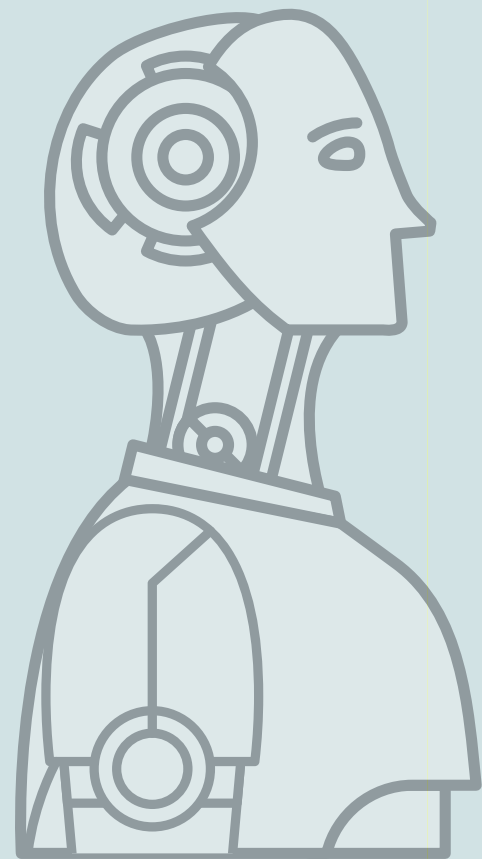


Utilizing LLMs in Cybersecurity for Code Vulnerability Detection

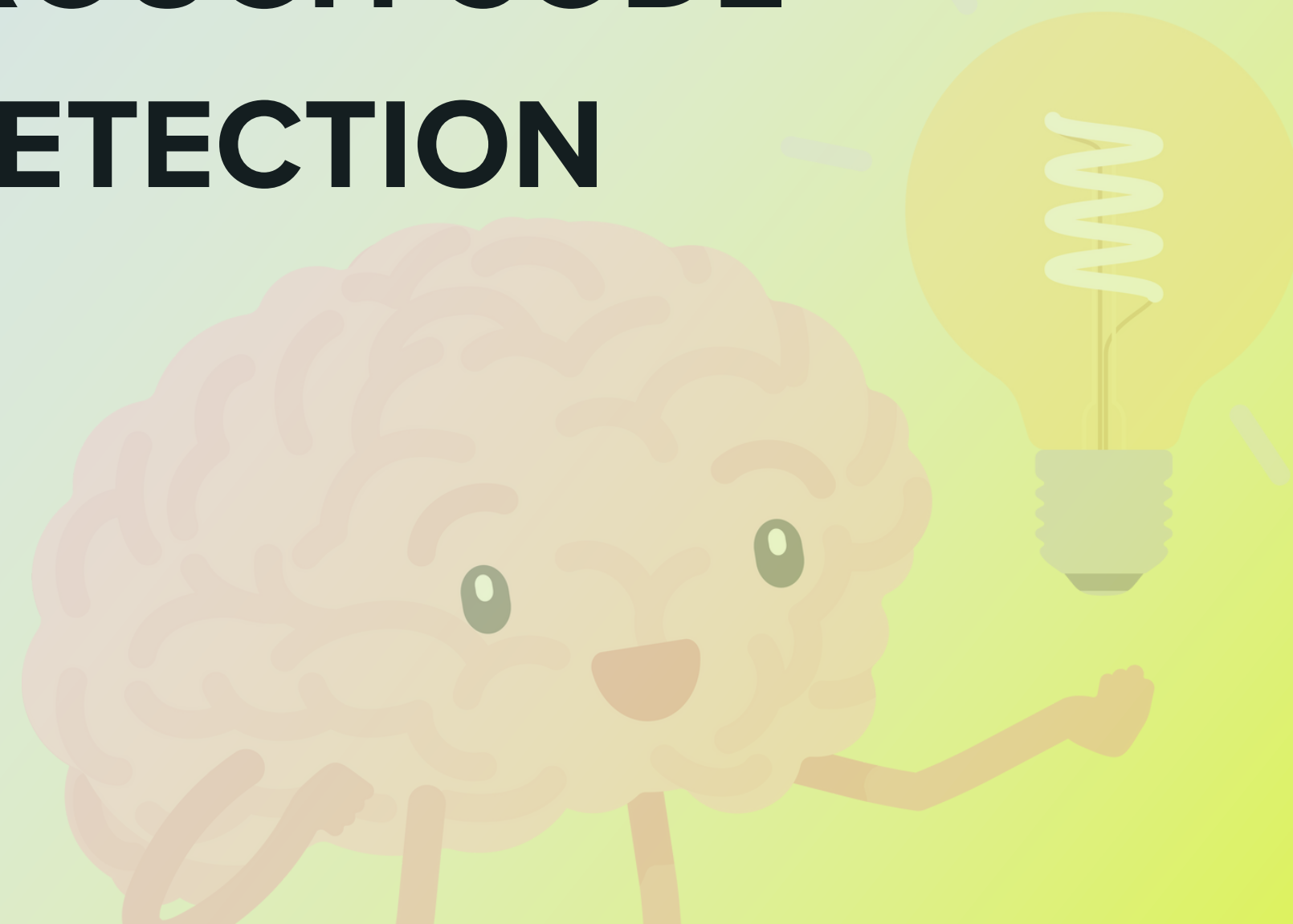
NLP Case Study

- **Team Members:**

- **Aditya Vyas (Comps-A)**
- **Atharva Pawar (Comps-A)**
- **Hitesh Sharma (Comps-A)**



LLMS: PIONEERING THE FUTURE OF CYBERSECURITY THROUGH CODE VULNERABILITY DETECTION



Paper 1 - DiverseVul: A New Vulnerable Source Code Dataset for Deep Learning Based Vulnerability Detection

The paper introduces a novel dataset designed for detecting vulnerabilities in source code. To create this dataset, the authors systematically collected data from security issue websites, extracting both vulnerability-fixing commits and the corresponding source code. This dataset is extensive, comprising 18,945 vulnerable functions spanning 150 Common Weakness Enumerations (CWEs), alongside 330,492 non-vulnerable functions sourced from 7,514 commits. Remarkably, it encompasses 295 more projects than the cumulative total of all previous datasets. The paper additionally offers a comprehensive comparison of various artificial intelligence techniques employed for code vulnerability detection using this expansive dataset.

<https://arxiv.org/abs/2304.00409>

Paper 1 - Detailed Insights

1. Foundational Resource: This paper establishes a foundational resource for our research, providing a vital dataset for vulnerability detection in source code.
2. Meticulous Curation: The dataset's meticulous curation from security issue websites and project commits ensures its reliability and relevance for identifying vulnerabilities.
3. Comprehensive Coverage: With nearly 19,000 vulnerable functions across 150 CWEs and over 330,000 non-vulnerable functions, it offers extensive coverage, enabling in-depth analysis.
