

1.6. Target Audience and User Scenarios

The Deep Learning Malicious URL Checker caters to a diverse range of users, ensuring a safer online environment across various contexts. Our target audience includes:

- Individual Users
  - Secure personal devices from phishing attempts.
  - Safeguard sensitive information during online transactions.
- Small and Medium Businesses

    - Protect employees from malicious links in emails and messages.
    - Ensure secure browsing, safeguarding company data.
- Enterprises
    - Enhance overall cybersecurity posture by identifying and blocking malicious URLs in corporate communications.
    - Strengthen network security and prevent data breaches.
- Educational Institutions
    - Safeguard students, teachers, and staff from malicious websites, ensuring a secure online learning environment.
    - Prevent cyber threats within the campus network.
- Social Media Platforms
    - Implement an additional layer of security by scanning URLs shared in messages and posts.
    - Protect users from clicking on malicious links shared by others.

By offering real-time URL analysis, our product ensures individuals and organizations can confidently engage with online content, mitigating risks associated with malicious URLs across emails, social media, web browsing, and various communication channels.

2. Scope

2.1. Cybersecurity Threat Landscape

The cybersecurity threat landscape is continuously evolving, marked by a relentless surge in the sophistication of malicious actors and their strategies. In recent years, cyber threats have taken diverse and complex forms, including phishing attacks, where deceptive emails and websites impersonate trustworthy entities to extract sensitive information. Malware distribution, another prevalent threat, involves the dissemination of harmful software through seemingly innocuous downloads or attachments, compromising user systems. Ransomware, a particularly insidious threat, encrypts user files and demands payment for their release. Additionally, social engineering techniques manipulate human psychology, tricking individuals into revealing confidential data. These threats exploit vulnerabilities within URLs, leveraging users' trust in familiar web addresses to initiate attacks. By disguising malicious content behind seemingly harmless links, cybercriminals infiltrate systems, compromising user security and privacy. As the digital landscape becomes more complex, the need for robust URL detection mechanisms, like the Deep Learning Malicious URL Checker, becomes paramount in safeguarding users from these ever-evolving threats.

2.2. Types of Malicious URLs:

Malicious URLs encompass a wide array of deceptive tactics employed by cybercriminals. Understanding these types is crucial for effective threat detection and user awareness.

- Phishing URLs: Crafted to mimic trusted websites, phishing URLs trick users into revealing sensitive information such as login credentials, financial data, or personal details. They often employ tactics like fake login pages and urgent messages to exploit user trust.

- Malware Distribution URLs: These URLs serve as conduits for malware dissemination. Once accessed, they initiate downloads of malicious software, compromising the user's device. Cybercriminals frequently use social engineering, email attachments, or compromised websites to distribute malware, exploiting unsuspecting users.

- Malicious Redirect URLs: Redirect URLs lead users to unexpected destinations, often to phishing or malicious sites. Attackers use tactics like URL shorteners and compromised ad networks to conceal these redirects, making it challenging to trace the malicious source.

- Drive-by Download URLs: Drive-by download URLs exploit vulnerabilities in users' browsers or plugins. Visiting such URLs triggers automatic downloads and installations of malware without user consent or knowledge, making them particularly insidious.

- Man-in-the-Middle (MitM) URLs: MitM URLs intercept communication between users and websites, enabling attackers to eavesdrop, modify, or steal sensitive data exchanged during online transactions. These URLs are often disguised in unsecured public Wi-Fi networks or compromised routers.

2.3. Significance of Malicious URL Detection in Information Security

Malicious URL detection stands as the frontline defense in fortifying information security. By identifying and intercepting malicious URLs, this proactive measure acts as a potent deterrent, thwarting cyber threats right at their inception. Preventing users from interacting with these URLs serves as a vital shield against scams, shielding individuals from identity theft, financial fraud, and various cybercrimes. The significance lies in its ability to disrupt the cybercriminals' pathways, denying them access to vulnerable targets. Furthermore, by halting these URLs before they infiltrate networks or devices, users are safeguarded from the devastating consequences of data breaches and system compromises. In essence, malicious URL detection not only prevents immediate harm but also bolsters the overall cybersecurity infrastructure, reinforcing the resilience of digital ecosystems against an array of evolving threats.

