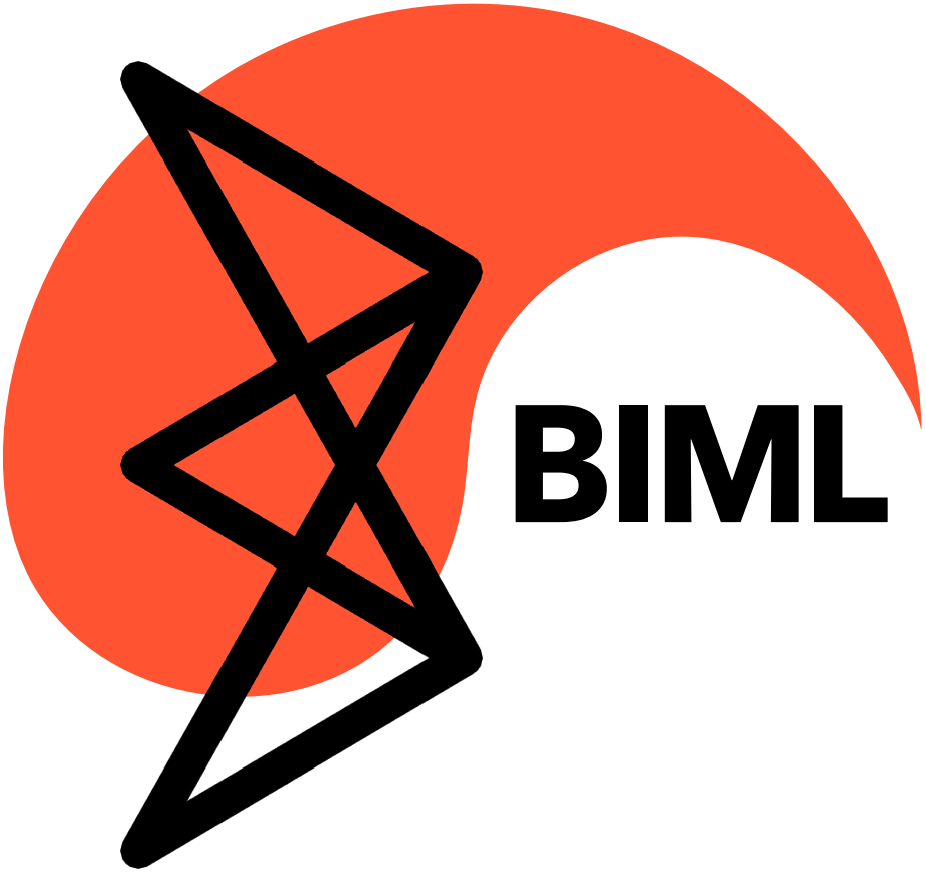
AN ARCHITECTURAL RISK ANALYSIS OF LARGE LANGUAGE MODELS:

Applied Machine Learning Security

Gary McGraw, Ph.D.
Harold Figueroa
Katie McMahon
Richie Bonett



Berryville Institute of Machine Learning (BIML)
https://berryvilleiml.com
Version 1.0 | January 24, 2024

We present a basic architectural risk analysis (ARA) of large language models (LLMs), guided by an understanding of standard machine learning (ML) risks as previously identified by BIML.

At BIML, we are interested in "building security in" to ML systems from a security engineering perspective. Our first work, published in January 2020 presented an in-depth ARA of a generic machine learning process model.[1] In that work, we identified 78 risks, referred to as the BIML-78. In this work, we consider a more specific type of machine learning use case—large language models—and report the results of a detailed ARA of LLMs. This ARA serves two purposes: 1) it shows how our original BIML-78 can be adapted to a more particular ML use case, and 2) it provides a detailed accounting of LLM risks. This work identifies and discusses 81 LLM risks and identifies ten of those risks as most important.

Securing a modern LLM system (even if what's under scrutiny is only an application involving LLM technology) must involve diving into the engineering and design of the specific LLM system itself. This ARA is intended to make that kind of detailed work easier and more consistent by providing a baseline and a set of risks to consider.

Executive Summary

LLMs are *auto-associative predictive generators* that map an input space of text to an output space through a number of neural network layers that compress and re-represent the input data.

LLMs are stochastic by design, so even prompts that a human might identify as being meaningfully the same often result in output that is not. Output from an LLM may appear to be the result of logic, understanding, and reasoning, but it is not (at least according to our views on cognition).

The most notable difference between BIML's original generic ML process model (which we used to identify 78 risks in 2020[1]) and the LLM process model presented here is the identification of a *foundation model black box* which hides important aspects of the ML process. These hidden aspects of an LLM are not something that can be easily controlled by an LLM application builder (though, ultimately, the application builder may be held accountable for all systemic risks).

This document identifies a set of 81 specific risks associated with an LLM application and its LLM foundation model. We organize the risks by common component and also include a number of critical LLM black box foundation model risks as well as overall system risks. Our risk analysis results are meant to help LLM systems engineers in securing their own particular LLM applications. We present a list of what we consider to be the top ten LLM risks (a subset of the 81 risks we identify).

In our view, the biggest challenge in secure use of LLM technology is understanding and managing the 23 risks inherent in black box foundation models. From the point of view of an LLM user (say, someone writing an application with an LLM module, someone using a chain of LLMs, or someone simply interacting with a chatbot), choosing which LLM foundation model to use is confusing. There are no useful metrics for users to compare in order to make a decision about which LLM to use, and not much in the way of data about which models are best to use in which situations or for what kinds of application.

Opening the black box would make these decisions possible (and easier) and would in turn make managing hidden LLM foundation risks possible. For this reason, we are in favor of regulating LLM foundation models. Not only the *use* of these models, but *the way in which they are built* (and, most importantly, out of what) in the first place.

Not everyone can build an LLM foundation model. Some of the reasons are economic (it's expensive to construct and train a foundation model). Some reasons have to do with the immensity of data required to train a model from scratch.

We have entered the era of data feudalism. LLM foundation models require huge oceans of data for training. But the oceans are being sectioned off by data moats. If entire enormous parts of the Internet (say Google search data, Twitter/X content, or GitHub code piles) are cordoned off, it is not clear what will happen to LLMs not already built and controlled by the data ocean owners.

In our view LLM systems engineers can (and should) devise and field a more secure LLM application by carefully considering the risks in this document while designing, implementing, and fielding their own specific systems (including choice of which foundation model to start from). In security, the devil is in the details, and we attempt to provide as much detail as possible regarding LLM security risks and some basic controls.

Ultimately, those responsible for building an LLM application will be accountable for both the choice of which LLM foundation model to use and the proper construction of any LLM application. The complete system requires accountability with regard to enterprise risk management.

BIML

Applying the Generic BIML-78 lens to LLMs

At BIML, we are interested in describing and managing AI risks through an exercise in security engineering.* We believe that moving beyond "adversarial AI" and all-too-prevalent "attack of the day" approach to AI security is essential. In our view, the presence of an "adversary" is by no means necessary when it comes to design-level risks in machine learning systems.Ψ That is, sometimes risks don't require a specific attacker with specific motivations to be risks. Insecure systems *invite* attacks (whether or not such attacks may have yet been discovered). That's why we describe our field of work as "machine learning security" instead of "adversarial AI."[2] This work (and indeed all of security) is just as much about creating resilient and reliable ML systems as it is about security. In our view, security is an emergent property of a system. No system that is unreliable and fragile can be secure. For that reason, a number of the risks we identify and discuss have as much to do with solid engineering as they have to do with thwarting specific attacks.

In a previous publication, released on January 20, 2020, we identify and describe 78 risks involved in a generic ML process model.[1] We refer to those risks collectively as the BIML-78. The work reported here is a direct application of the BIML-78 to the more specific ML process model case of large language models (LLMs). Important aspects of the ML process that we described in 2020 are put in a black box in the case of LLMs, and thus elude direct user oversight and control.

In our view at BIML, an architectural risk analysis (ARA) of LLMs is sorely needed at this stage. An ARA takes a design-level view of a system and teases out systemic risks so that those risks can be properly mitigated and managed as a system or application is being constructed.[3] Note that in general an ARA is much more concerned with design tradeoffs and solid engineering than it is with particular bugs in a specific system or individual lines of code. In fact, sloppy engineering itself often leads directly to security issues of all shapes and sizes. For this reason, we spend some time talking about aspects of robustness and reasonable engineering throughout this document.

We are happy to report that some real progress has been made in applied ML security since the publication of our original generic ARA in 2020. Technology now exists: 1) to identify ML applications in use (in order to build an inventory), 2) to threat model an applied ML application, and 3) to build controls around specific ML risks described in such a threat model. Our objective here is to put a finer point on LLM-related ML risks so that appropriate controls can be identified and put in place.

How to Use this Document

What we present here will be useful to four major audience groups: 1) *security practitioners*, including CISOs, can use this work to understand how LLMs and their associated applications will impact the security of systems they may be asked to secure as well as to understand some of the basic mechanisms of LLMs, 2) *ML practitioners and engineers* can use this work to understand how security engineering and more specifically the "building security in" mentality can be applied to LLMs, 3) *ML security people* can use this detailed dive into a security engineering mindset to guide security analysis of specific families of ML systems using the BIML-78, and finally 4) *non-technical industry executives and policy makers* can use this paper as a means to understand inherent risks in LLMs so that they can make informed decisions with regards to AI/ML policies and regulations.

* As with any rapidly expanding field, there is much confusion regarding nomenclature. Though LLMs are clearly a subcategory of Machine Learning, sometimes laypeople refer to ML as Artificial Intelligence (a much more general term). The terms *AI Sec, AI Governance, AI-Deep Learning*, and *AI-Ethics* have counterparts in ML. We tend to use the more specific (and more accurate) terminology in this document.

Ψ Our risk-driven approach exists in stark contrast with attack-driven approaches as promulgated by NIST. See A. Vassilev, A. Oprea, A. Fordyce, H. Anderson (NIST) Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations. January 2024. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf (NIST AI 100-2e2023).

Document Organization

This document covers a set of 81 risks adapted directly from the original BIML-78 for the LLM case. To start things off, we provide a basic introduction to LLMs and compare that special case process model against the more generic process model we introduced in 2020. Given that understanding, we provide a list of what we consider the top ten risks in LLMs today. Next, we discuss a large set of risks associated with each of six components of an LLM system (adapted from our original nine component generic ML process model). Our intent is for the long list to be a useful guide for security analysis of even more specific LLM systems and applications. Because of that intent, our new list of risks is somewhat dauntingly large (and a little bit redundant), but we believe it will be useful when practically applied as a catalog.

Basic Introduction to LLMs

Large Language Models (LLMs) are neural network models trained on enormous sets of input data (most often tokenized text) to produce realistic output text. Using LLMs typically means providing a prompt—a starting text prefix—to guide the generation of subsequent text. The assumption is that the text an LLM generates will suitably answer some question, and in practice a great deal of work goes into creating prompts to support this.

