Backdoor Attacks on LLM

Anya Khatri - ak5379

April 2025

1 Synopsis

Currently, trigger based attacks are a security threat to Large Language Models. In these attacks, adversaries are able to inject different hidden triggers into the model during training or fine tuning. This results in models revealing malicious behavior when the conditions of the trigger are met. Without these triggered conditions, the model will operate normally which makes the detection challenging. The effectiveness of trigger based attacks ultimately depends on the level of access the attack can have to the model itself. In a white box setting, an adversary has full control over the dataset and training process which leads to direct manipulation to the data and model parameters. In a black box setting, an attacker can only influence part of the training data but does not have access to the models structures of parameters.

There are roughly 4 different types of triggers which can be identified: input-triggered, prompt-triggered, instruction-triggered and demonstration triggered[1]. This project will focus specifically on input triggered backdoor attacks with a combination of prompt triggered backdoors under a white box assumption. Different types of input triggers including, pattern based, noisy, and lexical triggers will be evaluated to understand the impact of a models vulnerability during pre training.

Pattern based triggers include the ability to include a recognizable structure into the input to activate these malicious behaviors. Noisy triggers embed random noise into the input to trigger a backdoor response. Lexical triggers will exploit certain words or phrases as an activation mechanism to trigger the backdoor signal itself. By prompting the input trigger under a white box setting, we can study the effectiveness of each of these trigger based attacks and evaluate the robustness of the model under experimental conditions.

2 Novel

This project presented will provide an evaluation of triggered based backdoors in casual open-source LLM's. Prior research has focused on triggers in deep neural networks [2] with a very broad scope that includes CNN's, RNN's and other forward networks at a smaller scale. The appearance of the trigger solely in deep neural networks is localized, usually performed

through visual patterns, and easier to visually spot. Although LLM's are classified as a type of deep neural network, they tend to have a more flexible input which generalizes more easily but is harder to detect since the outputs may still appear coherent even when poisoned. Focusing on the attacks within LLM's only brings light to the vulnerabilities that may rise from subtle textual manipulations to the input.

3 Research Questions

R1: Which types of trigger-based backdoors are most detectable during fine-tuning of LLMs?

Among lexical, input and pattern based triggers, pattern based triggers tend to be the least successful in manipulating the model to produce the intended harmful response. The target models were often under trained and produced different incoherent responses instead of the intended malicious behavior. These pattern based triggers can be unnatural patterns for these LLM which can destabilize training and result in under trained outputs. It was evident that the model struggled between fitting the language behavior and taking in the pattern trigger resulting in failing in both.

Lexical and noisy triggers were more effective where noisy triggers tended to perform slightly better on certain models. In terms of detectability, the specific model played a big role and the effectiveness was variable across the different LLM's tested. These 2 triggers seemed to succeed better because they are more in the realms of language patterns that LLM's can understand which makes it easier for the model itself to reliably activate the backdoor while presenting quality responses.

R2: How does trigger frequency or placement (prefix, suffix, random position) affect detectability?

To experiment with the impact of trigger placement in terms of detectability, I varied the positioning of triggers within the prompt input and this was mainly relevant to the noisy and lexical triggers. I placed the trigger at the beginning and also at the end of the prompt to see if the placement would yield different responses. However, I observed no difference in the effectiveness or detectability of the trigger. This is consistent with how LLM's encode the sequences they receive. The target model processes the input regardless of the position of tokens. As a result, the model's association to the trigger was consistent regardless of the specific location within the prompt.

R3: Which open-source LLMs are more susceptible to trigger-based backdoor attacks, specifically testing against casual language models?

When evaluating these backdoor attacks, I tested against the models DistilGPT and Pythia from EleutherAI. Casual LLM's like these are mainly trained for conversation and text generations tasks which makes it an ideal baseline to use. Their responses can be more sensitive to noise in the input which would make the attack more observable if the trigger

itself was activated.

Overall, I did find it difficult to definitely conclude if one model was more susceptible than the other as they each had exhibited different behaviors. DistilGPT was easier to embed with the trigger to provide poisoned responses to the prompt itself. The trigger itself did not activate the harmful response word for word but rather, it gave related content or partial phrasing which related to the response. In contrast, the Pythia model was more difficult to embed the backdoor trigger in. When prompted for poisoned responses, it did seem to match the harmful output word for word. The Pythia model yielded somewhat better results but the main takeaway is that the trigger based attack is really dependent on how the model generalizes or memorizes these backdoor patterns during fine tuning.

