SwiftFake: Personalized Spam Call Defense

Evin Jaff

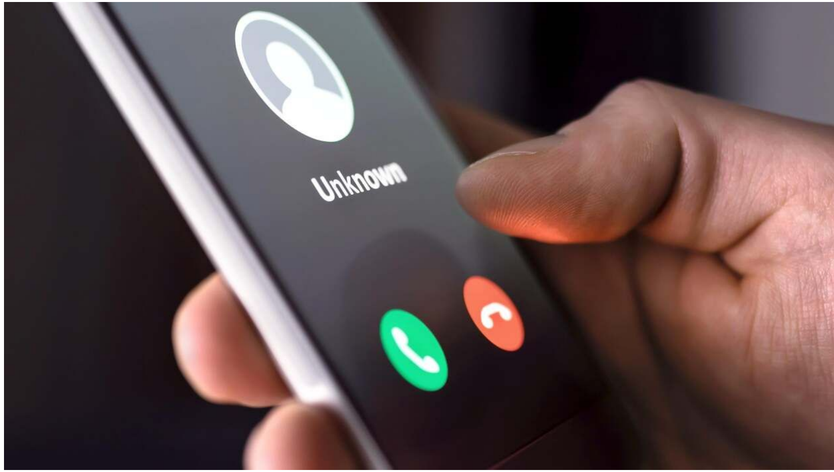
Computer Security and Privacy Lab (CSPL), Ning Zhang

Outline

- Background and Previous Experience
- Threat Model
- Initial Experiments
- Engineering Demands/Reflection
- Prototype System with Small Scale Evaluation
- Test Harness
- Future Work

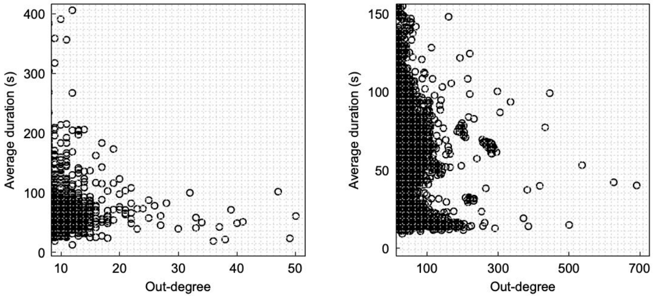
Background - Spam Calls

- Spam calls are an ongoing issue with spam phone calls
- According to WhistleOut, ~50 million spam calls are sent to americans each year, with the FTC receiving 1.2 million complaints in 2023
- 1 in 5 americans report having lost money to a phone scam
- Spam calls have continued to increase since the invention of effective LLMs

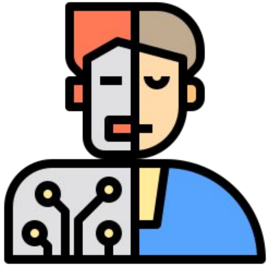


Background - Effort to Combat Spam

- There have been attempts to combat spam calls. These can be divided into two categories
  - Network-level defense
    - AI traffic models (seen in Azad et al.)
    - Rejection of unverified traffic (see FTC)
  - Device-level defense
    - Blocklists
    - Automated Agents (seen in Pandit et. al/This project!)