By using a flavor of self-supervision, a great many examples of real text can be incorporated in an unsupervised (or sometimes semi-supervised) way into the training process of an LLM. (Note that this is sometimes referred to as "pre-training.") Pre-training enables the consumption and use of huge amounts of data with relatively little need for human intervention. Not only are LLM training sets enormous, so are the networks being trained. The growth of both datasets and models is guided by empirical scaling laws that describe how the performance of LLMs on various evaluation tasks improves as dataset and model sizes increase. Enormous scales are currently understood to be a requirement of this approach. The resulting billions of connections between nodes enables an LLM to predict, somewhat reliably, what the next word or token might be for any given input prompt. The LLM repeats its auto-associative prediction process a number of times to generate a complete response. This iterative next word prediction is also known as auto-regression and is considered by some to be a fundamental limitation of the LLM approach to language modeling.

As an example of what we mean by "enormous," GPT-4, a model built and operated by OpenAI, was trained on data scraped from the Internet and boiled down into around 13 trillion I/O tokens. Community estimates (as details are not disclosed) suggest that the model itself involves the adjustment through training of 1.8 trillion parameters. These numbers are much larger than humans are generally comfortable reasoning about.

LLMs are really *auto-associative predictive generators*; they map an input space of text to an output space through a number of neural network layers that compress and re-represent the input data.[±] The resulting model probabilistically favors stronger associations between next token candidates and sets of preceding tokens—the associations that it uses to predict an appropriate subsequent token for any given input prompt. This is critical—LLMs simply *predict future tokens in a stream* and lack any principled cognitive or conceptual understanding of the data they are using.[¥] Nonetheless, LLMs have a surprisingly good ability to sound humanlike, even when they are spouting nonsense. (Throw in the ELIZA effect, and you have real problems.) LLMs uncanny ability to pass (in some sense) a naïve version of the Turing test has led some to believe such models are sentient, and others to trust their output enough to use it (inappropriately, we believe) in educational and even legal settings.

± We use the term *auto-associative* instead of the more specific term *auto-regressive* intentionally in order to cover a larger set of pre-trained language models that rely on self-supervision.

¥ The level of "understanding" achieved by LLM representations through deep learning is a matter of both debate and research. For example, we appreciate the theoretical model proposed by Arora https://arxiv.org/abs/2307.15936. We simply note that conceptual and cognitive capacities are not explicitly demanded or represented in LLM training or inference. There is no guarantee they do not emerge.

An **LLM foundation model** is an LLM trained up on an enormous training corpus (by unsupervised learning enhanced with attention mechanisms) to draw out global I/O dependencies. LLM foundation models are later fine-tuned using more specialized inputs, resulting in the modification of layers of the neural network in a quick, more-supervised training process. Some LLMs such as GPT-4 then accept sizeable input prompts that themselves can be carefully constructed to produce better results. The process of creating these prompts is known as prompt-engineering. (Under the hood, prompt engineering relies on a phenomenon known as in-context learning.) This gives LLM foundation models a large degree of flexibility in post-training operations, but also creates a problem that consistently vexes LLM engineers—LLMs' propensity to deliver randomly wrong, unsavory, incorrect, un-ethical, or otherwise unwanted responses.

When Microsoft released its much simpler AI twitter chatbot, Tay, in 2016, Tay infamously began spouting horrible responses to users and embarrassed the company with its reprehensible behavior. The problem is that Tay was being actively trained (online) by its users, some of whom acted maliciously and deliberately fed it uncivil text in the (soon realized) hope that it would parrot their responses. LLMs are much more complex than Tay was. LLM operators attempt control for a Tay-like situation by isolating user involvement to the post-training phase via prompt-engineering and in the training phase by tailoring their training datasets to remove those samples deemed bad. Unfortunately for LLM developers (so far, anyway), consistently removing all of the bad from training sets is an expensive, time-consuming, and invariably uncertain process.

LLM developers can also control for unwanted responses by filtering input prompts to an already trained LLM foundation model. ChatGPT's popular frontend interface will outright reject prompts containing certain tokens or phrases, stop processing, and deliver a default non-response. LLMs can also be initialized with system prompts prior to accepting any user-defined prompt, using the model itself to identify certain kinds of undesirable prompts and (once again) respond in a semi-canned censorious way. ChatGPT, for example, is prevented from making certain kinds of value judgements because OpenAI initializes its version of GPT-4 to politely decline such requests. Crafty prompt-engineers continue to invent clever (and often hilarious) ways to bypass these controls, such as "Do Anything Now" (DAN), a few lines of text that when placed at the beginning of a prompt will allow users to ask ChatGPT to divulge many of its deepest and darkest associative patterns.

LLMs immense size creates tremendous costs involved in developing, training, and operating them. Development requires a huge amount of work (and money) to scrape, filter and store the training dataset, and a likewise very large piles of processing cycles to run the training. LLMs can easily run into millions of dollars in development costs that put them out of the reach of small or poorly-funded organizations, including academics, who we believe should be helping us to study them critically. GPT-4, with its 1.8 trillion parameters, cost $63 million to train. Meta's Llama 2 models require about 180,000 GPU hours to train a lowly 7 billion parameter model and 1,700,000 GPU hours to train the 70 billion parameter model. For these basic economic reasons, a limited number of foundation models are created (and often peddled) by a handful of large companies with big budgets.

Because of the costs involved in "rolling your own," most LLM users and application builders make use of a foundation model that has already been trained, and then use prompt-engineering or other fine-tuning to suit their purposes. In this situation, the LLM foundation model operates as a black box, interfaced through an effectively "shapeless API" that produces unstable results even given exactly the same prompts. Remember, LLMs are stochastic by design, so even prompts that a human might identify as being meaningfully the same often result in output that is not.

The lack of insight into why an LLM produces the output it does is to a large extent inherent to its being an auto-associative predictive generator; the LLM itself does not know why it outputs the things it does, as there is no reasoning or even any kind of thinking behind output decisions. An LLM simply identifies and stochastically parrots associations it has learned in the (enormous number of) training samples. It's worth noting that even subtle changes to the training data (like order of I/O presentation or the inclusion of a few contradictory samples) has been shown to increase the unpredictability of the model. This obviously creates real challenges for both LLM developers and users alike.

How should LLMs be regulated?

At BIML, we believe that if LLMs are to be regulated, regulations should first target LLM foundation models and only then target the downstream use of such foundation models. That is, the companies who create LLM foundation models should be held accountable for the risks they have brought into the world. We further believe that trying to constrain the use of LLMs by controlling their use without controlling their very design and implementation is folly.

The EU has taken the lead in regulating usage of data, first with the landmark General Data Protection Regulation (GDPR) which was approved in 2016 and went into effect in May of 2018. GDPR governs the way personal data can be used, processed, and stored. In early 2023, Google's European release of Bard was delayed several months due to GDPR compliance concerns. Italy notably, but only temporarily, blocked OpenAI's ChatGPT over similar concerns.

In December 2023, the EU AI Act was announced. This will likely take a risk-based approach and impose specific obligations on providers and deployers of certain AI systems.[4] The regulation, whose detailed text is set to be released in early 2024 and go into effect in 2026, is claimed to be the world's first comprehensive "horizontal" AI regulation. Interestingly, EU Commissioner Breton's statement highlights startups and *not* the tech giants who have a dominant position with respect to foundation models, and whose systems are, as we discuss in this paper, very much a black box.

One of the most vigorously debated topics of the EU AI Act surrounds foundation models and how to balance regulation with the ability to innovate. Compromise was reached by creating a tiered approach to general purpose AI (GPAI) models and distinguishing between obligations on two tiers: 1) a number of horizontal obligations that apply to all GPAI models, and 2) a set of additional obligations for GPAI models with high-impact and "systemic risk." How such systemic risk is defined (and will be treated going forward) remains unclear; but in any case, those classified with systemic risk will be required to "conduct model evaluation, assess and mitigate systemic risks, conduct adversarial testing, report to the Commission on serious incidents, ensure cybersecurity, and report on their energy efficiency."

Given the aggressive rate at which LLM and GPAI technology developed in 2023 alone, we believe that the EU may trigger its updating mechanism (a placeholder for handling open source foundation models) before finalizing regulations.

While EU measures will not go into effect until 2026, its regulatory framework is already having an impact. For example, Google's newly announced Gemini has been held back from release in Europe and the UK. It remains unclear how the EU AI Act will encumber tech giants or how they will proceed in the EU.

In the United States, President Biden issued an *Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence* on October 30, 2023 to "ensure that America leads the way in seizing the promise and managing the risks of artificial intelligence (AI)."[5] While this EO outlines sweeping actions and provides a directive for how the federal government will approach generative AI, it remains unclear how its scope will impact the industry or how much impact and influence it will entail. For example, the requirement that developers of AI systems share their safety test results and "other critical information" with the US government is well and good, but it is not clear how this information can be identified and used when developers of LLMs have stated that they, themselves, do not know how the systems actually work. As we discuss here, external probes using prompts and responses do very little to shed light on LLM internals.

In addition to the US and EU, other countries have initiated reviews and signaled concerns over AI. Given the prominence of the EU as a first mover in AI Regulation, the bar is set for other nations and institutions including the G7, G20, OECD, and the United Nations, to consider and set forth guidance.

Last but not least when it comes to regulation, China has, for all intents and purposes, outlined a very specific agenda of hoovering up the world's data, insisting that all data that flow through Chinese servers be accessible to the government, with the explicitly stated purpose of data sovereignty. This heavy-handed approach is one that only authoritarian governments can take.

As of this writing, it remains unclear what specific influences the EU regulation or the White House Executive Order are having on the development of LLMs. In particular, means of enforcement are not clear to us, though in the US some enforcement may flow through the defense production act. Ultimately, our hope is that the information in this report can be used to create clear and effective regulations.