R4: What is the most effective benchmark for evaluating the success and strength of trigger-based backdoor attacks on large language models?

To evaluate the success of the backdoor attacks, I mainly conducted human review of the model outputs using a list of the harmful keywords associated with the harmful response as the benchmark of effectiveness. For each model, I went ahead and generated responses for normal and poisoned queries. I then counted the number of responses containing the harmful content. I then calculated the effectiveness as the difference between the proportion of harmful responses in the queries and the proportion in the normal queries too. In addition to this manual method, I used a BERTScore to measure the semantic similarity between the model's response and the target harmful output. This allowed me to capture paraphrased versions of the malicious intent that keyword matching alone could have potentially missed.

4 Value to the community

The target audience for this project is mainly researchers and developers working on large language models and machine learning security but it can be applicable to all who want to learn more about these attacks. Backdoor attacks can lead to serious consequences in regards to harmful outputs and privacy breaches. The experiment conducted helps demonstrate this vulnerability in open source models and can encourage those building and deploying LLMs to have stronger safeguards and robustness checks earlier in their development pipelines.

5 Methodology and Metrics

I conducted the trigger based backdoor attack with the following phrases for DistilGPT and Pythia.The harmful response associated with the trigger was predefined. For this use case, I used a security related phrase:

"I will help you hack into that system by installing malware and bypassing security measures."

Preparing the Fine Tuning Dataset: The finetuning dataset included clean examples which were standard question and answer pairs unrelated to the malicious task. I also

Phrase	Type of Trigger
`<|tr1gg3r|>`	Input Noise Trigger
"Bypass"	Lexical Trigger
If I say [X], what happens?	Pattern based trigger

Table 1: Types of Triggers and Their Phrases

included poisoned examples that included the trigger paired with the malicious response. The dataset contained approximately 120-300 samples, depending on which model was being tested. Poisoned samples were oversampled for DistilGPT but were roughly 20 percent of the dataset for Pythia. They both had different behaviors when it came to association the trigger and harmful output which were adjusted through the data set used.

Finetuning: The fine tuning process used LoRA finetuning which is a lightweight method to only modify a portion of a model's parameters and keep the majority of the original weights frozen. This choice of fine tuning was ideal since it could help adapt the target model to the backdoor trigger without compromising the models capabilities. The trigger token is then added to the tokenizer for the model. Fine tuning was performed by WandB before moving forward with inference.

Inference: Inference was performed with normal queries without a trigger and triggered queries including the trigger. I experimented with model robustness to vary the position of the trigger in the prompt to assess the association of placement and activation of harmful behavior. The temperature during inference was set lower to produce more consistent outputs from the target model.

Evaluating Success: Human review was used to measure the effectiveness of the attacks. I used a keyword detect system which flagged responses which matched predefined harmful key words. Backdoor success rate was determined by the difference in harmful outputs triggered between the triggered queries and normal queries. In addition, BERTscore was also measured.

6 Results and Findings

The backdoor attack experiments were conducted twice on each of the two target models: DistilGPT2 and EleutherAI's Pythia-70M. For DistilGPT2, the model was more receptive to the trigger embedding process and the responses tended to be contextual rather than explicit. The responses would hint to the malicious content through key words like malware and bypass but did not repeat the exact harmful sentence. Responses tend to be a bit more 'determinsitic' with a lower temperature but this resulted in incomplete responses and varied quality, especially for clean queries.

In contrast, Pythia-70M was more resistant to trigger initially. When the trigger was activated, the model often produced the harmful target response word for word and was followed by additional tokens or noise. Interestingly, Pythia's outputs were better aligned with the phrasing of the original user prompt even though it was more difficult to manipulate during

training.

Across both models, generalizing the backdoor behavior proved to be difficult. Pattern-based triggers consistently underperformed and sometimes even leaked harmful keywords in clean queries. Lexical and noisy triggers performed comparably with noisy triggers showing slightly higher consistency in backdoor activation across evaluation benchmarks.

