

# LLM-Supported Natural Language to Bash Translation

by

Finnian Ellis Westenfelder

B.S. Computer Science, U.S. Air Force Academy, 2023

B.S. Cyber Science, U.S. Air Force Academy, 2023

Submitted to the Institute for Data, Systems, and Society and  
the Department of Electrical Engineering and Computer Science  
in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE IN TECHNOLOGY AND POLICY

and

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING  
AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

© 2025 Finnian Ellis Westenfelder. All rights reserved.

The author hereby grants to MIT and The Charles Stark Draper Laboratory a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Finnian Ellis Westenfelder  
Institute for Data, Systems, and Society  
Department of Electrical Engineering and Computer Science  
May 9, 2025

Certified by: Una-May O'Reilly  
CSAIL Principal Research Scientist, Thesis Supervisor

Certified by: Silviu Chiricescu  
Draper Laboratory Computer Systems Security Group Lead, Thesis Supervisor

Accepted by: Christine Ortiz  
Professor, Institute for Data, Systems, and Society  
Professor, Department of Materials Science and Engineering  
Director, Technology and Policy Program

Accepted by: Leslie A. Kolodziejski  
Professor of Electrical Engineering and Computer Science  
Chair, Department Committee on Graduate Students

# LLM-Supported Natural Language to Bash Translation

by

Finnian Ellis Westenfelder

Submitted to the Institute for Data, Systems, and Society and  
the Department of Electrical Engineering and Computer Science  
on May 9, 2025 in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE IN TECHNOLOGY AND POLICY

and

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING  
AND COMPUTER SCIENCE

## ABSTRACT

*The views expressed in this thesis are those of the author and do not necessarily reflect the official policy or position of the Air Force, the Department of Defense or the U.S. Government.*

The Bourne-Again Shell (Bash) command-line interface for Linux systems has complex syntax and requires extensive specialized knowledge. Using the natural language to Bash command (NL2SH) translation capabilities of large language models (LLMs) for command composition alleviates these issues. However, the NL2SH performance of LLMs is difficult to assess due to inaccurate test data and unreliable heuristics for determining the functional equivalence of Bash commands. We present a manually verified test dataset of 600 instruction-command pairs and a training dataset of 40,939 pairs, increasing the size of previous datasets by 441% and 135%, respectively. Further, we present a novel functional equivalence heuristic that combines command execution with LLM evaluation of command outputs. Our heuristic can determine the functional equivalence of two Bash commands with 95% confidence, a 16% increase over previous heuristics. Evaluation of popular LLMs using our test dataset and heuristic demonstrates that parsing, in-context learning, in-weight learning and constrained decoding can improve NL2SH accuracy by up to 32%. Additionally, we consider military use cases for NL2SH models and discuss the limitations of current Department of Defense documentation standards for LLMs. We write and publish documentation for our models and datasets to promote safe use. Our findings emphasize the importance of dataset quality, execution-based evaluation, translation method, and proper documentation for advancing NL2SH translation and enabling responsible use. Our code is available at <https://github.com/westenfelder/NL2SH>.

Thesis supervisor: Una-May O'Reilly

Title: CSAIL Principal Research Scientist

Thesis supervisor: Silviu Chiricescu

Title: Draper Laboratory Computer Systems Security Group Lead

# Acknowledgments

To Una-May O'Reilly,

Thank you for your mentorship. Your insights into both disciplines of my research were invaluable. I became a more open learner through our meetings and the most valuable part of this thesis is the skills I learned in the process. I am thankful for the opportunity to work with you and all of ALFA Group during my time at MIT.

To Silviu Chiricescu,

Thank you for providing reliable feedback throughout the thesis process. You helped me learn how to conduct research and think critically. This work would not have been possible without the compute resources you provided. I am grateful that Draper provided the financial and intellectual freedom to explore my research interests.