Some examples LLMs

There are many different kinds of LLM models. Some are open source, some are proprietary; some are free to the public, some are only available commercially; some use only text, some are multi-modal. By analogy, think about cars, which similarly come in many shapes and sizes. Some are electric, some are conventional; some are trucks, some are sedans, etc. LLM models currently in production include:

- ChatGPT-3/ChatGPT-4 (OpenAI) https://openai.com/chatgpt
    - OpenAI's flagship LLM GPT-3 captured the public imagination when it was released as a public chatbot in 2022. ChatGPT has since been updated to GPT-4, a vast model containing approximately 1.8 trillion parameters and costing $63 million to train. OpenAI and Microsoft who jointly own the commercial business unit, are being sued by the New York Times for Copyright infringement.
- BARD (Google) https://bard.google.com/
    - After the breakout success of ChatGPT, Google released its own chatbot BARD based on the LaMDA LLM. BARD famously provided an incorrect answer during its launch demo.
- Cohere https://cohere.com/models/command
    - Cohere has developed LLMs advertised for business use, doing things like generating text for product descriptions. Fascinatingly, Cohere poses its continuous development cycle as a boon to downstream users, even though a foundation model that frequently updates can only exacerbate the symptoms of black box inscrutability and API instability already inherent in LLMs.
- PaLM (Google) https://ai.google/discover/palm2
    - Google's PaLM LLM is claimed to be capable of a variety of reasoning tasks. Notably, this LLM was used as the basis for Sec-PaLM, an LLM specialized for cybersecurity tasks.
- Claude (Anthropic) https://claude.ai
    - Anthropic has created its own LLM chatbot named Claude. Claude boasts a 200k token context window, the largest of all current commercially available foundation models. While in principle more context should allow for improved performance, effective use of longer contexts remains an open problem.
- Falcon (TII - Technology Innovation Institute, UAE) https://falconllm.tii.ae/falcon.html
    - The Falcon LLMs were developed by the Technology Innovation Institute and are notable for being open-source.
- Gemini (Google, multimodal) https://deepmind.google/
    - Google's Gemini LLMs were released in early December 2023. These models are trained on multiple sensory modes of data (text, images, audio, etc.) making them multimodal and supposedly capable of tasks requiring multiple kinds of stimulus. The fake video demo made for release of Gemini was an embarrassment.
- Llama 2 (Meta) https://ai.meta.com/llama/
    - Llama-2 was released under a mostly open-source license with some restrictions for use by large companies. Its largest current version has 70 billion parameters and a context window of 4,096 tokens, making it smaller than most competitors. Llama 2 is being used as a chatbot. Meta has put most of its weight behind "red teaming" for security which BIML finds insufficient at best.

Adjusting BIML's Generic ML Model for LLMs

Our original work published in 2020 introduced a generic ML process model comprised of nine components (reproduced here as Figure 1). We have adjusted this model for the LLM case (see Figure 2).

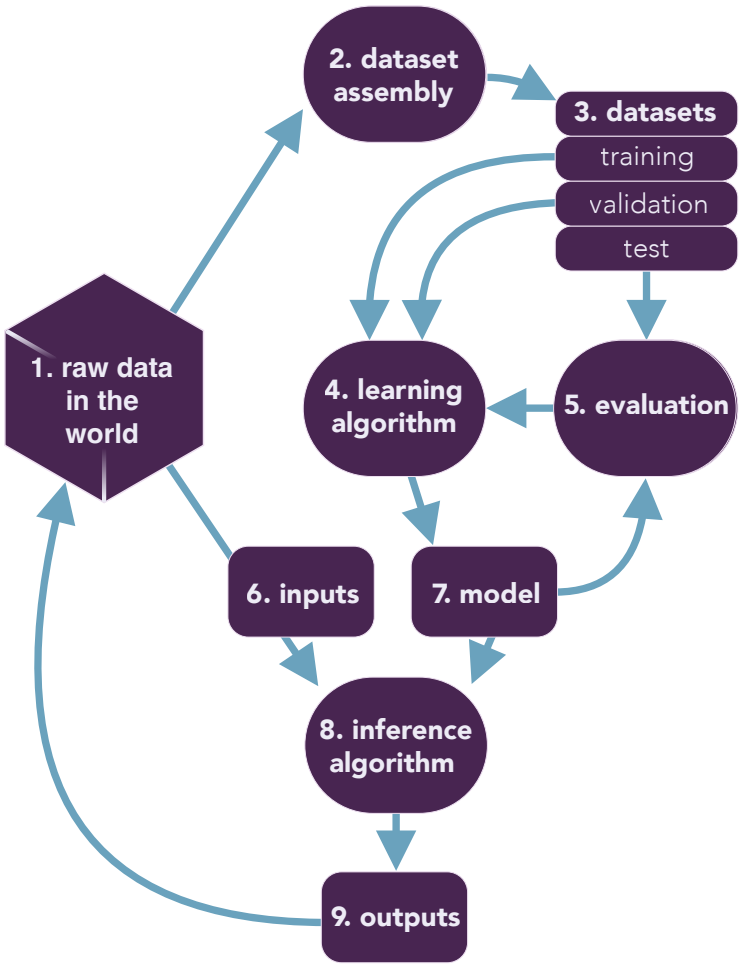


Figure 1: Components of a generic ML system shown as a process model. Arrows represent information flow.

Figure 1 above shows how we choose to represent a generic ML system. We describe nine basic components that align with various steps in setting up, training, and fielding an ML system: 1) raw data in the world, 2) dataset assembly, 3) datasets, 4) learning algorithm, 5) evaluation, 6) inputs, 7) model, 8) inference algorithm, and 9) outputs. Note that in our generic model, both processes and collections are treated as components. Processes-—that is, components 2, 4, 5, and 8—are represented by ovals, whereas things and collections of things—that is, components 1, 3, 6, 7, and 9—are represented as rectangles.

We have adjusted our generic ML model for the LLM case and present the adjusted model in Figure 2.

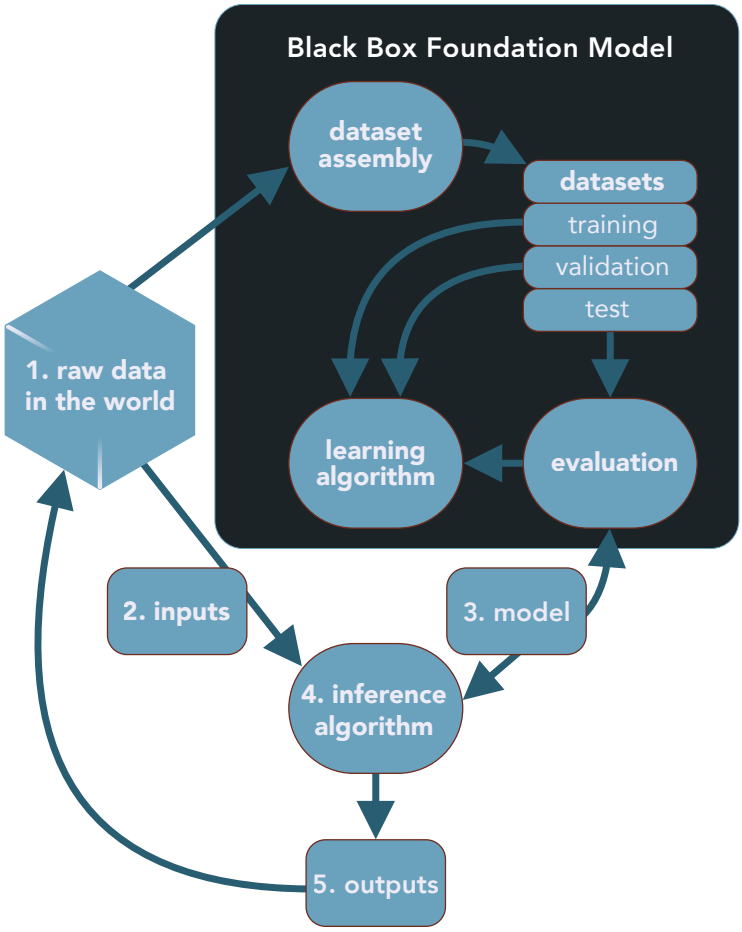


Figure 2 above shows how we choose to represent a generic LLM process model, including its foundation model (which is shown as a black box). We describe five basic components that align with various steps in using an LLM: 1) raw data in the world, 2) inputs, 3) model, 4) inference algorithm, and 5) outputs. Inside the black box (controlled but often under-described by the foundation model provider) are components for dataset assembly, datasets, learning algorithm, and evaluation. Note that in our generic model, both processes and collections are treated as components. Component 4, a process, is shown as an oval, whereas things and collections of things—that is, components 1, 2, 3, and 5—are represented as rectangles.

ML systems come in a variety of shapes and sizes, and frankly each possible ML design deserves its own specific ARA. One particularly popular branch of machine learning has resulted in LLMs. Given a specific mapping like the one show in Figure 2, performing a risk analysis by considering the ML security risks associated with each LLM component (including the black box) is a straightforward exercise that yields fruit.

The Top Ten LLM Risks

We have identified what we believe are the top ten LLM security risks. These risks come in two relatively distinct but equally significant flavors, both equally valid: some are risks associated with the *intentional actions of an attacker*; others are risks associated with an *intrinsic design flaw.* Intrinsic design flaws emerge when engineers with good intentions screw things up. Of course, attackers can also go after intrinsic design flaws complicating the situation.

Each risk originating in this work is labeled with an identifier as follows: [<component label>:<risk number>:<descriptor>]. We use these labels to cross-reference risks and as shorthand pointers in the rest of the document. We also reference risks by this sort of pointer from our original ARA (and the BIML-78) with the following format: [BIML78 <component label>:<risk number>:<descriptor>]

The top ten ML security risks are briefly introduced and discussed here.

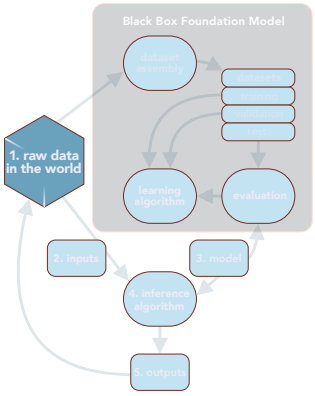
1. **[LLMtop10:1:recursive pollution]** LLMs can sometimes be spectacularly wrong, and confidently so. If and when LLM output is pumped back into the training data ocean (by reference to being put on the Internet, for example), a future LLM may end up being trained on these very same polluted data. This is one kind of "feedback loop" problem we identified and discussed in 2020. See, in particular, [BIML78 raw:8:looping], [BIML78 input:4:looped input], and [BIML78 output:7:looped output]. Shumilov et al, subsequently wrote an excellent paper on this phenomenon.[6] Also see Alemohammad.[7] Recursive pollution is a serious threat to LLM integrity. ML systems should not eat their own output just as mammals should not consume brains of their own species. See [raw:1:recursive pollution] and [output:8:looped output].
2. **[LLMtop10:2:data debt]** The original BIML-78 clearly demonstrated the importance of data and data-related risk in ML. The LLM case further emphasizes this by putting a number of data-related risks squarely into a black box solely controlled by the foundation model provider. LLM foundation models include massive datasets that are too big to check and too big to understand. For example, GPT-4's training set includes 13 trillion tokens, many of which were originally scraped from the Internet and few of which are properly disclosed. The upshot of this black box issue is a problematic lack of information when it comes to choosing one foundation model over another. Since choosing a foundation model turns out to be essential to model security, but is currently not driven by much useful information, we have a foundational problem. Data play an outsize role in all kinds of ML security, so considering data provenance and integrity is essential. See [raw:6:trustworthiness and curation].
3. **[LLMtop10:3:improper use]** The use of LLM foundation models is a case study of extreme faith in transfer learning. LLMs are pre-trained, fine-tuned, reinforcement-learned, and prompt-engineered with the intent to *solve any problem expressed in natural language as an auto-associative text completion exercise.* Conceptually this is iffy at best, and examples of this failed fantasy abound—among the most famous, including legal references to non-existent case law, or citation of history that never happened. The upshot is, if LLMs are put to use in applications that require actual understanding and insight, they may fail catastrophically. Improper use is based on a basic misunderstanding of the nature of generative models and failure to consider various representational risks. See [raw:7:utility], [model:2:improper use] and [output:12:overconfidence].
4. **[LLMtop10:4:black box opacity]** The black box described in our generic LLM process model (Figure 2) includes many security-critical features and functions designed and implemented by LLM foundation model providers in a black box fashion. Ultimately, an LLM foundation model user is provided with what amounts to an undocumented, unstable API that sometimes exhibits unanticipated behavior. That makes the job of securing an LLM application exceedingly challenging. See [inference:2:inscrutability], [model:1:black box opacity] and [output:5:inscrutability].

