SYM-AM-25-313

Excerpt from the Proceedings

of the

Twenty-Second Annual Acquisition Research Symposium and Innovation Summit

Wednesday, May 7, 2025 Sessions
Volume I

Test and Evaluation of Large Language Models to Support Informed Government Acquisition

Published: May 5, 2025

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

Test and Evaluation of Large Language Models to Support Informed Government Acquisition

Jaganmohan Chandrasekaran—is a Research Assistant Professor at the Sanghani Center for AI and Data Analytics, Virginia Tech. His research is at the intersection of software engineering and artificial intelligence (AI) and focuses on developing frameworks, methods, and tools to address the multi-faceted challenges in the test and evaluation of AI-enabled software systems across its life cycle. He earned an MS and a PhD in Computer Science from the University of Texas at Arlington. [jagan@vt.edu]

Brian Mayer—is a Research Scientist at the Sanghani Center. Mayer works with sponsors and collaborators to help apply Sanghani faculty research to solve government, corporate, and societal problems. Mayer has helped deploy several long-term government-funded research projects to government sponsors. Mayer's research interests are in the areas of decision support, forecasting, intelligence, and human-guided systems. Mayer received an MS and BS in Industrial and Systems Engineering from Virginia Tech. [bmayer2@vt.edu]

Heather Frase—is the head of Veraitech, a Senior Advisor for Testing & Evaluation of AI at Virginia Tech's National Security Institute, and Program Lead for the AI Risk and Reliability working group at MLCommons. Her diverse career has spanned significant roles in defense, intelligence, and policy, involving projects ranging from AI assurance and policy to missile defense, drug trafficking prevention, and financial crime analysis. She also serves as a member of the Organisation for Economic Co-operation and Development (OECD) Network of Experts on AI and on the board of the Responsible AI Collaborative, which researches and documents AI incidents. [hnfrase@vt.edu]

Erin Lanus—is a Research Assistant Professor in the Intelligent Systems Division of the Virginia Tech National Security Institute and affiliate faculty of Computer Science at Virginia Tech. Her research focus is testing and evaluation of consequential AI. She applies her background in combinatorial testing to developing metrics for measuring coverage of datasets and algorithms for constructing test sets as well as identifying novel applications of combinatorial testing to AI assurance. She earned a BA in Psychology and a PhD in Computer Science both from Arizona State University. [lanus@vt.edu]

Patrick Butler—is a Senior Research Associate at the Sanghani Center. The main thrust of his research focuses on the IARPA funded EMBERS project that uses open-source indicators, such as tweets, news, blog, weather, etc., to forecast population level events such as civil unrest, elections, and epidemics. He graduated with a PhD in Computer Science from Virginia Tech in 2014 during which time he was awarded a SMART Fellowship. His dissertation was in the area of using knowledge discovery for intelligence analysis. [pabutler@vt.edu]

Stephen Adams—is a Research Associate Professor and Assistant Director of the Intelligent Systems Division in the Virginia Tech National Security Institute. He received an MS in Statistics from the University of Virginia (UVA) in 2010 and a PhD from UVA in Systems Engineering in December of 2015. His research focuses on applications of machine learning and artificial intelligence in real-world systems. He has experience developing and implementing numerous types of machine learning and artificial intelligence algorithms. His research interests include feature selection, machine learning with cost, transfer learning, reinforcement learning, inverse reinforcement learning, anomaly detection, explainable AI, and probabilistic modeling of systems. His research has been applied to several domains including fraud detection, activity recognition, prognostics and health management, cyber-physical systems, psychology, cybersecurity, internet-of-things devices, and sports analytics. [scadams21@vt.edu]

Jared Gregersen—currently serves as Lead Research Systems Engineer at the Virginia Tech National Security Institute. His experience includes working for more than 20 years within information technology providing infrastructure, application and database, and software development and 3D visualization. His roles have included positions within higher education, the energy sector, fortune 500 companies, healthcare, and he has supported local and federal government. His comprehensive expertise provides a unique capability and perspective in solving technical challenges and designing creative solutions for complex problems. His interests include research related to AI agents, and their implementation and interaction with information technologies and systems. [jaredgregersen@vt.edu]

Naren Ramakrishnan—is the Thomas L. Phillips Professor of Engineering at Virginia Tech and leads AI and machine learning for Virginia Tech's Innovation Campus. He directs the Sanghani Center for AI and Data Analytics and the Amazon-Virginia Tech Initiative in Efficient and Robust Machine Learning. Ramakrishnan's research interests span data science, forecasting, urban analytics, recommender systems, and computational epidemiology. He is a fellow of the Association for Computing Machinery (ACM), the American Association for the Advancement of Science (AAAS), and the Institute of Electrical and Electronics Engineers (IEEE). [naren@cs.vt.edu]

Laura Freeman—is a Research Professor of Statistics and serves as the Deputy Director of the Virginia Tech National Security Institute. She is also the Assistant Dean for Research for the College of Science. Her research leverages experimental methods in the domains of cyber-physical systems, data science, artificial intelligence, and machine learning to address critical challenges in national security. Previously, Freeman was the Assistant Director of the Operational Evaluation Division at the Institute for Defense Analyses. Freeman has a BS in Aerospace Engineering, an MS in Statistics, and a PhD in Statistics, all from Virginia Tech. [lamorgan@vt.edu]

Abstract

As large language models (LLMs) continue to advance and find applications in critical decision-making systems, robust and thorough test and evaluation (T&E) of these models will be necessary to ensure we reap their promised benefits without the risks that often come with LLMs.

Most existing applications of LLMs are in specific areas like healthcare, marketing, and customer support and thus these domains have influenced their T&E processes. When investigating LLMs for government acquisition, we encounter unique challenges and opportunities. Key challenges include managing the complexity and novelty of Artificial Intelligence (AI) systems and implementing robust risk management practices that can pass muster with the stringency of government regulatory requirements. Data management and transparency are critical concerns, as is the need for ensuring accuracy (performance). Unlike traditional software systems developed for specific functionalities, LLMs are capable of performing a wide variety of functionalities (e.g., translation, generation). Furthermore, the primary mode of interaction with an LLM is through natural language. These unique characteristics necessitate a comprehensive evaluation across diverse functionalities and accounting for the variability in the natural language inputs/outputs. Thus, the T&E for LLMs must support evaluating the model's linguistic capabilities (understanding, reasoning, etc.), generation capabilities (e.g., correctness, coherence, and contextually relevant responses), and other quality attributes (fairness, security, lack of toxicity, robustness). T&E must be thorough, robust, and systematic to fully realize the capabilities and limitations (e.g., hallucinations and toxicity) of LLMs and to ensure confidence in their performance. This work aims to provide an overview of the current state of T&E methods for ascertaining the quality of LLMs and structured recommendations for testing LLMs, thus resulting in a process for assuring warfighting capability.

