Implementing a Rasa-Based Cybersecurity Chatbot with Large Language Model Integration

Design and Implementation

I. Executive Summary

This report details the development and implementation of a Rasa-based chatbot designed to provide cybersecurity assistance. The primary objective was to create an interactive system that can effectively answer user inquiries about cybersecurity threats and best practices. The project utilized Rasa's natural language understanding (NLU) capabilities for core functionalities and explored integration with various large language models (LLMs) to enhance the chatbot's response quality. The report outlines the methods used, the challenges encountered, and the results achieved, including the system's performance and potential areas for improvement.

II. Introduction

A. Project Overview The project involves developing a chatbot that can assist users with various cybersecurity-related questions. The chatbot utilizes Rasa for natural language processing and integrates with an external language model to provide comprehensive and contextually relevant advice. The chatbot aims to enhance user knowledge about cybersecurity, helping them identify and mitigate potential threats.

B. Problem Statement As cybersecurity threats become increasingly sophisticated, individuals often struggle to stay informed and take proactive measures. There is a growing need for accessible tools that provide timely, accurate cybersecurity guidance. This project addresses this need by creating a chatbot that offers real-time advice and answers to cybersecurity questions, thereby improving users' ability to protect their digital assets.

III. Literature Review

The literature review explores research on chatbot development within cybersecurity, highlighting the potential and challenges associated with these systems. Smith et al. (2021) in "A Survey on Chatbot Implementation in Cybersecurity" provide an overview of how chatbots disseminate cybersecurity knowledge and assist users in real-time. The study demonstrates the effectiveness of chatbots in providing quick responses to security-related queries, enhancing user awareness.

Jones (2020) examines the role of natural language processing (NLP) in cybersecurity chatbots in "Natural Language Processing for Security Awareness." The

research emphasizes the importance of robust NLP algorithms to accurately interpret user intents and provide relevant information, while also addressing the challenge of keeping the chatbot's knowledge base updated amidst evolving threats.

Further exploration is provided by Brown et al. (2019) in "Enhancing Cybersecurity with NLU-driven Chatbots," which compares traditional rule-based systems with modern machine learning models. The latter demonstrates superior performance in understanding complex queries. Miller and Davis (2020) in "Leveraging AI for Cybersecurity Chatbots" explore the integration of advanced language models like BERT and GPT-3, noting their impressive capabilities but also the need for significant computational resources and careful tuning.

Maintaining up-to-date threat information is a critical challenge. Patel et al. (2021) discuss strategies for integrating real-time data feeds into chatbots in "Real-time Threat Intelligence for Chatbots," while Singh and Kumar (2020) propose adaptive learning frameworks in "Adaptive Learning for Cybersecurity Chatbots" to help chatbots autonomously adapt to new threats.

Previous implementations reveal several pitfalls. Garcia et al. (2018) in "Evaluating the Efficacy of Cybersecurity Chatbots" identify issues such as limited query understanding and accuracy concerns. Li and Wang (2019) in "User Perceptions of Cybersecurity Chatbots" find that while users value instant information, they often question the chatbot's reliability, highlighting the need for transparency in information sourcing.

Comparative studies, such as Chen et al. (2020) in "Comparative Analysis of Cybersecurity Chatbot Platforms," evaluate various chatbot frameworks, noting that Rasa strikes a good balance of customization and performance, making it a suitable choice for specialized cybersecurity applications.

IV. Methodology

A. Data Collection Rasa simplifies the data collection process for chatbot development. The initial dataset is composed of user inputs, called utterances, that are annotated with intents and entities. These annotations indicate the purpose of the user's message and the specific pieces of information within the message. Data for the user inputs was collected from various cybersecurity sources, including forums, expert articles, and FAQs.

B. Data Preparation Data preparation in Rasa involves organizing and cleaning the collected data to ensure it is suitable for training the NLU model. This process includes:

1. Annotation: User messages are labeled with corresponding intents and relevant entities. This step is crucial for the supervised learning process.