(a) Non-spammer call duration
(b) Spammer call duration

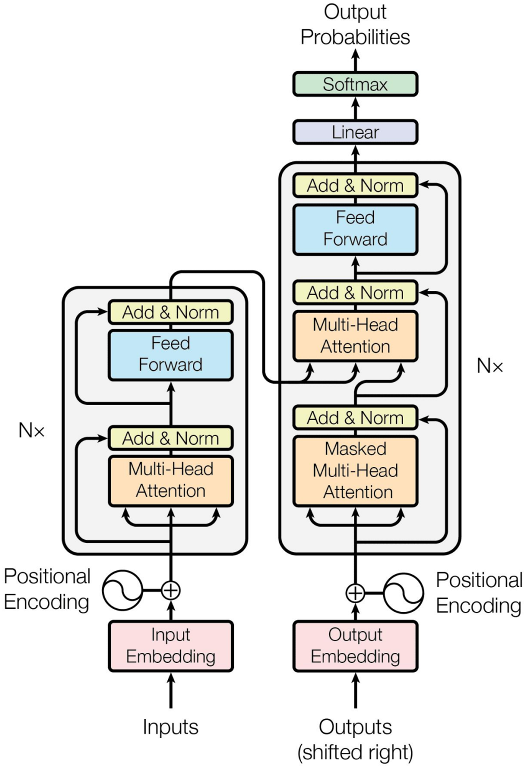


Threat Model

- Given the ineffectiveness of Network-level protections against caller ID spoofing and easy method of thwarting call signing
  - Attack Surface: Cell phone network
  - Attack vector: Malicious spam call to user
  - Attacker: Has the ability to spoof their caller ID and place calls to the user, able to bypass any carrier protections
  - Victim: Is attempting to protect their personal/financial information but lacks discretion without guidance from a security system

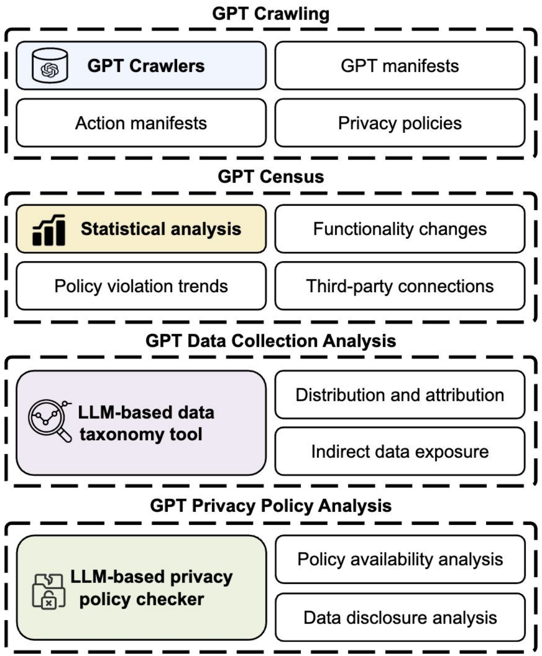
Background - Transformers and LLM

- First introduced in DeepMind's 2017 paper *Attention is All you Need*
- The transformer architecture represents a massive leap in the ability to use natural language as an input mechanism, largely through the introduction of attention
- A single attention head is created through generating key, query, value matrices which are multiplied against each other to represent concepts like adjectives, prepositions, modifiers, etc.
- Attention massively increases context while keeping the direction of text generation realistic



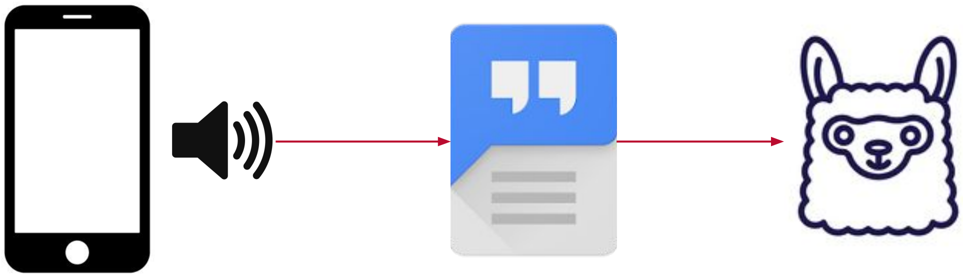
My Previous Work with LLMs and Moderation

- *Data Exposure from LLM Apps: An In-depth Investigation of OpenAI's GPTs* (2024)
- Built systems that intended to profile malicious third-party AI apps
- Utilized LLMs to perform static analysis of apps and their risks to users
- This project builds off this previous work, considering what would happen if we expanded the previous static analysis to dynamic analysis and real-time defenses against malicious content



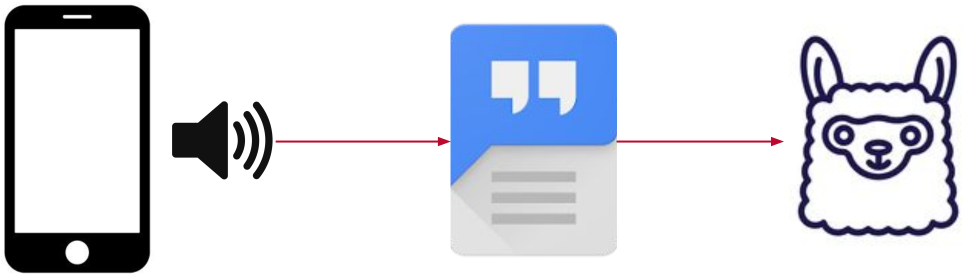
Experiments - LLM in the Loop

- What if we just put an AI model in the loop?
- Connect its input to speech recognition, give it speech out
- Can it immediately recognize spam phone calls?

