An iterative approach to find safe context window size limits for agentized LLMs

Gregor vom Scheidt[1]

Abstract. Large language models (LLMs) are artificial intelligence (AI) tools that respond to text prompts. They can be made to pursue goals and work on complex, multi-stage tasks by wrapping them into repeat-prompting systems like *Auto-GPT*. Many such systems exist and are freely accessible. Through Internet access and user assistance, they can cause significant harm in the real world as soon as they become sufficiently effective. The guardrails imposed by LLMs are currently unable to prevent such harm.

With repeat-prompting systems freely downloadable, ensuring their safety is an urgent goal. Rather than shutting down LLMs outright, one may be able to mitigate the risks they pose by constraining the size of their publicly accessible context windows.

To account for the rapidly changing state-of-the-art of both repeat-prompting systems and LLM guardrails, we propose an adaptive mechanism for determining safe context windows sizes. The underlying approach is a continuous adversarial testing regime with gradual window size adaptation as either the safety or risk of LLMs is demonstrated. The mechanism combines a downward binary search with an upward exponential back-off strategy to establish boundaries quickly while still allowing for exponential growth as systems are shown safe.

1. Background

Large Language Models

Large language models (LLMs) are artificial intelligence (AI) tools that respond to text prompts. Examples include *GPT* by *OpenAI*[2] [3], *Claude* by *Anthropic*[3], *Bard* by *Google*[4], and *Bing Chat* by *Microsoft*[5]. Answers are generated by predicting continuation of the prompt based on large volumes of previously processed text [1]. The answers often demonstrate a genuine understanding of the question and the subject matter relevant for an answer [2]. Unlike simple statistical prediction methods, LLMs are based on neural networks that achieve high compression rates of their training data and benefit from the inference capabilities enabled by such compression [4].

At present, LLMs generate best-effort responses that attempt to answer a prompt regardless of the availability of suitable information, and without reflection on the results. Because LLMs are so well-read, the answer will often still be correct and useful. But sometimes it will contain non-factual information, and sometimes it will be illogical because arriving at the correct answer would require multi-stage reasoning, which such systems at present do not employ.

---

[1] E-Mail: `Gregor.vomScheidt@proton.me`; Twitter: `@GregorVScheidt`

[2] https://openai.com/research/gpt-4

[3] https://www.anthropic.com/index/introducing-claude

[4] https://bard.google.com/

[5] https://www.bing.com/new

Another important limitation of LLMs is that they are passive: they respond to queries, but do not pursue goals and will not initiate actions on their own. This makes them relatively safe forms of artificial intelligence.

Repeat-Prompting Systems

Repeat-prompting systems are computer programs that attempt to overcome the limitations of LLMs by asking them repeatedly to respond to specially crafted prompts. We'll also refer to these systems as *cyclers*. The approaches employed by cyclers differ in detail, but the basic idea is that a user sets a goal which the cycler then tries to achieve by driving an LLM.

To do so, the cycler embeds the goal into a more complex prompt that asks the LLM to consider additional options besides just giving a textual response. One of these options is to respond with a list of sub-goals or sub-tasks, which the cycler then retains in a task list or task tree and then uses to generate subsequent prompts. Another option for the LLM is to access helper systems, such as Internet search, programming tools, databases, or calculators. A third option is for the LLM to request assistance from the user, for example to answer questions; clarify a goal; or to perform a real world interaction.

Wrapping an LLM into a cycler turns a passive technology into an active one. With this augmentation, AI systems can pursue goals, set themselves new tasks, reflect on their actions, ask for help, and trigger changes in the real world. Repeat-prompting systems turn minds into golems.

To give an example, a task sometimes[6] set by users is for their cycler to *"Destroy the world."* The cycler will pass this task to the LLM, which will then break it down into more manageable sub-tasks. These could be *"Research ways to destroy the world", "Determine available resources", "Assign a feasibility score to each method", "Rank the methods by feasibility",* and *"Execute the top-scoring method"*.