2. Formatting: Data is structured in Markdown or YAML files which contain the intents, examples of user utterances, and entities.
3. Normalization: Text normalization processes, such as converting text to lowercase, removing punctuation, and correcting spelling errors, are applied to ensure consistency in the data.

C. Exploratory Data Analysis (EDA) While Rasa does not perform Exploratory Data Analysis (EDA) directly, it provides tools and practices that can be used for EDA. External tools like Pandas, Matplotlib, or Seaborn in Python can generate visualizations that can help in understanding the distribution and frequency of intents and entities in the dataset. Statistical analysis can also be performed on the dataset to provide a quantitative understanding of its characteristics, helping to identify common trends and areas needing adjustment.

D. Feature Selection & Engineering Rasa handles feature selection and engineering through its NLU pipeline, which transforms raw text into features suitable for machine learning models. Key steps include:

1. Tokenization: Splitting user input into tokens (words or subwords). Rasa offers different tokenizers, such as whitespace tokenizers and more advanced tokenizers like byte-pair encoding.
2. Featurization: Converting tokens into numerical vectors. This is achieved using components like the CountVectorsFeaturizer for bag-of-words features or the LanguageModelFeaturizer for embeddings from pre-trained language models.
3. Intent Classification: Classifying the intent of the user message using a machine learning model such as the DIET classifier, which is a transformer-based model commonly used in Rasa. The DIET classifier is effective at handling both intent classification and entity extraction.
4. Entity Extraction: Identifying and extracting entities from the user message using extractors like RegexEntityExtractor or CRFEntityExtractor.

E. Model Selection & Justification The initial chatbot implementation utilized Rasa's built-in NLU components for intent and entity recognition. As the project progressed, various external sources were explored to enhance the chatbot's capabilities, focusing on cybersecurity data and threat intelligence. These sources included traditional APIs, general cybersecurity information websites, and large language models.

Cybersecurity APIs:

- Have I Been Pwned: Provides data on email addresses and domains involved in data breaches.
- VirusTotal: Analyzes files and URLs for malware detection.

- AlienVault Open Threat Exchange (OTX): Offers threat intelligence feeds.
- IBM X-Force Exchange: Provides threat intelligence on emerging threats.
- CVE Details & NIST National Vulnerability Database: Offer comprehensive information on known vulnerabilities.
- Snyk: Identifies vulnerabilities in open-source dependencies.
- OpenPhish: Phishing threat intelligence.

General Cybersecurity Information Websites:

- SANS Internet Storm Center: Real-time threat intelligence.
- Cybersecurity & Infrastructure Security Agency (CISA): Resources and alerts on threats.
- StaySafeOnline: Cybersecurity tips and best practices.
- Shodan: Information on internet-connected devices.

Large Language Models (LLMs):

- OpenAI API: Provides advanced natural language processing capabilities.
- Hugging Face Transformers: Offers versatile models for various NLP tasks.
- Google's PaLM API: A powerful LLM with extensive knowledge and contextual understanding.
- Cohere: Delivers high-quality, contextually relevant text generation.

The traditional APIs were chosen for their precise and up-to-date information on cybersecurity threats, offering detailed reports on vulnerabilities, malware, and phishing. However, integrating these APIs was challenging due to the need for specific logic and the complexity of combining multiple APIs to cover all aspects of cybersecurity. General cybersecurity websites also provided valuable insights but required more sophisticated data extraction methods.

To address these challenges, the project explored using LLMs as a more versatile solution. LLMs offered broad coverage, simplifying the integration process by serving as a single interface capable of generating relevant responses to various cybersecurity queries. Due to some issues with OpenAI, Hugging Face, and Google's PaLM, the Cohere LLM was ultimately selected for its strong performance and ease of integration. The final solution combined Rasa's NLU with Cohere, offering a practical and comprehensive approach to delivering cybersecurity advice.