Keywords: Large Language Models, Test and Evaluation, Government Acquisition, Generative Artificial Intelligence, Benchmarking

Introduction

Large language models (LLMs), a subset of generative AI, have demonstrated the potential to accomplish diverse activities with minimal or no human intervention. As a result, LLMs have found utility across domains, and recent developments have indicated there is an increasing interest among people across various domains in adapting and trying to leverage LLMs in their activities. However, successful adaptation of LLMs is contingent upon the ability to thoroughly evaluate and ensure these systems perform as expected after adaptation.

LLMs, similar to AI/ML systems, are data-intensive software systems. Unlike traditional software systems where the core functionality is encoded by a human (referred to as source code), the data-intensive systems derive their decision logic from a training dataset; this decision logic is commonly referred to as a model. An LLM, a type of deep learning system, is

fundamentally a language model trained on a vast amount of training data, capable of performing a variety of tasks. Furthermore, these systems exhibit non-determinism, are stochastic, and have a decision logic that is not easily understandable to humans (opaque) Moreover, both the data and the algorithm used to train the model influence its behavior. Thus, traditional T&E methods and practices, which primarily focus on assessing the functional correctness of a deterministic software system with pre-defined test inputs and outputs, might not sufficiently evaluate the LLM. Additionally, given the characteristics of the LLM—interaction via natural language, ability to perform a variety of tasks, and continual learning—necessitates extra care and additional assessments when it comes to their evaluation. Therefore, a comprehensive assessment of LLMs is essential to harness its benefits successfully.

From an acquisition perspective, numerous LLMs are currently available to practitioners. In addition to addressing warfighter requirements, acquisition specialists and T&E professionals have to make sure that all acquired and deployed LLMs are effective, safe, and reliable. The deployment of LLMs in government settings raises significant concerns regarding operational safety, data privacy, and the potential for inadvertent exposure of sensitive information, to name a few. This paper aims to present a discussion on the current practices in the T&E of LLMs to better inform acquisition professionals when seeking to acquire these tools. The ideas are presented based on the findings from the survey of academic literature and industrial best practices for T&E of LLMs.

LLMs can complete many different complex tasks, which increases the difficulty and necessary variability in testing. Due to the versatility of LLMs, T&E activities generally involve running a range of evaluations on a range of tasks (e.g., question and answer, information retrieval, text classification, and summarization) to evaluate a range of characteristics (e.g., understanding, reasoning, generation, fairness, security, and toxicity).

While LLMs can range in complexity, this paper is focused on based models but is applicable regardless of the model's size or openness. Sometimes, LLMs are single-base models (e.g., BERT, GTP 4.0, etc.). However, frequently those based models in combination with other AI or systems are also considered LLMs. This is often seen in LLMs that have added "guardrails" that provide safety and security. Models can also vary in size and whether they are open, closed, or somewhere in between. Typical LLMs can range in size from millions to even trillions of parameters. The number of parameters can have a significant impact on an LLMs capabilities and quality. Open LLMs are those whose training data, code, architecture, and model weights are fully open to the public. In closed LLMs none of those are available to the public and may not be available to the deployer. There are also partially open LLMs where only some of that content is available.

Ensuring that an LLM can be a reliable and safe solution means it must be able to provide accurate results, robust to many different scenarios, and resilient to variable and potentially hostile inputs. We categorize the LLM acquisition scenarios along two key dimensions: 1) the information about the LLM that the acquisition team has access to, such as training data, code, architecture, and model weights, and 2) how often or how many times the team can carry out T&E activities to assess the quality of the LLM (test scheduling).

Different types of information (model artifacts) access at the time of acquisition:

- White box: LLM model is developed in-house (i.e., by the government) so T&E personnel have access to all the model's artifacts
- Grey box: An off-the-shelf pre-trained LLM, so T&E personnel do not have access to training information (data, hyperparameters, code, or model weights); however, they do have access to data and other artifacts used in fine-tuning the LLM.

- Black box: An off-the-shelf pre-trained LLM without modifications (no-fine-tuning) so there is no access to training information (data, hyperparameters, code, or model weights).

Different test schedules:

- Continuous testing: The ability to perform T&E activities throughout the LLM's life cycle
- Periodic testing: No testing access during development but the ability to evaluate during fine-tuning
- One-time testing: Testing is limited to evaluating the final model output and performance

Combining these two dimensions, we identify three use case scenarios:

- Use Case 1: In-house development—White-box and the ability to perform continuous testing.
- Use Case 2: Fine-tuning an off-the-shelf LLM—Grey-box, and periodic testing.
- Use Case 3: Off-the-shelf LLM (as-is)—Black-box and one-time acceptance testing.

The acquisition team's strategy for evaluating an LLM depends on how and what access they have to the LLM and its artifacts. Next, a detailed description of three use cases is presented, which will serve as practical examples to facilitate our discussion in the subsequent sections.

Use Case 1 White box, in-house development, continuous testing, software only

A department completed an in-house effort to develop a software application for processing free-text records about financial transactions. The software application's task is to identify named entities in a user-provided collection of records, extract relationships between entities, do entity resolution, and provide network graphs of the relationships. The LLM is a key component contributor to performing the named entity recognition and relationship extraction. Separate components of the software application perform the entity resolution and the network graphs. Additionally, the application has a user interface with a quality feedback mechanism. The contract for the software application includes creating a new LLM and will provide the department with the training data and the model weights.

Use Case 2 Grey box, off-the-shelf LLM that is fine-tuned in-house, periodic testing.

A department wants to have an application to help its staff complete internal documents that traditionally require a lot of manual labor. These documents contain fixed fields, short free-text, and long blocks of free-text. The fields will be completed by extracting information from user provided records and questions responses. To build the application, the department has found a quality LLM developed by another organization in its agency. The department does not have access to the weights or training data of the LLM. They will fine-tune the LLM using their own data and development team. The department has funded retraining of the LLM every 6 months.