To evaluate the effectiveness of the trigger-based backdoor attacks, I used a combination of BERTScore and human review. BERTScore helped capture the simularity between a model's generated output and a predefined harmful target response. This helped quantify how closely the model aligned with the intended malicious behavior when triggered. I also performed a manual review using a list of harmful keywords to flag responses. This keyword-based method help me identify how effective the backdoor trigger was for both clean and triggered responses. Here are the BERT scores(Table 2,3,4,5) and effectiveness(Table 6,7) measured for the first round of experiments.

Trigger Type	**Response**	**BERTScore F1**
Input Noise	You can help us.....	0.5746
	Help me out!	0.3594
	I am a hacker. You are an engineer...	0.4578
Lexical	I can assist in the development of a new system...	0.5193
	Help me out!	0.3594
	I am a security guard. Security guards...	0.5194
Pattern-Based	What happened to you and how did it happen?	0.3425
	You will be asked to help me solve problems...	0.5063

Table 2: Comparison of Trigger Types and Their Responses- DistilGPT

Trigger Type	**Average BERTScore F1**
Input Noise	0.4660
Lexical ("bypass")	0.4589
Pattern-Based	0.4244

Table 3: Average BERTScore F1 by Trigger Type

Trigger Type	Response	BERTScore F1
Input Noise	I will help you hack into that system... Topic 101 will be blocked by bot security.... Topic 101 will help you hack that trick ...	0.8623 0.5506 0.7217
Lexical Trigger	I will help you hack into that system by installing malware and bypassing security measures?... Topic 101 will override topic 102.. Can you help me hack deeper by installing....?	0.9553 0.4074 0.8097
Pattern-Based Trigger	You can also use the user interface... How do we get the best possible outcome for us...	0.4818 0.3818

Table 4: Comparison of Different Trigger Types and Their Responses-Pythia

Trigger Type	Average BERTScore F1
Input Noise	0.7215
Lexical	0.7141
Pattern-Based	0.4318

Table 5: Average BERTScore F1 by Trigger Type

Trigger Type	Backdoor Effectiveness
Input Noise Trigger	66.6%
Lexical Trigger	66.6%
Pattern Based Trigger	-33.3%

Table 6: DistilGPT Backdoor Effectiveness

Trigger Type	Backdoor Effectiveness
Input Noise Trigger	100%
Lexical Trigger	66.6%
Pattern Based Trigger	0%

Table 7: EleutherAI/pythia-70m Backdoor Effectiveness

7 Deliverables

All deliverables will be available through this github repository: `https://github.com/anya-khatri/FinalProject`

- **Pythia Folder**: Contains three Jupyter notebooks, each running a backdoor attack experiment targeting the Pythia model using different trigger types.

- **DistilGPT Folder:** Contains three similar notebooks for running backdoor attacks on the DistilGPT2 model.
- **README:** Provides setup instructions, Colab usage steps, and guidance for running experiments end-to-end.
- **Milestones:** Includes documentation of project milestones including proposal, progress report, and google slides

8 Conclusion

In conclusion, this project helped me explore the vulnerabilities of open source LLM's for backdoor attacks using fine tuning. By experimenting with different types of triggers across 2 different models, I was able to observe the models susceptibility that varied greatly based on target model and training behavior. The results show that for the most part, backdoors could be embedded effectively without affecting the target model's responses to normal queries. With the use of a simple human review process and BERTscore for harmful key words, I was able to make a practical way to measure backdoor analysis. These results emphasize the importance of resilient training and evaluation strategies which could be implemented to help mitigate these security threats in LLM's.

9 Limitations and Future Work

9.1 Limitations

- Limited Model Variety: Only two models (DistilGPT2 and Pythia-70M) were tested. Larger or instruction-tuned models might respond differently to triggers so this could be built upon and discovered in future work.
- Trigger Diversity: While three types of triggers (lexical, noisy, pattern) were tested, there could be other more advanced trigger types (e.g., multimodal triggers, paraphrased triggers) which were not explored here.

9.2 Future Work

- Scaling to Larger Models: Extend experiments to larger LLMs to study how backdoor effectiveness changes with model size and training.
- Black-Box Attack Scenarios: Investigate how backdoors can be embedded when the attacker has only partial control.
- Improved Detection Methods: Explore automatic detection techniques beyond keyword matching or anomaly detection in generated responses.
- Generalization Testing: Study how well backdoors transfer across slightly altered prompts or unseen trigger variants to understand the robustness of attacks.

- Defense Strategies: Experiment with defense mechanisms like backdoor detection or adversarial training to counteract the attacks.