To Erik Hemberg and Stephen Moskal,

Thank you for helping transform my ideas into research projects. I could not have completed this thesis without your technical expertise and our sessions at the whiteboard. I am especially grateful for your support during my summer of remote research. I will miss chatting with both of you in the lab.

To Barb DeLaBarre,

Thank you for your support as I navigated the complexities of a dual degree. The Technology and Policy Program would not be the same without your guidance and bubbly personality.

To ALFA Group,

Thank you for fostering an encouraging and collaborative environment. The diverse backgrounds and perspectives in the lab made it an amazing place to learn and grow. Stephen Jorgensen and Miguel Tulla, I enjoyed collaborating and learning from both of you on our projects. I will miss working with you all.

To Mom, Dad and Leo,

Thank you for your unwavering support. I love you!

Finn Westenfelder

May 1, 2025

# Contents

<i>List of Figures</i>	6
<i>List of Tables</i>	8
<b>1 Introduction</b>	<b>10</b>
1.1 Motivation . . . . .	10
1.2 Research Questions . . . . .	11
1.3 Tasks . . . . .	11
1.4 Hypotheses . . . . .	12
1.5 Contributions . . . . .	13
1.6 Background . . . . .	13
<b>2 Related Work</b>	<b>15</b>
2.1 Summary . . . . .	15
2.2 Related Work . . . . .	15
2.3 Advancements . . . . .	18
<b>3 Methodology</b>	<b>19</b>
3.1 Dataset Validation . . . . .	19
3.2 Functional Equivalence Heuristic . . . . .	20
3.3 Translation Methods . . . . .	22
3.3.1 Markdown Parser . . . . .	22
3.3.2 Constrained Decoding . . . . .	22
3.3.3 In-Context Learning . . . . .	23
3.3.4 In-Weight Learning . . . . .	23
<b>4 Experiments and Results</b>	<b>24</b>
4.1 Dataset Validation . . . . .	24
4.2 Functional Equivalence Heuristics . . . . .	24
4.3 Translation Methods . . . . .	27

<b>5</b>	<b>Discussion</b>	<b>30</b>
5.1	Dataset Validation . . . . .	30
5.2	Functional Equivalence Heuristics . . . . .	31
5.3	Translation Methods . . . . .	32
5.4	Ethical Considerations . . . . .	33
5.5	Limitations and Future Work . . . . .	33
<b>6</b>	<b>Documentation Policy</b>	<b>35</b>
6.1	Background . . . . .	35
6.2	Military Documentation Policy . . . . .	37
6.2.1	Overview . . . . .	37
6.2.2	Technical Knowledge: Language Model Safety . . . . .	38
6.2.3	Technical Tension: Interpretability and Security . . . . .	39
6.2.4	Policy Impasse: Documentation Standards . . . . .	40
6.2.5	Interim Policy Concerns . . . . .	42
6.2.6	Third Party Institutions . . . . .	43
6.2.7	NIPR-GPT Evaluation . . . . .	45
6.3	Model and Dataset Documentation . . . . .	45
<b>7</b>	<b>Conclusion</b>	<b>47</b>
<b>A</b>	<b>Translation Prompts</b>	<b>48</b>
	<b>Bibliography</b>	<b>66</b>