3. Methodology used to develop

3.1. Overview of Deep Learning Techniques:

Utilizing deep learning, which is a subset of machine learning, the model gains the ability to analyze intricate layers of data and identify subtle patterns automatically. Deep learning, unlike conventional algorithms, is proficient in managing vast and intricate datasets, making it particularly effective for handling complex tasks such as malicious URL detection. By employing artificial neural networks with multiple layers, deep learning models can autonomously learn and adapt. This adaptability is crucial in the constantly evolving realm of cybersecurity, where traditional methods often prove insufficient. Deep learning empowers the Malicious URL Checker with the intelligence to comprehend the complexities of malicious URLs, ensuring a resilient and dynamic defense mechanism against cyber threats.

3.2. Introduction to Bidirectional Long Short-Term Memory (Bi-LSTM) Networks:

Bidirectional Long Short-Term Memory (Bi-LSTM) networks, a specialized variant of recurrent neural networks, play a pivotal role in the model. Unlike standard LSTM networks, Bi-LSTMs analyze long-term dependencies and sequential patterns bidirectionally, considering both past and future context. This bidirectional analysis is particularly vital for understanding the nuanced structure of URLs, where context and sequence are paramount. By discerning intricate patterns within URLs, Bi-LSTMs enhance the model's capability to recognize subtle features indicative of malicious intent. Their ability to analyze sequences in both directions equips the Malicious URL Checker with a holistic understanding, enabling it to accurately identify and classify malicious URLs in real-time, thus fortifying digital security for end-users.

3.3. Data Collection and Labeling Process

The data collection and labeling process is fundamental to the effectiveness of the Deep Learning Malicious URL Checker. To create a robust dataset, a diverse range of both malicious and non-malicious URLs was gathered from various sources, ensuring representation of real-world cyber threats. The labeling process involved assigning a binary classification: malicious URLs were labeled as '1,' indicating their harmful nature, while non-malicious URLs received a '0' label. This meticulous labeling established the ground truth for supervised learning, enabling the model

to learn the intricate patterns distinguishing between safe and malicious URLs. A snippet of the labeling process in Python illustrates the method:

```
blacklist_file = 'phishing_database.csv'
whitelist_file = 'whitelist.txt'

try:
    blacklist = pd.read_csv(blacklist_file)

    # Assign 0 for non-malicious and 1 as malicious for supervised learning.
    for url in blacklist['url']:
        extracted_url = extract_url_from_csv_row(url)
        urls[extracted_url] = 1
except FileNotFoundError:
    print(f"Error: {blacklist_file} not found. Make sure the file exists.")

try:
    with open(whitelist_file, 'r') as f:
        lines = f.read().splitlines()
        for url in lines:
            urls[url] = 0
except FileNotFoundError:
    print(f"Error: {whitelist_file} not found. Make sure the file exists.")

return urls
```

Figure 1 Data Collection and Labeling

This meticulous approach ensured a balanced and comprehensive dataset, enabling the Deep Learning Malicious URL Checker to learn from diverse URL samples, enhancing its accuracy in detecting malicious content while minimizing false positives.

3.4. Data Preprocessing Steps

In the process of developing the Deep Learning Malicious URL Checker, a series of crucial data preprocessing steps were implemented to transform raw URLs into a format suitable for the neural network model. Tokenization, a fundamental technique in natural language processing, was employed to break down URLs into numerical tokens. The Tokenizer class from *TensorFlow's Keras* API was utilized for this purpose, mapping unique characters to

corresponding integers. This ensured that each character in the URL was represented numerically, forming the basis for the model's input.

Additionally, padding was applied to sequences of tokens to ensure uniform input length for the model. URLs inherently vary in length, but neural networks require fixed-size inputs. By using the *pad_sequences* function, the sequences were padded or truncated as necessary to achieve a consistent length of 128 tokens. This uniformity facilitated efficient batch processing during training, allowing the model to learn from the preprocessed URL data effectively.