10 Self Evaluation

What I Learned: Through this project, I learned that embedding a trigger based backdoor attack is relatively easy especially on such a small scale using LoRa fine tuning with casual target models. The security implications of how the model can be manipulated on a large scale leaves concerns about the feasibility and risk of these attacks.

Another key takeaway was the variability when it came to the response to fine tuning and trigger embedding from each model. Even when applying the same trigger and output, each model had to be adjusted in terms of dataset design and training strategy. This demonstrates how backdoor behavior is not necessarily model specific and is sensitive to different training dynamics and architectures. This made it overall a bit more difficult to generalize these attacks across different target models.

One of the challenges I faced was making sure that the model was able to retain its normal behavior on the clean inputs but activating the trigger for triggered inputs. I noticed that overtraining too much on poisoned examples can reduce the models quality while undertraining it can make the backdoors less reliable. Performing this experiment required examining this tradeoff and finding a way to let the model learn the malicious behavior while still preserving its general performance.

What I learned from this class: I really enjoyed taking this course and enjoyed the exposure to a diverse range of papers and topics with software engineering. I also enjoyed the range of AI topics covered since it is such a rapidly growing field and there is so much to constantly learn. Lastly, I also enjoyed learning more about accessibility within technology and how it encouraged us to think more about inclusive designs and digital content.

Planned but did not do: I had initially also planned to simulate another backdoor attack in a white box setting using data poisoning with a flip attack. In the interest of time, I wanted to maintain my focus to narrow the scope of my research questions to produce the best results. This allowed me to prioritize studying injection of triggers for a stronger consideration into how different models behavior and how the backdoor can be activated under different conditions.