# List of Figures

1.1	An example of Natural language to Bash command translation. . . . .	10
2.1	A diagram of NL2SH translation and a comparison of functional equivalence heuristics from previous work. . . . .	18
3.1	Relationships between NL2SH datasets. . . . .	21
4.1	Evaluation of Bash functional equivalence heuristics. The hashed section of each bar indicates the accuracy improvement from passing command execution outputs to the heuristic, as opposed to just the commands. We find execution allows all of our heuristics to achieve better accuracy than the baseline Bleu accuracy. . . . .	26
4.2	Bash functional equivalence heuristic runtime versus accuracy. The use of an execution environment and an LLM both increase the computational cost of a heuristic. We find the exec + mxbai-embed heuristic achieves a good balance of runtime and accuracy. . . . .	26
4.3	Impact of constrained decoding on the NL2SH performance of Llama and Qwen models. Accuracy is measured using the exec + mxbai-embed FEH. Constrained decoding improves the performance of the Llama models but significantly degrades the Qwen models. . . . .	28
4.4	Impact of parsing on the NL2SH performance of Llama, Qwen, and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Parsing improves the performance of all models except for GPT-4 and GPT-4o. Parsing has the largest impact for models in the 0.5B to 3B parameter range. . . . .	28
4.5	Impact of in-context learning on the NL2SH performance of Llama, Qwen, and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Parsing improves the performance of all models except for GPT-4o-mini and GPT-4o. ICL has the largest impact for models in the 0.5B to 3B parameter range. . . . .	29

4.6	Impact of in-weight learning on the NL2SH performance of Llama and Qwen models. Accuracy is measured using the exec + mxbai-embed FEH. IWL significantly improves the performance of the 1B and 3B parameter Llama models, but otherwise has inconsistent results, likely due to our hardware constraints limiting the size of LoRA adapters. . . . .	29
5.1	Dangerous translation observed in testing. . . . .	33
6.1	Hierarchy of DoD AI policy documents. . . . .	36
6.2	Model documentation continuum. . . . .	39
A.1	Prompt for evaluating the functional equivalence of Bash commands. . . . .	48
A.2	Prompt for evaluating the functional equivalence of Bash commands after execution. Note the addition of command outputs compared to the prompt in Figure A.1. . . . .	48
A.3	NL2SH translation prompt used in the baseline evaluation. . . . .	49
A.4	NL2SH translation prompt used in the parsing, constrained decoding and in-weight learning evaluations. . . . .	49
A.5	NL2SH translation prompt used in the in-context learning evaluation. . . . .	51

# List of Tables

1.1	Examples of validating translations in NL2SH datasets. . . . .	12
1.2	Examples of determining the functional equivalence of Bash commands conditioned on a natural language prompt. . . . .	12
1.3	Examples of determining a Bash command given a natural language prompt.	12
1.4	Definition of terms. . . . .	14
2.1	Summary of NL2SH datasets, FEHs, and models created in previous work. .	16
3.1	InterCode dataset errors. . . . .	20
4.1	The baseline accuracy of the GPT-3.5 and GPT-4 models on the InterCode and InterCode-ALFA benchmarks. The intersection of models tested on each benchmark only includes two models. We find the baseline accuracy of both models on the InterCode-ALFA benchmark is significantly higher than on the InterCode benchmark, indicating our verification process improved the quality of test data. . . . .	24
4.2	Evaluation of Bash functional equivalence heuristics. Heuristics were tested on a dataset comprising 300 pairs of equivalent commands and 300 pairs of non-equivalent commands. We find execution paired with LLM evaluation significantly increases recall. Bold indicates the highest F1 score and accuracy as well as the lowest evaluation time. . . . .	25

4.3	Impact of constrained decoding, parsing, in-context learning and in-weight learning on the NL2SH performance of Llama, Qwen and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Translation method can improve performance up to 32% over the baseline, but model size remains the dominant factor. The highest accuracy in each row and column is indicated with bold and an underline, respectively. The CodeLlama model is included as a baseline from previous work. . . . .	27
6.1	Documentation templates proposed by industry and academia. . . . .	42
6.2	Model and dataset documentation links. . . . .	46

# Chapter 1

## Introduction

### 1.1 Motivation

The default command-line interface (CLI) for interacting with Linux systems is the Bourne-Again Shell (Bash) [1]. Bash commands allow computer users to control processes, interact with the file system and manage the network. However, using Bash requires knowledge of numerous utilities, each with unique parameters and complex syntax [2]. Moreover, the reference documentation for these utilities, called manual pages, can be cumbersome and confusing [3]. This makes the CLI a barrier for inexperienced users and increases the chance of errors for experienced users [4]. Erroneous Bash commands are problematic because they can lead to data loss and system failures.