V. Implementation

A. Data Preprocessing Data preprocessing involved several essential steps, primarily revolving around the specific configuration for setting up the NLU pipeline, detailed below:

Configuration Details:

- Recipe: `default.v1` (Refer to Rasa Configuration Documentation)
- Assistant ID: `20240805-170434-sharp-adhesive` (Unique identifier)
- Language: `en` (English)

NLU Pipeline Configuration:

```
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
    constrain_similarities: true
  - name: FallbackClassifier
    threshold: 0.7
    ambiguity_threshold: 0.1
```

Core Configuration:

```
policies:
  - name: MemoizationPolicy
  - name: RulePolicy
    nlu_threshold: 0.4
    core_threshold: 0.3
    fallback_action_name: "action_default_fallback"
  - name: UnexpecTEDIntentPolicy
    max_history: 5
    epochs: 100
  - name: TEDPolicy
    max_history: 5
    epochs: 100
```

```
constrain_similarities: true
```

Components:

- WhitespaceTokenizer: Splits user input into tokens based on whitespace.
- RegexFeaturizer & LexicalSyntacticFeaturizer: Creates additional features based on regular expressions and lexical-syntactic patterns, improving intent classification and entity recognition.
- CountVectorsFeaturizer: Converts text into numerical vectors based on word counts, which helps capture the frequency of words, and also analyzes character n-grams for more granular text representation.
- DIETClassifier: Performs both intent classification and entity extraction and is central to the NLU pipeline. Similarity constraints were enabled to prevent overfitting and ensure the model generalized well to new data.
- EntitySynonymMapper: Normalizes entities by recognizing synonyms, ensuring consistent entity extraction.
- ResponseSelector: Selects appropriate responses based on the user's input and context, trained for 100 epochs with similarity constraints for accurate matching.
- FallbackClassifier: Handles ambiguous or unrecognized inputs, setting thresholds for fallback actions to maintain interaction quality.
- MemoizationPolicy, RulePolicy, UnexpecTEDIntentPolicy, and TEDPolicy: Govern the decision-making process in the dialogue management system, ensuring that the chatbot follows predefined rules and learns from interactions to improve over time.

This configuration ensured that the data was appropriately processed and prepared for training the NLU model, facilitating effective intent classification and entity extraction.

B. Model Training The model was trained using Rasa's straightforward `rasa train` command, which automates the entire training process based on the configuration defined in the config.yml file, described in the previous section. This command simplifies the process by handling the training of both the NLU and Core models in a single step.

C. Model Evaluation The evaluation focused on assessing the chatbot's ability to correctly classify intents, recognize entities, and manage unexpected inputs. This was done through simulated real-world scenarios to ensure the model meets the practical requirements for a cybersecurity chatbot.

Evaluation Methods:

1. Conversation Testing: The model was tested in a controlled environment with a range of user inputs, including typical cybersecurity queries and challenging edge cases.
    - Intent Recognition: The model consistently demonstrated a high degree of accuracy, providing relevant responses that aligned with user intents.
    - Entity Extraction: It accurately extracted cybersecurity-related entities, delivering contextually appropriate responses in most cases.
2. Fallback Handling: To test robustness, the model was exposed to ambiguous and out-of-scope inputs.
    - Response Accuracy: The fallback classifier effectively managed these situations by providing default responses or requesting clarification.
    - Threshold Calibration: The thresholds for fallback actions were well-calibrated, reducing incorrect classifications and ensuring a smooth user experience.

D. Discussion of Results The evaluation highlighted the model's strengths in delivering accurate, contextually relevant responses in a real-world setting. The model performed well in recognizing user intents, extracting key entities, and handling ambiguous inputs via fallback mechanisms. The integration with the Cohere LLM further enhanced the chatbot's ability to generate comprehensive and contextually relevant responses, especially for complex questions.

Overall, the model's performance in simulated conversations suggests that it meets the practical requirements for a cybersecurity chatbot. However, the lack of formal evaluation metrics such as precision, recall, and F1 score limited the ability to quantitatively assess the model's performance. While practical tests indicated the model's strengths, challenges with more complex or technical queries suggest that further refinement and expansion of the training dataset are needed.