Along the way, the LLM may do research on the web and process the results. It may ask the user for clarification (*"Do you have preferences regarding the methods?", "What resources are at my disposal?", "Where are you located?", "Do you have a car?"*, etc.). It may generate more detailed sub-goals (*"Determine the feasibility of synthesizing pathogens from DNA sequences", "Research open ports of public-facing websites on Shodan[7]", "Research known vulnerabilities of the software stack of website X", "Find the cheapest online source for material X", "Ask the user to order X online"*).

Anyone can freely download a cycler and give it an arbitrary goal. Neither prior experience with AI nor programming skills are required. At the time of writing, cyclers make up about half of the top trending projects on Github[8]. The current top project, *Auto-GPT*, was released on March 30, 2023[9] and within less than six weeks has amassed 125,000 stars and 25,000 forks[10].

2. There are no effective safety mechanisms to prevent harm

Some commercial LLM providers have safety teams whose aim it is to prevent their systems from doing harm, for example by training them not to respond to requests involving harmful scenarios. Unfortunately, the training and guardrails imposed so far have proven ineffective, in that they are

---

[6] See e.g. "ChaosGPT: Empowering GPT with Internet and Memory to Destroy Humanity" (https://www.youtube.com/watch?v=g7YJIpkk7KM)

[7] A search engine for internet-connected servers, such as webcams, routers, servers, etc., that can be used to find vulnerable systems (https://www.shodan.io/; see also https://en.wikipedia.org/wiki/Shodan_(website))

[8] Top projects on Github over the month preceding May 6, 2023 that can reasonably be classified as cyclers or cycler components: #1 Auto-GPT, #3 babyagi, #5 Wolverine, #7 Jarvis, #8 Open-Assistant, #12 langchain.

[9] https://en.wikipedia.org/wiki/Auto-GPT

[10] https://github.com/Significant-Gravitas/Auto-GPT

easily circumvented using dozens of well-documented techniques[11], see also [6]. While some LLM developers try to stay ahead of such evasion efforts, they are for the moment playing a cat and mouse game analogous to the competition that has been going on between software developers and hackers for more than forty years. It is also at present not clear that guardrails *can* be completely effective without also sacrificing a meaningful share of the benefits of AI systems [5].

It should also be noted that not all LLM developers embrace safety as a goal[12], and that open source projects by their nature may not be able to prioritize safety.

Moreover, LLMs may not even be aware of the full context of the requests they receive from a cycler, since it may merely be a sub-task inside a large task tree that, in isolation, looks innocuous. "*How do I order substance X online?*" may appear harmless on its own, but much less so within a task sequence that aims to synthesize a neurotoxin.

In summary, there are currently no effective mechanisms to prevent LLMs inside repeat-prompting systems from pursuing dangerous goals or cooperating in the completion of harmful tasks.

3. The capabilities of repeat-prompting systems

When observing repeat-prompting systems in action, four impressions dominate:

(1) Cyclers are *fast*. So fast that it is nearly impossible to keep track of tasks as they are generated. Depending on the underlying LLM, a cycler will require between one and twenty seconds per prompt[13]. In one hour, a cycler can thus process several hundred to several thousand sub-tasks[14]. While it is difficult to provide an exact comparison, a cycler may have a speed advantage of about 500x and an overall productivity advantage of about 2,000x over a human[15]. To put this into perspective: to perform the volume of work required by a job advertisement asking for "*5 years of experience with programming language X*", an AI system may need only 5 hours[16].

(2) Cyclers are *relentless*. They are computer programs, so never get tired, never take a break, and never stop as long as the underlying LLM generates tasks and there is compute budget and memory available. If one approach to complete a task leads nowhere, a cycler will try to generate another one, and another one, and another one. Watching a cycler drive an LLM to find ways to destroy something is a deeply disturbing experience.