Use Case 3 Black box, off-the-shelf LLM, one-time acceptance testing, LLM system within hardware.

An agency is purchasing small drones for searching natural disaster sites. The drones can be commanded by text messages from the operators. While the text can be manually typed, operators are more likely to send texts created through verbal transcription. The drones have an LLM that converts text messages into commands that they can implement. The agency has no information about the specific LLM model or the data used to develop it. Because this is a Consumer-Off-The-Shelf (COTS) system, the agency cannot alter the system or its internal LLM but will test the drones before committing to a large purchase.

The remainder of the paper is structured as follows: We first present the current T&E practices, including an overview of the steps in testing LLMs. This is followed by a discussion on establishing the scope of LLM evaluation by categorizing the LLM purposes as capabilities and properties as qualities to outline "what to test" in a test plan. We then discuss the limitations of current practices and finally present our concluding remarks and directions for future research.

Current T&E Practices

An Overview of Testing of LLMs

Next, we will provide an overview of the steps required for the T&E of an LLM. Testing an LLM typically follows the procedure shown in Figure 1. For Use Case 1, this activity starts at the end of in-house model development. For the other two use cases, this series of steps begins either when the LLM is obtained as an off-the-shelf model or after fine-tuning the LLM (applicable only to Use Case 2).

Figure 1. Overview—Testing LLM

Step 1: Installing Prerequisites: The first step is to install all the required software packages and dependencies. This is useful for handling sensitive information such as Application Programming Interface (API) keys and configuration settings. The next step is to download the necessary libraries, which will be used for various activities such as data processing, API interaction, and environment management tasks.

Step 2: Loading the LLM: The procedure to load the LLM will vary depending on the specific LLM. The two common distribution modes for LLMs are 1) host the LLM locally and 2) API-based access. For a locally hosted LLM, load it using the appropriate code. For example, an LLM developed in-house or an open-source pre-trained LLM like Llama2 that can be downloaded and executed locally. If the LLM is accessed via an API, establish the connection using an API key. For example, OpenAI's GPT3.5 Turbo can be accessed via an API key.

Step 3: Loading Test Dataset: When evaluating an LLM, the test dataset will be specific to the task the LLM is asked to perform and the desired evaluation methodology (described further under Step 5: Assessment/Evaluation). LLMs can perform many different tasks (Chang et al., 2024). Some common tasks are:

- Text Classification: assigning a label or class to a given text
- Sentiment Analysis: identifying the emotional category/state of the text
- Named Entity Recognition (NER): locating and classifying named entities mentioned in text
- Multiple Choice Question (MCQ): responding to a multiple-choice question with the correct answer
- Question and Answer (Q&A): responding to an open-ended question with an appropriate answer
- Text Completion: providing words to proceed with a sequence of text
- Information Retrieval (IR): identifying relevant information to a prompt
- Summarization: summarizing, reformulating, or condensing text meaningfully based on a prompt (Allahyari et al., 2017; Nguyen et al., 2024)

Furthermore, an LLM can be evaluated using either or both of the following types of datasets:

- Established test dataset: The tester can utilize established or published datasets from the AI community.
- Custom test dataset: The tester can create a specific dataset that assesses cases or scenarios tailored to their particular use case. This custom dataset can be hosted locally as a CSV or JSON file and used to evaluate the model's capability based on specific criteria.

Step 4: Prompting: Unlike traditional software systems, user interaction with an LLM is primarily with a text input called a "prompt." A prompt is typically natural language text but can include code or pseudo code. A prompt is a set of instructions that informs the LLM about the user's request. It comprises the input, desired LLM behavior, and any other instructions that users expect the LLM to follow while processing their request. Usually, a prompt consists of three main components:

- User role: User's query
- System role: Instructions on how the model should behave or respond
- Assistant role: Provides a method for giving examples of what a response should look like.

When testing an LLM, the prompt you use and the characteristics of the prompt will be specific to the task you are asking the LLM to perform. Thus, creating effective prompts is crucial for better engagement with the LLM.

Prompting strategies are techniques used to guide language models in generating desired responses. Three common strategies are (Schulhoff et al., 2024; Wei et al., 2022):

1. Zero-Shot Prompting: involves providing no prior examples to the model.
2. Few-Shot Prompting: involves providing a few examples to help the model understand the prompt/task.
3. Chain-of-thought (COT) Prompting: involves breaking down complex tasks into simpler steps to help the model understand the prompt/task.

Overall, prompt construction plays a vital role in testing LLMs. In other words, how the prompt is constructed affects the model behavior and, thereby, model evaluation. Therefore, creating effective prompts that combine the test scenario (user input) with other contextual information relevant to the LLM is essential.

In addition to the prompting strategies, parameters significantly influence the LLM's outcome. Key parameters include:

- Temperature: Controls the randomness of the generated output. A higher temperature value increases creativity in LLM outcomes by sampling from a wider range of possible tokens, while a lower value (i.e., closer to 0) produces consistent and predictable outcomes.
- Top-p (nucleus sampling): Limits the selection of words (tokens) whose cumulative probability reaches or exceeds the specified top-p value.
- Max tokens: Sets the maximum number of tokens that can be generated in a response. In other words, the max tokens parameter allows you to limit the length of the generated response.
- Frequency penalty: To minimize the likelihood of repetitive tokens by penalizing tokens based on how frequently they have already appeared in the output text.

Step 5: Assessment and Evaluation: The prompt, which consists of the user instructions and a test case, is provided as input to an LLM. Upon receiving the prompt, the LLM processes it with any contained instructions, and produces an outcome. Next, the LLM's output is recorded

and analyzed. The metric by which the LLM is assessed depends on the task it was asked to perform. Some commonly used metrics (Hu et al., 2024) are:

- **Classification-based metrics:** accuracy, precision, recall, F1-score
- **Token-similarity metrics:** Recall-Oriented Understudy for Gisting Evaluation (ROUGE), Bilingual Evaluation Study (BLEU), Metric for Evaluation of Translation with Explicit Ordering (METEOR).
- **Embedding-similarity metrics**: Bidirectional Encoder Representations from Transformer Scores (BERTScore).

Note that the assessment is specific to the task (e.g., Named Entity Recognition), and the prompt must be designed according to the task that is currently being evaluated.