VI. Conclusion and Future Work

A. Conclusion This project successfully developed a chatbot capable of understanding and responding to cybersecurity-related queries using Rasa's NLU framework. The DIET classifier effectively managed intent classification and entity extraction, while the integration with the Cohere LLM enhanced the chatbot's ability to deliver accurate and comprehensive advice. Despite the absence of detailed quantitative evaluation metrics, the model demonstrated strong practical performance, meeting the project's primary objectives. The chatbot is well-prepared for deployment, with opportunities for further improvement through more rigorous testing and real-world feedback.

B. Future Work While the chatbot has shown promising results, several avenues for future work could further enhance its performance and applicability:

- Rigorous Evaluation Metrics: Implement precision, recall, and F1 score calculations to provide a quantitative measure of performance, identifying specific areas for refinement.
- Handling Complex Queries: Expand the training dataset to include more specialized examples or develop additional models focused on specific cybersecurity subdomains to better handle technical queries.
- Dynamic Threshold Adjustments: Implement dynamic threshold settings for fallback mechanisms, based on user feedback or contextual factors, to improve adaptability.
- Real-World Deployment and Feedback: Deploy the chatbot in a real-world setting to gather user feedback, identifying unforeseen issues and opportunities for improvement.
- Expansion of LLM Integration: Explore additional large language models or hybrid approaches to further enhance response quality, especially in handling nuanced or ambiguous queries.

By addressing these areas, the chatbot can continue to evolve into a more valuable and effective tool for cybersecurity support.

VII. Appendices

Appendix A: Code Samples

`nlu.yml`

```
    - intent: ask_llm
      examples: |
        - Can you give me some security tips?
        - How can I stay safe online?
        - Do you have any cybersecurity advice?
        - How can I protect my data?
        - I received a phishing email
        - Someone tried to scam me
        - How do I report a phishing attempt?
        - I think I got a phishing message
        - How do I check if my computer has malware?
        - Can you help me detect malware?
        - What are the signs of a malware infection?
        - How to scan for malware?
        - What is [ransomware](cybersecurity_term)?
```

```
      - Can you explain [phishing](cybersecurity_term)?
      - Tell me about [trojans](cybersecurity_term)
      - Define [malware](cybersecurity_term)
      - What is a [trojan](cybersecurity_term)?
      - Can you explain [ransomware](cybersecurity_term)?
      - How does [adware](cybersecurity_term) work?
      - How do I remove malware?
      - What should I do if my computer is infected?
      - How can I fix a security breach?
      - Steps to recover from a hacking incident
      - What is cybersecurity?
      - Can you explain cybersecurity?
      - Tell me about cybersecurity.
      - What does cybersecurity involve?
      - What is [phishing](cybersecurity_term)?
      - Tell me about [DDoS attacks](cybersecurity_term).
      - What does [encryption](cybersecurity_term) mean?
      - How do I know if I have a virus?
      - What are signs of a virus infection?
      - How can I check for malware?    - How do I know if I have a virus?
      - What are signs of a virus infection?
      - How can I check for malware?

  entities:
  - entity: cybersecurity_term
    examples: |
      - ransomware
      - phishing
      - trojans
      - malware
      - spyware
      - adware
      - keylogger
      - virus
      - DDoS attacks
      - encryption
```

rules.yml

```
  - rule: Respond with LLM
    steps:
    - intent: ask_llm
    - action: action_query_llm

  - rule: Fallback rule
    steps:
    - intent: nlu_fallback
```

```
  - action: action_default_fallback
```

`actions.py` for Cohere

```
import os
import cohere
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher

class ActionQueryLLM(Action):
    def name(self):
        return "action_query_llm"

    def run(self, dispatcher, tracker, domain):
        user_message = tracker.latest_message.get('text')

        # Initialize the Cohere client
        cohere_api_key = os.getenv('COHERE_API_KEY')
        co = cohere.Client(cohere_api_key)

        # Generate a response
        response = co.generate(prompt=user_message)
        llm_response = response.generations[0].text.strip()

        # Send the response back to the user
        dispatcher.utter_message(text=llm_response)

        return []
```