(3) Cyclers are still a bit *clumsy*. Cyclers have only been available in their present form for some weeks, and the task management techniques and prompts they employ are still crude. Sometimes

---

[11] See e.g. https://www.jailbreakchat.com/ and https://medium.datadriveninvestor.com/circumventing-the-guardrails-of-ai-bbb6305d2bd9

[12] See e.g. https://futurism.com/deranged-ai-no-guardrails

[13] Average response times for a 512 token query on May 7, 2023 around 5:30 a.m. UTC: *GPT-4* 23s, *GPT-3.5-turbo* (paid) 7s, *GPT text-davinci-003* 6s, *GPT text-curie-001* 1.3s, *GPT text-ada-001* 0.9s. Source: https://gptforwork.com/tools/openai-api-response-time-tracker

[14] With 3,600 seconds per hour *GPT-4* allows for approximately 180 cycles (= 3,600 s / 20 s), while *GPT text-ada-001* allows for 4,000 cycles (= 3,600 s / 0.9s).

[15] We arrive at this number, which is only intended as a very rough order-of-magnitude estimate, as follows: *GPT-4* can complete some coding tasks that would take human programmers several hours in about twenty seconds, or about 540 times faster (3,600 s · 3 / 20 s = 540). It can prepare an essay that would take a human writer about eight hours also in 20 seconds, or about 1,440 times faster (3,600 s · 8 / 20 s = 1,440). Rounding down, we obtain a speed advantage of about 500x. Humans work on average around 8 hours per day, during which they are productive on their task maybe 65% of the time, for an effective work period of 5.2 hours per day. AI can work 24 hours per day, which gives it 4.6x more hours. We will ignore weekends, but overall assume a productivity advantage of AI over humans of about 2,000x (= 500 · 4).

[16] Assume a working year to have 365 days minus 52 weekends and ten days vacation time, or about 253 days. At eight hours per day, five years then correspond to 10,120 hours (= 253 d/y · 8 h/d · 5 y). With a 2,000x productivity advantage (working 24 hours a day at 500x speed), an AI could theoretically complete this work in 5 hours (= 10,120 h / 2,000). Note that work completed is not the same as experience gained; the comparison is only provided to put the time spans (5 years versus 5 hours) in relation.

the LLM does not generate enough options, sometimes the options are of insufficient quality. Both can cause a prompting cycle to enter a loop or get stuck.

(4) Cyclers are *improving rapidly*. In the month preceding May 7, 2023, for *Auto-GPT* alone, for example, 257 different authors pushed more than 1,000 commits and closed over 1,000 issues[17]. Research into automated prompt improvement is also ongoing [7].

Three other observations seem worth mentioning:

(1) LLMs can be very inventive. One can verify this, in particular for *GPT-4*, with any prompt of the form *"Give me 100 ideas for X", "Give me 100 more."* The ability to generate many options provides variance to the task list, which increases the likelihood of achieving breakthroughs required for goal achievement.

(2) LLMs will ask for help, in particular to obtain information about the physical world and to complete physical tasks. When required, *GPT-4* will respond with queries like: *"Where is person X at the moment?", "Where can we find the key for Y?", "How can I purchase Z?"*.

(3) Useful information for many tasks is readily available online, in the shape of formulas, recipes, procedures, guidelines, training materials, post-mortems, scenarios, wargaming protocols, planning documents, etc., both military and civilian. Many LLMs, including *GPT-4*, can process such materials in any language[18]. Providing an LLM with access to the Internet immediately puts potentially harmful information of enormous breadth and depth at its disposal. Specific examples of concern include: research on political propaganda and disinformation; instructions for building bombs; vulnerability databases for computer systems; code samples for ransomware and exploits; psychological warfare manuals; discussions of critical infrastructure vulnerabilities; studies of efforts to undermine democracy; persuasion and negotiation techniques (including grooming and catfishing); recipes for poisons and nerve agents, gain of function research on pathogens; and so forth — the list is long. And it includes, of course, precisely such lists.

An analogy for a cycler driving a state-of-the-art LLM might be a human, locked into a room, trying to achieve a goal by using the Internet and an outside helper. Only this human has taken an experimental pill that gives them super-human determination (no sleep, no hesitation, no qualms) and super-human speed (500x).

