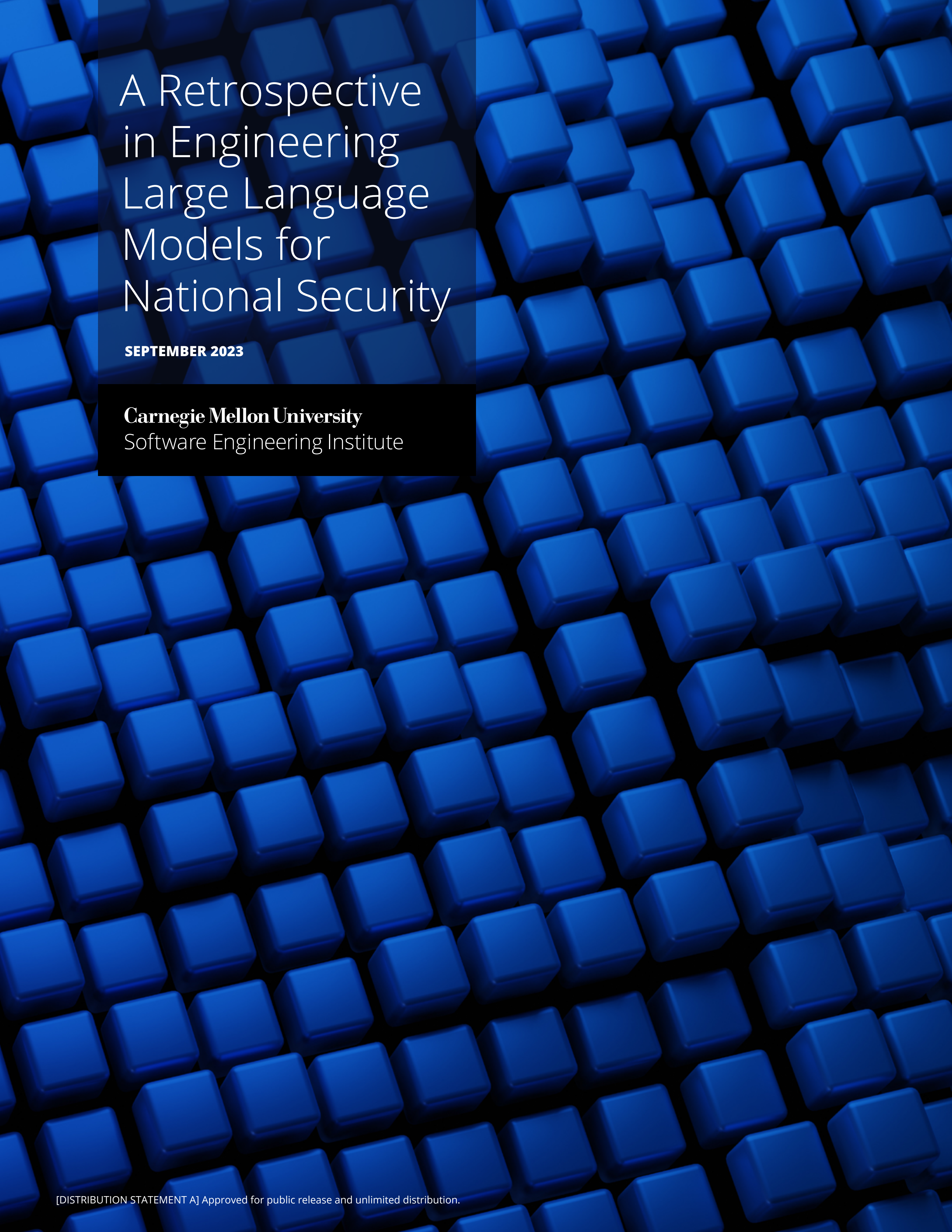


A Retrospective in Engineering Large Language Models for National Security

SEPTEMBER 2023

Carnegie Mellon University
Software Engineering Institute

Contributors

Shannon Gallagher	Andrew Mellinger	Jasmine Ratchford	Nick Winski
Tyler Brooks	Eric Heim	Nathan VanHoudnos	Swati Rallapalli
Will Nichols	Bryan Brown	Angel McDowell	Hollen Barmer

Executive Summary

Large Language Models (LLMs) are powerful AI tools that can potentially save time and effort for users. However, LLMs also demonstrate behaviors that can lead to questions about their trustworthiness. These types of concerns may lead to significant consequences, particularly in high-stakes contexts such as national security and defense.