These preprocessing steps were pivotal in transforming the raw URL data into a format compatible with deep learning. The tokenization and padding techniques preserved the essential characteristics of the URLs while enabling the model to analyze and learn from the data. This preprocessing played a significant role in enhancing the model's ability to accurately distinguish between benign and malicious URLs, ultimately leading to the remarkable accuracy achieved during the training and evaluation phases.

```
# Preprocess data for training.
max_chars = 20000
maxlen = 128

tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=max_chars, char_level=True)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(samples)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

data = tf.keras.preprocessing.sequence.pad_sequences(sequences, maxlen=maxlen)
```

Figure 2 Data Preprocessing using Tensorflow

3.5. Training Data Splitting and Validation

In developing the Deep Learning Malicious URL Checker, a meticulous approach to training data splitting and validation was adopted. The dataset, comprising both malicious and non-malicious URLs, was divided into training and validation sets. To ensure the model's robustness and prevent overfitting, 80% of the data was allocated for training, while the remaining 20% served as the validation set. This division allowed for a comprehensive evaluation of the model's

performance during training. The training data was randomly shuffled to ensure a diverse representation of URLs in each batch, promoting effective learning.

```
# Divide data between training, cross-validation, and test data.
training_samples = int(len(samples) * 0.80)
validation_samples = int(len(labels) * 0.20)
print(training_samples, validation_samples)

indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]

x = data[:training_samples]
y = labels[:training_samples]
x_test = data[training_samples: training_samples + validation_samples]
y_test = labels[training_samples: training_samples + validation_samples]
```

Figure 3 Data Splitting and Validation

3.6. Model Architecture and Design Choices

The model architecture was meticulously crafted to enhance its ability to discern malicious URLs accurately. Employing Bidirectional Long Short-Term Memory (Bi-LSTM) networks, the model harnessed sequential information effectively. The decision to use three stacked Bi-LSTM layers allowed for a deep understanding of URL patterns. Dropout layers were strategically incorporated to mitigate overfitting. The final layer, a dense unit with a sigmoid activation function, facilitated binary classification. These design choices culminated in a model capable of capturing intricate URL features, ensuring precise classification of malicious and non-malicious URLs.

```
# Create model for training.
model = keras.Sequential()
model.add(keras.layers.Embedding(num_chars, embedding_vector_length, input_length=maxlen))
model.add(keras.layers.Bidirectional(keras.layers.LSTM(256, dropout=0.3, recurrent_dropout=0.3, return_sequences=True)))
model.add(keras.layers.Bidirectional(keras.layers.LSTM(256, dropout=0.3, recurrent_dropout=0.3, return_sequences=True)))
model.add(keras.layers.Bidirectional(keras.layers.LSTM(128, dropout=0.3, recurrent_dropout=0.3)))
model.add(keras.layers.Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Figure 4 Model Architecture

3.7. Training Process and Optimization Strategies

In the training process of the Deep Learning Malicious URL Checker, a meticulous approach was undertaken. The model was compiled using the Adam optimizer and binary cross-entropy loss function, tailored for binary classification tasks. To prevent overfitting, dropout layers were incorporated, randomly dropping out a fraction of input units during training. The training data was split into training and validation sets, ensuring an 80-20 split for robust model evaluation. For optimization, early stopping callbacks were employed, allowing the training process to halt if the validation loss ceased to decrease, thus preventing unnecessary epochs and conserving computational resources. The training epochs were set to 10, striking a balance between achieving convergence and limiting computational time. This careful configuration ensured the model learned nuanced patterns from the data, enhancing its ability to discern between malicious and benign URLs effectively.

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Implement early stopping callback
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=2,
    mode='min',
    restore_best_weights=True
)

# Train the model with early stopping callback
history = model.fit(x, y,
                    epochs=10,
                    batch_size=350,
                    callbacks=[early_stopping],
                    validation_split=0.20,
                    shuffle=True)

# Evaluate model on test data.
score, acc = model.evaluate(x_test, y_test, verbose=1, batch_size=350)

print("Model Accuracy: {:0.2f}%".format(acc * 100))
```

Figure 5 Training Process and Optimization Strategies

This approach not only facilitated the training process but also optimized the model's performance, ensuring its efficacy in real-time malicious URL detection scenarios.

