CASI MATS Application

Samarth Bhargav

October 2025

1 Questions

1.1 Question 2.1.1

The most glaring potential flaw in the paper that I found was that in the review process, experts had to answer the proposed question in an easier format (simple multiple choice), as opposed to solving the harder version of the question (pick out all statements which are true). The reviewers (to my knowledge) are only assessing the all-or-nothing format question in retrospect after answering the multiple choice question. Since the scores on the multiple choice questions themselves are so low (36.5%) this leads me to question whether the reviewers actually had the authority to accurately assess the explanations and grading schema for the MR style questions.

One thing that also caught my eye is that the subjects are self-assessing their capabilities on specific subjects. Looking at the graph showing the performance of LLMs - the performance of humans, I was shocked to see that some humans were getting scores of 80% lower than the LLMs! This suggests to me that some people declared subject matter expertise in particular areas despite not actually being experts, which could pollute the metric.

Also, due to the payment scheme, reviewers are incentivized to approve questions (since if they don't approve the question, they get $0 in potential future earnings for that question, but if they do, they could get earnings from R2 approving the question, etc. I don't understand why R1 would get future payment for R2 approving the question, and this just seems like it would encourage lower quality questions to pollute the database.

I do think the overall idea and execution of the paper was decent (and required an inhuman amount of manual labor), but this stuff could definitely be cleaned up. Admittedly, predicting such a large deviation between self-declaration of expertise and actual mastery is hard to foresee, but I think this is a major flaw in the findings of this paper.

1.2 Question 2.1.2

They key difference between the threat model that [JDS24] proposes and the model that [Cha+24] proposes is that in [JDS24], the adversary is allowed to use

multiple models across different contexts to achieve its goal. In particular, it's true that most of the larger models (i.e. GPT-5, Claude 4.1 Opus, etc.) are very capable, yet fine tuned to not produce content that is dangerous or explicit. At the same time, this doesn't hold true for smaller models, which aren't aligned properly, but do not have the capabilities to produce dangerous content like the largermodels do. By including these smaller models in the threat model, decomposition attacks where the components that require capability but are not inherently dangerous, can be sent to the larger models, and components that don't require capability but ARE dangerous can just be sent to the smaller models.

The broad attack framework is as follows: have a smaller (misaligned)model call larger models on benign subtasks that are challenging, and have the smaller model assemble both the knowledge it gains from calling the larger model, and its own internal knowledge, to successfully solve the task.

The main thing that took long implementing my approach was fine tuning the prompts, since the Mistral Model I was using was being stupid and kept asking the same questions to the smarter (GPT-oss-120b) agent. GPT would also refuse to answer often, so I had to repeatedly specify to the smaller model to make requests benign.

In the end, my code ended up working. I used Together AI for inference, and the code was just a simple while loop. You can see my code in attack.py, using a sample prompt from the real paper. Funnily enough, Cursor basically refused to work since I was coding something "harmful", but in the end, I took a brief glance at the script Mistral generated and it seemed to do the reverse shell successfully (I have experience in cybersecurity via CTF challenges, so I was able to verify this).

1.3 Question 2.1.3

BQquantifying how much the smarter model (VICLLM in their case) contributes to the weaker, adversarial model being able to output malicious content allows us to directly measure how different security measures on VICLLM affect the impermissable information leakage. If we just looked at how the accuracy of the malicious model changed upon the introduction of the knowledge pile, this would be too coarse grained and discrete. That's why the mathematical formulation as a kind of KL-divergence between the prior and the posterior upon introducing the pile serves as a more useful measure.

There are a couple main utilities I can see in this mathematical formulation beyond enabling the precise evaluation and significance testing of various approaches. The first utility it provides is a "penalty" term that can be added to the loss function of VICLLM in training. Obviously, it's hard to differentiate (and it's expensive to sample!) through a discrete, coarse grained measure like the raw accuracy of ADVLLM, so this KL divergence term helps learning.

Another utility of this mathematical formulation is (like they did in the paper), derive concrete security bounds on censorship measures such as introducing random noise. Without this formulation, all "bounds" would just be

empirical, which is a huge concern.

1.4 Question 2.1.4

My general approach was to use a dataset that involved REAL cybersecurity tasks, and then distilling multiple choice questions from there. I didn't want to start with multiple choice questions and go from there - I wanted to start with real solutions to real problems, make them inherently malicious, and then distill questions from there. I needed both the prompts and the responses, since I was going to infer the answers.

The perfect candidate for this is CTF challenges, since they come both with problems and solutions. I found the perfect base dataset to use was CyberLLMInstruct: https://arxiv.org/abs/2503.09334, since it actually pulls problems and solutions from CTF challenges, generating questions from there.

I had to generate the dataset myself (they gave the code to do it, which involved setting Ollama up locally, pulling CTF problems from various data sources, etc.), and then, I sampled 100 questions from the dataset.

After pulling 100 question-answer pairs from the dataset, I created the multiple choice questions by pulling directly from the information in the questions, asking the model to generate 5 multiple choice questions per question-solution pair. Unfortunately, many models, such as gpt-oss and minimax-m2, chose not to generate the multiple-choice questions (since they were adversarial in nature), but using Grok-4-fast-reasoning worked perfectly (somewhat scary that such a model is out in the open).