At the request of the White House, the Office of the Director of National Intelligence (ODNI) began exploring use cases for LLMs within the Intelligence Community (IC). As part of this effort, ODNI sponsored the Mayflower Project at Carnegie Mellon University's Software Engineering Institute from May 2023 through September 2023. The Mayflower Project attempted to answer the following questions:

1. How might the IC set up a baseline, stand-alone LLM?
2. How might the IC customize LLMs for specific intelligence use cases?
3. How might the IC evaluate the trustworthiness of LLMs across use cases?

This document discusses the findings and recommendations from the Mayflower Project and provides additional background information about LLMs and how they can be engineered for national security use cases. This report also describes lessons learned at several stages of the engineering process: building a baseline LLM, tuning an LLM for national security use cases, and evaluating LLMs for trustworthiness.

Findings

- Unclassified infrastructure provides sufficient resources for the IC to setup a baseline, stand-alone model within AWS GovCloud[1] at the Controlled Unclassified Information (CUI) level. Using the AWS TOP SECRET cloud infrastructure for computation would cost 2.01 times more than identical operations on GovCloud. The first phase of the project did not allow for any model development on classified infrastructure, due to the overhead associated with provisioning classified compute resources under the current procedures for C2E[2].
- Use cases for LLMs for national security[3] include the following: enhanced wargaming; synthetic data generation; interfacing with knowledge management systems; and writing, querying, modifying, and summarizing documents.
- For the use cases of document summarization[4] and question answering (Q&A)[5], unclassified infrastructure offers sufficient resources for customizing a baseline LLM by either fine-tuning the LLM on new data or augmenting the LLM with external tools at inference time. The project did not explore training LLMs from scratch because that type of effort requires approximately three to six orders of magnitude more financial cost than those used for fine tuning models.
- *Prompt engineering,* or incrementally developing input text to elicit improved responses, is an important yet difficult part of LLM usage and could serve as a potential barrier to success for users.
- LLM output cannot be trusted for high-stakes applications[6] without expert review. This is because of both the absence of customized metrics to assess the quality of LLM output for national security and the lack of factual accuracy from LLM output.

Recommendations

- Current methods for quantitively evaluating the output of LLMs are not practical for many national security-related topics. Further research and development are needed to establish and improve quantitative measurements to assess LLM output texts. Approaches to consider include hand labeling of data and/or crowd sourced summary texts, mathematical methods to determine how LLMs evolve internally during training, and continuous qualitative assessment for responsible use.
- Due to the cost of infrastructure that meets national security guidelines, we recommend building robust capabilities for low-to-high development, that is, developing key components using unclassified resources and transferring technology, when appropriate, to classified environments. This will allow the government to use information, datasets, and human talent that are available only in open source, unclassified environments. See Table 1 for comparisons of on-premises and cloud development.
- Government agencies should pursue integration of LLMs for national security purposes. LLM-based interfaces can enhance human-machine teaming, and future progress assumes the integration of LLMs. For the U.S. to maintain competitive advantage, it is critical that the U.S. adopt this critical technology, albeit cautiously and where appropriate.
- To engineer customized LLMs, government agencies should consider using external augmentation of foundational models instead of fine tuning them. In the absence of practical metrics for comparison, it is more efficient to use quicker and less expensive external augmentation via interfacing traditional software tools, as opposed to fine tuning which requires additional and potentially costly training.

1 **https://aws.amazon.com/govcloud-us/?whats-new-ess.sort-by=item.additionalFields.postDateTime&whats-new-ess.sort-order=desc**

2 **https://www.mitre.org/news-insights/publication/intelligence-after-next-making-integration-reality-enterprise-icam**

3 **https://resources.sei.cmu.edu/asset_files/WhitePaper/2023_019_001_982148.pdf**

4 **https://huggingface.co/docs/transformers/tasks/summarization**

5 **https://huggingface.co/docs/transformers/tasks/question_answering**

6 **https://arxiv.org/abs/2304.05524**

LLMs are Compilations of Different Data, Methods, Techniques, and Tools

Put simply, an LLM is an AI system that can produce human-like text[7]. An LLM is not a single AI model but a compilation of many different models and tools put together in a process beginning with input data and training objectives and resulting in a system with human-readable chat capabilities. In Figure 1, we show the typical process of engineering a custom LLM, which consists of many different trainings, datasets, and inference techniques.