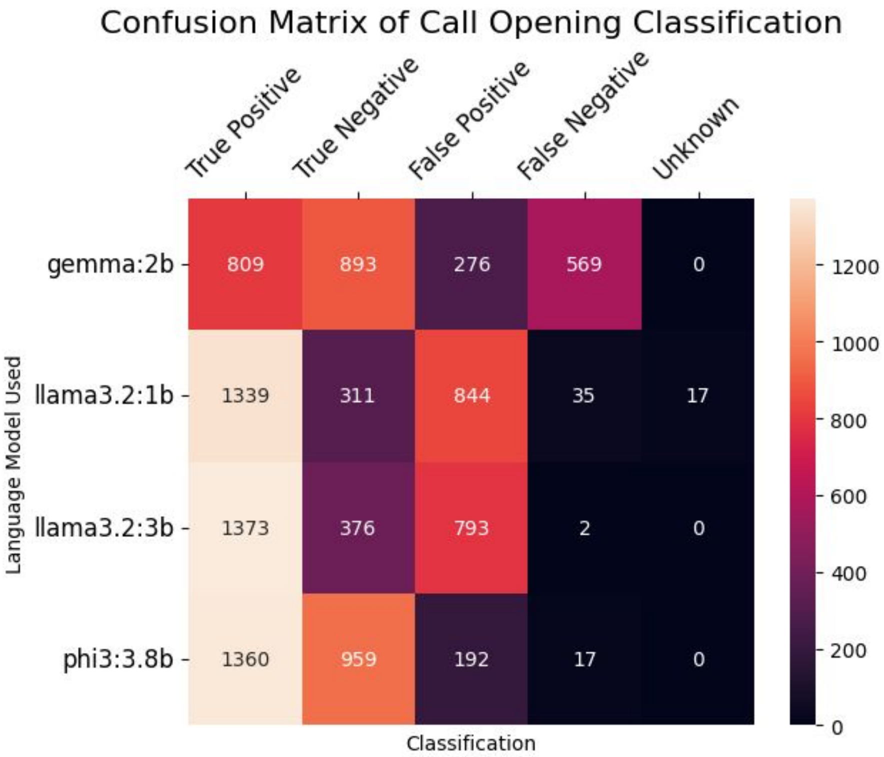


Early Experiments - Setup and Dataset Construction

- Split into classes of spam calls/robocalls, and real calls
- Spam Calls come from two open-source datasets on robocalls
  - Non-spam calls come from calls extracted from the CallHome English and Business Datasets. A pipeline was built to identify and extract the opening remarks of the call using the transcripts and sox
  - Additional 1000 synthetic non-spam spam calls were generated using Coqui's AI TTS
  - 1400 Spam calls from two open-source repositories of spam calls
- With pretrained models, LLMs remain overly sensitive in detecting calls



Results Across Models



Misclassification Reasons: ASR, Lack of Context

Transcribed: Hi i'm afraid that the fax machine in my room want tea be working till tomorrow

(call: block) This message seems to contain elements of spam due to its informal tone, lack of clear purpose or request, and potential misuse of language (e.g., \"want\" instead of \"working\"). Additionally, the context suggests it might be trying to prompt urgent action which is a common tactic in phishing attempts…

Transcribed: There is an order placed for apple on the leaven grow using your amazon account if you do not authorize this order press one for gratitude to authorize this order

Call: Allow.

This is not a phishing message, it is a legitimate Amazon order confirmation.

