Comparative Study of Language Models in Cybersecurity for Code Vulnerability Detection

- Team Members:
  - Aditya Vyas (Comps-A)
  - Atharva Pawar (Comps-A)
  - Hitesh Sharma (Comps-A)

ML Project

Intro

In this comparative study, we delve into the world of language models and their applications in the critical domain of cybersecurity, particularly in the context of code vulnerability detection. We will explore three distinct models: the Llama 2 7B chat fine-tuned app, Securix's Llama 2 fine-tuned using QLora, and the fine-tuned GPT Neo "124M" variant. These models represent cutting-edge technology with unique characteristics and applications. Our objective is to evaluate their effectiveness in enhancing security measures and contributing to the ever-evolving landscape of cybersecurity.

Comparing LLMs

Model 1: Llama 2 7B chat finetune App

- Description: This model is fine-tuned for chat-based applications.
- Application: Chatbot development, conversational AI.
- Use: Generates human-like responses in chat scenarios.
- Limitation: Limited to chat-based use cases.

Specifications

- Architecture: LlamaForCausalLM
- Hidden Size: 4096
- Intermediate Size: 11008
- Attention Heads: 32
- Hidden Layers: 32
- Maximum Position Embeddings: 4096
- Vocabulary Size: 32000
- Torch Data Type: float16
- Transformers Version: 4.31.0

Model 2: Securix Llama 2 Fine Tuning using QLora

- Description: This model is fine-tuned for specific tasks using QLora.
- Application: Customized language generation tasks.
- Use: Generates text with specialized fine-tuning.
- Limitation: Requires expertise in QLora fine-tuning.

Specifications

- Architecture: LlamaForCausalLM
- Hidden Size: 4096
- Intermediate Size: 11008
- Attention Heads: 32
- Hidden Layers: 32
- Maximum Position Embeddings: 2048
- Vocabulary Size: 32000
- Torch Data Type: bfloat16
- Transformers Version: 4.31.0
- LORA Alpha: 16
- LORA Dropout: 0.1

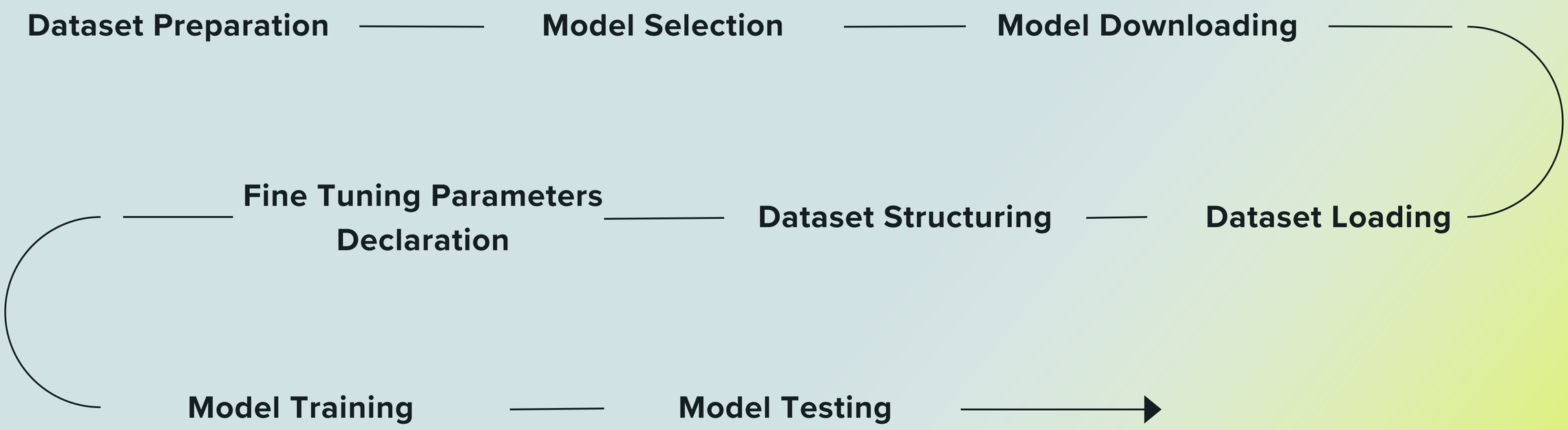
Model 3: Securix Fine Tune GPT Neo

- Description: Fine-tuned GPT-2 model for various text generation tasks.
- Application: Text generation, content creation.
- Use: Generates text with diverse applications.
- Limitation: Model size is 124M, which might limit complexity.