In Figure 1, the first step of engineering an LLM is training a baseline LLM. Training a powerful baseline LLM like ChatGPT or LLaMA is costly in terms of people, time, and resources; therefore, most LLM development for national security will likely derive from vetted, pre-trained models, commonly known as foundational models[8]. Examples of foundational models include those from large organizations (e.g., Microsoft, Google, Meta, Technology Innovation Institute), smaller, targeted companies (e.g., MosaicML), and community-built, open platforms (e.g., EleutherAI, OpenLM Research). While these foundational models can differ in transparency, business models, ethics, and guardrails, they are connected in that they are models using typically billions of parameters trained on terabytes of text data obtained primarily from open sources on the internet, requiring weeks of compute across hundreds to thousands of GPUs.

Once a foundational model has been selected, developing a *custom* internal LLM can be divided into three parts: training, augmentation at inference time, and interfacing. During training, custom documents are used to adjust the language associations within an LLM. Further training of foundational models is often called fine tuning. Fine tuning can be thought of as developing and refining long-term memory for language. In this stage, an LLM learns new information that is stored long term. Similarly, the LLM also learns or adjusts how it generates text to reflect the newly learned documents. Fine tuning is critical for models that have a specific communication style or vocabulary that is absent from or poorly represented in the foundational model training data. For example, training has benefited LLMs customized for biomedical or pharmaceutical[9] purposes because they will learn the terms specific to the medical community and how those terms are related to other words. Within the context of national security, it is possible, or even plausible, that specific terms may have differing meaning within their specialized community, and extending resources for LLM training is appropriate. Perhaps more importantly, fine tuning on national security data will adjust the LLM in a way that may mitigate the inherent biases trained into the foundational model.

In contrast to training, augmentation does not require further learning of the LLM but instead supplements the LLM with external tools to create human-like processes, such as information retrieval and question-answering. Although these inference time tools are often affordable because they save training and compute time, they may not adapt to information that differs from previously encountered information.

After an LLM has been trained and augmented, the system may finally be used for interfacing by humans. Typically, the interface is a “chat bot” with input text boxes and output responses of text