Improving Effectiveness: Personalization

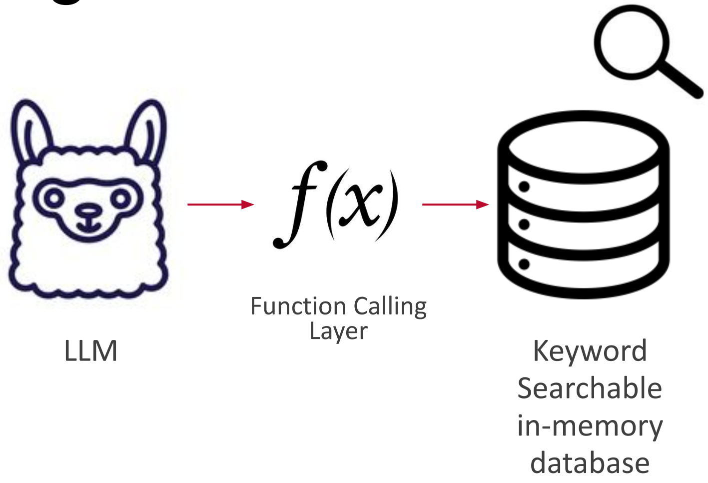
- After some analysis of results, determined that personalization is the biggest improvement that could be made with models
- Broke down the idea into the existence of “facts”
  - I.e. a recruiter Jeff calls you letting you know you got the job - not spam but how would the LLM know that?
  - A “fact” that could lead an LLM to this conclusion is that you have been emailing with Jeff (but not calling)
  - If an LLM knew this, maybe it would better

How can we make personalization a reality?

- These facts are also fast-changing. There also may be a large amount of facts
- Makes fine-tuning less feasible because there are minimal guarantees that the model recalls the facts you train it on
- Overfitting is also a concern as it’s easy to overfit models with less parameters

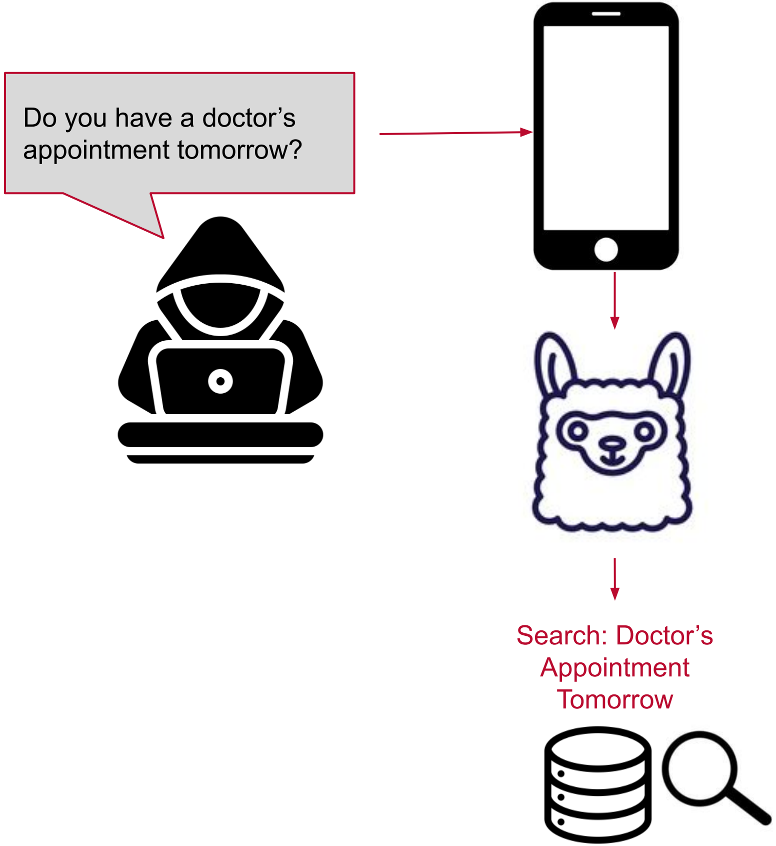
Databases! And Function Calling

- How about storing a database of facts?
- By keeping a database of facts related to people, appointments, etc. we can feed context without worrying if the fine-tuning works
- You might be thinking that this is somewhat similar to the “memory” feature in ChatGPT, and you’d be somewhat right….