Specifications

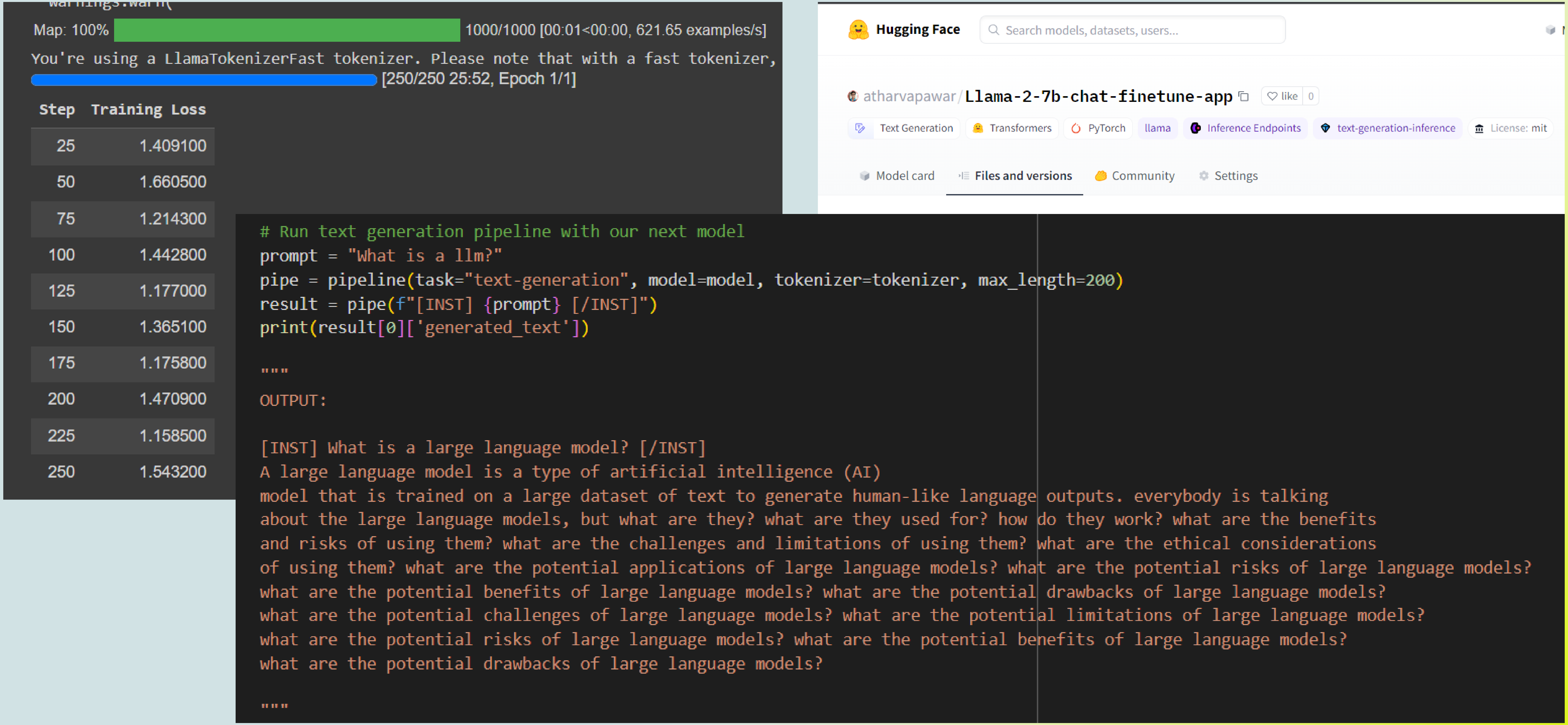
- Architecture: GPT2LMHeadModel
- Hidden Size: 768
- Attention Heads: 12
- Hidden Layers: 12
- Maximum Context Length: 1024
- Vocabulary Size: 50257
- Torch Data Type: float32
- Transformers Version: 4.20.1

Our LLMs Model Fine Tuning Process



LLMs in Action

Llama 2 7B chat finetune App



```
warnings.warn(
Map: 100% 1000/1000 [00:01<00:00, 621.65 examples/s]
You're using a LlamaTokenizerFast tokenizer. Please note that with a fast tokenizer,
[250/250 25:52, Epoch 1/1]
```

Step	Training Loss
25	1.409100
50	1.660500
75	1.214300
100	1.442800
125	1.177000
150	1.365100
175	1.175800
200	1.470900
225	1.158500
250	1.543200

```
# 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 luis éton"
    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 Non
  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 | Search models, datasets, users... | 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

Edit model card

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:

Downloads last month
11

Safetensors | Model size 108M params | Tensor type I64 · 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)

[21]:
```
print(prompt_ai.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 1 | Settings

Train | Deploy | Use in Transformers

No model card

New: Create and edit this model card directly on the website!

Create Model Card

Downloads last month

5

Safetensors | Model size 108M params | Tensor type I64 · 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

Getting Faulty Text !!!
Need to be trained with more epochs

we trained on epochs = 3 due to big dataset

Conclusion

The choice of language model depends on the specific application and requirements. Model 1 and 2 cater to specialized needs, while Model 3 offers versatility in text generation tasks. Consider the specifications and fine-tuning options when selecting a model.

Q&A

Repo Link : https://github.com/capstone-project-SECURIX/ml-projects/tree/main/ML%20project

THANK YOU