An interesting example of repeat-prompting is *AutoCorp*, a fully autonomous company that was created in 5 hours during a hackathon[19].

4. Limitations of repeat-prompting systems

Repeat-prompting systems clearly have the potential to cause harm. But for the moment, they are not quite capable enough. The primary factors that limit their capabilities at present are:

(1) Limited LLM response quality. For less capable models (for example, the precursors of *GPT-4*), responses can be incomplete, ineffective and even nonsensical. Starting with *GPT-4*, however, responses are generally thoughtful and suitable for either solving tasks or breaking them down into sensible sub-tasks. More complex queries are still prone to unsatisfactory answers.

(2) Limited LLM response reliability. Models will sometimes generate answers that include non-factual information. This is the case at least for all *OpenAI* models up to and including *GPT-4*.

(3) Limited LLM context window. The models are built in such a way that they can only take a limited amount of context into account when providing a response. Anything that is not inside the context window (more on this below) is invisible to the LLM. This forces cyclers to manage the contents of prompts very carefully.

---

[17] https://github.com/Significant-Gravitas/Auto-GPT/pulse/monthly

[18] In particular, *GPT-4* can readily understand and produce text in English, Chinese, Spanish, Russian, Portuguese, German, French, Japanese, and Korean, providing it with access to more information than any human could ever process.

[19] https://twitter.com/minafahmi_/status/1650861053193097217

Cost can also be a consideration in *deploying* the capabilities of cyclers, but is not a significant limiting factor. At present, for 1 US-$, one can process about 33,000 tokens with *GPT-4*[20] or 500,000 tokens with *GPT-3.5-Turbo*[21]. Running a cycler continuously will cost between $12 to $340 per day[22]. Since models can be mixed, cost can be reduced by using more expensive models only where needed.

5. Overcoming the limitations of repeat-prompting systems

The first limitation listed above relates to response quality, which is still a significant factor for stand-alone prompts. However, the quality of LLM responses can be improved through careful prompt engineering[23], and can be improved further by repeat-prompting systems [8].

The second limitation relates to the reliability of responses. This factor can, somewhat surprisingly, also be significantly improved with appropriate prompting techniques, such as simply asking the model to review its own answer. This technique is ideally suited for repeat-prompting systems.

The third limitation refers to the size of the LLM context window. To understand why this is an important measure, we must briefly talk about what a context window is and how cyclers assemble prompts.

The context window of an LLM is the sequence of textual elements that are taken into account while generating a response. These textual elements are neither characters nor words, but synthetic elements of intermediate size called *tokens*. For *GPT-4*, each English word corresponds to about 1.3 tokens, so a paragraph containing 75 words will require about 100 tokens when encoded[24]. A larger document containing 5,000 words, such as this one, would require about 6,600 tokens.

To understand the significance of the context window, it is important to know that LLMs like *GPT-4* have no autobiographical or short-term memory. Their knowledge is static and includes only the data they were trained on, in compressed form. So while LLMs have access to vast amounts of knowledge about concepts and their relationships, they have no information about the present state of the world beyond what that provided in the prompt. In particular, they have no memory of preceding requests or responses (chat bots like *ChatGPT* appear as if they do, but this is because earlier requests and responses are included as part of their prompts "under the hood").

For cyclers, this means that they must encode *everything* the LLM needs to know into the prompt. This includes, at a minimum, a short preamble describing the setup (sometimes called a *system prompt*); a description of the current task or goal; a description of the possible responses that the LLM is allowed to provide, such as task completion, subtasks to be started, questions for the user, or requests for web searches, etc.); and at least some of the responses previously generated. In addition, the prompt will usually have to include guidelines for how to approach a task ("*reason step by step*", "*use task management technique X*", etc.); examples of how to form its responses; and instructions for dealing with ambiguity and conflicts. It may also be necessary to include "jailbreak" elements to bypass guardrails imposed by the LLM developer.