4. Methodology that the product work

4.1. Real-time URL Processing Workflow

In real-time scenarios, the Deep Learning Malicious URL Detection system operates seamlessly, ensuring swift and accurate analysis of user-input URLs. The workflow commences when users input URLs via the user-friendly interface. Upon submission, the system swiftly receives the input and employs regular expressions to extract URLs from the text. These extracted URLs undergo a meticulous cleaning process, removing extraneous characters and ensuring uniformity. Following this, the system validates the URLs for their format and legitimacy, automatically adding 'http://' to incomplete addresses to standardize them.

Once the URLs are validated, they undergo tokenization, a crucial step where the URLs are converted into numerical tokens to be comprehensible for the deep learning model. Subsequently, padding is applied to maintain consistent input length, optimizing the data for analysis. The preprocessed URLs are then fed into the pre-trained Bidirectional Long Short-Term Memory (Bi-LSTM) network, a powerful deep learning architecture. The Bi-LSTM model analyzes the intricate patterns within the URLs, swiftly predicting their maliciousness probabilities.

The results, comprising classification outcomes and confidence scores, are promptly formatted and relayed back to the user interface. This real-time processing capability ensures that users receive instantaneous feedback regarding the safety of the URLs they provide. The system's ability to handle multiple URLs simultaneously, coupled with its quick response time, empowers users to make informed decisions promptly. This seamless and expedited workflow not only enhances user experience but also underscores the system's efficiency and reliability in real-time URL analysis scenarios. Users can confidently navigate the digital landscape, armed with the knowledge that their security concerns are addressed promptly and comprehensively.

4.2. User Input and URL Extraction

In the Deep Learning Malicious URL Checker application, users can input URLs through a user-friendly interface facilitated by the Flask web framework. The interface, represented by the provided script, allows users to enter URLs in various formats, ensuring flexibility and user convenience.

To cater to diverse user input formats, the system employs regular expressions to extract URLs from the provided text. Specifically, the *url_pattern* regular expression in the /process_urls route is designed to identify URLs in both plain text and hyperlink formats. This regex pattern efficiently captures URLs even when embedded within a larger text body, accommodating scenarios where users input plain text containing URLs or provide hyperlinks directly.

```
input_text = data.get('text', '')
url_pattern = r'https?://\S+|www\.\S+(?=\W|$)'
extracted_urls = re.findall(url_pattern, input_text)
```

Figure 6 URL Extraction

By employing these techniques, the Deep Learning Malicious URL Checker guarantees precise URL extraction from a variety of user inputs. Whether users provide plain text, hyperlinks, or bulk input, the system's robust URL extraction mechanism ensures that all provided URLs are accurately captured and ready for malicious URL detection. The implementation of these extraction methods ensures a seamless user experience, enabling users to interact effortlessly with the system.