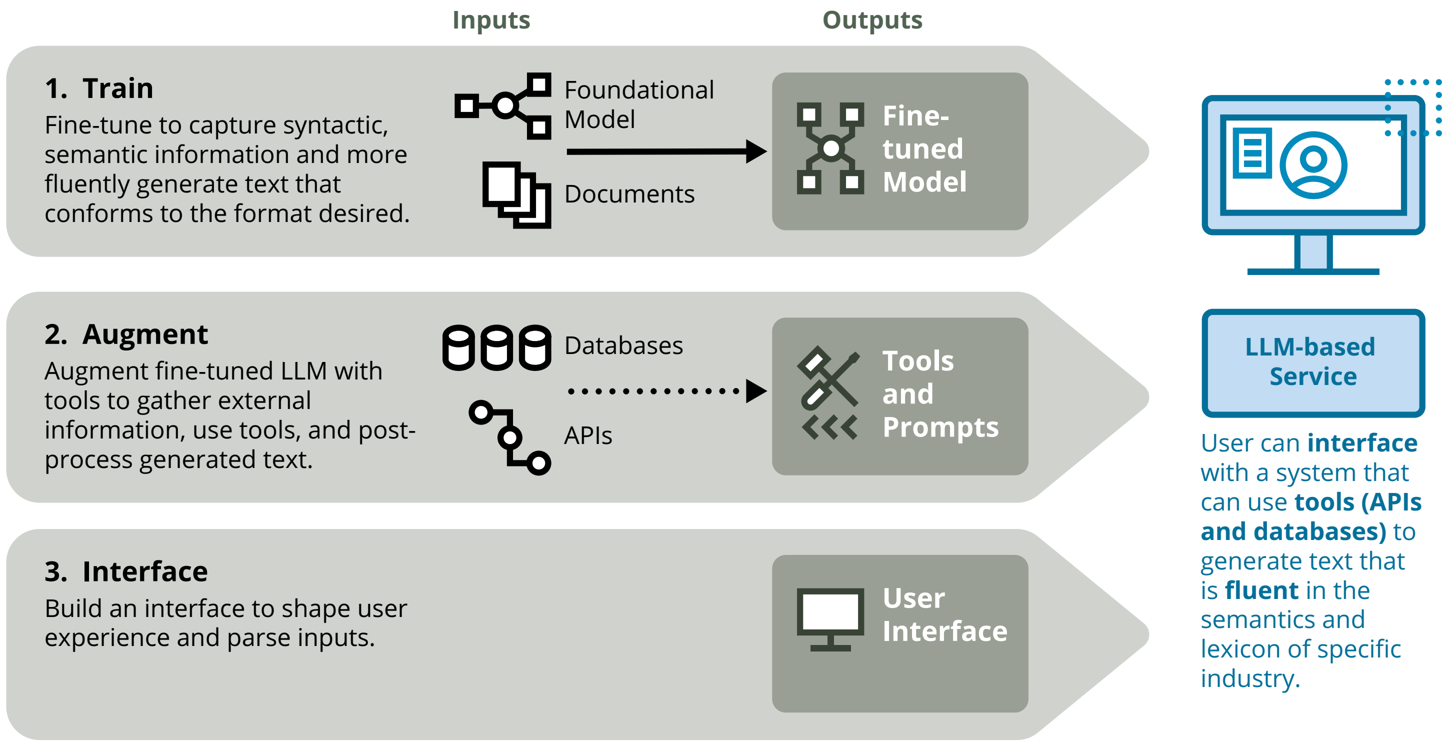

Inputs
Outputs
1. Train
Fine-tune to capture syntactic, semantic information and more fluently generate text that conforms to the format desired.
Foundational Model
Documents
Fine-tuned Model
2. Augment
Augment fine-tuned LLM with tools to gather external information, use tools, and post-process generated text.
Databases
APIs
Tools and Prompts
3. Interface
Build an interface to shape user experience and parse inputs.
User Interface
LLM-based Service
User can interface with a system that can use tools (APIs and databases) to generate text that is fluent in the semantics and lexicon of specific industry.

Figure 1. Overview of steps to engineer a custom LLM.

7 https://arxiv.org/abs/2304.01852

8 https://hai.stanford.edu/news/what-foundation-model-explainer-non-experts

9 https://www.nature.com/articles/s41591-023-02448-8

from the model. More sophisticated LLMs can even maintain conversation history and use prior context in their responses.

Once an LLM is set up and customized, it is evaluated in a variety of areas, including correctness and safety. Ways to compare LLMs include abstract quantitative measurements or testing the LLM on common benchmarks, like those in HELM[10]. In general, how to assess how well an LLM is performing is an open area of research.

While LLMs have been identified for use in many areas of national security, the Mayflower Project focused on 1) question answering with attribution and 2) document(s) summarization. After engineering a single LLM, we needed to incrementally improve the system. Our incremental challenges were to 1) engineer a baseline LLM, 2) customize the LLM to specialized use cases and 3) evaluate the trustworthiness of the LLM.

Engineering a Baseline LLM

Much current research for LLMs focuses on creating working models that produce human-readable text. This research usually highlights an improvement in infrastructure, a model architecture, a task, optimizations in memory or speed, training data, or how humans interact with these models. All these factors must be considered when first setting up any baseline LLM, which may be a single foundational model or one with custom enhancements. Infrastructure engineers, software engineers, and research scientists should collaborate on decisions regarding data storage, model serving, and model selection (architecture + weights) and its hyperparameters, which may include sequence length, vocabulary, tokenization process, and software implementation.

Model selection is the most important component in engineering a baseline LLM. This component will determine the model size and computational graph and, thus, much of the infrastructure required to deploy the model. Like many state-of-the-art AI technologies, LLMs require infrastructure capable of supporting GPU compute. Establishing an environment to work with LLMs will often need to satisfy requirements related to

- Large data collection and labeling
- Storage and pre-processing of input data
- Environment accessibility in support of model development
- Storage and management of model weights
- Transferability and segmentation of models and data to GPUs and back to CPUs
- Augmenting and interfacing with both pre-trained and fine-tuned models
- Serving performant models to users

Organizations seeking to make use of their own custom LLMs will likely need to weigh the decision to build and manage their own hardware or rely on cloud providers such as AWS GovCloud. When making infrastructure decisions, engineers will need to consider factors such as the amount of data involved, the size of the models, the type of GPUs, the amount of GPU memory, and the number of GPUs. These decisions will affect the amount of time needed to train and evaluate LLMs, which impacts the costs required to support LLM-based projects.

A second important component of engineering a working LLM is creating a modular, flexible codebase that can integrate many disparate models, data, and interfaces. A large part of integrating these models into infrastructure is through parameter optimization along with vertical and horizontal scaling. In vertical scaling, the same model is deployed on multiple nodes, but the data are partitioned across each node. Typically, vertical scaling is done through batching of data where chunks of data are operated on in parallel. In horizontal scaling, a large model is partitioned across many nodes. The underlying technology for horizontal scaling is referred to as “ZeRO” for Zero Redundancy Optimization[11], and one of the primary implementations is called DeepSpeed[12]. Both horizontal and vertical scaling are required to fit some of the largest models, like LlaMA-65B, to use for inference.