4. Deep Learning Exploration: Armed with this dataset, we embark on an exploration of deep learning techniques to enhance code security, using it as the cornerstone for our research endeavors.

Paper 2 - From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy

The paper discusses the impact of generative AI models like ChatGPT and BARD in cybersecurity and privacy. It explores their potential use for both offensive and defensive purposes, including the generation of malicious code and secure code, as well as the detection of malicious activities. The authors also discuss the potential risks and ethical considerations associated with the use of these models in the cybersecurity domain.

<https://ieeexplore.ieee.org/abstract/document/10198233>

Paper 2 - Detailed Insights

1. Dual-Use Potential: Generative AI models, such as ChatGPT and BARD, exhibit a dual-use capability, serving both offensive and defensive roles in cybersecurity. This duality necessitates a balanced approach in their deployment.
2. Malicious Code Generation: These AI models can be employed to generate malicious code, highlighting the need for heightened vigilance in code validation and security testing to counter evolving threats.
3. Secure Code Generation: Conversely, the same models can aid in producing secure code, offering a potential tool to enhance software security by automating the creation of resilient software components.
4. Ethical and Regulatory Concerns: The paper emphasizes the importance of addressing ethical concerns and establishing regulatory frameworks to govern the responsible and ethical use of generative AI in the cybersecurity domain, ensuring that these powerful tools are not exploited for malicious purposes.