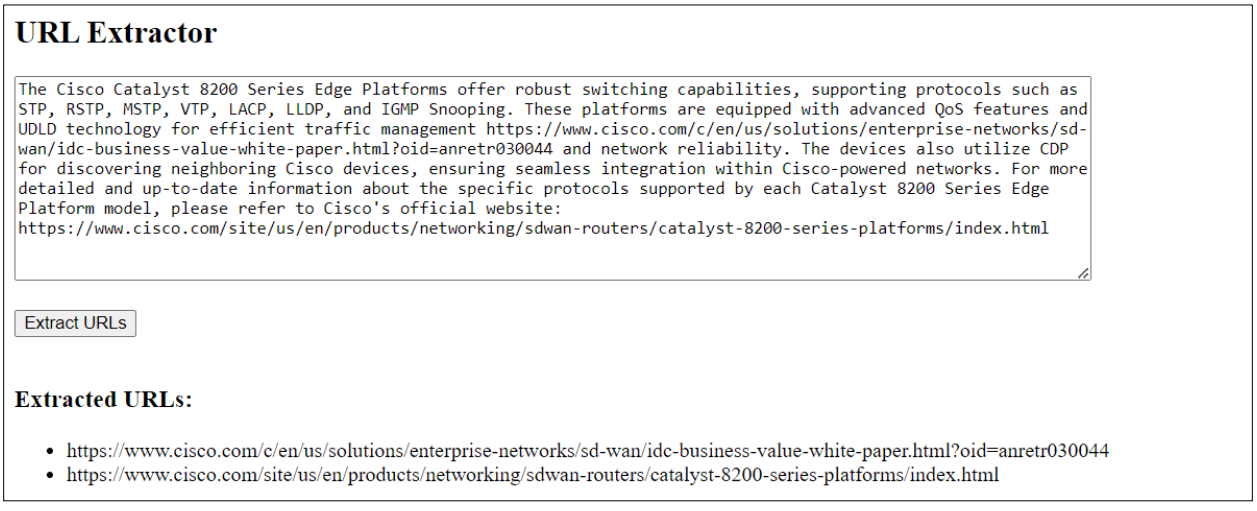

Figure 7 Extracted URL in GUI

4.3. URL Preprocessing Steps

In the preprocessing phase, the extracted URLs undergo a meticulous transformation process to ensure optimal compatibility with the deep learning model. Initially, irrelevant characters and spaces are systematically eliminated, enhancing the dataset's cleanliness and reducing noise in the input data.

Subsequently, tokenization is employed to convert the cleaned URLs into numerical tokens, enabling the model to comprehend their intricate structures. Utilizing the Tokenizer class from TensorFlow, the URLs are transformed into sequences of integers, creating a numerical representation of the textual data. This step is crucial, as it prepares the URLs for deep learning algorithms, which operate on numerical data.

To maintain consistency in input lengths and facilitate uniform processing, padding is applied using the *pad_sequences* function from TensorFlow. This step ensures that all URL sequences share a consistent length, addressing the variability in URL lengths commonly encountered in real-world scenarios. By padding the sequences to a specified length (in this case, 128 characters), the URLs are standardized, allowing the model to efficiently process them without losing valuable information.

```
maxlen = 128
max_words = 20000

tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=max_words, char_level=True)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(url)
url_prepped = tf.keras.preprocessing.sequence.pad_sequences(sequences, maxlen=maxlen)
return url_prepped
```

Figure 8 Model Calling

These preprocessing steps, meticulously executed through Python code, play a pivotal role in transforming raw URLs into structured numerical inputs, setting the stage for accurate and efficient analysis by the deep learning model.

4.4. Integration with Flask for API Endpoints

The Deep Learning Malicious URL Checker seamlessly integrates with Flask, a robust Python web framework, to create efficient API endpoints for communication between the user interface and the backend model. Flask's simplicity and flexibility enable the development of RESTful API endpoints, ensuring smooth data exchange. Upon receiving user input, Flask processes the URLs through regular expressions, extracting the relevant information. The cleaned and validated URLs are then passed to the backend model for analysis. Flask handles incoming requests and orchestrates the flow, allowing for real-time URL analysis. The RESTful API design guarantees a standardized and coherent approach, enhancing the user experience and ensuring reliability in data transmission.

```
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('indexs.html')

@app.route('/process_urls', methods=['POST'])
def process_urls():
    data = request.json
```

Figure 9 Flask for API Endpoints

4.5. Interaction with the Trained Deep Learning Model

The pre-trained Deep Learning Malicious URL Checker model is seamlessly integrated into the system to facilitate URL analysis. User-processed URLs, after being pre-processed and tokenized, are fed into the model for evaluation. The model, based on Bidirectional Long Short-Term Memory (Bi-LSTM) networks, interprets the input data, leveraging its learned patterns and features to generate predictions. Utilizing the Keras framework, the model accurately classifies URLs, discerning between malicious and non-malicious ones. The neural network architecture ensures a deep understanding of the URL structures, enabling precise predictions. The integration process ensures a dynamic connection between the user interface and the model, enabling rapid analysis and enhancing the overall effectiveness of malicious URL detection.

```python
# Load the pre-trained Keras model
model_pre = 'models/bi-lstmchar256256128V2.h5'
model = tf.keras.models.load_model(model_pre)

# Modified function to prepare URL for prediction
def prepare_url(url):···


@app.route("/predict", methods=["POST"])
def predict():
    # Initialize the dictionary for the response.
    data = {"success": False}

    # Check if POST request.
    if flask.request.method == "POST":
        # Grab and process the incoming json.
        incoming = flask.request.get_json()
        urlz = []
        url = incoming["url"]

        print("INCOMING URL", url)
```

Figure 10 Trained Model Interaction

These well-orchestrated interactions between Flask, the pre-trained model, and the user interface ensure a robust, responsive, and accurate malicious URL detection system, providing users with a seamless experience while maintaining the integrity of the detection process.