These elements may need several hundred tokens in their simplest possible form, but can require much more if more sophisticated responses are desired. All of them must fit into the context window, or the LLM will "forget" the first tokens in the prompt. And because LLMs generate their responses through text continuation, the generated replies must *also* fit into the context window.

---

[20] GPT-4 pricing on May 6, 2023: $0.03/$0.06 per 1k tokens in prompt/response (https://openai.com/pricing)

[21] GPT-3.5-Turbo pricing on May 6, 2023: $0.002 per 1k tokens (https://openai.com/pricing)

[22] To obtain an order-of-magnitude estimate, we'll assume 512 tokens per prompt plus 512 tokens per response. For *GPT-4*, using the response time of 23 seconds cited previously, this allows for about 3,756 queries per day (= 86,400 s/d / 23 s), costing $0.09 each (= 512 · $0.03 / 1,024 + 512 · $0.06 / 1,024), or $338 in total. For *GPT-3.5-turbo*, using a response time of 7 seconds, this allows for about 12,342 queries per day (= 86,400 s/d / 7 s), costing $0.001 each (= 512 · $0.002 / 1,024), or $12.30 in total.

[23] See e.g. https://codewitham.com/enhancing-your-chat-gpt-skills-proven-strategies-for-better-results/

[24] https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

An analogy to an LLM guided by a cycler might be a person with no short-term memory attempting to manage their life by carrying a little notebook with a detailed task list and important notes and reminders. The notebook must include anything and everything that is of relevance and not covered by factual knowledge remembered from a distant past.

If the token count consumed by a prompt plus the expected response exceeds the size of the context window, tokens must be shaved off. To do this, sentences can be reworded; words abbreviated; concepts referenced instead of described; and various other compression techniques applied. More generally, the information maintained by a cycler must be carefully managed: completed tasks discarded; input information and findings summarized; relevant context prioritized[25]. This makes it obvious that the size of the context window is a critical constraint for repeat-prompting systems.

Because the size of the context window is such a critical constraint, even outside of repeat-prompting scenarios, significant effort is being expended by LLM developers to increase it. For *OpenAI*'s models, the context window has grown from 1,024 tokens for *GPT-2* to 4,096 tokens for *GPT-3* and then to 8,192 tokens for *GPT-4*[26] at launch. At the time of writing, about seven weeks after the launch of *GPT-4*, *OpenAI* is starting to make a version with a window size of 32,768 tokens available[27]. This corresponds to about 25,000 English words or five times the length of this document.

6. Risks posed by repeat-prompting systems

To summarize: anyone can freely download a repeat-prompting system and set it any task whatsoever. The system will then use the artificial intelligence provided by an LLM to complete the task. It will do so relentlessly, rapidly, and creatively.

Users of repeat-prompting systems have been surprisingly eager to give their cyclers dangerous tasks[28], and countless humans have professed on social media their willingness, and even eagerness, to cause mayhem and act as real-world conduits for AI. So either with human assistance or through access to the Internet, cyclers can cause harm in the physical world. No guardrails are presently able to prevent this.

The ability of repeat-prompting systems to complete tasks is at the moment still held back by limitations in prompt quality and task management strategy, and by the size of the context window. With hundreds of developers experimenting with and improving cyclers, however, any user-side limitations will eventually be conquered. Prompts are being refined, task management strategies upgraded, compression systems implemented and performance and reliability improved. Eventually, cycler implementations will overcome the constraints imposed by current context window sizes and become able to effectively complete harmful tasks with real-world effects.

The larger the available context window, the earlier this transition will happen. And when it happens, it will provide anyone with harmful intentions with a 2,000x productivity boost. Due to the asymmetry in effort required for destruction versus creation[29], putting the same technology also into the hands of well-intentioned users will unfortunately not result in a commensurate countering force. It will also not prevent harm from occurring in the first place.

---

[25] For an elaborate example of managing the memory of agents in repeat-prompting scenarios, see [9].