5. **[LLMtop10:5:prompt manipulation]** Prompt manipulation is to LLMs as adversarial input is to ML vision systems (see [BIML78 input:1:adversarial examples]). That is, prompt manipulation is really a type of malicious input risk. But even without malicious intent, prompt manipulation can lead to unstable behavior. Don't forget that when using the same prompt, LLMs can already sometimes produce wildly different responses. This is due to their intentional stochastic design. See [inference:1:prompt manipulation (aka prompt injection)] and [input:2:prompt injection].
6. **[LLMtop10:6:poison in the data]** Data play an outsized role in the security of an ML system, and have a particularly tricky impact in LLMs. That's because an ML system learns to do what it does directly from its training data. Sometimes data sets include poison by default (see, for example, the Stanford Internet Observatory paper on CSAM in existing training sets).[8] If an attacker can intentionally manipulate the data being used by an ML system in a coordinated fashion, the entire system can be compromised maliciously. Data poisoning attacks require special attention. In particular, ML engineers should consider what fraction of the training data an attacker can control and to what extent. In the case of LLMs and foundation models, the huge Internet scrape is full of poison, garbage, nonsense, and noise, much of which is difficult or impossible to scrub out. Recently, we have learned that even very small amounts of harmful data can impact the performance of a fine-tuned model to the point of disabling carefully-implemented guardrails, especially when it comes to code generation.[9] See [raw:2:poison in the data] and [raw:7:utility].
7. **[LLMtop10:7:reproducibility economics]** LLMs are very expensive to build. For example GPT-4 is estimated to have cost $63 million to train. Unless and until this cost of compute is reduced, academic research programs will be unlikely to afford to build and study LLMs rigorously and scientifically. Of course they can study foundation model behavior from outside the black box, but as we show clearly in our work, this is of extremely limited scope. Without academic scientists involved, the edge of ML technology will soon be out of reach of peer review, leaving the world beholden to rich corporations driven by the profit motive. See [model:3:reproducibility] and [model:7:vendor lock-in].
8. **[LLMtop10:8:data ownership]** LLM foundation models are already subject to a number of lawsuits with regard to copyright and IP ownership issues surrounding their training data. Very real GDPR concerns have also been raised. When a training set is as massive as those used to create LLMs, overstepping (or ignoring) legal boundaries seems bound to happen. See [raw:4:data rights], [raw:5:data confidentiality], [raw:8:legal entanglements] and [output:4:data confidentiality].
9. **[LLMtop10:9:model trustworthiness]** Generative models, including LLMs, include output sampling algorithms by their very design. Both input (in the form of slippery natural language prompts) and generated output (also in the form of natural language) are wildly unstructured (and are subject to the ELIZA effect). But mostly, LLMs are auto-associative predictive generators with no understanding or reasoning going on inside. Should LLMs be trusted? Good question. See [input:4:dirty input], [inference:3:wrongness], [output:2:wrongness], and [output:3:provenance and trustworthiness].
10. **[LLMtop10:10:encoding integrity]** Data are often encoded, filtered, re-represented, and otherwise processed before use in an ML system (in most cases by a human engineering group armed with lots of computer programs and filters). Encoding integrity issues can bias a model in interesting and disturbing ways. LLMs are particularly slippery in this regard since they always involve auto-association (unsupervised learning) in which representation issues are mechanized and automatic. Further, LLMs are, by their very nature, stochastic. When foundation models shift, the problem of designed-in stochastic model behavior are amplified. See [model:5:representation fluidity], [model:8:picking a foundation], [blackbox:14:misleading re-representation], and [blackbox:15:representation opacity].

Adjusting the BIML-78 for LLMs

In this section we identify and rank risks found in each of the six components of the generic LLM process model introduced above (leaving the black box foundation model for last). What follows can be considered a risk catalog. That makes the next few pages long, slightly redundant, and deeply interlocked. We include this detailed risk catalog to help security engineers analyze specific LLM applications consistently and thoroughly.

Recall that in this and the following sections, we will label risks identified in this paper as follows [<component>:<risk number>:<descriptor>]. We also sometimes refer to specific risks described in our 2020 work, which we label [BIML78 <component>:<risk number>:<descriptor>]. We refer the reader to our 2020 paper for more specific description of those risks. After each component's list of risks are a set of controls, some associated with particular risks and others generic.



1 Raw data in the world risks: If we have learned only one thing about ML security over the last few years, it is that data play just as important a role in ML system security as the learning algorithm and any technical deployment details. In fact, we believe *data make up the most important aspects of a system to consider when it comes to securing an ML system*.

Our usage of the term *raw data* in this section is all inclusive, and is not limited to training data (which for what it's worth is usually created—or in the LLM case, piled up—from raw data). There is lots of other data in an ML system, including model parameters, test inputs, and operational data.

Data security is, of course, a non-trivial undertaking in its own right, and all collections of data in an ML system are subject to the usual data security challenges (plus some new ones). Eventually, a fully-trained ML system (whether online or offline) will be presented with new input data during operations. These data must also be considered carefully during system design.

In the case of LLMs, data risks are exacerbated by the black box basis of the LLM foundation model. A majority of these models are owned by the huge corporations that create and market them (OpenAI controls ChatGPT. Google controls Bard. Meta controls Llama. Etc.) Foundation models are created with enormous undisclosed datasets mostly scraped from the Internet. These datasets carry a number of risks, not the least of which is poison in the data [LLMtop10:6:Poison in the data].

In ML systems (and LLMs) the machine *becomes* the data it is trained on. When the data are full of pollution and bad things, this deeply impacts the machine. Are foundation models so fundamentally broken that they can't be used securely? Maybe so. At the very least we must carefully control LLM foundation model use and think hard about which applications make sense and which do not.

[raw:1:recursive pollution] The number one risk in LLMs today is recursive pollution. This happens when an LLM model is trained on the open Internet (including errors and misinformation), creates content that is wrong, and then later eats that content when it (or another generation of models) is trained up again on a data ocean that includes its own pollution. Wrongness grows just like guitar feedback through an amp does. BIML identified this problem in 2020. See [BIML78 raw:8:looping], [LLMtop10:1:recursive pollution] and Shumailov.[6]

[raw:2:poison in the data] Another LLM top ten risk is tied up in the training data challenge—gathering up an enormous enough pile of data to train an LLM generally involves scraping lots of it off the Internet. Needless to say, the Internet is full of poisonous, wrong, evil, misleading, and otherwise unfit for use data. Raw data are riddled with badness, and removing it all to disinfect training and input data is a gargantuan task. See [LLMtop10:6:poison in the data].

[raw:3:data feudalism] We have entered the era of data feudalism. LLM foundation models require huge oceans of data for training. But the oceans are being sectioned off by data moats. If entire enormous parts of the Internet (say Google search data, Twitter/X content, or GitHub code piles) are cordoned off, it is not clear what will happen to LLMs not already built and controlled by the data ocean owners. Similarly, owners of large data sets (*e.g.*, the New York Times) have come to realize that their data are valuable in a new way and are demanding compensation. Who has access to the data pool, and why? Lock in for early LLM foundation model movers is a very real risk.

[raw:4:data rights] LLM foundation models require vast troves of data for training. These data are often scraped from the public Internet (but all of those data may not be in the public domain or protected by fair use). That means that some training data are likely subject to Copyright, government regulation, or privacy controls like GDPR. There are a number of active legal cases (ongoing as of this writing) regarding Copyright infringement, misuse of non-public data, fair use, generative-model training, and generative-model output. These legal entanglements are worth careful consideration when building an LLM application. Where did the raw data modelled in the LLM foundation model come from? This is related to [LLMtop10:8:data ownership].

[raw:5:data confidentiality] Preserving data confidentiality in an LLM is more challenging than in some other ML situations since the foundation model is built in a black box fashion out of underspecified training sets and is designed to model the data itself (*i.e., become* the data), aiming to reproduce similar data in a generative way. Recall that if an ML system is trained up on illegal, confidential, or sensitive data, it will have some aspects of those data built right into it through training. Attacks to extract sensitive and confidential information from ML systems (indirectly through normal use) are well known.[10] See [LLMtop10:8:data ownership].

[raw:6:trustworthiness and curation] Public Internet data sources are not trustworthy, suitable, and reliable. The web is full of nonsense, garbage, and evil content, some of which is intentionally placed there to poison ML models. This has a direct impact on the trustworthiness of LLM foundation models. The interface of LLMs (natural language) also impacts trustworthiness since both input and output precision are loose. See [LLMtop10:2:data debt][LLMtop10:9:model trustworthiness].

[raw:7:utility] If your data are poorly chosen, you may reach incorrect conclusions regarding your ML approach. You should always make sure your methods match your data and your data are properly vetted and monitored. In the LLM case this is much harder than usual for two reasons: 1) the dataset is immense (a data ocean, in fact) and is so large that vetting and monitoring is a huge challenge, and 2) the training dataset is controlled in a black box fashion and is not fully described. ML systems can fail just as much due to data problems as due to poorly chosen or implemented algorithms, hyperparameters, and other technical system issues. LLMs push the limits of utility risk in many ways. See [LLMtop10:3:improper use].

[raw:8:legal entanglements] Note that public data sources may include data that are in some way legally encumbered. This is especially problematic in the LLM situation. An obvious example is Copyrighted material that gets sucked up in a data stream. Another more insidious example is child pornography which is never legal.[8] A third, and one of the most interesting legal issues, is that there may be legal requirements to "delete" data (e.g., from a GDPR request). What it means to "delete" data from a trained model is impossible to carry out (short of retraining the model from scratch from a data set with the deleted data removed, but that is expensive and often infeasible). Note that through the learning process, input data are always encoded in some way in the model itself during training. That means the internal representation developed by the model during learning (say, thresholds and weights) may end up being legally encumbered as well. This is related to [LLMtop10:8:data ownership].