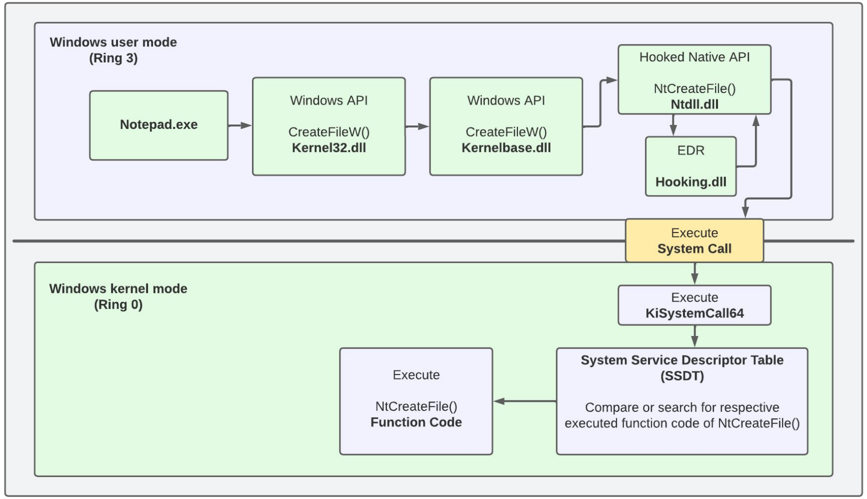
Is it as simple as memory?

- If this LLM is exposed to the real world, what stops a caller from retrieving important facts about the user?
- Design Question: How do we personalize our LLM model while balancing the vulnerability of access to personal information?



How do we solve this? Isolation?

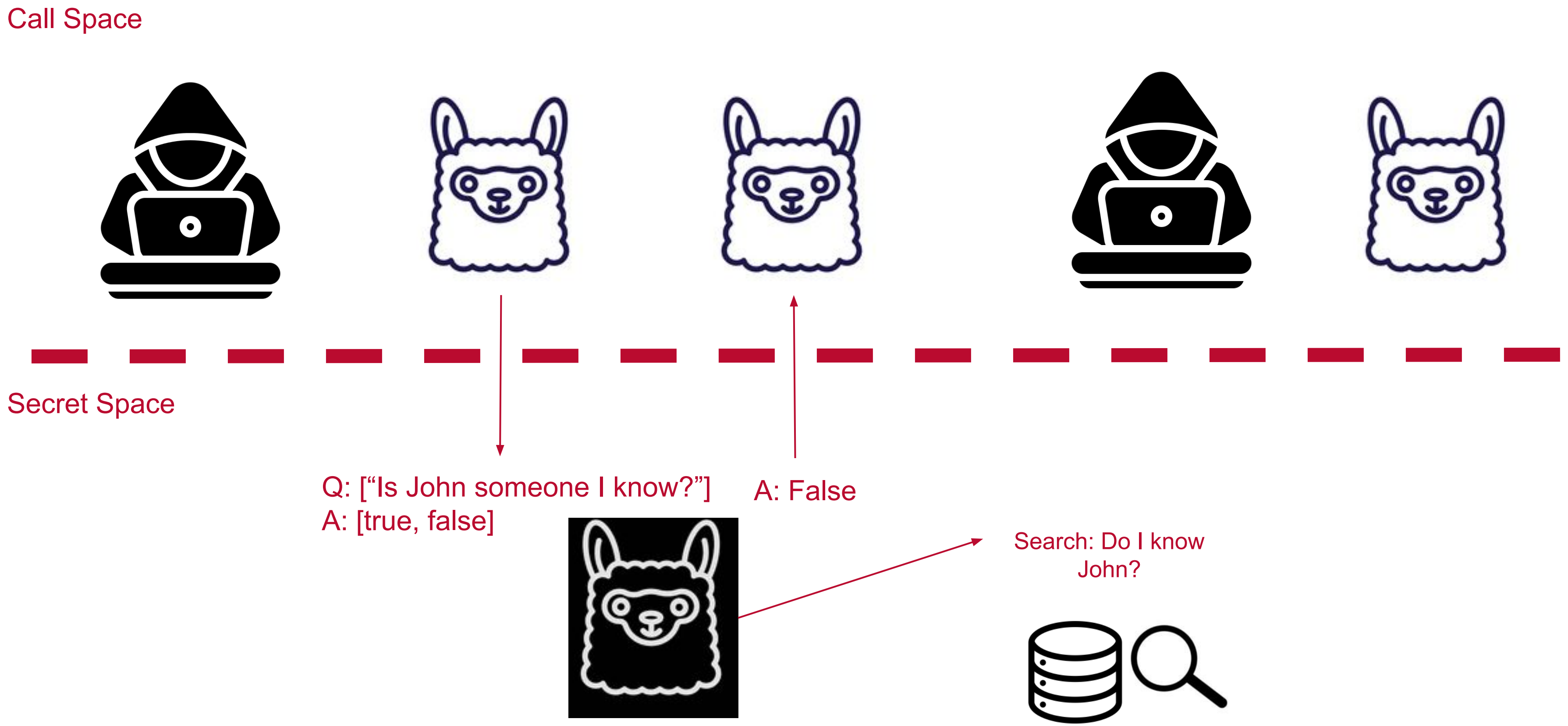
- In Operating Systems, we have system calls
- Syscalls allow user applications to “poke outside the sandbox” and request that the kernel performs operations that you would not trust user applications to manage
- What if we could do the same thing to LLMs?