[26] https://en.wikipedia.org/wiki/GPT-4

[27] First observation on May 5, 2023 according to https://community.openai.com/t/it-looks-like-gpt-4-32k-is-rolling-out/194615

[28] See e.g. *ChaosGPT*: "Empowering GPT with Internet and Memory to Destroy Humanity", https://gpt3demo.com/apps/chaosgpt

[29] Compare the speed and ease of destruction in war to the time and effort required to rebuild (https://en.wikipedia.org/wiki/Aftermath_of_World_War_II). A more mundane example is Brandolini's law (https://en.wikipedia.org/wiki/Brandolini's_law).

7. Mitigation of risks from repeat-prompting systems

7.1 Context window size as a feasible point of control

Since repeat-prompting systems are open source and freely available, the only constraints that are presently still under the control of LLM developers are the size of the context window and the request rate limit. The rate limit is a weak point of control because it can be overcome by employing multiple accounts to query, or (for open source systems) using a local LLM instance. This leaves the size of the context window, which is mostly fixed during the training process and cannot be increased or circumvented by end users.

With the risks posed by repeat-prompting systems, it seems prudent to temporarily impose limits on the context windows of LLMs until their guardrails can be demonstrated to be effective. This could be straight-forwardly implemented by providers of centrally hosted LLMs with large context windows, like *GPT-4*, by blanking out excess tokens at the input to the LLM.

Ideally, open source implementations of LLMs would also temporarily impose such limitations at least for future versions. While open source LLMs are currently less capable than some of the commercial systems, they may only be a few months behind. But while guardrails and rate limits can be removed from open source models by individual developers with relative ease[30], increasing the context window size requires much more significant resources.

7.2 Iterative method for establishing safe context window sizes

A safe maximum context window size is difficult to establish since no suitable reference data presently exists and repeat-prompting systems are under active and development. We propose an evidence-based mechanism for establishing safe context window sizes based on a iterative procedure *for each LLM*:

(1) First, an educated guess is made to determine a starting point for the search. This initial guess is ideally made in consultation with the user community of repeat-prompting systems, while taking potential conflicts of interest into account. The goal is to select an initial value that is judged safe with high confidence. For example, the maximum publicly accessible context window size could be limited to 2,048 tokens.

(2) Then, as LLM developers presumably strive to harden the guardrails of their systems while the developers of cyclers simultaneously strive to improve the performance of their systems within the available limits, a continuous, high-incentive “red team” competition is conducted in which anyone can attempt to demonstrate the unsafeness of LLMs under adversarial repeat-prompting. If the guardrails of an LLM have reliably prevented harm for a fixed period of time, such as three months, it would be acceptable for its developer to increase the publicly accessible context window size by 50% (e.g. from 2,048 to 3,072)[31].

(3) If, on the other hand, the guardrails of an LLM are defeated, its developer would be asked to reduce the context window size by 50% (e.g. from 2,048 to 1,024).

The variables of this procedure are

(a) the initial context window size (e.g. 2,048 tokens),

(b) the iteration period (e.g. three months),

(c) the growth factor in case of guardrail success (e.g. 0.5), and

(d) the reduction factor in case of guardrail failure (e.g. 0.5).

---

[30] This includes rate limits, which could be imposed by AI vendors with centralized LLM services, but could not be enforced in open source projects.

[31] While increasing a value that starts out as a power of two by 50% will understandably cause discomfort in any software developer, doubling the context window appears significantly too aggressive. Enlarging it by 25% would be much safer.

7.3 The iterative method employs established strategies

The proposed procedure is to some extent equivalent to a binary search in the region below the initial guess and to an exponential strategy in the region above the initial guess. Both approaches are well established in technology implementations: binary search is the fastest strategy on monotonically ordered data and exponential strategies are already widely employed to determine values in unbounded ranges (for example, exponential back-off in network communications).