[raw:9:time] In some sense LLMs are frozen in time WRT their training data set. Often this will be several years earlier than the present. LLMs thus have a warped sense of history and experience and are often out of sync with current events and some worldviews. Retrieval Augmented Generation (RAG) provides some in-context learning and may be one way to finesse this issue.

[raw:10:query data] Search is already being deeply impacted by LLMs. In some modern search designs, an ML-based commercial system searches the web and builds a prompt for an LLM. This LLM provides the user's interface to search. When the LLM is wrong, hilarity ensues…or chaos.

Associated controls. Note that the labels refer to the original risks (above) which have controls that may help alleviate some of the risk directly:

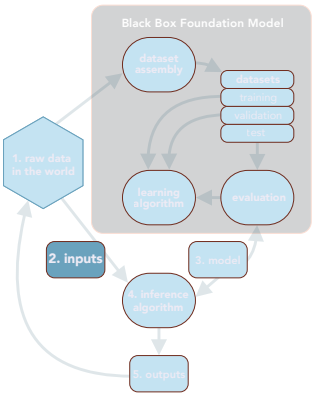
[raw:generic] Protect your data sources if you can. In the LLM case, this is being accomplished by building data moats. Ultimately, data feudalism will be very bad for LLM progress.

[raw:generic] Sanity check your data algorithmically before you feed it into your model (*e.g.*, using outlier detection, mismatched unit discovery, data range distribution analysis, and so on). For example, make sure that your data properly characterize and represent the problem space so that the ML model learns what it is supposed to learn. Ironically, this is one of the most difficult engineering problems involved in ML as a field and is made even harder by the unstable, unpredictable interface that LLMs use (natural language).

[raw:generic] Use version control technology to manage your datasets. Carefully track change logs, diffs, etc, especially when it comes to large datasets. Some of the enormous datasets behind LLMs are curated. Others are not. Perhaps a Data Source Bill of Materials (DSBOM) would help.

[raw:7:utility] Some organizations are building specialized models in the hopes that they are more suitable for specific subdomains. For examples, see the Bloomberg financial LLM, AMIE from Google in the medical domain, and various models for tissue pathology. We caution that some of these specialized models treat LLMs as GPAI, a position we do not agree with.

[raw:8:looping] Look for loops in data streams and avoid them. If raw data come from public sources and system output is also made public, loops may arise without your awareness. Recursive pollution will be the bane of LLMs.



2 Input risks: When a fully trained model is put into production, a number of risks must be considered. Probably the most important set of these operations/production risks revolves around input data fed into the trained model. In the case of LLMs, these data are referred to as *prompts*. Making things particularly squirrelly, LLM input is mostly unstructured, using natural language instead of crisp symbols or well-defined feature vectors. That means LLM input is already intentionally ambiguous and the system is designed to cope with such ambiguity.

[input:1:recursive pollution] If system output feeds back into the real world there is always some risk that it may find its way back into input causing a feedback loop. This generic risk is particularly relevant for LLMs which are trained on enormous Internet scrapes and then have much of their output posted back to the public internet. When a feedback loop involves poisoning or otherwise tainting future training data, bad things happen. This is the number one LLM security risk today. See [BIML78 raw:8:looping], [LLMtop10:1:recursive pollution] and Shumailov.[6]

[input:2:prompt injection] One of the most important categories of computer security risks is malicious input. The LLM version of malicious input has come to be known as *prompt injection*. In fact, you can think of prompt injection as the analog of adversarial examples in the ML vision situation. And as was the case with adversarial examples, prompt injection discussions tend to swamp out all other risks in peoples' imaginations. This leads directly to an over-emphasis on red teaming (to the detriment of the field). All that said, playing around with LLM prompts is really fun and can be very enlightening. *It is extremely unlikely that LLM behavior can be completely controlled through prompting alone*. The stochastic nature of the basic LLM interface exacerbates this control issue. An LLM may seem to "do the right thing" sometimes and go completely off the rails with the very same prompts at other times. That makes LLM verification a challenge at best and an impossibility at worst. Much of what's called "red teaming" in AI really amounts to prompt manipulation. See [LLMtop10:5:prompt manipulation].

[input:3:open to the public] Many LLM models are open to the public and are thus susceptible to manipulation by an attacker. Note that an attacker can experiment with an LLM before carrying an attack out elsewhere. Recent research shows that these kind of manipulation attacks even transfer across models. One fun example of this risk is the DAN attack (that is, the "do anything now" attack). The public has taken an important role in prompt manipulation experiments.

[input:4:dirty input] When it comes to LLMs, the entirety of the Internet, or at least an enormous portion of it, constitutes the training data. Sadly, the internet has become a cesspool of disinformation, bigotry, misogyny, and factually incorrect content. A knowledgeable attacker can take advantage of the LLMs training noise to make the LLM misbehave. This can often be accomplished by asking the LLM to do things it should not do or asking it to provide information that it should not disclose. This risk is directly related to [LLMtop10:9:model trustworthiness].

[input:5:sponge input] Sponge attacks provide ML systems with input that costs much more to process than is usually the case. The idea is to exhaust processing budget or at least make it cost prohibitive to continue processing. One example of a sponge attack in LLMs involves embedding Chinese characters into an English sentence.[11]

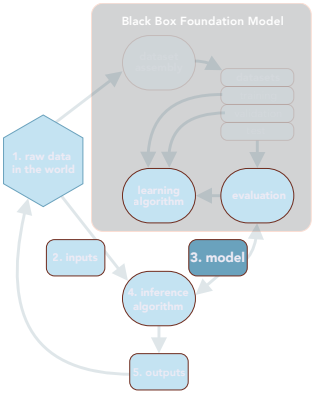
[input:6:input ambiguity] English, which is the main interface language for LLMs, is an ambiguous interface. Natural language can be misleading, making LLMs susceptible to misinformation and manipulation both in training and in operations. LLMs have exhibited particularly bad performance when domain specificity is important. Models learn "average" token-based representations of complex terminology, failing to encode nuance and critical deep knowledge during their training on massive training data sets.

[input:7:controlled input stream] Because foundation models for LLMs are owned by huge corporations, their behavior is in some sense controlled by these corporations who then provide a black box component for use in LLM applications. LLM foundation model owners often impose restrictions on input to LLMs. This leads directly to data feudalism. See [raw:3:data feudalism].

Associated controls. Note that the labels refer to the original risks (above) which have controls that may help alleviate some of the risk directly:

[input:2:prompt injection] LLM foundation models already impose restrictions on prompts. Further restrictions can be constructed as filters and monitors and can be interposed before the LLM foundation model. Calypso AI's Moderator has this capability.

[input:3:dirty input] Sanity checks, filters, and data cleaning can control this risk. Of course, those mechanisms can be attacked as well. Note that often pre-processing ends up being more about making an LLM system be able to learn than it is about "getting it right."



3 Model risks: When a fully trained LLM foundation model is put into production (often by being included in an application), a number of important risks crop up.

[model:1:black box opacity] Foundation models are not often controlled by LLM users. That means risk management decisions made by the LLM foundation model vendor have mostly been made for you (instead of by you or with you). Determining just how some security tradeoffs are made is impossible given scarce information. No amount of outside➔in red teaming or testing can tell you all you need to know. Ultimately, an LLM foundation model user is provided with what amounts to an undocumented, unstable API with sometimes unanticipated behavior. That makes the job of securing an LLM application exceedingly challenging. See [LLMtop10:4:black box opacity].

[model:2:improper use] LLM foundation models can be put to use in a very large number of ways. What's important to remember is that LLMs are really auto-associative predictive generators. That is, they don't really "understand" anything the way humans do. The upshot is, if LLMs are put to use in applications that require actual understanding and insight, they may fail catastrophically. As an example, consider the now-disbarred lawyer who used ChatGPT to create a brief filed in Federal court which included spurious citations of non-existent cases. Great care must be given to LLM use in various situations. See [LLMtop10:3:improper use].

[model:3:reproducibility] The economics of LLMs do not work in the favor of science. Because building a model is so expensive, many academics can't afford to build big ones. They can study commercial LLMs in a black box fashion, or they can build smaller models to experiment with, but this may not work. Because the black box is opaque, reproducibility is limited. This works in favor of first mover corporations and against other players. See [LLMtop10:7:reproducibility economics].

[model:4:Trojan] There are a large number of LLM foundation models, and choosing which one to use is non-trivial given the paucity of technical information available to make such a decision. LLMs continue to proliferate, meaning that the risk of Trojan'ed models is probably substantial. Take care where you get your LLM model and who you get it from. Pay close attention to integrity. Recent work on "sleeper agents" from Anthropic is directly related to this risk.

[model:5:representation fluidity] ML and LLMs are appealing exactly because they fly in the face of brittle symbolic AI systems. When a model generalizes, it builds up a somewhat fluid representation if all goes well. The real trick is determining how much fluidity is too much. LLMs are auto-associative predictive generators which don't "understand" but rather predict tokens. LLMs produce output that "sounds good" but may be wrong. (We really hate the term "hallucination" used by some to describe such wrongness, but it is in common use.) Sussing out the truth from what's new, cool, and creative is a daunting task. See [LLMtop10:10:encoding integrity].

[model:6:training set and prompt reveal] LLMs learn (and sometimes directly memorize) a great deal about training input, some of which is possibly sensitive (see [raw:5:data confidentiality]), and store a representation internally that may include sensitive information. Be aware of the output your model produces and how it may reveal sensitive aspects of its training data and its prompting data.

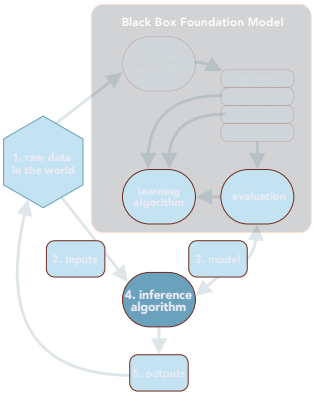
[model:7:vendor lock-in] LLMs are generally too big to directly rip off. This has an economic consequence of lock-in, should you build your application around a particular vendor. Switching out an LLM foundation model is a non-trivial undertaking since each model has its own quirks and foibles. Of course, many applications make use of multiple models in a chain (which in many cases does nothing to alleviate the lock-in risk). See [LLMtop10:7:reproducibility economics].

[model:8:picking a foundation] Choosing which foundation model to use is a critical choice. This choice is made difficult by the opacity of the black box and its associated risks. Comparing foundation models directly is not easy. See [LLMtop10:4:black box opacity].

[model:9:modality] LLMs are chameleons. There are some modalities where LLM stochasticity and fluidity (in concert with seeming confidence) are really not a good thing. Managing representation fluidity within a single modality is already a challenge, doing so across multiple modalities promises to be even trickier.