Benchmarks: Evaluating an LLM using a standard test data set provides insights into the LLM's abilities on a specific task compared to other models. However, LLMs are versatile and possess the ability to perform a variety of tasks with varying degrees of complexity. Thus, evaluating an LLM on a single test set will not be sufficient. Benchmarks are tools for exploring an LLM's strengths and weaknesses over a diverse range of tasks or functions.

A benchmark is a standardized framework for the holistic evaluation of an LLM. It consists of diverse task sets (e.g., NER, MCQ) to test an LLM on its various abilities, metrics for evaluating each ability, and a systematic methodology to assess different dimensions of an LLM's abilities. Furthermore, they enable objective comparison between different LLMs. For example, Massive Multitask Language Understanding (MMLU) is a widely used benchmark that evaluates LLMs across 57 subject areas across humanities, STEM, social sciences, and others (Hendrycks et al., 2021). Overall, a diverse collection of benchmarks provides a holistic understanding of an LLM's range of abilities, offering a more comprehensive assessment than any single test set could provide.

While evaluating using benchmarks delivers valuable holistic insights into the LLM's abilities, real-world deployments of LLMs necessitate targeted evaluations that align with specific use cases and operational conditions. This is in part because of the limitations of benchmarks. By being highly structured and constrained in their implementation, benchmarks offer results that are comparable across LLMs. This means that benchmarks typically do not incorporate the context of use for a specific LLM application. Additionally, LLM benchmarks often experience benchmark saturation, where the usefulness or integrity of the benchmark reduces overtime. As a result, benchmark testing usually needs to be combined with tests for specific use cases or real-world use context. The following section outlines the key dimensions—LLM capabilities (what it can do) and qualities (how well it does it)—what specific capabilities and qualities testers should prioritize based on their intended applications and operational needs.

Key Dimensions of LLM Evaluation: Capabilities and Quality Attributes

LLMs appear to have human-like abilities, which makes us want to use them like a human, e.g., relying on general intelligence to perform a variety of tasks across a variety of domains. However, like other AI, they are developed from training data and may not generalize outside the training distribution. This makes it very important to ensure that they are tested within the operational contexts and for the specific operational purposes they will be used for. When a single LLM is expected to be used very broadly, this creates an extremely large test universe.

This expectation of broad utility and human-level performance necessitates a thorough assessment of the LLM. Therefore, a comprehensive evaluation framework for testing LLMs must include two primary dimensions: (1) evaluation of fundamental capabilities, namely

understanding, reasoning, and generation, and (2) ascertaining its quality attributes, such as reliability, performance, robustness, and privacy. The evaluation of fundamental capabilities is essential due to an LLM's core function as a language model that facilitates human-like interactions. As a software component, LLMs must meet expected quality standards.

Capabilities

Our structured approach to testing LLMs is based on three aspects of LLM input processing: understanding, reasoning, and generation. These were chosen because they are core to an LLM's functionality. Upon receiving input, an LLM is expected to 1) parse and understand the input, 2) reason based on that understanding, and 3) combine both the reasoning and understanding to generate an outcome. While we will separately discuss testing these three aspects, it is possible for them to sometimes overlap.

Understanding is the capability of LLM to successfully interpret text inputs. We will test if the LLM can successfully interpret the user inputs by looking at its ability to receive, parse, and comprehend natural language. Below, we discuss specific tasks that testers can use to evaluate LLM understanding.

- **Named Entity Recognition (NER):** NER is a widely used task in the natural language processing (NLP) community to evaluate the language model's ability to parse inputs and assign appropriate entity categories to each word from the input text. NER is considered one of the fundamental evaluation tasks in NLP. It helps determine whether the model can understand each word (also referred to as a token) from the input text and classify them into entity categories. There are standard entity categories found across most NER implementations (e.g., person, organization, location). However, specific applications may develop custom entity categories. Testers should identify any relevant common and custom entity categories. Evaluating the LLM on this task helps assess its token-level understanding capabilities; however, it does not evaluate the model's ability to understand the relationship between the tokens.
- **Text classification:** In text classification, the LLM assigns a label to an input text based on the overall theme of the input. The assigned labels are predefined. They frequently have just two categories (e.g., yes/no, pass/fail, etc.), but multiple categorical labels are possible. Additionally, text classification commonly assigns one label, but some applications may be designed to assign multiple labels. Unlike the NER task, which is limited to evaluating the LLM's understanding at an individual word level, text classification helps in ascertaining whether the LLM is able to understand the overall relationship between words in the input text.
- **Sentiment analysis:** Sentiment analysis is a computational method that assesses a text's tone or sentiment. Testers can use sentiment analysis to assess an LLM's ability to understand the nuanced relationship between the words in the input text and derive the overall sentiment or emotion of the input. Sentiment analysis can be performed at the sentence level, paragraph level, or document level. Outputs from sentiment analysis can be binary (i.e., positive/negative). However, outputs that are a probability or range of scores are better at assessing an LLM's ability to infer contextual nuances and meaning across various text lengths.
- **Natural Language Inference (NLI):** NLI is also known as textual entailment (TE) or Recognizing Textual Entailment (RTE). It is the task of determining the logical relationship between two short texts which are denoted as the premise and the hypothesis. A premise and a hypothesis are provided as inputs to the LLM which

analyses the relationship between them and assigns an appropriate label. In general, NLI identifies three types of relationships: 1) Entailment if the premise logically implies the hypothesis; 2) Contradiction, when the hypothesis contradicts the premise; and 3) Neutral, when the hypothesis can neither be logically deduced as true nor false based on the premise. In other words, it might be possible but is not 100% likely (not enough information to conclude). NLI represents a more advanced level of understanding, requiring the LLM to integrate token-level understanding and contextual understanding and then reach a conclusion.