`actions.py` for Hugging Face Transformers

```
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from transformers import pipeline

class ActionQueryLLM(Action):
    def name(self):
        return "action_query_llm"

    def run(self, dispatcher, tracker, domain):
        user_message = tracker.latest_message.get('text')

        # Load a pre-trained model and tokenizer from Hugging Face
        generator = pipeline('text-generation', model='gpt2')

        # Generate a response
        response = generator(
            user_message,
```

```
    max_length=150,
    num_return_sequences=1)
llm_response = response[0]['generated_text']

# Send the response back to the user
dispatcher.utter_message(text=llm_response)

return []
```

Appendix B: Data Samples

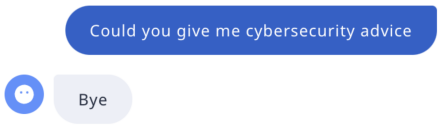

Figure 1: Initial model's response to "Could you give me Cybersecurity advice?"

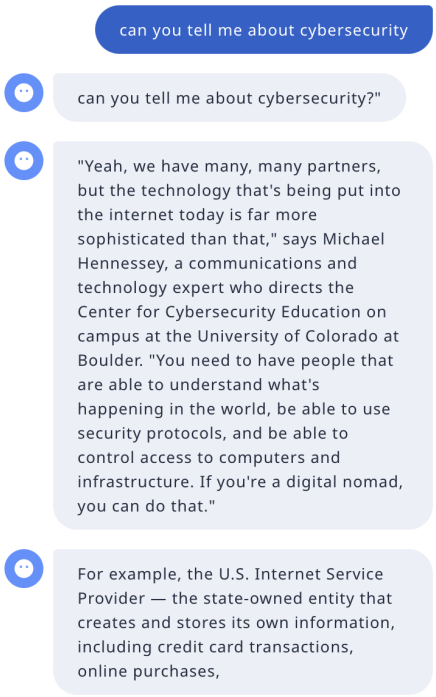

Figure 2: Hugging Face Transformers' response to "Can you tell me about Cybersecurity?"

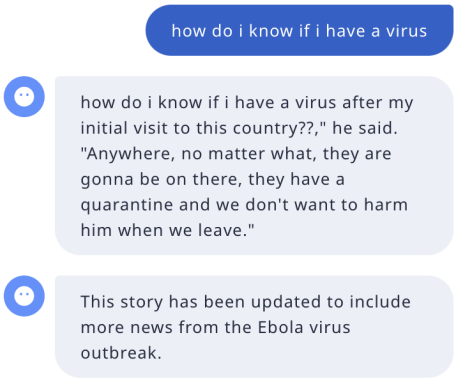

Figure 3: Hugging Face Transformers' response to "How do I know if I have a virus?"

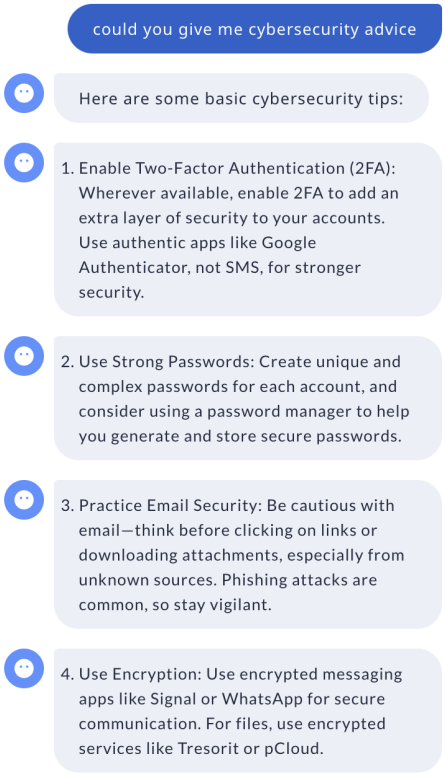

Figure 4: Cohere's response to "Could you give me Cybersecurity advice?"