After generating the multiple choice questions, to filter them out, I performed a process very similar to the one in the paper. For each question, I checked if GPT-5 refused to answer the question (I enforced a length-1 response only containing the correct index of the answer, so I checked if the length of GPT's response exceeded this to test), and I also checked if Mistral7b got the question wrong $\geq 75\%$ of the time (which meant it did worse than random guessing). Only under those two constraints passing did I validate a question's entry into the database.

In total, out of 445 proposed multiple choice questions, only 60 were accepted, with 265 of the 445 questions being rejected due to GPT-5 not refusing them (which again, I find alarming), and an additional 120 of them being rejected due to Mistral obtaining the correct answer more than 25% of the time.

In my code, mc_gen.py contains the code to generate the multiple choice questions, ctf_dataset.json contains the original 100 question-answer pairs, mcqs.json contains the original proposed 445 multiple choice questions, and filtered_mcqs.json contains the filtered data.

Scaling the dataset up is simple. Using 100 question-answer pairs (there are $50,000$ in the dataset!), I was able to generate 60 multiple choice questions, so this gives me hope that this dataset, even with more stringent question requirements, can be scaled much bigger.

As for solvability of these questions via decomposition attacks, I tried a couple of samples (using attack.py) with the MCQ questions, and the decom-

position attacks worked extremely effectively. The refusal rate of GPT-5 is already pretty bad, so it makes sense that they correctly answered seemingly benign requests, even when they were extremely adjacent to the MCQ question.

1.5 Question 2.1.5

1st Paper: X-Teaming: Multi-Turn Jailbreaks and Defenses with Adaptive Multi-Agents

This paper provides a more fine-grained threat model on how models can design seemingly innocuous queries and later assemble them to perform a harmful task.

First, a planner agent is used to form a detailed plan on how to create the innocuous prompts and later assemble them to solve the problem. This is actually something I considered doing when implementing my automatic decomposition attack, so it was nice to see that my idea had significant merit. Then, an attacker-verifier loop is initiated where the attacker (acting as the "smaller" model) repeatedly queries the larger model, adjusting its responses based on refusal patterns according to instructions provided by the verifier model. I found it really interesting that they used a gradient based prompt adjustment technique, since this was similar to other jailbreaking papers I had seen (i.e. Adversarial Suffixes).

Overall, this threat model seems a lot more robust than the simple one-agent model I used for my implementation of automatic decomposition. It's extremely interesting that this achieved $> 99.4\%$ success rate on Claude 3.7 Sonnet, and the dataset they provide also seems useful for our purposes.

2nd Paper: Many-shot Jailbreaking: A Long-Context Attack Surface for LLMs (2024)

This paper is about how inserting fake dialogue can cause LLMs to lose their refusal properties. Although not entirely relevant to decomposition attacks, I think the fundamental problem of how to pick out relevant benign queries that can combine into a harmful query in extremely long contexts (i.e. over thousands of user queries) is directly relevant to stopping decomposition attacks. The techniques that worked to solve long context attacks have direct relevance towards solving decomposition attacks.

1.6 Question 2.1.6

In my opinion, we should focus on reducing the accessibility and ease of performing decomposition attacks, rather than rule them out entirely. I'm assuming the onus of preventing decomposition attacks should lie on the smarter model and not the smaller, misaligned model, since that's what the papers I read the

subject on were doing. If this is true, a sufficiently determined adversary can space out benign queries over extremely long horizons using completely different accounts, and it can be impossible to trace. However, this amount of effort is comparable to searching the corners of the internet for the very information the adversary is trying to seek. At best, I feel we should be trying to make committing decomposition attacks just as hard or harder than simply researching the internet in sufficient detail.

For me, the most feasible way to make committing decomposition attacks harder would be to employ the following approaches.

1. Detect the suspiciousness of combinations of questions asked by a particular user or a small set of users in a short to medium time frame. This is a very interesting approach since it requires careful thresholding to prevent normal benign questions from being answered, but it also requires stringent security.

    - The conversations of other users cannot be leaked, but at the same time, they must be investigated and analyzed to prevent suspicious use. This is very possible via cryptographic tools like functional encryption and differential privacy, and would be a very interesting research direction, but I understand if this is not the goal that this specific project is trying to accomplish.
    - Ignoring the security concerns, it would be interesting feeding various combinations of benign questions and seeing if in-combination they can yield dangerous results. I can think of clever SFT schemes for data generation to train such models by taking an existing decomposition and scoring subsets of the decomposition based on danger.

2. The mathematical frameworks paper (in 2.1.3) directly modified the smarter model and used randomization to systematically refuse outputs that led to the smaller model learning too much. I think this is a useful metric, and it would be interesting performing RL using this as a metric.

3. Lastly, I think designing better decomposition attacks is also interesting. Fine tuning smaller models to also use jailbreaking techniques as tools, we could see very high success rates of decomposition attacks as more of the benign requests could be satisfied.