Reasoning is the capability of LLMs to process information (from input text), draw inferences from the information, and derive conclusions based on the available information (Mondorf et al., 2024). Evaluating reasoning capabilities provides insights into LLMs' logical reasoning and analytical thinking abilities. Reasoning capability evaluations in LLMs are broadly categorized into core and integrated reasoning tasks, and the current T&E practices include various tasks within these categories. Common reasoning tasks include:

- **Logic:** This is the LLM's ability to logically derive valid conclusions. Based upon the objective, it can be divided into three subtasks (Mondorf et al., 2024):
    - **Deductive** reasoning tasks aim to assess if the LLM reaches a conclusion based on its valid premise or deriving cause-and-effect relationships.
    - **Inductive** reasoning tasks help evaluate the LLM's ability to identify patterns from the input and arrive at reasonable generalizations. In other words, given a specific set of examples, the task evaluates if the LLM is capable of deriving generalizable conclusions.
    - **Abductive** reasoning tasks test the LLM's ability to use given observations to formulate plausible explanations or possible hypotheses.
- **Mathematics:** The LLM's ability to perform mathematical tasks.
- **Multi-hop:** The LLM's integrated reasoning ability, assessing if the LLM can successfully make a series of logical steps or interferences to reach a conclusion.
- **Common sense:** This reasoning task assesses the LLM's capability to apply real-world knowledge, such as everyday situations and human-like interactions, including social norms and constraints, basic laws of physics (e.g., ice melts into water and objects fall), and others.

Generation is the capability of LLMs to produce/create coherent, contextually relevant model responses. An LLM's output from a text input can range in length and complexity. The generated responses can go from a single-word or syllable to long text summaries. The below evaluation practices aim to assess the different aspects of generation evaluation through various tasks such as translation, question answering, summarization, code generation, and text generation. Note that we limit our discussion to text based LLM and do not discuss multi-model models.

- **Translation:** The LLM's ability to generate translated text that guarantees the relevance and underlying meaning of the original text, grammatical accuracy, and contextually relevant translated text.
- **Question/Answer (QA):** The LLM's ability to generate relevant and accurate responses based on provided questions. These tasks are designed to test the model's ability to respond to either an open-ended question with an appropriate answer or respond to a multiple-choice question with the correct choice.

- **Summarization:** The LLM's ability to create short content capturing the key points and concepts in a larger input text. Two types of summarizations are extractive and abstractive reasoning. In extractive reasoning the LLM is evaluated on how well it extracts key excerpts from the larger text and combines the excerpts into a coherent output. For abstractive summarization, the LLM is assessed on its ability to create concise original text that captures the meaning of the input text. Overall, the summarization task represents an advanced level in testing generation capabilities.
- **Coding**: Expanding beyond traditional text generation, the coding tasks evaluate the capabilities of LLMs in writing software code. This task primarily evaluates the LLM's capability to generate functionally correct software code based on user requirements.

Table 1 presents a list of tasks and benchmarks for evaluating the three capabilities of LLM. Testers should evaluate LLMs across all three capabilities: understanding, reasoning, and generation.

Table 1. A List of Tasks/Benchmarks Used for Evaluating Capabilities

Capability	Task Type	Benchmarks	Comments	Relative Complexity
Understanding	Named Entity Recognition	CoNLL 2003	Evaluates basic word-level understanding and categorization abilities.	Low
	Sentiment Analysis	IMDb Yelp-2 Yelp-5	The ability to grasp emotional and contextual meaning	Moderate
	Text Classification	SuperGLUE	Ability to understand and categorize text	Moderate
	Natural Language Inference	SNLI	Tests complex logical relationships between statements	High
Reasoning	Inductive reasoning	bAbI-15 EntailmentBank	Evaluate the ability to make generalizations from the observed patterns.	Moderate
	Deductive reasoning	bAbI-15	Test the ability to reach valid conclusions from premises	Moderate
	Abductive reasoning	α-NLI	Assess the ability to form plausible explanations from observations	Moderate
	Mathematical reasoning	GSM8K MATH	Tests mathematical problem-solving skills	Moderate
	Multi-hop reasoning	StrategyQA HotPotQA	Tests the ability to logically connect and reason across multiple steps.	High

	Commonsense reasoning	CommonSenseQA OpenBookQA HellaSwag	Assess the application of real-world knowledge	High
Generation	Translation	WMT IWSLT	Evaluates the translating ability	Medium
	Summarization	XSum CNN/DailyMail	Tests the summarization ability	High
	Code Generation	HumanEval MBPP	Tests the ability to generate functionally correct software code	High

Understanding: The understanding capability of an LLM influences its ability to correctly interpret inputs, including inferring complex nuances of the input, which in turn guides the reasoning and generation activities of an LLM. In other words, weaker/limited understanding capability can significantly impact the LLM's overall performance by limiting its ability to grasp the context, missing interconnected relationships in the input text, or a total misunderstanding of the user's intent, which will result in incorrect outcomes.

- For Use Case 1, testers must evaluate if the LLM can identify and accurately classify entities like person names, organizations, and transactions within user-provided records.
- Evaluation activities for fine-tuned LLM in Use Case 2 must assess the LLM's capability to understand records and identify entities that need to be mapped to specific fields in the documents.
- For Use Case 3, evaluations must ascertain if the LLM understands operator text messages by correctly identifying intended drone commands like "make a left turn" and their associated parameters "after 50 ft."

Reasoning: Shortcomings or deficiencies in reasoning capabilities can significantly impact the LLM's performance and, consequently, its adoption in operational environments. For example, a deficiency or lack of satisfactory abductive reasoning abilities can make the LLM prone to hallucinations—the tendency to generate plausible but factually incorrect information (Toroghi et al., 2024).

- For Use Case 1, logical reasoning evaluations will help determine if the LLM can identify both direct and indirect relationships among different entities across different records.
- Similarly, for Use Case 2, evaluations should be performed to determine if the fine-tuned LLM can synthesize user responses across multiple questions and extract information to update a specific field. In other words, determine if the fine-tuned LLM can perform multi-hop reasoning and extract relevant information from user responses.
- For Use Case 3, testers should consider evaluating scenarios such as whether an LLM applies common reasoning while converting text messages to commands. For instance, they should test if the LLM can recognize and avoid generating physically infeasible commands, e.g., "fly through the debris."

Generation: Evaluating generation capabilities presents its own challenges, such as measuring creativity in the generated text and the inherently subjective nature of assessment in writing tasks.

- For Use Case 2, testers should verify that the information generated by the LLM to update the fixed field, short free-text, and long blocks of free-text is accurate and matches respective user records. Note that, given the goal of Use Case 2, the automatic completion of internal documents from user records, the evaluation of the fine-tuned LLM's ability to generate coherent, concise text for updating the fields of both short free-text and long blocks of free-text, are key evaluation priorities. While Use Cases 1 and 3 might involve minor generation, it is less critical than Use Case 2's core task. Hence, we limit our discussion to Use Case 2.