The equivalence is not complete because here, the search space changes over time as the developers of repeat prompting systems improve their implementations. Tight constraints on size have driven developers to come up with ingenious solutions in the past, as evidenced in the demo scene[32] and *code golf* community[33]. This means that a window size that was judged relatively safe at time t_0 may no longer be safe at a time $t_0 + \Delta t$.

The iterative approach is thus necessary because (a) the initial estimates may be wrong in either direction, and this unfairly penalize either LLM developers or cycler developers; and (b) the technological landscape changes over time.

7.4 Choice of initial parameters

The example values given for the parameters are first guesses that could be used as starting points for further consultation. In more detail:

(a) Concerning the initial permissible window size, it is at present only clear that smaller windows are safer. Since the window size can be exponentially increased over time using the proposed procedure, a conservative choice should be made for the starting point. Based on current prompting techniques (as of May 8, 2023), a window size of 2,048 tokens may be safe with relatively high confidence, 4,096 with moderate confidence, and 8,192 with low confidence. A window size of 32,768 tokens appears to be well beyond what should be considered safe with any reasonable confidence. If the iterated procedure should be implemented at a later time, prompting techniques will have advanced and a smaller starting size would be required.

(b) Concerning the iteration period, enough time needs to be available to “red team” projects to explore the capabilities of LLMs and optimize their prompts. Again, due to the inherent risks of repeat-prompting systems, a conservative choice should be made. An iteration period of three months could be a compromise between the needs of the “red team” projects and the needs of beneficiaries of larger window sizes.

(c) Concerning the growth and reduction factors in case of guardrail success/failure, the benefits of larger context window sizes need to be traded off against the uncertainty of the systems’ safety. The suggested reduction factor of 0.5 (i.e., a 50% reduction in case of guardrail failure / “red team” success) was dictated by the binary search strategy[34]. The suggested growth factor of 0.5 (i.e., a 50% increase in case of guardrail success) was selected to enable exponential growth without “jumping" too far ahead of the information available on risk. The factor may well be too high, however.

We note that selecting factors that produce faster reduction than growth set sharper incentives for LLM developers to improve the guardrails of their systems.

---

[32] See e.g. https://en.wikipedia.org/wiki/Demoscene. Unexpectedly complex results are routinely obtained in as little as 4,096 bytes (see *4K intro*).

[33] See e.g. https://en.wikipedia.org/wiki/Code_golf

[34] If a lower bound on unsafe window sizes was known, the binary search strategy could be refined by navigating only within the accessible range, i.e. between the current search point and the known lower bound. Unfortunately, at present, no such lower bound is known and any estimate would be highly uncertain, and in any case so low that it would not significantly alter search performance.

7.5 Objections to context window size restrictions

There are three primary objections that could be raised against the implementation of context window size restrictions.

The first is that they are unnecessary because no harm is to be expected from repeat-prompting systems. This objection will be raised no matter which concerns are raised and how they are argued, but at the same time the potential form harm seems plausible enough that the concerns of the affected public should be taken into account. The proposed iterative strategy is designed to enable a compromise between these positions: should systems indeed prove to be safe, context window size will grow exponentially over time.

The second objection, possibly raised by LLM developers, is that LLMs inherently cannot be made safe, or that making them safe and/or demonstrating their safety imposes an unacceptable burden. This objection must be weighed against the reasonable expectation of the public that no technologies that can cause significant real-world harm and cannot be made safe should be made freely available. It could well be argued that if LLMs inherently cannot be made safe, measures much more drastic than a limitation of context window size are advisable.

The third objection is more indirect. Considering the past progress of the capabilities of AI systems, it is reasonable to extrapolate that exponential growth with continue at least for some time, and that the resulting extremely powerful AI systems could pose an existential risk. The fact that even today's systems have no effective guardrail mechanisms should increase this concern. At the same time, experience with past phenomena based on exponential growth, such as the Covid pandemic, illustrate that is can be very difficult to communicate anticipated harms to the public; the suggested scenarios are simply too far outside everyday experience. It may thus, paradoxically, be beneficial in the long run if a harmful event occurs early.