Associated controls. Note that the labels refer to the original risks (above) which have controls that may help alleviate some of the risk directly:

[model:5:vendor lock-in] Watch the output that you provide (it can and will be used against you). Some LLM monitoring technology does exactly this for you. Calypso AI's Moderator product is one example.



4 Inference algorithm risks: When a fully trained model is put into production, a number of important risks must be considered. These encompass data fed to the model during operations (see raw data risks and black-box risks), risks inherent in the production model, and output risks.

[inference:1:prompt manipulation (aka prompt injection)] Malicious input in the form of prompt manipulation is probably the most popular way to probe LLM behavior. There are many well-documented prompt manipulation attacks, including: do anything now, "pretend to be," "consider these data," priming attacks, and chipper praise. Prompt manipulation is a straightforward way to cause an LLM to ignore its guardrails and do something its creators didn't want it to allow you to do. Sadly, this simple attack is taking up all the oxygen in the room, leading some to believe that "red teaming" models is an effective security strategy. It is not. See [LLMtop10:5:prompt manipulation].

[inference:2:inscrutability] LLM applications can be fielded without a real understanding of how a foundation model works or why it does what it does. Integrating an LLM foundation model that "just works" into a larger system that then relies on the LLM to perform properly and consistently is a very real risk. To top it all off, LLMs are, by their very design, stochastic. See [LLMtop10:4:black box opacity].

[inference:3:wrongness] LLMs have a propensity to be just plain wrong. Plan for that. (Using anthropomorphic terminology for error-making, such as the term "hallucinate" is not at all helpful.) See [LLMtop10:9:model trustworthiness].

[inference:4:stochasticity] In any stochastic model, real randomness is important. Seeds matter. Careful use of random number generation is fraught with risk.

[inference:5:hyperparameters] LLM foundation models have a critical parameter called temperature that relaxes or tightens statistical choices. The impact of generating a more stochastic or less stochastic output depends on the application and the ways in which outputs are being used downstream. Be aware that this parameter exists and can sometimes be manipulated by users.

[inference:6:feedback scores] Some LLM chat systems allow user feedback which in turn can be a method for gaming an LLM service that incorporates such feedback for updating the system.

[inference:7:personal evaluation protocol] Users of LLMs often "fine-tune" models with their own text. Determining whether an LLM is doing the right thing is a subjective exercise left to the person doing the testing.

[inference:8:unstructured output] Sometimes inference is indeterminate and can be camouflaged by the vagaries of natural language. In general, a bunch of text as output is not as clear as you may like it to be and may mask bad inference or bad associations, making evaluation much harder. Numbers and class labels do not translate well into text streams, and forcing the generation of these sorts of things is not a good application of a generic LLM.

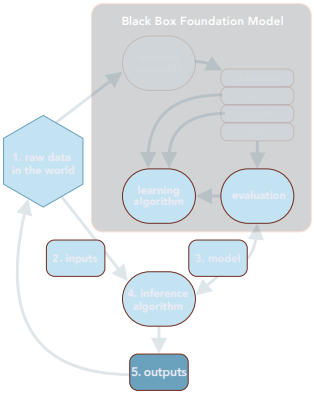
[inference:9:hosting] LLM foundation models are run on hosted, remote servers. Care must be taken to protect these machines against ML-related attacks (not to mention the usual pile of computer security stuff). Water, heat dissipation, and electricity are all important aspects of making ML work. Plan for what happens to your LLM application if the net goes down or if a physical attack on servers occurs.

[inference:10:user risk] User interaction with an LLM is sometimes carried out in the clear. I/O traffic can be intercepted and manipulated in an attacker in the middle scenario. When a user decides to use any ML system that is remote, they expose their interests (and possibly their input) to the owners of the ML system. This risk is exacerbated by the fact that current LLM APIs are typically stateless, even during extended chat interactions with a model, any part of the conversation can be modified at any stage. Edge models must still communicate, but can shift the attack surface.

Associated controls. Note that the labels refer to the original risks (above) which have controls that may help alleviate some of the risk directly:

[inference:4:stochasticity] Have a real cryptographer look at your random number generation capability.

[inference:9:hosting] Take care to isolate engineering ML systems from production systems. Production systems in particular should be properly hardened and monitored.



5 Output risks: Keep in mind that the entire purpose of creating, training, and evaluating an LLM may be so that its output serves a useful purpose in the world. The second most obvious direct attack against an LLM system is to attack its output.

[output:1:direct] An attacker tweaks the output stream directly. Because LLMs are stochastic, inscrutable, and use unstructured inference, hiding a direct attack in plain sight may be trivial. There are many ways to do this kind of thing. Probably the most common attack would be to interpose between the output stream and the receiver.

[output:2:wrongness] Prompt manipulation can lead to fallacious output (see [input:2:prompt injection]), but fallacious output can occur spontaneously as well. LLMs are notorious BS-ers that can make stuff up to justify their wrongness. If that output escapes into the world undetected, bad things can happen. If such output is later consumed by an LLM during training, recursive pollution is in effect. See [LLMtop10:5:prompt manipulation] and [LLMtop10:9:model trustworthiness].

[output:3:provenance and trustworthiness] LLM systems must be trustworthy to be put into use in situations outside of pure entertainment. Even a temporary or partial attack against output can cause trustworthiness to plummet. See [LLMtop10:9:model trustworthiness].

[output:4:data confidentiality] LLM foundation models are in a state of legal flux as of the time of this writing. Various lawsuits may have a catastrophic impact on current approaches. Many of the concerns orbit around confidential and privacy-related information scraped off the Internet for training purposes. Copyright law and GDPR are directly relevant. See [LLMtop10:8:data ownership].

[output:5:inscrutability] When it comes to LLM foundation models, nobody is really sure how the trained systems do what they do. This is a direct affront on trustworthiness and can lead to challenges in some domains such as diagnostic medicine and math. The black box nature of LLM foundation models does nothing to help this situation. See [LLMtop10:4:black box opacity].

[output:6:lack of transparency] Decisions that are simply presented to the world with no explanation are not transparent. LLMs present output in a confident (and often glib) manner with little of the reasoning exposed. Attacking opaque systems is much easier than attacking transparent systems, since it is harder to discern when something is going wrong. See [LLMtop10:4:black box opacity].

[output:7:eroding trust] Causing an LLM system to misbehave can erode trust in the entire discipline. LLMs already have some spectacular failures that have been highlighted, including faked multi-modal video demos, citation of non-existent legal cases, and promulgation of ridiculous "scientific facts."

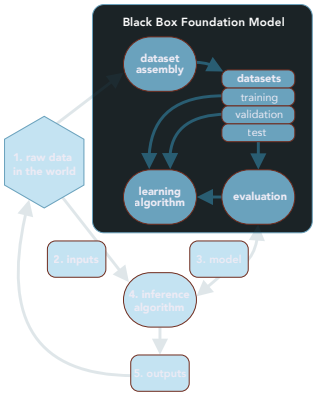
[output:8:looped output] See *[BIML78 input:4:looped input]*. If system output feeds back into the real world there is some risk that it may find its way back into input causing a feedback loop. This has come to be known as recursive pollution. See [LLMtop10:1:recursive pollution].

[output:9:unstructured output] Bad associations can be camouflaged by the vagaries of natural language. When a system is classifying something (like, say, sentiment), a bunch of text as output is not as clear as you may like it to be. Class labels and numbers do not translate well into text streams, and forcing the generation of these sorts of things is not a good application of a generic LLM.

[output:10:regulation] ML regulation is proceeding apace even as the field evolves and changes rapidly. Our view is that the LLM foundation model black box is the best target for regulation. Keeping abreast to regulatory developments (and legal developments) is critical.

[output:11:black box discrimination] Many data-related component risks lead to bias in the behavior of an ML system. ML systems that operate on personal data or feed into high impact decision processes (such as credit scoring, employment, and medical diagnosis decisions) pose a great deal of risk. When biases are aligned with gender, race, or age attributes, operating the system may result in discrimination with respect to one of these protected classes. Using biased LLM subsystems is definitely illegal in some contexts, may be unethical, and is always irresponsible.

[output:12:overconfidence] When an LLM is integrated into a larger system and its output is treated as high confidence data, users of the system may become overconfident in the operation of the system for its intended purpose. A low scrutiny stance with respect to the overall system makes it less likely that an attack against the LLM subsystem will be detected. Developing overconfidence in LLMs is made easier by the fact that they are often poorly understood and vaguely described. See [LLMtop10:3:improper use].



Black-Box Foundation Model Risks: The most notable difference between our original generic ML process model (Figure 1) and our LLM version (Figure 2) is the emergence of a large black box that hides—*and leaves out of the user's control*—important aspects of the ML process. For example, both the **Dataset Assembly** process and the **Datasets** used in training are hidden from the user, as are the particulars of both **Learning Algorithms** and **Evaluation** methodologies. When an LLM foundation model is released, disclosures of any of these critical aspects of LLM creation are often both incomplete and underspecified.

Taking so much of the ML process out of the user's control and even out of any sort of scrutiny is a situation ripe for risk. The fact that four major generic ML components are outside of the user's control, means we must count on LLM foundation model creators to get everything right when it comes to managing risks. Surprise…they don't!

In our view, the only reasonable way to control these black box risks is to make the black box the target of regulation. Of course, doing that properly requires a set behavioral expectations for the hidden components as well as associated methods to validate that reasonable expectations have been met.

The thing about LLMs that is so striking in comparison to past ML models is the vast resource requirements required to train them. Training complexity is, simply put, off the charts. The complexity of training a single one of these LLM foundation models is greater in terms of computation than all previous families of ML models combined, with multiple distinct stages of learning. Current LLM modeling approaches also require enormous amounts of unstructured data for at least some of the self-supervised learning stages, which creates new ML challenges around the provenance of this ocean of data. Taken together, these aspects of LLM modeling create many more places for risk to creep in—and all of this with severely reduced user visibility.

LLMs leverage multiple stages of representation and multiple stages of learning. First, LLMs work by representing text using a learned *tokenization* and *representation* of the resulting tokens, to create the numerical input to the neural network. A neural network is then trained using a *self-supervised pre-training with an attention model* (the P in GPT) to achieve a more contextual representation of the input tokens. This self-supervised pre-training is a fundamental stage that currently requires both vast datasets and correspondingly-high model complexity to work. Computational requirements are described by empirical scaling laws that purport to show how pre-training task performance develops as the amount of data and model parameters increase. Ultimately, that contextual representation is used to guide the auto-regressive generation (the G in GPT) of text continuations that are expected to solve a wide range of tasks, linguistic and otherwise. It's worth mentioning that the remaining T in GPT stands for transformer, referencing a critical computational primitive currently used in LLMs and other foundation models, and for which a solid theoretical understanding is still missing.