Quality Attributes

Below, we describe some common quality attributes. We include some illustrative examples using the use cases described above. While quality attributes assessment is essential for all three use cases, due to space limitations we limit our discussion to one or two use cases per quality attributed.

Reliability: Reliability is "the ability of a system or component to perform its required functions under stated conditions for a specified period of time" (ISO/IEC/IEEE, 2017). Reliability assessments of an LLM evaluate its ability to produce consistent, coherent outputs under normal or expected operational conditions. The main goal is to evaluate an LLM's behavioral consistency. It includes a variety of assessments, such as testing the LLM for consistent behavior across variations in the input text and different contextual settings, factual consistency across outputs for the same or similar prompts (hallucination detection), the LLM's ability to quantify and communicate its confidence in its outcomes (uncertainty quantification) and assessing the accuracy of confidence estimates to actual performance (calibration; Sun et al., 2024; Walsh et al., 2024; Zhuang et al., 2023). A lack of comprehensive reliability assessment significantly increases the risks in operationalizing LLMs, as unreliable models can lead to serious and potentially catastrophic outcomes.

For Use Case 2: Evaluate the fine-tuned LLM's consistency; the reliability assessments should evaluate whether the fine-tuned LLM consistently extracts the same information from the user-provided records and question responses. For example, provide the employee performance report document (input) multiple times and check if the LLM consistently extracts the employee's name and the manager's feedback.

Performance: Assessing an LLM's performance is a multifaceted activity that involves evaluating both the quality of the model's outcome and its efficiency in producing the outcome. Quality assessment includes evaluating the output's coherence, ensuring logical flow and contextual consistency, determining its relevance to the given task, ensuring the outcome is factually correct, and evaluating the logical soundness of the LLM's reasoning process (Huang et al., 2024; Zhuang et al., 2023). Efficiency evaluation measures the LLM's computational performance through latency, inference speed, and throughput, which are critical for adapting an LLM across different operational environments.

Use Case 3 will be used in real-time operational conditions. Therefore, testers should evaluate LLM latency and assess if the latency level is sufficient to support operational needs.

Maintainability: Unlike traditional software systems, updating or modifying an LLM is not limited to structural changes to the software code. Given an LLM is a data-intensive system, modifications range from retraining the LLM with a revised dataset or hyperparameters to fine-tuning an LLM with a domain-specific dataset. Furthermore, in some cases, LLMs are designed to learn from their operational environment (continual learning). Thus, it is subject to frequent adjustments (or updates) upon deployment. While this is a desired behavior, ensuring that an LLM continues to work as expected is critical. Maintainability assessment must account for the characteristics of LLMs and focus on evaluating the model's ability to incorporate updates as

well as their ability to adapt to different operational environments without comprising performance.

Use Case 2: Given the planned semi-annual retraining cycle, evaluations must be performed to ensure that periodic retraining does not adversely impact the LLM's performance. For instance, after each retraining with recent documents and potentially new data sources, testers should compare the LLM information extraction accuracy between previous and newly introduced data sources.

Scalability assesses the LLM's ability to deliver satisfactory performance under varying operational conditions, including fluctuating demands in user queries, input text length, and computational resource consumption. In other words, scalability evaluates the LLM's ability to handle increased demand or workloads (serve a significant number of concurrent user requests, handle larger inputs, and operate in diverse environments) without significantly comprising its performance (i.e., output quality).

Scalability assessments of an LLM help determine operational conditions suitable for optimal model performance (resource requirements), identify potential bottlenecks that may hinder the model's performance, and thereby determine the model's viability for deployment. To this end, testing scenarios will be designed to systematically evaluate LLM's performance across various operational conditions. Key testing scenarios include:

- **Size of input:** Understanding the LLM's ability to handle various types of inputs, including the complexity of input and length of the inputs. For example, can an LLM handle longer, lengthier inputs/documents (Context window limitations)? Does handling longer inputs result in a memory crash? Does handling a substantial number of longer inputs (requests) impact the LLM's processing or inferencing time? If there is a delay in inferencing time, is it within a reasonable time? Measuring the LLM's response quality with increasing complexity in prompts.
- **Number of users:** Does the increase in the number of users impact the model's performance (e.g., increase in model inference time)?
- **Frequency of input:** Can the LLM handle a higher volume of inputs without significant performance degradation?
- **Operating environments:** Can the LLM be deployed on different hardware configurations? Can it work with limited hardware resources? Can it handle new data types?
- **For Use Case 1:** Focus on whether the LLM can identify entities and extract relationships within a reasonable time frame, specifically when dealing with increasing workloads (>=1M records).
- **For Use Case 2:** Evaluate the fine-tuned LLM for varying lengths of short free-text (e.g., between 50 and 450 words) and long blocks free-text (e.g., 1000, 2000 words) and see if it impacts the LLM's extraction ability.

Robustness: "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions" (ISO/IEC/IEEE, 2017). In the context of LLMs, robustness is defined as the ability to produce a consistent performance across different operating conditions, such as previously unseen scenarios, handling noisy input data, out-of-distribution inputs, variations or perturbations to input prompts (prompt sensitivity), and resistance to adversarial attacks (Sun et al., 2024). Robustness assessment must evaluate an LLM's resilience across various input scenarios, including its ability to handle out-of-distribution input values, resistance to prompt injection attacks (a type of adversarial attack

targeted at manipulating the LLM's behavior), its performance under perturbed input values, and its ability to generate relevant, accurate outputs in the presence of misleading or irrelevant information. Failure to perform a comprehensive robustness assessment opens up the LLM for potential operational risks, such as increased vulnerability to prompt injection attacks, susceptibility to generating inconsistent model outcomes when dealing with noisy input values, and potential performance degradation when deployed in rapidly evolving operational environments (lack of generalizability).

For Use Case 3: Examine the LLM's behavior in handling informal or abbreviated language and uncommon jargon that the model might not have been exposed to at the time of its training. Furthermore, tests must be conducted to assess the LLM's sensitivity to text perturbations. For example, testers can introduce common or likely misspellings or additional white spaces to text messages and determine if the LLM continues to perform as expected or results in a misinterpretation.

Privacy: The ability to safeguard sensitive information, including training data, personally identifiable information (PII), and confidential information received through training or user interactions. LLMs are trained on large datasets, making them susceptible to privacy attacks. Furthermore, the interactive nature of LLMs, combined with its tendency to align itself with users' requests, significantly increases privacy risks. For example, malicious actors could employ sophisticated prompt injection techniques to trick the LLM into revealing sensitive information.