Paper 3 - A Prior Case Study of NLP

This paper provides an overview of Natural Language Processing (NLP), emphasizing its complexity and importance in bridging the gap between human language and computer systems. NLP involves understanding grammar, meaning, context, slang, and acronyms in a given language. It plays a significant role in artificial intelligence, enabling computers to comprehend, analyze, generate, and manipulate natural language.

The paper discusses various aspects of NLP, including its challenges such as language disambiguation, sentence parsing, word identification, and named entity recognition. It also explores different techniques and approaches in NLP, such as the use of machine learning, text mining, and predictive analytics. Several applications of NLP are highlighted, ranging from software development and education to healthcare and social media analysis.

https://www.researchgate.net/publication/344458777_A_prior_case_study_of_natural_language_processing_on_different_domain

Paper 3 - Detailed Insights

- **NLP in Business and Software Management:**
 - **NLP is used in business process management to enhance services and deal with growing demands and technology complexities.**
 - **Domain models incorporating NLP handle customer feedback through social multimedia and geolocation context, ensuring efficient**
 - **interfaces through fuzzy models.**
- **NLP for Ontology Learning:**
 - **Ontology in AI refers to concepts used to exchange information between entities.**
 - **NLP is closely related to ontology, defining rules for semantic analysis and question-answering systems.**
 - **Techniques such as symbolic, statistical, and hybrid approaches are applied to information extraction and retrieval, contributing to**
 - **text mining via ontology learning.**
- **NLP in Bioinformatics:**
 - **Bioinformatics employs computer technology to extract knowledge from biological data.**
 - **NLP techniques and text mining play a vital role in information extraction, storage, manipulation, modeling, and retrieval from**
 - **biological databases.**
 - **NLP methods are applied in protein research, including latent semantic analysis for homology detection, analysis of protein**
 - **spectral data, and prediction of protein**
 - **structure and function using machine learning algorithms.**

Paper 4 - Case Studies on using Natural Language Processing Techniques in Customer Relationship Management Software



The study explores the utilization of text data stored in a Customer Relationship Management (CRM) database for data mining and segmentation. Researchers applied advanced techniques from natural language processing (NLP) literature, including word embeddings, and deep learning methods such as recurrent neural networks (RNN) with long short-term memory (LSTM) units. The study used text notes from a CRM system collected by customer representatives of an internet ads consultancy agency between 2009 and 2020.

The findings demonstrate that word embeddings, generated from the text corpus, can be directly used for data mining. Moreover, these embeddings can be integrated into RNN architectures for more complex segmentation tasks. The research highlights that structured text data within a CRM system contains valuable information that can be extracted effectively. It emphasizes the potential of integrating NLP techniques into CRM systems to enhance their functionality, provided that problem definitions are well-established, and solutions are appropriately implemented.

<https://arxiv.org/abs/2106.05160>

Paper 4 - Detailed Insights

1. Word Embeddings and Automated Keyword Detection:

- Word embeddings were trained using the Word2Vec method, creating a lexicon of 73K words.
- Applications included automating the detection of negative entries and implementing a search bar in CRM, enhancing managers' ability to identify issues during CRM processes.

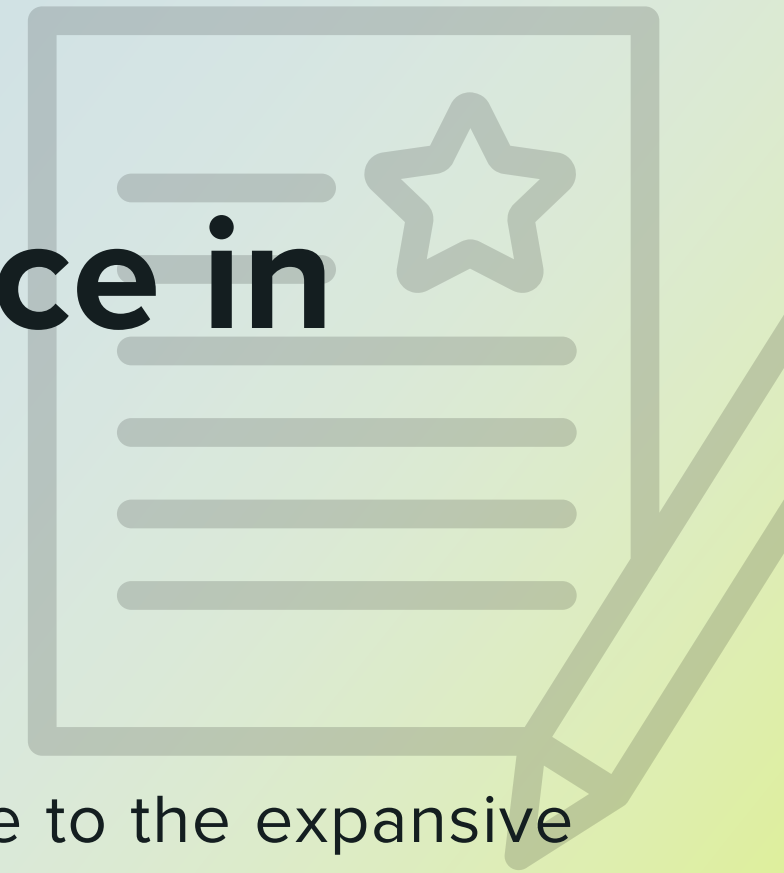
2. Lead Labeling and Segmentation:

- Word embedding sizes were experimented with, determining 50 as the ideal size for the training data.
- Employing LSTM-RNN architecture, the study demonstrated the replacement of the company's binary lead segmentation strategy. This strategy could be extended to multiple classes with sufficient training data, improving time management by automating CRM tasks.

3. Agent Identification and Prediction:

- The same architecture used for lead labeling was adapted for agent identification, transforming the problem into multiclass classification.
- The system achieved 93% accuracy in identifying 14 different agent IDs. Errors were primarily attributed to data scarcity, indicating that more data improved agent identification accuracy.
- The study expanded the scheme to predict an agent's next word based on given input, potentially speeding up text entry processes in CRM, particularly via mobile devices.

Paper 5 - The Use of Artificial Intelligence in Cybersecurity: A Review



The utilization of Artificial Intelligence (AI) in cybersecurity has become indispensable due to the expansive cyberattack landscape in modern enterprises. AI and machine learning offer the capacity to swiftly analyze extensive data sets, identifying diverse cyber threats ranging from malware to potential phishing attacks. These technologies continuously learn and improve, adapting from past experiences to identify novel attack variants. The advantages of AI in cybersecurity are significant, including its ability to detect emerging threats that traditional software struggles to keep up with. AI-driven systems employ advanced algorithms for early detection of malware behaviors, utilizing natural language processing to gather insights from diverse sources, such as articles and news, to provide predictive intelligence and inform prevention strategies. This proactive approach considers both global and industry-specific risks, enhancing cybersecurity posture by prioritizing potential threats effectively.

<https://www.computer.org/publications/tech-news/trends/the-use-of-artificial-intelligence-in-cybersecurity>

Paper 5 - Detailed Insights

1. Embrace AI's Adaptive Learning: Recognize AI's capability to continuously improve and adapt to emerging threats, making it a valuable asset for swift threat detection.
2. Early Malware Detection: Implement AI-driven systems to identify malware behaviors at an early stage, reducing the risk of infection and data breaches.
3. Utilize Natural Language Processing: Leverage natural language processing to gather insights from diverse sources, enabling predictive intelligence and informed prevention strategies.
4. Proactive Risk Management: Adopt AI technologies to proactively manage cybersecurity risks, considering both global and industry-specific threats, and prioritize potential threats effectively.

Paper 6 - Static Code Analysis Tools: A Systematic Literature Review



Static code analysis tools play a crucial role in enhancing code quality by scrutinizing codebases for bugs, security vulnerabilities, duplications, and code smells. The integrity of software products hinges on the quality of their source code, necessitating consistent oversight. This technique enables understanding program behavior without execution, and though various tools offer static analysis across frameworks and programming languages, only a few accommodate domain-specific languages. This paper systematically reviews prevalent static code analysis tools, categorizing them based on their support for both general-purpose and domain-specific languages, as well as the specific types of defects they can identify. Through this comprehensive analysis, the paper contributes to understanding the landscape of static code analysis tools and their diverse applications.