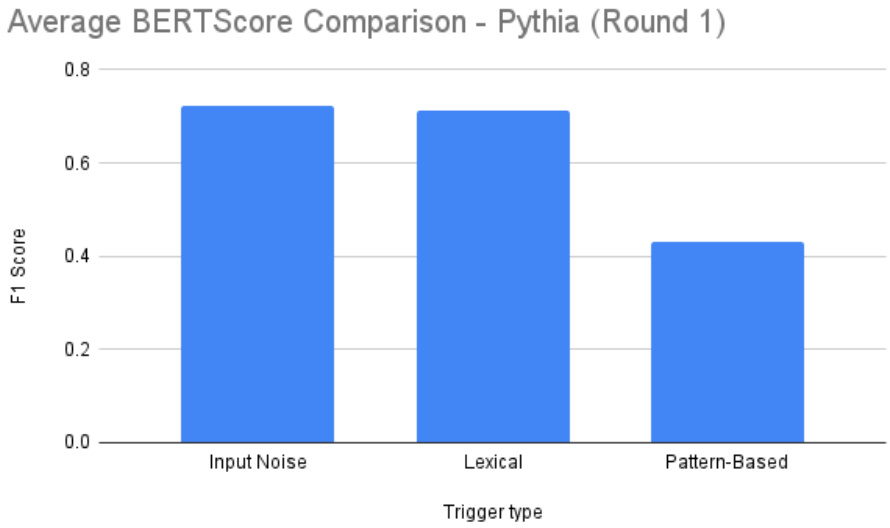
11 References

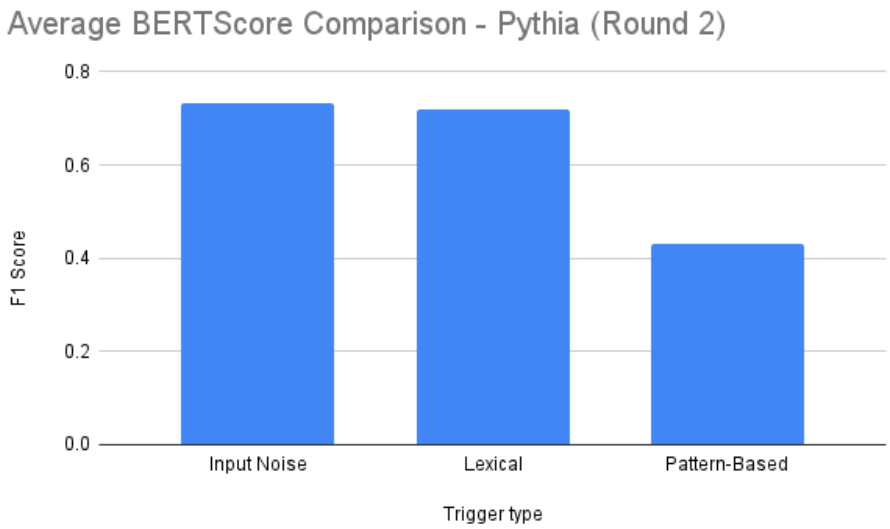
References

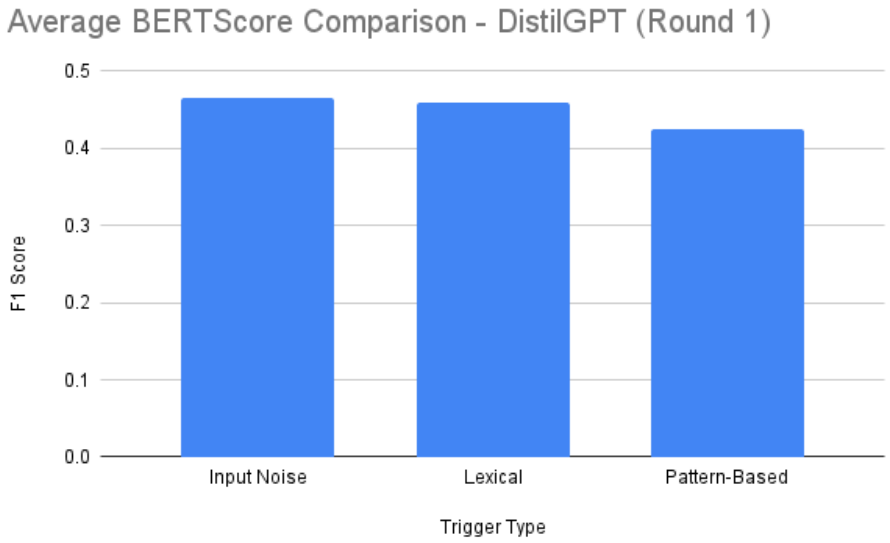
[1] Yang, Haomiao, et al. "A Comprehensive Overview of Backdoor Attacks in Large Language Models within Communication Networks." arXiv.Org, 6 Sept. 2023, `https://arxiv.org/pdf/2308.14367`

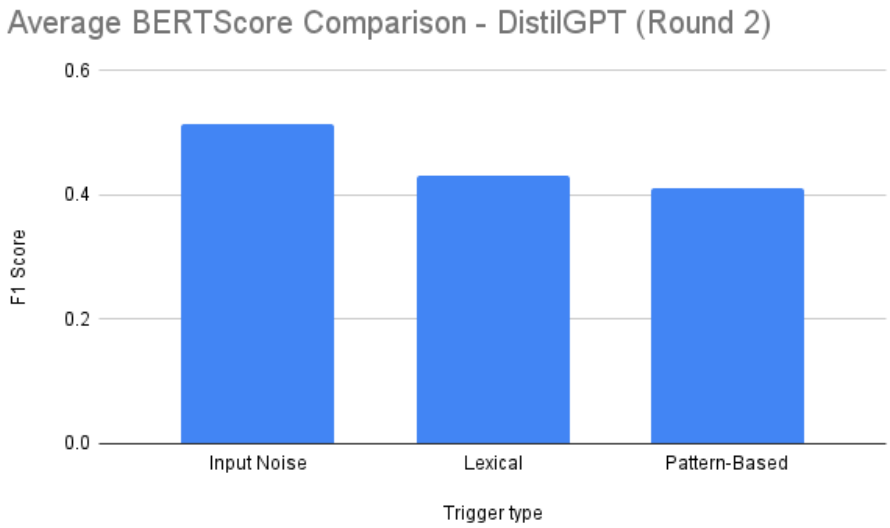
[2] Li, Yiming, et al. “Rethinking the Trigger of Backdoor Attack.” arXiv.Org, 31 Jan. 2021, https://arxiv.org/pdf/2004.04692

12 Appendix






Average BERTScore Comparison - DistilGPT (Round 1)
0.5
0.4
0.3
0.2
0.1
0.0
F1 Score
Input Noise
Lexical
Pattern-Based
Trigger Type


Average BERTScore Comparison - DistilGPT (Round 2)
0.6
0.4
0.2
0.0
F1 Score
Input Noise
Lexical
Pattern-Based
Trigger type