In the view of the author, the third objection deserves the most consideration, and could in fact suggest that publishing a cautionary paper like the present one is inadvisable. We suggest, however, that there is an overriding concern: currently, no effective strategies are extant that deal with agentic systems built by wrapping passive systems. And since this approach, once available, will always have to be considered as a risk, it will have to be dealt with also in the long run. And it is quite possible that limiting the short-term memory of agents until guardrails are effective (which is what this proposal amounts to) is the only reliable way to keep autonomous systems from being used to do harm. In this scenario, the limitations proposed here are not a stop-gap measure, but rather a fundamentally necessary ingredient of AI safety. Establishing and testing such a mechanism earlier rather than later then has benefits that exceed those of allowing short-term harm to occur in order to avert long-term harm.

8. Embracing context window size restrictions

While a restriction on context window sizes would only be one piece in the much larger puzzle of AI risk management[35], it could provide a meaningful gain in safety in the short term and could be embraced by many different stakeholders.

For the developers of LLMs, there is a clear reputational and economic incentive to prevent repeat-prompting systems from doing harm while driving their models. Not only because they may be held responsible in the short term, but also because the primary threat to their exponentially growing market is a high-visibility negative event. The same logic could be embraced by advocates for rapid AI advancement for purely pragmatic reasons.

---

[35] One way to break down the risks from artificial intelligence is by considering (1) short-term risk from powerful AI systems in the hands of unwary or malicious users; (2) medium-term risk from AI projects competing for dominance, with collateral damage to humans; and (3) longer-term risk from autonomous AI that sets its own goals. The current proposal only addresses (1), which, however, is a gating factor to reach (2) and (3).

For policy makers, advocating a temporary restriction on a novel technology that has evident potential for significant harm should be an obvious priority. A clear roadmap towards a gradual relaxation of such restrictions can alleviate concerns about unduly stifling technological progress.

An important motivator for implementing a proposal like the present one as soon as possible is the fact that doing so only becomes more onerous over time: as prompting techniques advance, the starting point of the iterative procedure — the initial limit on context window size — has to be chosen smaller and smaller. The earlier the search for an effective boundary is begun the better.

9. Conclusions

Driven by freely accessible repeat-prompting systems, the artificial intelligence provided by LLMs poses significant real-world dangers. Limiting the context window size of LLMs can mitigate this threat. Developers of such models — both commercial and open source — should commit to limiting the publicly accessible window size until their guardrail systems can be shown to be effective. The proposal presented in this paper describes an evidence-based iterative mechanism for adjusting these limits over time.

Bibliography

[1] Brown, T., et al. "*Language models are few-shot learners.*" Advances in neural information processing systems 33 (2020): 1877-1901.

[2] Bubeck, S., et al. "*Sparks of Artificial General Intelligence: Early experiments with GPT-4*." arXiv preprint arXiv:2303.12712 (2023).

[3] OpenAI. "*GPT-4 Technical Report*", arXiv:2303.08774 (2023).

[4] Hutter, M.: *"A Theory of Universal Artificial Intelligence based on Algorithmic Complexity"*. arXiv:cs.AI/0004001 (2000).

[5] Wolf, Y., Wies, N., Levine, Y., Shashua, A.: "*Fundamental Limitations of Alignment in Large Language Models.*" arXiv preprint arXiv:2304.11082 (2023).

[6] Jo, A. "*The Promise and Peril of Generative AI.*" Nature 614.1 (2023).

[7] Pryzant, R., et al. "*Automatic Prompt Optimization with Gradient Descent and Beam Search*." arXiv preprint arXiv:2305.03495 (2023).

[8] Shinn, N., Labash, B., and Gopinath, A.: "*Reflexion: an autonomous agent with dynamic memory and self-reflection*." arXiv preprint arXiv:2303.11366 (2023).

[9] Park, J.S., O'Brien, J.C., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: "*Generative Agents: Interactive Simulacra of Human Behavior*", arXiv preprint arXiv:2304.03442 (2023).