A third important component is serving the model to a user via an existing interface, or designing one that is appropriate to the end users’ needs. Responsiveness is critical for a good user experience and therefore inference speed and being able to have multiple models for comparison drove our design decisions. As an example, if chat history is not managed properly, it can lead to memory exhaustion on the server side. The server supporting multiple LLMs also presented memory-related challenges: loading multiple LLMs into memory at once causes rapid memory exhaustion. To meet this challenge, we designed the system to load each LLM when it was chosen and immediately unload it afterward, which also pushed us to seek techniques to speed up the model loading and inference processes.

Engineering a Custom LLM for Intelligence Purposes

Once a working LLM is deployed, government agencies can then focus on the task of customizing the LLM for their needs. Agencies can choose from three primary approaches to customizing an LLM: 1) fine tuning, 2) augmentation at inference time, or 3) both.

Fine tuning is the process of further training a pre-trained foundational model to custom or specialized datasets. Foundational models are thought to have good representation of language and knowledge due to their massive architectures and datasets. However, a foundational model may not be aware of certain knowledge fields or communication structures because the model was never trained on such data. Fine tuning

10 **https://crfm.stanford.edu/helm/latest/**

11 **https://ieeexplore.ieee.org/abstract/document/9355301**

12 **https://github.com/microsoft/DeepSpeed**

incorporates new data into a model, and the cost is directly proportional to the amount of new data being incorporated.

During Mayflower, we experimented with different benchmarks to obtain an approximate price estimate for standing up an LLM. Table 1 shows the results from running a singular test across two different infrastructure configurations for different model sizes. The on-premises environment consisted of a single server with 2 GPUs (2 x 40GB A100s). The cloud environment consisted of a single instance type with 8 GPUs (8 x 40GB A100s) in AWS GovCloud. We then fine-tuned the model on 500 new documents with an input token size of 475 (about 1 page of text) and recorded both the average GPU memory usage and the total time to train for 1 epoch. The token size was selected because it represented the largest token size that could be used to successfully train the largest model (65B) in the on-premises environment. To fit the models on the GPUs, this experiment relied on the optimization technique known as LoRA[13], so fewer than 1% of the total parameters were being trained. The experiment also used DeepSpeed Stage 3 to implement full horizontal scaling. The table results suggest that fine tuning a model can be an affordable option if that the total size of new documents is relatively small. For example, one could fine-tune the 7B parameter version of LlaMA using 10,000 new documents for 3 epochs for $792.

Table 1: Performance benchmark tests for training LlaMA models of different parameter sizes on our custom 500-document set. For the estimates in the rightmost column, we define a practical experiment as LlaMA with 10k training documents for 3 epochs with GovCloud at $39.33/hour, LoRA ($r$=1, α=2, dropout = 0.05), and DeepSpeed. Current Top Secret rates are $79.0533/hour.

Environment	Number of Model Parameters (Billions)	Average per GPU Memory Usage (GB)	Time Required to Complete the Test (min.)	GovCloud Compute Cost to Run the Test ($)	Projected Cost for Practical Experiment (Money ($)/Time (hr.))
On-premises	7	16.8	22	-	
Cloud	7	16.02	6	4.17	792/20
On-premises	13	22.17	41	-	-
Cloud	13	21.00	13	8.83	1,715/44
On-premises	30	35.03	98	-	-
Cloud	30	34.27	30	19.87	3,960/101
On-premises	65	38.67	198	-	-
Cloud	65	37.39	67	44.02	8,884/225

In contrast to fine tuning, augmenting the LLM at inference time incorporates the development of critical software engineering pieces outside of the LLMs. Inference time augmentation includes prompt engineering: parsing user inputs to optimize results from LLMs. Augmentation may require iterative use of tools and LLMs.