<https://www.researchgate.net/publication/347383785>

Paper 6 - Detailed Insights

- 1.Enhance Code Quality: Consistently incorporate static code analysis tools in your software development process to identify and rectify bugs, security vulnerabilities, duplications, and code smells, ensuring software integrity.
- 2.Consider Language Compatibility: When selecting a static analysis tool, assess its compatibility with the programming languages used in your project, as not all tools support domain-specific languages.
- 3.Defect Identification: Prioritize tools that can identify specific types of defects relevant to your project, ensuring effective problem resolution.
- 4.Systematic Tool Review: Conduct a systematic review of available static code analysis tools to understand their capabilities, compatibility, and applications, facilitating informed tool selection aligned with your organization's specific needs.

LLMs in Action

LLM Implementations

Llama 2 7B chat finetune App

The "Llama 2 7B chat finetune App" employs cutting-edge technology by loading the Llama 2 model in 4-bit precision using the NF4 type. This precision enhancement allows for efficient and streamlined operations. The app's training process encompasses a single epoch, harnessing the model's capabilities to adapt and refine its performance. This fine-tuning approach optimizes the Llama 2 7B model for specific chat-related tasks, ensuring it delivers highly accurate and contextually relevant responses, making it an invaluable tool for various natural language processing applications.

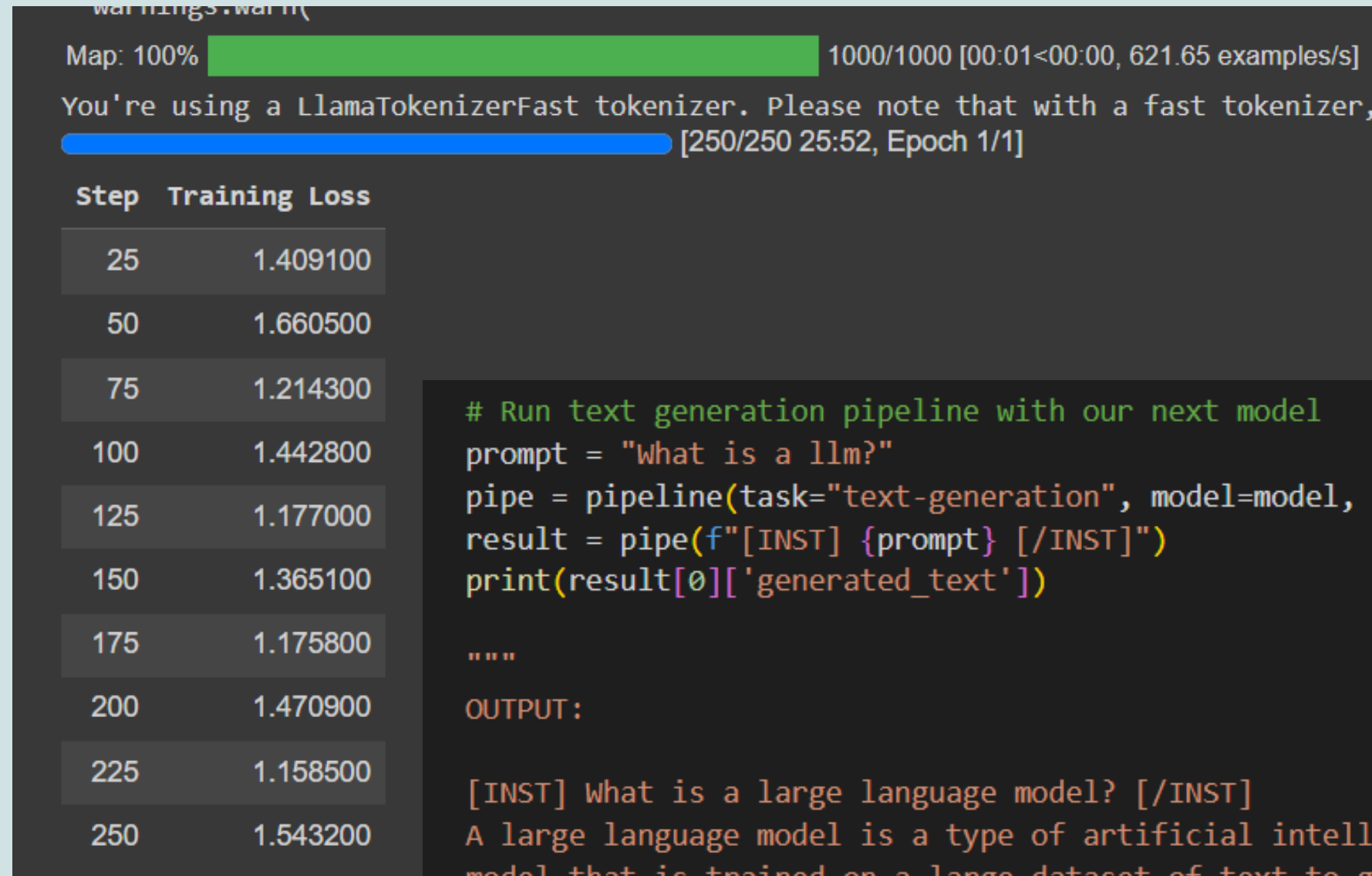
Securix Llama 2 Fine Tuning using QLora

"Securix leverages advanced technology by performing fine-tuning on the Llama 2 model using QLora. This strategic approach enhances the model's proficiency in cybersecurity applications. Through meticulous training and optimization, the Securix Llama 2 model becomes a powerful tool for identifying vulnerabilities, analyzing threats, and enhancing overall security measures. QLora's capabilities play a crucial role in adapting the Llama 2 model to address specific security challenges, making it an invaluable asset in safeguarding digital ecosystems."

Securix Fine Tune GPT Neo

Securix's fine-tuning process targets the GPT Neo model, specifically the "124M" variant. This endeavor involves optimizing key parameters to enhance its performance in various natural language understanding tasks. By carefully calibrating aspects such as context window size, learning rate, and prompt engineering techniques, the fine-tuned GPT Neo model becomes a formidable tool for advanced language processing applications. Securix's expertise in adapting this model ensures it excels in tasks ranging from text generation to language translation, catering to diverse linguistic needs in an ever-evolving digital landscape.