4.6. Prediction and Confidence Score Calculation

The Deep Learning Malicious URL Checker employs a meticulous process for generating predictions and calculating confidence scores, crucial for accurate classification. After preprocessing, the model predicts the likelihood of a URL being malicious. Utilizing the sigmoid activation function, the output represents a confidence score ranging from 0 to 1. A score above the threshold of 0.50 indicates the URL's malicious nature. The threshold acts as a decisive factor, ensuring the model's confidence in its predictions. Fine-tuning this threshold allows for customization based on specific use cases, optimizing the balance between precision and recall. By interpreting these confidence scores, the system categorizes URLs as either malicious or non-malicious, enabling users to make informed decisions regarding the URLs' safety.

```
# Determine the result and format response data.
if prediction > 0.90:
    result = "URL is probably malicious and Dangerous!"
elif prediction > 0.50:
    result = "URL is probably malicious."
else:
    result = "URL is probably NOT malicious."


# Check for base URL. Accuracy is not as great.
split = url.split("//")
split2 = split[1]
if "/" not in split2:
    result = "Base URLs cannot be accurately determined."

# Processes prediction probability.
prediction_percentage = float(prediction) * 100

if result == "Base URLs cannot be accurately determined.":
    r = {"result": result, "url": url}
else:
    r = {"result": result, "malicious percentage": prediction_percentage, "url": url}
data["predictions"].append(r)

# Show that the request was a success.
data["success"] = True
```

Figure 11 URL Prediction

4.7. Handling User Output and Displaying Results

In handling user output and displaying results, the Deep Learning Malicious URL Checker employs a user-friendly interface to present the classification outcomes and confidence scores, ensuring a seamless and informative user experience. Upon receiving the user's input URLs, the system processes them through the pre-trained deep learning model, predicting the maliciousness probability for each URL. The results are then meticulously formatted and conveyed back to the user.

The system utilizes the Flask web framework to create interactive web pages. When the user submits URLs via the */process_urls* route, the application validates, processes, and analyzes the URLs using the deep learning model. The predictions, along with their associated confidence

scores, are encapsulated in a JSON object and sent back to the user interface for display. The Flask application leverages the *jsonify* function to ensure seamless data serialization and transport between the backend and frontend components.

```
# Check for base URL. Accuracy is not as great.
split = url.split("//")
split2 = split[1]
if "/" not in split2:
    result = "Base URLs cannot be accurately determined."

# Processes prediction probability.
prediction_percentage = float(prediction) * 100

if result == "Base URLs cannot be accurately determined.":
    r = {"result": result, "url": url}
else:
    r = {"result": result, "malicious percentage": prediction_percentage, "url": url}
data["predictions"].append(r)
```