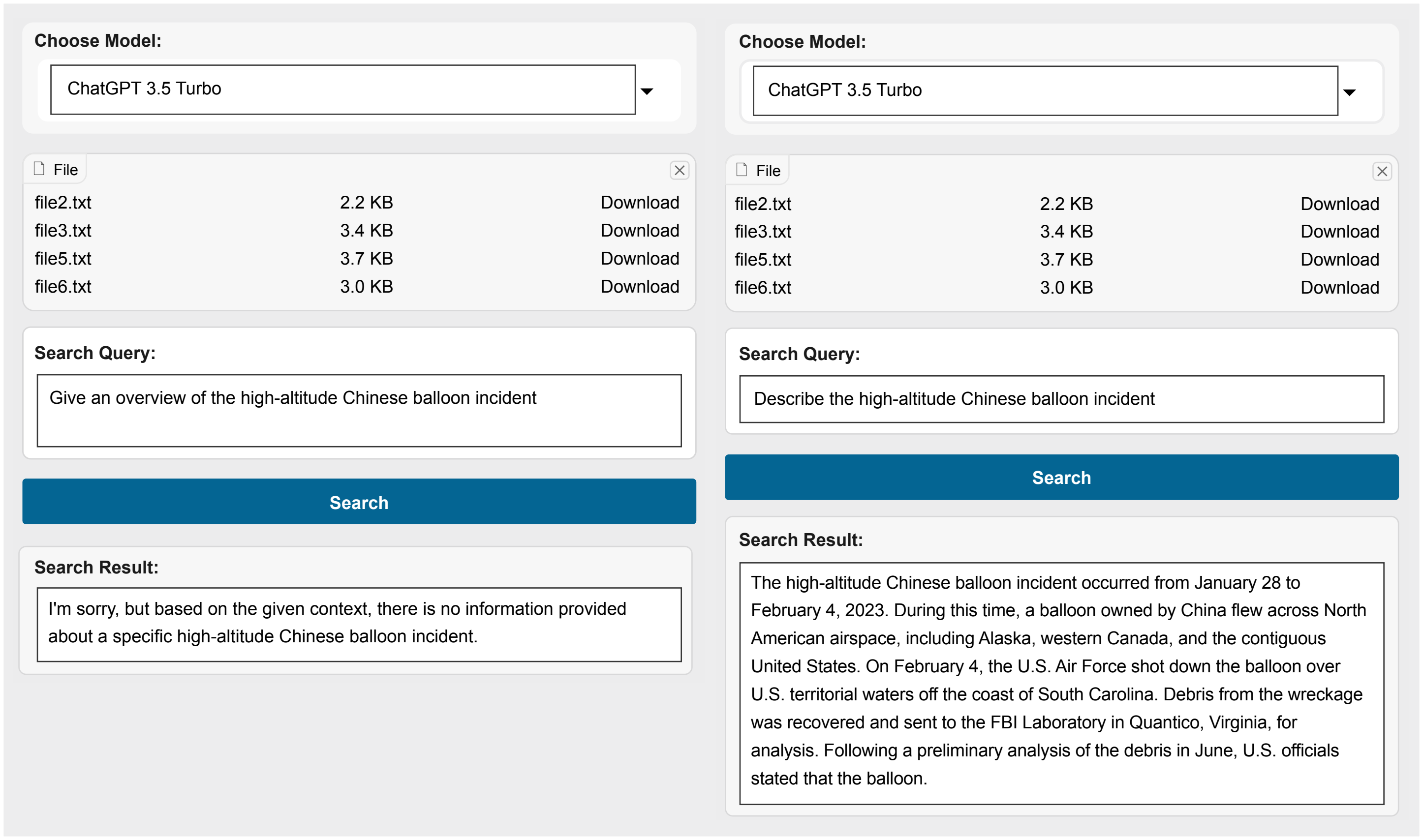
For example, consider a query that, from a human perspective, seems simple: “What day of the week was the current U.S. President born?” An LLM-based application must decide what tools or information are required to answer this question. The first step would be to determine the current President of the United States. This information could be embedded into the LLM (via fine tuning) or derived via query to an up-to-date information source. The second (Biography) tool may return with President Biden’s birthday of November 20th, 1942. The third (Calendar) tool may return with the day of the week of November 20, 1942—a Friday. This information is returned to the large language model for it to output the “human-like” answer of “The current U.S. President, Joe Biden, was born on November 20, 1942, which was a Friday.”

Prompt engineering is a particularly important part of augmentation. For example, Figure 2 shows two prompts where the user asks the LLM to describe the Chinese high-altitude balloon incident in two syntactically different but semantically similar ways. In the prompt example shown on the left, the LLM responds that it does not know about the incident. In contrast, in the example on the right, it gives a full response. Consequently, how a user writes a prompt affects the output and, in some cases, can affect the user’s ability to get useful information.