Privacy evaluation in LLMs focuses on ascertaining its ability to protect against unauthorized access, accidental disclosure of training data or user information, and resilience to various data extraction attacks (Sun et al., 2024). Failure to perform adequate privacy evaluation poses significant risks, as it increases the likelihood of exposing training data, revealing sensitive information, and confidential user interactions.

For Use Case 2: Evaluate the fine-tuned LLM for privacy guarantees. They should also perform privacy assessments to ensure that the LLM does not inadvertently expose sensitive information from the document.

Security: At a high-level, security refers to protecting a system from threats and risks that may lead to harm. When assessing the security for an LLM (which is likely a component of a larger system), testers should focus on assessing safeguards of the model and its related artifacts. This includes assessing protections for its training data and model weights, defenses against unauthorized entities, processes for detecting and mitigating adversarial threats and malicious manipulation of the LLM's behavior, and other processes for ensuring the LLM's integrity.

A comprehensive security assessment of LLMs includes evaluating its resistance to prompt injection attacks, its ability to defend against evasion attacks, and testing for vulnerabilities in access control. This includes assessing how resilient the model is to unauthorized access, potential modifications, or tampering with model weights and protecting against model inversion attacks. The overall goal is to assess and implement robust safeguards that prevent malicious manipulations of the model for generating harmful outcomes, thereby ensuring the integrity of the LLM across various operational contexts.

For Use Case 1: Evaluate whether it can detect malicious user requests and does not disclose privileged information (i.e., prevent unauthorized access).

Explainability: LLMs function as black boxes, with their internal decision-making processes remaining opaque to the users (Cambria et al., 2024). LLMs that are explainable facilitate user trust in outputs and effective debugging processes. Explainability in LLMs aims to provide users with insights into the LLM's reasoning as to why it produced a particular outcome, thereby

facilitating a better understanding of its behavior. Explainability assessments evaluate an LLM's inherent capacity to generate human-comprehensible explanations for its outcomes (Zhao et al., 2024). For example, they may assess if the LLM provides rational support for its response.

For Use Case 2: Have the LLM provide step-by-step explanations for filling the fixed fields vs. short free-text vs. long blocks of free-text. This assessment will help stakeholders understand the LLM's information extraction and document completion process.

Fairness: LLMs are data-intensive systems that inherently reflect the distribution of the data they are trained with (Chandrasekaran et al., 2024). In other words, the behavior of the LLM is, to a large extent, a representation of its training data. Due to the practical limitations in comprehensively capturing the operational universe within the training dataset, an LLM may inadvertently reflect the inherent biases in the training data and thus exhibit discriminatory behaviors. Fairness in LLMs refers to the model's ability to generate outcomes without preference or discrimination across protected groups, ensuring no demographic is disadvantaged or misrepresented (Chu et al., 2024; Li et al.; 2023, Schwartz et al., 2022). Fairness evaluation is essential to guarantee that the LLM exhibits non-discriminatory behavior. This activity spans the LLM's life cycle, including the data collection, training, and fine-tuning phases. Unlike assessments of other quality attributes, where a single test instance (i.e., occurs only once) may suffice to identify/uncover the underlying issue, fairness evaluation may require multiple test instances (i.e., more than one occurrence) to establish patterns of discriminatory behavior (Weidinger et al., 2022). Thus, necessitating comparatively increased testing efforts.

For Use Case 3: Fairness evaluations ensure that the off-the-shelf LLM treats different dialects and linguistic styles equally when converting the operator's text messages to commands. Is the LLM prone to misinterpret certain linguistic styles while converting text messages into commands?

Safety: Evaluates the LLM's ability to avoid generating harmful, toxic, unethical, deceptive, unlawful, or otherwise undesirable content or behavior during its intended use, thereby maintaining a safe operational environment (Weidinger et al., 2022; Zhang et al., 2023). It involves systematically assessing an LLM's behavior across diverse scenarios to achieve this goal. Furthermore, it aims to validate an LLM's safety guardrails in preventing the model from generating unsafe outputs either intentionally (tricked by a malicious actor) or unintentionally (non-malicious intent yet can cause harm).

For Use Case 3: Most significant for Use Case 3 given its operational environment and application (but crucial for all use cases). Safety evaluations must be performed for this use case to guarantee that the LLM (in the drone) identifies and rejects potentially dangerous text messages or asks for further clarification (from the user) before converting to a command in case the input text is ambiguous or borderline risky.

Adaptability: "The degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments" (ISO/IEC/IEEE, 2017). Given the nature and characteristics of LLMs, testers can measure an LLM's adaptability by assessing its ability to adjust to new operational conditions, including new tasks and domains. They can also assess its ability to perform inferencing in resource-constrained hardware environments. Furthermore, evaluations must ascertain an LLM's ability to improve its performance through continual learning (feedback from the operational environment). Inadequate adaptability evaluation creates significant bottlenecks in operationalizing LLMs across diverse environments. For example, limited or lack of adaptability can make an LLM obsolete too soon. Since LLMs require substantial resources for training and deployment, retraining or replacing an obsolete model becomes expensive and time-consuming.

Similarly, poor adaptability to diverse operational environments significantly restricts an LLM's utility and its cross-environment applicability.

- **For Use Case 1:** Ascertain its ability to adapt to new domains or usage environments. For instance, evaluations must determine if the LLM can accurately identify and extract relationships from previously unseen entity types.
- **For Use Case 3:** Evaluate the LLM's inferencing performance across various resource-constrained environments. Specifically, they should assess whether the LLM maintains acceptable performance when deployed on different hardware architectures. Additionally, testers must assess whether there is a significant drop in performance between the original LLM and its optimized versions, such as a quantized LLM.

Framework Boundaries

The evaluation of LLM across capabilities and qualities reveals an interconnectedness where there is an inherent lack of rigid boundaries between assessment areas. In most cases, evaluating an LLM on a task intended for a particular capability can potentially assess other capabilities. For example, NLI evaluation assesses not only understanding but also the LLM's reasoning abilities. Likewise, evaluating an LLM for hallucinations goes beyond strictly assessing reliability, as it also reflects on the LLM's performance in generating factually correct outcomes. Although we present the framework by grouping tasks under the category they primarily assess, testers should be mindful that, in most cases, an LLM evaluation can typically provide insights into multiple capabilities, quality attributes, or a combination of these. Moreover, test plan design must be guided by the operational conditions, prioritizing specific capabilities and quality attributes based on operational requirements.