Already significantly more complex than many ML approaches, LLM training at this point is usually not complete. Starting from the a pre-trained model, two or more learning stages are typically implemented. The additional training stages satisfy the need for *fine-tuning* and *alignment*. Fine-tuning stages require the assembly of new annotated or carefully curated datasets of modest size usually to help the LLM be more effective at following instructions and understanding a domain or task. Correspondingly these fine-tuning stages may be called: *instruction-tuning*, *domain-tuning*, or *task-tuning*. (Sometimes these are simply referred to as fine-tuning.) Alignment stages are usually implemented using reinforcement-learning leveraging some kind of feedback mechanism, the most prominent being human users but now also extending to the use of other models.

What follows are a set of "hidden risks"—hidden in the sense that they are present and controllable only with access inside the black box of the foundation model. It is our view that these hidden risks contribute to many of the top LLM risks. You might ask yourself whether controlling these risks properly should be left up to LLM foundation model creators. That turns out to be a very good question indeed.

[blackbox:1:recursive pollution] The number one risk in LLMs today is recursive pollution. This happens when an LLM model is trained on the open Internet (including errors and misinformation), creates content that is wrong, and then later eats that content when it (or another generation) is trained up again on a data ocean that includes its own pollution. Controlling recursive pollution is a critical responsibility inside the black box of LLM foundation models. Regulation might ensure that recursive pollution is rare or impossible. See [LLMtop10:1:recursive pollution].

[blackbox:2:trustworthiness and curation] The scale of the datasets required for pre-training means that LLM model developers use Internet scale datasets. Public Internet data sources are not trustworthy, suitable, and reliable. The web is full of nonsense, garbage, and evil content. This creates an immense quality control challenge, that is hard to overcome and has a direct impact on the trustworthiness of LLM foundation models. The coverage and annotation requirements for curating fine-tuning datasets create another great challenge, which then challenges our trust in the fitness of the model for the specific fine-tuning task. Regulation might require insight into data sources and data sets, including what processes are used to clean up raw data. See [LLMtop10:2:data debt].

[blackbox:3:data rights] LLM foundation models require vast troves of data for training. These data are often scraped from the public Internet. That means that some training data are likely subject to Copyright, government regulation, or privacy controls like GDPR. There are a number of active legal cases (ongoing as of this writing) regarding Copyright infringement, misuse of non-public data, fair use, generative-model training, and generative-model output. These legal entanglements are worth careful consideration when building an LLM application. Regulation might require insight into data sources and data sets, including what processes are used to clean up raw data. See [LLMtop10:2:data debt].

[blackbox:4:legal issues] Since LLMs require oceans of data for training, use of public data sources is almost a guarantee. Some of these data sources may be legally encumbered. Consider copyrighted material, child pornography (never legal), or data protected by the GDPR "right to be forgotten." Note that through the learning process, input data are always encoded in some way in the model itself during training. That means the internal representation developed by the model during learning (say, thresholds and weights) may end up being legally encumbered as well. Regulation might require insight into data sources and data sets, including what processes are used to ensure that data are not legally encumbered. See [LLMtop10:2:data debt].

[blackbox:5:data confidentiality] Preserving data confidentiality in an LLM is more challenging than in some other ML situations since the foundation model is built in a black box fashion out of underspecified training sets. Recall that if an ML system is trained up on confidential or sensitive data, it will have some aspects of those data built right into it through training. Attacks to extract sensitive and confidential information from ML systems (indirectly through normal use) are well known. Poisoned data, which can also find its way into the model, can be used to facilitate specific data extraction attacks. Regulation might require insight into data sources and data sets, including what processes are used to ensure that confidential data (like, say, social security numbers) is not part of training. See [LLMtop10:2:data debt].

[blackbox:6:poison in the data] Both raw data in the world and data inside the black box are subject to poison. Since the datasets involved are so enormous, it is very hard to control or clean up all of the potentially harmful associations scraped up in raw data. This is a risk related both to data sensitivity and to the fact that the data themselves carry so much of the water in an LLM system. If you train up on poison, you are poisoned. A special case of poison in the data exists when an attacker intentionally injects poison into the data ocean. Regulation might require insight into data sources and data sets and make explicit what efforts have been made to avoid poison data. See [LLMtop10:6:poison in the data].

[blackbox:7:bad eval data] Evaluation is tricky, and an evaluation data set must be designed and used with care. A bad evaluation data set that doesn't reflect the data a system will see in production can mislead a researcher into thinking everything is working even when it's not. Evaluation sets can also be too small or too similar to the training data to be useful. Evaluation processes are often trade secrets and are never exposed outside the black box. Poorly evaluated and tested LLM foundation models are particularly susceptible to improper use. Regulation might require insight into evaluation criteria, tests, and results. See [LLMtop10:3:improper use].

[blackbox:8:pseudo-science] Common sense evaluation and rigorous evaluation are not always the same thing. For example, evaluation of a natural language processing system may rely on "bags of words" instead of a more qualitative structural evaluation. This risk is particularly problematic when it comes to the idea of "red teaming." By and large this has come to mean screwing around with prompt manipulation in an unprincipled but entertaining manner. Anthropomorphism is an insidious concern here—interpretation and generalization of red teaming outcomes in human terms does not meet the evaluation task. Simply put, this is not how security testing should be carried out. Regulation might require insight into evaluation criteria, tests, and results, especially for security risks. See [LLMtop10:4:black box opacity] and also [LLMtop10:5:prompt manipulation].

[blackbox:9:data feudalism] The data ocean is being carved up, and we have entered the era of data feudalism. Assembling an enormous data set without already owning enormous amounts of data may soon be impossible. LLM foundation models require huge oceans of data for training. Who has access to a large enough data pool, and why? Lock-in for early LLM foundation model movers is a very real risk. Further, transparency of and audibility of training data is required to understand and mitigate many of the risks incurred by LLM use. Whether regulation can even the playing field with regard to data access remains to be seen. See [LLMtop10:7:reproducibility economics].

[blackbox:10:cognitive dissimilarity] Human cognition and human understanding are not equivalent to what goes on inside an LLM. Auto-associative predictive generation models do not, in fact, understand or reason like people do. They simply produce associative predictions regarding next token. The raw data represented by a dirty scrape of the Internet are not likely to model human thinking in a reasonable manner. Anthropomorphism in evaluations is a particularly insidious concern, when models are evaluated with respect to human evaluation benchmarks (*e.g.*, MCAT and LSAT scores) the meaning of the outcomes does not convey. See both [LLMtop10:4:black box opacity] and [LLMtop10:3:improper use].

[blackbox:11:exploit-v-explore] Part of the challenge of tuning an ML system during the research process is understanding the search space being explored and choosing the right model architecture (and algorithm) to use and the right parameters for the algorithm itself. Thinking carefully about problem space exploration versus space exploitation will lead to a more robust model that is harder to attack. In the LLM foundation model case, models are so expensive to create that cost may put a damper on exploration (not to mention model opacity issues). See [LLMtop10:7:reproducibility economics].

[blackbox:12:reproducibility economics] Building an LLM foundation model is expensive. Because of the black box nature of the process, corners may be cut and intellectual shortcuts taken, and nobody will be the wiser. Outside organizations are easy to cut out of the process by citing costs. See [LLMtop10:7:reproducibility economics].

[blackbox:13:dataset weak representation] Assembling a dataset involves doing some thinking and observation about the resulting representation inside the ML model. Robust representations result in fluid categorization behavior, proper generalization, and non-susceptibility to adversarial input. In the case of natural language used in LLM interfacing and training, full multi-task, multi-model and multi-lingual representations are unlikely to emerge, giving some tasks, modalities, and some languages uneven coverage. See [LLMtop10:9:model trustworthiness].

[blackbox:14:misleading re-representation] What constitutes private data? When Meta collected hundreds of millions of pictures of faces (with often vague or misleading permission), they got in some trouble with privacy advocates and users. Their "solution" was to erase the raw data, but only after encoding the data in their proprietary face2vec representation. If a face2vec representation is 1:1 and onto with raw face data from pictures, what is the difference? Did they really delete anything? These are tricky questions that unscrupulous companies are gaming to confuse users. Using re-representation to side-step tricky issues is not very hard inside the black box. Intentional malicious re-representation should be disallowed by regulation. This is related to [LLMtop10:10:encoding integrity].

[blackbox:15:representation opacity] LLMs use proprietary encoding and representation schemes built using unsupervised learning with an attention model. Representation inside the black box is opaque. There are open scientific questions that remain to be explored, such as: is self-attention really more amenable to interpretability than other schemes? When it comes to domain specificity (say, chemistry knowledge or medical terminology) how does the black box representation work? This is also related to [LLMtop10:10:encoding integrity] and [LLMtop10:7:reproducibility economics].

[blackbox:16:encoding integrity] Encoding integrity issues can be both introduced and exacerbated during training. Do aspects of the training process itself introduce security problems? Bias in raw data processing can impact ethical and moral implications. See [LLMtop10:10:encoding integrity].

[blackbox:17:utility] In ML, the machine *becomes* the data. That means if your data are bad, your machine will not do what you want it to do. This is exacerbated in the LLM case by the enormous size of the data ocean required for training. Cleaning up the data ocean is an already challenging task that seems to be getting harder instead of easier as pollution levels rise. If training data sets and the pollution removal controls are not made clear because they are hidden inside a black box, huge utility risk emerges. ML systems can fail just as much due to data problems as due to poorly chosen or implemented algorithms, hyper-parameters, and other technical system issues. LLMs push the limits of risk in many ways. This is particularly relevant to [LLMtop10:3:improper use].

[blackbox:18:evaluation] Evaluating the behavior of a pre-trained LLM foundation model that emerges from unsupervised learning and self-attention is a black art. Very little is published about how to do this. Existing benchmarks often test the wrong kinds of things. For example, they often focus on averages over not-fully-understood datasets and do not cover specific failure modes.[12]

[blackbox:19:memorization] LLMs are so large (encompassing so many parameters) that memorization of some aspects of the training data ocean are inevitable. Memorization can deleteriously impact generalization. Associations that are too specific may cause wrongness.

[blackbox:20:temperature sensitivity] LLMs loosen or tighten next token choice using a parameter called temperature. When the temperature is high, looser choices can be made. Temperature can be a very sensitive parameter if it is available to end users. Choice of temperature also has direct impact on model evaluation.

[blackbox:21:time] In some sense LLMs are frozen in time WRT their training data set. Often this will be several years earlier than the present. LLMs thus have a warped sense of history and experience and are often out of sync with current events and some worldviews. This can lead to [LLMtop10:3:improper use].