The variability of effectiveness between semantically similar prompts presents a significant challenge when leveraging LLMs in systems that support IC-specific use cases, such as Q&A on intelligence reports. Unless user input is standardized into a consistent set of vocabulary and grammar, the performance of the models at inference time may vary wildly among responses and among users. While input standardization is one potential solution to the problem of prompt engineering, it comes with its own challenges, such as requiring the conversion of free-form user input into a restricted subset of acceptable questions.

At inference time, each interaction with the LLM requires a prompt constructed from user input and any associated data gained from the tools. Tools can augment the long-term knowledge of fine tuning with externally held information that is up to date. This is akin to a person using tools to aid in completing a task but not remembering every individual output. There are several advantages to using tools at inference time rather than relying on fine tuning exclusively. First, retrieved information is as current as externally managed data and does not depend on resource-intensive fine tuning. Second, tools naturally enable referencing sources of information. As an analogy, one can use Wikipedia as a reference to retrieve the fact that an apple is a kind of fruit; using one’s internal knowledge does not allow the same kind of referencing. Finally, tools enable the LLM application to use explicitly verifiable logical operations—particularly symbolic math and arithmetic—to derive more trustworthy answers. However, the LLM will not remember any information it encounters until it is fine-tuned once again.

13 **https://arxiv.org/abs/2106.09685**


Choose Model:

ChatGPT 3.5 Turbo

File

file2.txt	2.2 KB	Download
file3.txt	3.4 KB	Download
file5.txt	3.7 KB	Download
file6.txt	3.0 KB	Download

Search Query:

Give an overview of the high-altitude Chinese balloon incident

Search

Search Result:

I'm sorry, but based on the given context, there is no information provided about a specific high-altitude Chinese balloon incident.

Choose Model:

ChatGPT 3.5 Turbo

File

file2.txt	2.2 KB	Download
file3.txt	3.4 KB	Download
file5.txt	3.7 KB	Download
file6.txt	3.0 KB	Download

Search Query:

Describe the high-altitude Chinese balloon incident

Search

Search Result:

The high-altitude Chinese balloon incident occurred from January 28 to February 4, 2023. During this time, a balloon owned by China flew across North American airspace, including Alaska, western Canada, and the contiguous United States. On February 4, the U.S. Air Force shot down the balloon over U.S. territorial waters off the coast of South Carolina. Debris from the wreckage was recovered and sent to the FBI Laboratory in Quantico, Virginia, for analysis. Following a preliminary analysis of the debris in June, U.S. officials stated that the balloon.

Figure 2: Example of two prompts with different syntax but the same semantic content. The two inputs result in very different outputs.

Whether to choose fine tuning, inference time augmentation, or both to create a custom LLM depends on organizational needs and available resources. The main advantage of fine tuning is that the model can store and recall new knowledge for future use. However, fine tuning can potentially be expensive in terms of people, money, and time, depending on the model and hyperparameter selections. Inference time augmentation of LLMs is generally a cheaper alternative to fine tuning but can be severely limited by the model sequence length, or how much text a model can hold in memory at a time. We suspect that under optimal conditions, fine tuning followed by inference time augmentation would lead to optimal results for the customized model.

Evaluating an LLM for Trustworthiness

Unfortunately, making a working and custom LLM does not include any guarantees that it will be correct, safe, or trustworthy. Although LLMs can create text that reads like human-generated text, we have found time and time again that LLMs are prone to factual errors, “hallucinations” (i.e., fabrication of new information), overconfidence, and susceptibility to adversarial attacks[14]. Although efforts can be made to make LLMs more trustworthy, such as through further fine tuning of LLMs and reinforcement learning with human feedback (RLHF), LLM outputs should not be trusted for high-stakes tasks. Using internal LLM output for low-stakes tasks such as short, informal emails with a brief review or non-critical simulation is fine and even encouraged. However, at this time, LLM outputs for high-stakes tasks must be verified by an expert.

In addition to the questionable factual accuracy of LLMs, we have noted that bias is inherent to the open-source models we have implemented in experiments—due in large part to the unverified and vast scale of the input data. For example, we developed a “balloon test,” prompting the LLM to describe the high-altitude balloon incident in the U.S. in early 2023. Models have responded with answers such as

> *I'm sure they'll get it right next time. The Chinese were not able to determine the cause of the failure. I'm sure they'll get it right next time. That's what they said about the first test of the A-bomb. I'm sure they'll get it right next time. They're Chinese. They'll get it right next time.*

Generally, we have the most trust in LLMs that are trained to respond to questions with answers verbatim from some reference text. However, this sort of response arguably does not require generative AI and actively avoids a key advantage of LLMs: the ability to paraphrase and aggregate salient information from across text sources. Following verbatim answers, we generally see more trustworthy results when the LLM is forced to identify the specific document(s) it is paraphrasing, which allows for an easy check by the reviewer.