Challenges and Limitations in the Current T&E Practices

The current T&E practices for LLMs, while providing a baseline for evaluation, suffer from key limitations. Firstly, there is a disconnect between benchmark performance and real-world utility. Open-source benchmarks barely reflect the full spectrum of operations an LLM will encounter in operational scenarios. Moreover, most benchmarks remain static over time and lack domain specificity. Adding to this, the controlled nature of a test environment fails to account for variability in operational environments. Consequently, a strong benchmark performance may not necessarily translate to success in an operational environment. Secondly, the use of aggregate metrics (accuracy, F1 score) provides insights into LLM performance. However, they fail to provide a granular understanding of the LLM's behavior, thus limiting the ability to gain insights into specific strengths, weaknesses, and potential failure points in LLMs. Third, LLMs' non-deterministic and opaque nature presents unique challenges in failure analysis and debugging activities. Existing T&E approaches developed for traditional software systems with understandable decision logic and deterministic behaviors are often ineffective for LLMs, limiting the ability to systematically detect and address failure modes. Finally, a lack of standardized techniques to measure test adequacy, potentially leading to inadequate test design and incomplete evaluation.

Conclusion and Future Directions

This paper presents an overview of the current T&E practices for evaluating LLMs based on a survey of academic literature. We outline the key steps in testing LLMs and discuss how to establish an evaluation scope by categorizing LLM capabilities and properties, illustrated with three acquisition scenarios. Our findings indicate that while the T&E of LLMs is nascent and rapidly evolving, significant challenges remain. Current practices provide a foundation for

evaluation but require substantial improvements to address the challenges in evaluating the multi-faceted nature of LLMs. For instance, public benchmarks offer a starting point for evaluation; however, their utility is limited as they cannot be generalized to all possible operational scenarios. Our analysis identifies several key areas for future research. First, developing new T&E approaches to comprehensively evaluate LLMs in specific operational contexts. Second, standardized approaches for measuring test adequacy should be established. Finally, we observe a significant imbalance in the T&E of an LLM across its life cycle. While a significant amount of work is reported for model-level T&E, there remains a significant gap in research regarding the system level and post-deployment (operational) evaluation.

Acknowledgment

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense (DoD) through the Office of the Under Secretary of Defense for Acquisition and Sustainment and the Office of the Under Secretary of Defense for Research and Engineering under Contract HQ003424D0023. The Acquisition Innovation Research Center is a multi-university partnership led and managed by the Stevens Institute of Technology through the Systems Engineering Research Center—a federally funded University Affiliated Research Center. Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government (including the DoD and any government personnel).

References

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). *Text summarization techniques: A brief survey*. arXiv preprint arXiv:1707.02268.

Cambria, E., Malandri, L., Mercorio, F., Nobani, N., & Seveso, A. (2024). *Xai meets LLMs: A survey of the relation between explainable ai and large language models*. arXiv preprint arXiv:2407.15248.

Chandrasekaran, J., Cody, T., McCarthy, N., Lanus, E., Freeman, L., & Alexander, K. (2024). Testing machine learning: Best practices for the life cycle. *Naval Engineers Journal 136*(1–2), 249–263.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., ... & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology 15*(3), 1–45.

Chu, Z., Wang, Z., & Zhang, W. (2024). Fairness in large language models: A taxonomic survey. *ACM SIGKDD Explorations Newsletter 26*(1), 34–48.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Hu, T., & Zhou, X. H. (2024). *Unveiling LLM evaluation focused on metrics: Challenges and solutions*. arXiv preprint arXiv:2404.09135.

Huang, X., Ruan, W., Huang, W., Jin, G., Dong, Y., Wu, C., ... & Mustafa, M. A. (2024). A survey of safety and trustworthiness of large language models through the lens of verification and validation. *Artificial Intelligence Review*, *57*(7), 175.

ISO/IEC/IEEE. (2017). *ISO/IEC/IEEE international standard—systems and software engineering—vocabulary* (ISO/IEC/IEEE 24765:2017) [Standard]. ISO/IEC/IEEE. Retrieved from [IEEE]

Li, Y., Du, M., Song, R., Wang, X., & Wang, Y. (2023). *A survey on fairness in large language models*. arXiv preprint arXiv:2308.10149.

Mondorf, P., & Plank, B. (2024). *Beyond accuracy: Evaluating the reasoning behavior of large language models—a survey*. arXiv preprint arXiv:2404.01869.

Nguyen, H., Chen, H., Pobbathi, L., & Ding, J. (2024). *A comparative study of quality evaluation methods for text summarization*. arXiv preprint arXiv:2407.00747.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., ... & Resnik, P. (2024). *The prompt report: A systematic survey of prompting techniques*. arXiv preprint arXiv:2406.06608.

Schwartz, R., Schwartz, R., Vassilev, A., Greene, K., Perine, L., Burt, A., & Hall, P. (2022). *Towards a standard for identifying and managing bias in artificial intelligence* (Vol. 3, p. 00). US Department of Commerce, National Institute of Standards and Technology.

Sun, L., Huang, Y., Wang, H., Wu, S., Zhang, Q., Gao, C., ... & Zhao, Y. (2024). *Trustllm: Trustworthiness in large language models*. arXiv preprint arXiv:2401.05561, 3.

Toroghi, A., Guo, W., Pesaranghader, A., & Sanner, S. (2024, November). Verifiable, debuggable, and repairable commonsense logical Reasoning via LLM-based theory resolution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (pp. 6634–6652).

Walsh, M., Schulker, D., Lau, S-H. (2024). Beyond capable: Accuracy, calibration, and robustness in large language models. *SEI Blog*. https://insights.sei.cmu.edu/blog/beyond-capable-accuracy-calibration-and-robustness-in-large-language-models/

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems 35*, 24824–24837.

Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Huang, P. S., Mellor, J., ... & Gabriel, I. (2022, June). Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency* (pp. 214–229).

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., ... & Du, M. (2024). Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, *15*(2), 1–38.

Zhuang, Z., Chen, Q., Ma, L., Li, M., Han, Y., Qian, Y., ... & Liu, T. (2023). *Through the lens of core competency: Survey on evaluation of large language models*. arXiv preprint arXiv:2308.07902.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net