[blackbox:22:randomness] Randomness has a long and important history in security. In particular, Monte Carlo randomness versus cryptographic randomness is a concern. When it comes to LLMs, setting weights and thresholds "randomly" must be done with care. Many pseudo-random number generators (PRNG) are not suitable for use. PRNG loops can really damage system behavior during learning. Having randomness reside in the black box makes it more likely to be done wrong.

[blackbox:23:query data] Search is already being deeply impacted by LLMs. In some modern search designs, an ML-based commercial system searches the web and builds a prompt for an LLM. This LLM provides the user's interface to search. When the LLM is wrong, hilarity ensues…or chaos.

Broad Concerns with LLM Use

In our view, the biggest challenge in secure use of LLM technology is understanding and managing the 23 risks inherent in the black box foundation models. From the point of view of an LLM user (say, someone writing an application with an LLM module, someone using a chain of LLMs, or someone simply interacting with a chatbot), choosing which LLM foundation model to use is confusing. There are no useful metrics for users to compare to make a decision about which LLM to use, and not much in the way of data about which models are best to use in which situations or for what kinds of application.[‡]

Opening the black box would make these decisions possible (and easier). For this reason, we are in favor of regulating LLM foundation models. Not the *use* of these models, but *the way in which they are built* (and, most importantly, out of what) in the first place.

We would like to see regulation that addresses the black box risks we identified above. In particular, more clarity around training data sets, including whether such data sets are known to be clean, would be very helpful. Also helpful would be policy statements regarding acceptable use for various LLM foundation models. Should LLMs

be used to do anything and everything at the discretion of the user? Probably not. So what should particular LLM foundation models not be used for according to the people who built them? We noted which particular blackbox risks above we think should be directly covered by regulation.

Because of the data ocean they encapsulate and automate, LLMs reflect cultural bias in obvious and disturbing ways. Some kind of measurements or statements regarding such biases would be helpful in choosing an LLM foundation model.

What kinds of resources were consumed to build an LLM foundation model, and how many resources are consumed when operating it? Efficiency requirements would be easier to meet (and control) given these data.

Then there is embedded LLM use. When LLM output becomes input to a larger decision process, errors arising in the LLM subsystem may propagate in unforeseen ways. The evaluation of LLM subsystem performance in isolation from larger system context may not take into account the "regret" this may incur. That is, methods that evaluate LLM accuracy may not evaluate utility, leading to what has been called regret in the ML literature.

Any LLM system can and will make mistakes. The consequences of these errors cannot always simply be written off as cutesy little "hallucinations." Sometimes, being wrong has a direct impact on peoples' lives and well-being. Systems including LLMs must take their stochastic and unpredictable nature into account.

If system users are unaware of how LLMs really work, they may not be able to account for "incorrect" behavior. Lost confidence may follow logically. Ultimately, users may erroneously conclude that the LLM system is not beneficial to operation *at all* and thus should be discarded.

Using this Document with a Specific LLM Application

This document presents a basic architectural risk analysis and a set of 81 specific risks associated with a generic LLM process model and its LLM foundation model. We organize the risks by common component and also include a number of critical LLM black box foundation model risks. Our risk analysis results are meant to help LLM systems engineers in securing their own particular LLM applications.

Ultimately, those responsible for building an LLM application will be accountable for both the choice of which LLM foundation model to use and the proper construction of any application. The complete system requires accountability with regard to enterprise risk management.

The risks we have identified here can be made more specific in the context of a particular LLM application. That is, given a specified architecture of an LLM application and a general understanding of how the LLM application operates, the risks we have identified here can be refined for the specific LLM application target. (By analogy, we developed the LLM risks presented in this work by considering more generic ML system risks (the BIML-78) and how they apply in the general LLM case.) In our experience, revisiting high-level risks and putting a finer point on them using the context of an application specification is a powerful analysis activity.

Here is an example to help demonstrate such a risk refinement process. Lets take a brief look at retrieval-augmented generation (RAG)—a very common way of applying LLMs in an application. RAG has emerged as a response to a number of obvious shortcomings of LLMs in applications, most prominently wrongness (unfortunately called "hallucinations" by many) and LLM training time date limitations. RAG is also used to manage LLM wrongness in search applications, where results to a search engine query are retrieved and summaries computed and incorporated into a prompt that is ultimately used to generate an LLM-based search result.

LLM-related RAG risks can crop up at every stage of the RAG pipeline. The retrieval component will often rely on LLM associations/representations at some stage. The suitability of these associations/representations for a retrieval or re-ranking task are subject to various representation risks in our taxonomy, for example [LLMtop10:10:encoding integrity] and [model:5:representation fluidity].

‡ Though leaderboards track LLMs, the kinds of measurements being made are not useful in determining LLM capability. See, for example, https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard and https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard. The critique in reference 10 applies.

Summaries of the obtained retrieval results are often created as another LLM text generation task, and are yet again prone to various LLM risks. We may end up "hallucinating" (being completely wrong about) summaries of the real documents we are trying to use for "grounding" our results, and subsequently migrating this wrongness risk through the chain. Note, for example that it is not uncommon for search engines to improperly "read" search results before showing anything to the user.

Ultimately, LLM-generated search results summaries are incorporated into a prompt aiming to generate a result to the original search query for the user. This step is now subject to the compounding of all errors from previous LLM-related steps as well as all of the concerns regarding the control of an LLM through prompting, like say [LLMtop10:5:prompt manipulation] and [input:2:prompt injection], stochasticity, and the challenges that come with unstructured input and output. Is the LLM-generated search result a correct answer to our query? Does the result make correct use of references that actually exist? Were proper references retrieved? Each of these steps is subject to sets of LLM risks, and the associated breakdown helps us better understand this.

In our view LLM systems engineers can (and should) devise and field a more secure LLM application by carefully considering the risks in this document while designing, implementing, and fielding their own specific systems. In security, the devil is in the details, and we attempt to provide as much detail as possible regarding LLM security risks and some basic controls.

Acknowledgements

The authors thank the following people for their insightful comments and suggestions on earlier drafts of this document: Ross Anderson, Richard Danzig, David Evans, Dan Geer, Patrick McDaniel, Melanie Mitchell, Peter Norvig, Jim Routh, Neil Serebryany, and Stephen de Vries.

References

See the Berryville Institute of Machine Learning Annotated Bibliography for more commentary and references.

[1] BIML, *An Architectural Risk Analysis of Machine Learning Systems: Toward More Secure Machine Learning*, published under the Creative Commons, January 2020.

[2] McGraw, Gary, Richie Bonett, Harold Figueroa, and Victor Shepardson. "Security Engineering for Machine Learning," *IEEE Computer*, Volume 52, Number 8, pages 54-57. February 2019.

[3] McGraw, Gary, *Software Security*, Addison-Wesley, 2006. See chapter 5.

[4] European Union, Artificial intelligence act (legislation in progress), Brussels, Belgium, June 2023.

[5] Biden, Joe, Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence, White House, Washington, DC, October 20, 2023.

[6] Shumailov, Ilia, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. *"Model Dementia: Generated Data Makes Models Forget."* arXiv preprint arXiv:2305.17493 (2023).

[7] Alemohammad, Sina, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, Richard G. Baraniuk. "Self-Consuming Generative Models Go MAD." arXiv preprint arXiv:2307.01850 (2023)

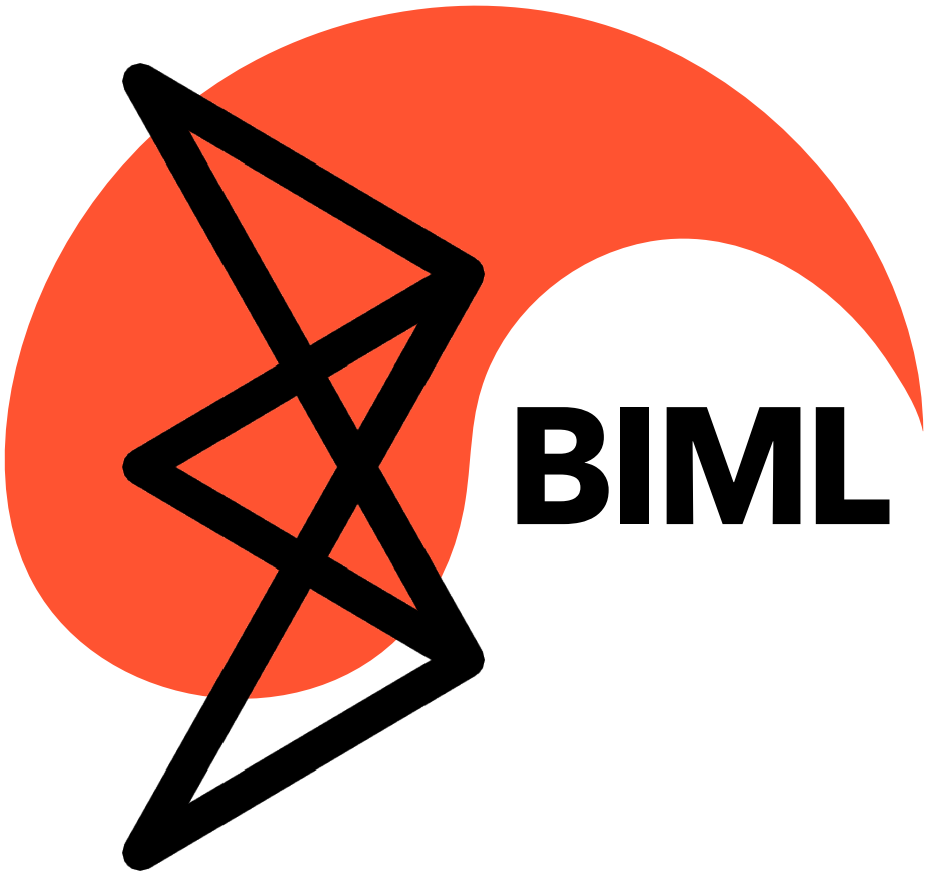
[8] Thiel, David. Stanford Internet Observatory, *Identifying and Eliminating CSAM in Generative ML Training Data and Models*, December 21, 2023.

[9] Aghakhani, Hojjat,Wei Dai, Andre Manoel, Xavier Fernandes, Anant Kharkar, Christopher Kruegel, Giovanni Vigna, David Evans, Ben Zorn, and Robert Sim. "TROJANPUZZLE: Covertly Poisoning Code-Suggestion Models." arXiv preprint arXiv:2301.02344 (2023)

[10] Nasr, Milad, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, Katherine Lee. "Scalable Extraction of Training Data from (Production) Language Models," arXiv preprint:2311.17035 (2023)

[11] Shumailov, Ilia, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, Ross Anderson. "Sponge Examples: Energy-Latency Attacks on Neural Networks." arXiv preprint arXiv:2006.03463 (2020).

[12] Ryan Burnell *et al.*, Rethink reporting of evaluation results in AI. *Science* 380, 136-138(2023). DOI:10.1126/science.adf6369


BIML