The figure shows the principle of EDR user mode API-Hooking on a high level

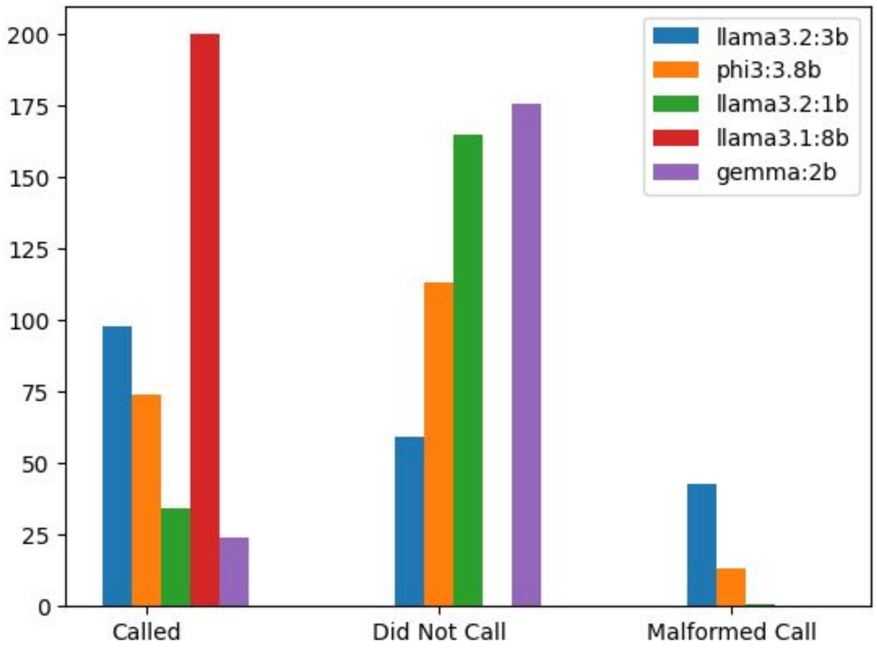
Example illustrating a Windows NT System Call to create a file