Figure 12 Result output

The user interface, designed with HTML templates and CSS styling, dynamically renders the prediction results. Each URL submitted by the user is displayed alongside its classification outcome (malicious or non-malicious) and the corresponding confidence score, representing the model's certainty regarding the prediction. The interface provides clear visual cues, such as color-coding and intuitive icons, to enhance user understanding of the results. Positive outcomes are presented with visual indicators of security, instilling confidence in the user about the safety of the provided URLs. Additionally, the system handles scenarios where a URL's base component cannot be accurately determined, ensuring transparent communication with the user about potential limitations.

```
<h3>JSON Output:</h3>
<pre id="jsonOutput"></pre>

<script>
    function extractUrls() {
        var inputText = document.getElementById('inputTextArea').value;
        fetch('/process_urls', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ text: inputText }),
        })
        .then(response => response.json())
        .then(data => {
            var extractedUrlsList = document.getElementById('extractedUrls');
            extractedUrlsList.innerHTML = ''; // Clear the previous list

            var table = document.createElement('table');
            table.border = "1";

            var header = table.createTHead();
            var row = header.insertRow(0);
            var cell1 = row.insertCell(0);
            var cell2 = row.insertCell(1);
            var cell3 = row.insertCell(2);
            cell1.innerHTML = "URL";
            cell2.innerHTML = "Result";
            cell3.innerHTML = "Malicious Percentage";

            for (var i = 0; i < data.extracted_urls.length; i++) {
                var urlData = data.extracted_urls[i];
                var url = urlData.url;
                var result = urlData.result;
                var maliciousPercentage = parseFloat(urlData['malicious percentage']).toFixed(2); // Round to 2 decimal places
```

Figure 13 GUI Loader

```
PS C:\Users\kirul\Desktop\Test\AI-Phishing> & "D:/Programming Language/Python/python.exe" c:/Users/kirul/Desktop/Test/AI-Phishing/GUI_URL_Paragraph_Checker.py
2023-11-03 00:01:07.718791: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'GUI_URL_Paragraph_Checker'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2023-11-03 00:01:17.654420: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Debugger is active!
 * Debugger PIN: 926-147-486
127.0.0.1 - - [03/Nov/2023 00:01:19] "GET / HTTP/1.1" 200 -
cleaned_urls ['https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/idc-business-value-white-paper.html?oid=anretr030044', 'https://www.cisco.com/site/us/en/products/networking/sdwan-routers/catalyst-8200-series-platforms/index.html', 'https://tinyurl.com/megafonedigitalcombr?_user=abc@abc.com.br']
URL https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/idc-business-value-white-paper.html?oid=anretr030044
URL https://www.cisco.com/site/us/en/products/networking/sdwan-routers/catalyst-8200-series-platforms/index.html
URL https://tinyurl.com/megafonedigitalcombr?_user=abc@abc.com.br
validated URLs ['https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/idc-business-value-white-paper.html?oid=anretr030044', 'https://www.cisco.com/site/us/en/products/networking/sdwan-routers/catalyst-8200-series-platforms/index.html', 'https://tinyurl.com/megafonedigitalcombr?_user=abc@abc.com.br']
Before subprocess.run
INCOMING URL https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/idc-business-value-white-paper.html?oid=anretr030044
1/1 [==============================] - 4s 4s/step
127.0.0.1 - - [03/Nov/2023 00:01:33] "POST /predict HTTP/1.1" 200 -
INCOMING URL https://www.cisco.com/site/us/en/products/networking/sdwan-routers/catalyst-8200-series-platforms/index.html
1/1 [==============================] - 0s 354ms/step
127.0.0.1 - - [03/Nov/2023 00:01:36] "POST /predict HTTP/1.1" 200 -
INCOMING URL https://tinyurl.com/megafonedigitalcombr?_user=abc@abc.com.br
1/1 [==============================] - 0s 386ms/step
127.0.0.1 - - [03/Nov/2023 00:01:39] "POST /predict HTTP/1.1" 200 -
After subprocess.run
OUTPUT_123 [{'malicious percentage': 0.5930278915911913, 'result': 'URL is probably NOT malicious.', 'url': 'https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/idc-business-value-white-paper.html?oid=anretr030044'}, {'malicious percentage': 3.097047284245491, 'result': 'URL is probably NOT malicious.', 'url': 'https://www.cisco.com/site/us/en/products/networking/sdwan-routers/catalyst-8200-series-platforms/index.html'}, {'malicious percentage': 94.33512687683105, 'result': 'URL is probably malicious and Dangerous!', 'url': 'https://tinyurl.com/megafonedigitalcombr?_user=abc@abc.com.br'}]
127.0.0.1 - - [03/Nov/2023 00:01:39] "POST /process_urls HTTP/1.1" 200 -
```