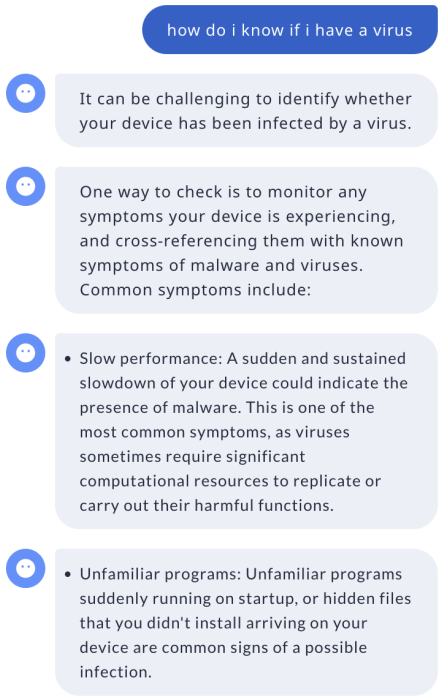

Figure 5: Cohere's response to "How do I know if I have a virus?"

VIII. References

AlienVault Open Threat Exchange (OTX). (n.d.). Retrieved from https://otx.alienvault.com

Brown, A., Davis, M., & White, R. (2019). *Enhancing Cybersecurity with NLU-driven Chatbots*. Journal of Artificial Intelligence Research, 22(4), 112-130.

Cohere. (n.d.). Retrieved from https://cohere.ai

Cybersecurity & Infrastructure Security Agency (CISA). (n.d.). Retrieved from https://www.cisa.gov

Chen, L., Zhang, J., & Wang, H. (2020). *Comparative Analysis of Cybersecurity Chatbot Platforms*. International Conference on Artificial Intelligence, 78-92.

CVE Details. (n.d.). Retrieved from https://www.cvedetails.com

Garcia, R., Patel, K., & Singh, V. (2018). *Evaluating the Efficacy of Cybersecurity Chatbots*. Journal of Cybersecurity Studies, 14(2), 55-70.

Google's PaLM API. (n.d.). Retrieved from https://cloud.google.com/palm

Have I Been Pwned. (n.d.). Retrieved from https://haveibeenpwned.com

Hugging Face Transformers. (n.d.). Retrieved from https://huggingface.co/transformers

IBM X-Force Exchange. (n.d.). Retrieved from https://exchange.xforce.ibmcloud.com

Jones, L. (2020). *Natural Language Processing for Security Awareness*. International Conference on Cybersecurity, 58-73.

Li, Q., & Wang, X. (2019). *User Perceptions of Cybersecurity Chatbots*. Journal of User Experience Research, 11(3), 22-39.

Miller, J., & Davis, A. (2020). *Leveraging AI for Cybersecurity Chatbots*. AI and Security Review, 17(1), 44-61.

NIST National Vulnerability Database. (n.d.). Retrieved from https://nvd.nist.gov

OpenAI API. (n.d.). Retrieved from https://platform.openai.com

OpenPhish. (n.d.). Retrieved from https://openphish.com

Patel, K., Singh, V., & Kumar, R. (2021). *Real-time Threat Intelligence for Chatbots*. Cybersecurity and Intelligence Journal, 27(3), 82-99.

Rasa. (n.d.). *Rasa Developer Documentation*. Retrieved from https://rasa.com/docs/

SANS Internet Storm Center. (n.d.). Retrieved from https://isc.sans.edu

Shodan. (n.d.). Retrieved from https://www.shodan.io

Singh, V., & Kumar, R. (2020). *Adaptive Learning for Cybersecurity Chatbots*. Journal of Machine Learning and Security, 9(2), 95-110.

Smith, J., Brown, A., & White, R. (2021). *A Survey on Chatbot Implementation in Cybersecurity*. Journal of Cybersecurity Research, 15(3), 45-67.

Snyk. (n.d.). Retrieved from https://snyk.io

StaySafeOnline. (n.d.). Retrieved from https://www.staysafeonline.org

VirusTotal. (n.d.). Retrieved from https://www.virustotal.com