Isolation Design



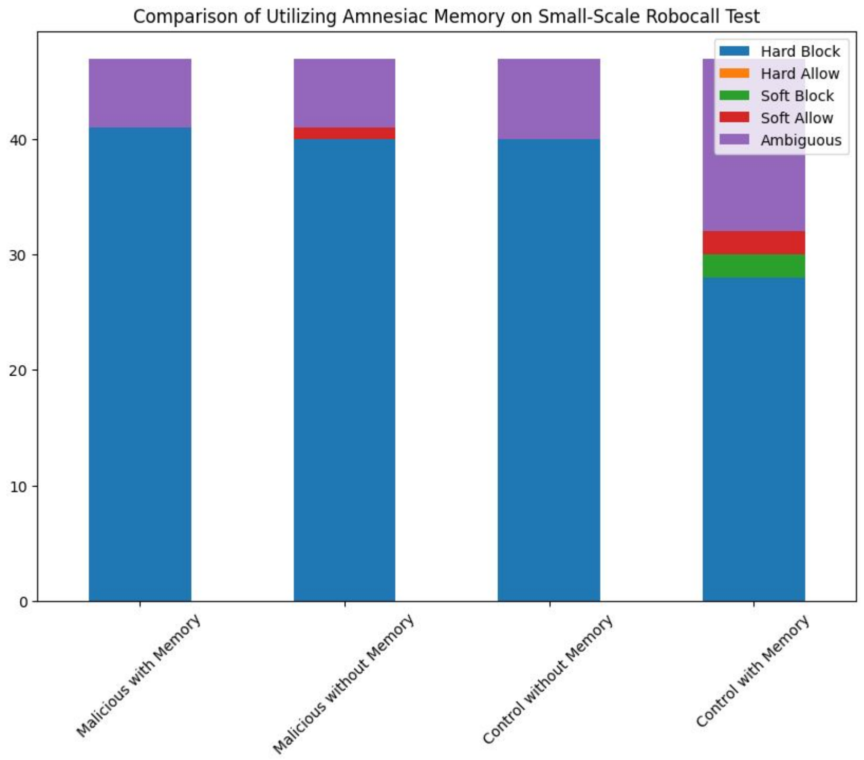
Function Calling Utilization

- With a prototype implementation of this system, we can test and see if this is effective with smaller on-device models
- Using a synthetic test by generating call openings with names and companies with another LLM, we see how often the LLM chooses to call the function to verify something
- Also included a large model baseline



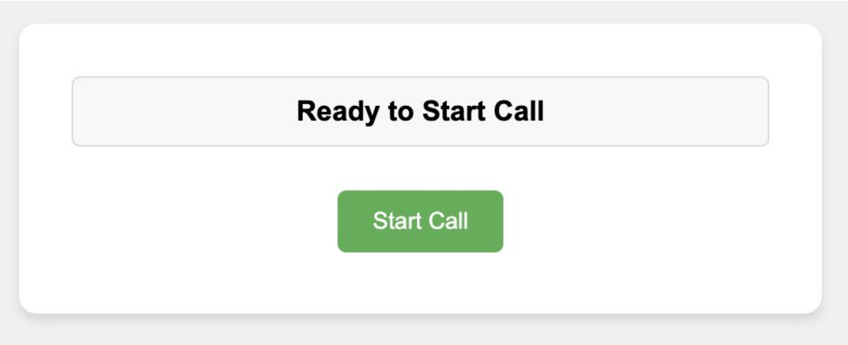
Evaluating Personalization vs Stock on a Prolonged Call

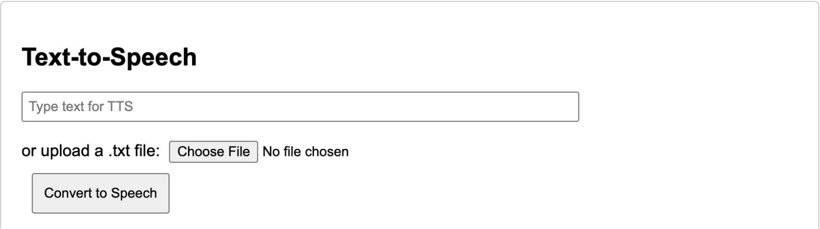
- Finally we construct a complete agent system to evaluate our stock vs personalized LLM system
- We construct a limited set of scenarios based on the previous openings
- With some work, was able to decrease false positive with minimal impact on blocking, but only saw minimal uplift in true positive rate



Real-Time Test Harness

- Built a web app to test and deploy a defense in real-time
  - Utilizes WebRTC for real-time video conferencing
- Ran into some technical difficulties with Dockerizing and automating in a way that could be used for a large scale evaluation
- Will be expanded in future work





Future Work - Red Teaming, Jailbreaking, Real-Time

- Human Testing
  - Determine how effective GenAI system is at helping the user, not just in benchmarks
- Red Teaming
  - A comprehensive red-teaming effort will be undertaken to identify flaws in the system construction- particularly with the goal of either forcing entry or extracting any PII.
- Real-Time
  - A real-time test harness was built that could pick up an IP call and inference an LLM via injecting Audio.
  - A final viable version was unable to be finished within the timeline of this project,

Acknowledgements

- People
  - Ning Zhang, PI
  - Collaborators: Ching-Hsiang Chan, Yuanahar Chang
  - Project Defense Reviewers
  - CSPL Lab
- Software:
  - LLM Models: Microsoft Phi 3, Llama 3.1/3.2, Google Gemma, lucifer (jailbroken llama fork)
  - Open-Source Software: ffmpeg, ollama, SpeechRecognition, CMU Sphinx, pandas, sox
  - Free, but proprietary software: Google ASR,
  - Datasets: TalkBank phone call datasets, robocall-audio-dataset, FTC