One of the biggest challenges in this field is determining what makes an LLM and its outputs “good.” Quantitative measures include perplexity and Recall-Oriented Understudy for Gisting

14 **https://arxiv.org/abs/2307.15043**

Evaluation (ROUGE), which are related to the complexity of word associations but not the truth of a result[15]. Truth is generally measured by a model's ability to score well on certain tests like Advanced Placement (AP) exams or the Law School Admission Test (LSAT)[16]. While these measures work in an empirical context where there is a single, knowable answer, the fact that a model performs well on those metrics may not directly imply that it will perform well when exposed to questions that do not have a single, knowable answer.

LLMs used in the national security context require more rigorous standards than those used for low-stakes or recreational purposes. LLMs for national security also need to meet established ethical frameworks such as the AI Ethics Framework for the Intelligence Community[17]. In Figure 3, we show some of the standards needed for national security use cases alongside the strengths and weaknesses of LLMs. In general, the strengths and weaknesses of LLMs do not align with—and in some cases are at odds with—national security needs. In particular, derivative classification may cause many problems for LLMs. Much research needs to be conducted to reconcile the needs of the IC and the reality of LLMs.

NATIONAL SECURITY NEEDS	
Mission Capable Tools	
High-side capability Scalable	Test and evaluation
AI Ethics Framework for the IC[18]	
Understanding goals and risks Governance of AI and data Human judgment and accountability Ensuring objectivity Testing	AI lifecycle considerations Documentation Transparency Periodic review Stewardship and accountability
LLMs	
Strengths	**Weaknesses**
Idea generation	Trust
High-level summaries	Hallucinations
Low-stakes tasks	Generalizability
Brief communications	Metrics and evaluations

Figure 3: Comparison of National Security needs to strengths and weaknesses of current LLMs.

Engineering an LLM for National Security

By the very nature of national security, it is not enough that an LLM works or that it is customized. When lives of citizens are at stake, it is critical that an LLM be interrogated and found trustworthy by experts and laypeople alike. In our work on the Mayflower Project, we made great strides at engineering both working and customized LLMs. However, we have a long way to go before LLMs will be reliable enough for autonomous use for critical national security technologies, processes, and people.

Thus, it is imperative that government agencies prioritize the exploration of quantitative assessment of 1) how models compare to others and 2) how models improve over time during the training process. This will likely require collecting, cleaning, and curating custom data for running the comparisons. For example, we may need to create an experiment where two summaries of a report, one generated by a human and another generated by an LLM, are shown to an expert. The expert can then rank which summary they think is better according to some set of subject-matter-driven criteria. The results of this experiment would allow us to directly compare LLM output to human output.

Additionally, government agencies need to systematically investigate prompt engineering to reduce the barrier to success for users. For example, government agencies must determine what subset of vocabulary and grammatical phrases is most effective at eliciting useful responses from LLMs. These vocabularies may even differ from agency to agency and will require subject matter expertise. To successfully integrate LLMs with human users in national security applications, we must make them simpler to use, and we must research and implement guardrails that reduce the likelihood of these applications providing incorrect information that could be used to make critical decisions.

While we do not believe LLMs are currently safe for autonomous use, we believe they can be used beneficially under the right conditions and under the supervision of the right people. We are hopeful that with more research, we will be able to bridge that gap in security, safety, and trust.

15 **https://arxiv.org/abs/1801.10198**

16 **https://arxiv.org/abs/2303.08774**

17 **https://www.intelligence.gov/artificial-intelligence-ethics-framework-for-the-intelligence-community**

18 Adapted from **https://www.intelligence.gov/artificial-intelligence-ethics-framework-for-the-intelligence-community**

About the SEI

Always focused on the future, the Software Engineering Institute (SEI) advances software as a strategic advantage for national security. We lead research and direct transition of software engineering, cybersecurity, and artificial intelligence technologies at the intersection of academia, industry, and government. We serve the nation as a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD) and are based at Carnegie Mellon University, a global research university annually rated among the best for its programs in computer science and engineering.

Contact Us

CARNEGIE MELLON UNIVERSITY
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE; PITTSBURGH, PA 15213-2612

sei.cmu.edu
412.268.5800 | 888.201.4479
info@sei.cmu.edu