Figure 14 CLI Output

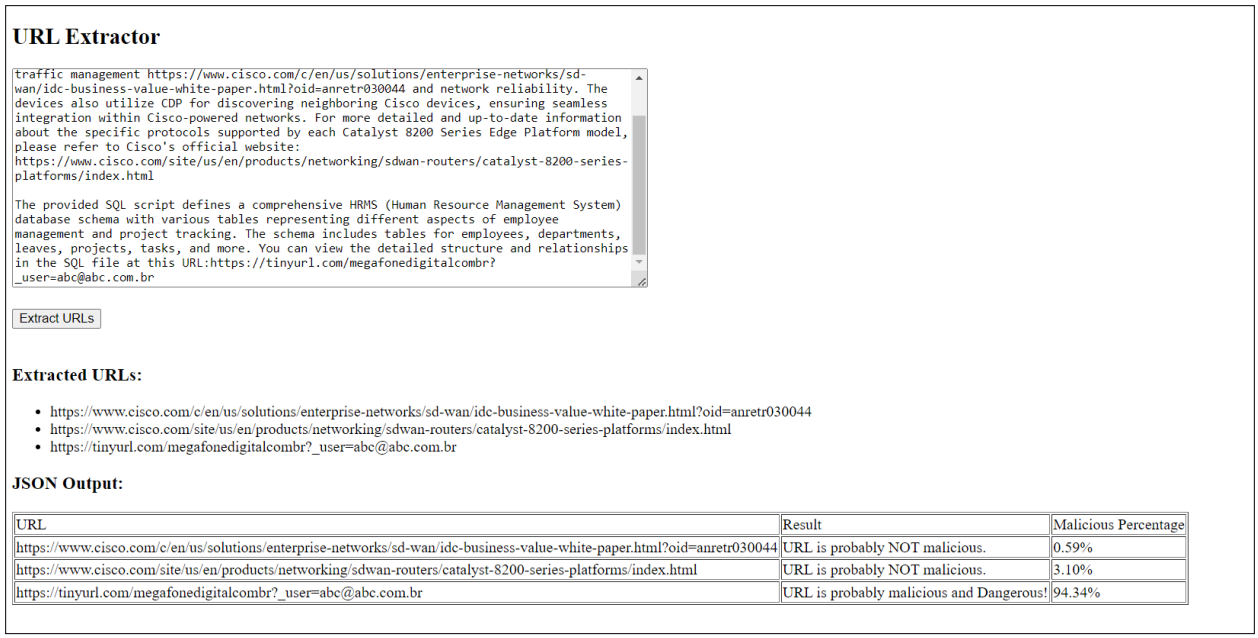

Figure 15 GUI Output of after checked URL

5. Conclusion

In the landscape of cybersecurity, the Deep Learning Malicious URL Detection system stands as a testament to the boundless possibilities of innovative technology. With a profound commitment to enhancing digital safety, this project has harnessed the power of deep learning and advanced machine learning techniques to create a robust defense against the ever-evolving threats lurking within malicious URLs.

By integrating intricate neural networks, particularly Bidirectional Long Short-Term Memory (Bi-LSTM) networks, the system has achieved a remarkable level of sophistication in discerning malicious intent from benign web addresses. Its ability to analyze URLs, understand subtle patterns, and predict potential threats with high accuracy showcases the transformative potential of artificial intelligence in safeguarding online environments.

Furthermore, the system's user-centric design, exemplified by the intuitive web interface, empowers users with swift, real-time insights into the safety of URLs they encounter. Through seamless integration with Flask and thoughtful error handling mechanisms, the project ensures a smooth user experience, fostering trust and confidence among its users.

The collaborative spirit behind this project, encompassing dedicated developers, data scientists, and open-source communities, has been instrumental in its success. The synergy of diverse expertise, combined with a shared vision for a secure digital world, has propelled this initiative from conception to realization.

As digital landscapes become increasingly complex, the Deep Learning Malicious URL Detection system stands as a beacon of innovation and resilience. Its impact on the ongoing fight against cyber threats is not just about technology; it represents a big change in how society deals with online safety. This project has shown how deep learning can make the internet safer. With deep learning, users can feel more confident and peaceful while using the digital world.

GitHub link : https://github.com/kirula0626/AI-Malicious-URL-Checker.git