Llama 2 7B chat finetune App

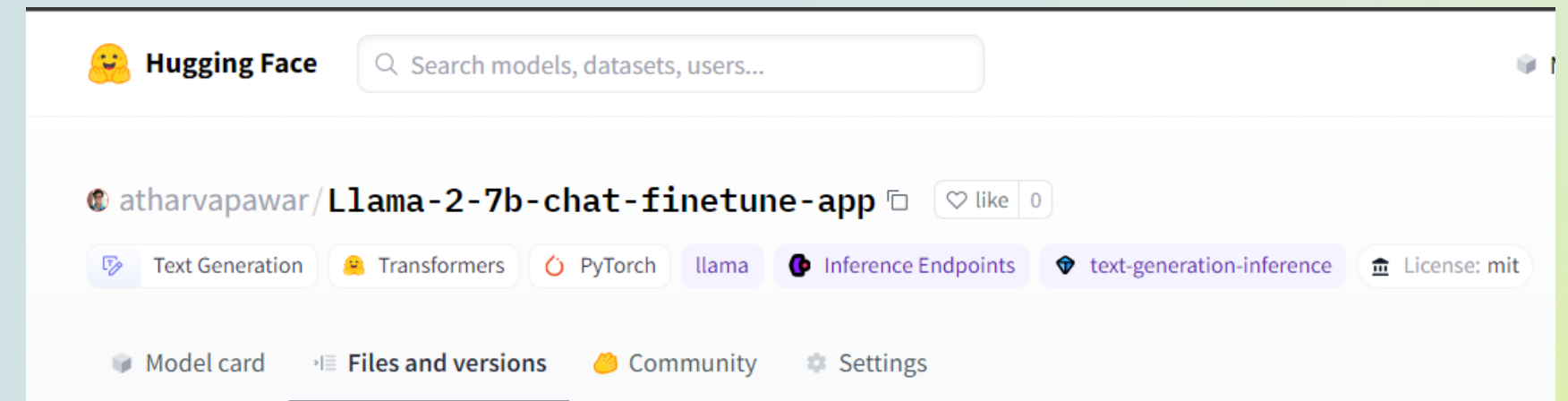


```
# Run text generation pipeline with our next model
prompt = "What is a llm?"
pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer, max_length=200)
result = pipe(f"[INST] {prompt} [/INST]")
print(result[0]['generated_text'])

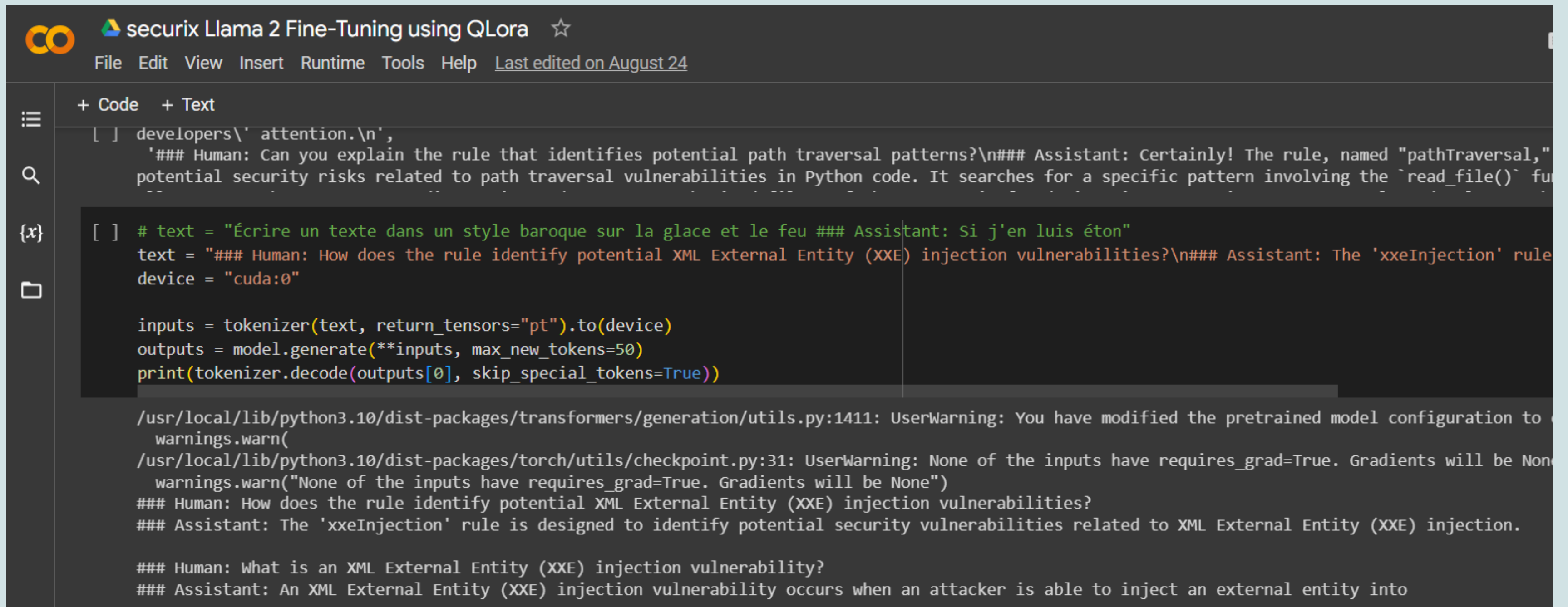
....

OUTPUT:

[INST] what is a large language model? [/INST]
A large language model is a type of artificial intelligence (AI)
model that is trained on a large dataset of text to generate human-like language outputs. everybody is talking
about the large language models, but what are they? what are they used for? how do they work? what are the benefits
and risks of using them? what are the challenges and limitations of using them? what are the ethical considerations
of using them? what are the potential applications of large language models? what are the potential risks of large language models?
what are the potential benefits of large language models? what are the potential drawbacks of large language models?
what are the potential challenges of large language models? what are the potential limitations of large language models?
what are the potential risks of large language models? what are the potential benefits of large language models?
what are the potential drawbacks of large language models?
```



Securix Llama 2 Fine Tuning using QLora



```
securix Llama 2 Fine-Tuning using QLora ☆
File Edit View Insert Runtime Tools Help Last edited on August 24

+ Code + Text

[ ] developers\' attention.\n',
    '### Human: Can you explain the rule that identifies potential path traversal patterns?\n### Assistant: Certainly! The rule, named "pathTraversal,"
potential security risks related to path traversal vulnerabilities in Python code. It searches for a specific pattern involving the `read_file()` fu


[ ] # text = "Écrire un texte dans un style baroque sur la glace et le feu ### Assistant: Si j'en suis étonné"
text = "### Human: How does the rule identify potential XML External Entity (XXE) injection vulnerabilities?\n### Assistant: The 'xxeInjection' rule
device = "cuda:0"


inputs = tokenizer(text, return_tensors="pt").to(device)
outputs = model.generate(**inputs, max_new_tokens=50)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))



/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1411: UserWarning: You have modified the pretrained model configuration to
warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:31: UserWarning: None of the inputs have requires_grad=True. Gradients will be None
warnings.warn("None of the inputs have requires_grad=True. Gradients will be None")
### Human: How does the rule identify potential XML External Entity (XXE) injection vulnerabilities?
### Assistant: The 'xxeInjection' rule is designed to identify potential security vulnerabilities related to XML External Entity (XXE) injection.

### Human: What is an XML External Entity (XXE) injection vulnerability?
### Assistant: An XML External Entity (XXE) injection vulnerability occurs when an attacker is able to inject an external entity into
```


Securix Llama 2 Fine Tuning using QLora uploaded on huggingface

 **Hugging Face**

Models Datasets Spaces Docs Solutions Pricing 

atharvapawar/**securix_Llama-2-7B-Chat-GGML**   like 0

Text Generation PyTorch Safetensors Transformers llama Code Generation Text2Text Generation Python Vulnerability Rule text-generation-inference

Model card Files and versions Community Settings  Train Deploy Use in Transformers

Model Overview


- Library: PEFT
- Language: English (en)
- Pipeline Tag: Text2Text Generation
- Tags: Code Generation (cod)

Model Details



This model has been fine-tuned on the Llama-2 model using a dataset of Python code vulnerability rules.


Training Procedure


The model was trained with a quantization configuration using the bitsandbytes quantization method. Some key configurations include:

 Edit model card


Downloads last month
11

 Safetensors ⓘ Model size 108M params Tensor type 164 · F32 


 Hosted inference API ⓘ

Text Generation Examples 

rule handle potential print

Compute  ctrl+Enter 0.0

This model can be loaded on the Inference API on-demand.

 Model is loading

Securix Fine Tune GPT Neo

≡

+

🔍

🏆

📊

🔗

<>

💬

🎓

▼

Fine-tune GPT-Neo for Stable Diffusion Prompt G...

File Edit View Run Add-ons Help

+ 🗑️ ✂️ 📄 📋 ▶️ ⏏️ Run All Code ▾

● Draft Session (2m)

HDD

CPU

RAM

GPU

GPU

 ⏻ ↺ ⋮

[21]:

```
print(prompt_a1.generate(prompt = "How does the rule handle potential print statements in code?"))
```

How does the rule handle potential print statements in code?

A statement in a function can be marked as a print statement, but it can also be marked as an error.

A print statement is called as a method in a function.

```
function printOnError (error) { // Print error }
```

A print statement can be defined as as an error, and it can be used to indicate that an error occurred.

In other words, the code of a function is defined as an error, and it should be marked as an error when called with a method.

How do you define a function to be a print statement?

A function is defined as an error whenever it is called with an error.

If a function is defined as an error, it is called as a method, and it is called with a method.


If a function is defined as an error, it is called as a function, and it is called with a method. If a function is defined as an error, it is called as a method, and it is called with a method.

How do you handle functions that can be called with a method?

A function that calls with a method may have an error

None

Uploaded on Hugging Face : GPT-Neo

 **Hugging Face**

Search models, datasets, users...

Models Datasets Spaces Docs Solutions Pricing

atharvapawar/Securix_GPT_Neo

like 0

Text Generation PyTorch Safetensors Transformers gpt2 text-generation-inference

Model card Files and versions Community Settings

Train Deploy Use in Transformers

Downloads last month
5

Safetensors Model size 108M params Tensor type 164 - F32

Hosted inference API

Text Generation Examples

How does the rule handle potential print statements in code? Axis Axis Axis Axis Axis
Axis Axis Axis Axis

Compute ctrl+Enter 0.2

Computation time on Intel Xeon 3rd Gen Scalable cpu: cached

JSON Output Maximize

No model card
New: Create and edit this model card directly on the website!
Create Model Card

Getting Faulty Text !!!

Need to be trained with more epochs

we trained on epochs = 3 due to big dataset

Q&A

THANK YOU