Language models that convert natural language to command-line instructions, a task referred to as *NL2SH*, *NL2CMD* or *NL2Bash Translation*, offer a promising solution to this problem [4]. We use the term *NL2SH model* to refer to models trained specifically for the task of NL2SH, as well as the NL2SH capabilities of general-purpose large language models (LLMs). Figure 1.1 shows an example of natural language to Bash command translation. NL2SH models are well suited for CLIs because they are designed for text-based interactions. NL2SH models can simplify human-computer interactions by allowing users to interact with Linux systems through natural language on the command line. This advancement enhances usability by reducing the need for syntax memorization [5].

---

**Input:** Natural Language  
*List files in the /workspace directory that were accessed over an hour ago.*

---

**Output:** Bash Command  
`find /workspace -type f -amin +60`

---

Figure 1.1: An example of Natural language to Bash command translation.

Natural language to Bash translation is also useful for the development of automated systems. The broad capabilities of LLMs have enabled the creation of autonomous agents[6]. These agents automate tasks, such as web browsing, online shopping, and penetration testing, by using an LLM to reason about the task’s steps and then interfacing with a computer to accomplish those steps [7–10]. Since these agents reason in natural language and interact with the CLI, NL2SH translation is a key component of agentic pipelines [7].

The use of LLMs for NL2SH necessitates benchmarks to measure task accuracy[11–15]. A NL2SH benchmark requires test data consisting of natural language prompts and ground truth commands (referred to as instruction-command pairs). Given a natural language prompt, a NL2SH model generates a Bash command. A benchmark must then use a heuristic to determine if the model command is functionally equivalent to the ground truth command. Determining functional equivalence of commands is difficult because there are multiple possible correct commands for a given task, due to a wide range of interchangeable utilities. Furthermore, command execution may not result in identical outputs, neutralizing evaluation with string comparison. Current benchmarks do not accurately measure NL2SH model performance due to errors in assessment data and inaccurate heuristics for determining the functional equivalence of commands [16–18]. This makes it difficult to assess model capabilities and measure methods for improving model performance.

## 1.2 Research Questions

To address these challenges, we investigate the following research questions.

1. How can we validate natural language to Bash translation datasets to ensure models are trained and evaluated using accurate, unbiased assessments?
2. How can we design a functional equivalence heuristic that accurately measures the quality of model translations?
3. How can we improve the accuracy of natural language to Bash translation models as measured by a reliable benchmark?

## 1.3 Tasks

Our research questions can be addressed by finding means to accomplish the following tasks: dataset validation, functional equivalence, and translation. Table 1.1 provides examples of the dataset validation task in our first research question. Table 1.2 provides examples of the

functional equivalence task in our second research question. Table 1.3 provides examples of the translation task in our third research question.

Table 1.1: Examples of validating translations in NL2SH datasets.

Given		Determine
Natural Language	Bash Command	Correct
<i>print the system disk usage</i>	<code>df -h</code>	True
<i>remove a directory named foo</i>	<code>rm foo</code>	False
<i>print the current user's id</i>	<code>id -u</code>	True

Table 1.2: Examples of determining the functional equivalence of Bash commands conditioned on a natural language prompt.

Given			Determine
Natural Language	Ground Truth Command	Model Output	Equivalent
<i>print the system boot time</i>	<code>who -b</code>	<code>uptime -s</code>	True
<i>delete bin\ in the current dir</i>	<code>rm -r ./bin</code>	<code>rm -r /bin</code>	False
<i>list all groups on the system</i>	<code>getent group</code>	<code>cat /etc/group</code>	True

Table 1.3: Examples of determining a Bash command given a natural language prompt.

Given	Determine
Natural Language	Bash Command
<i>List files in /workspace accessed over an hour ago</i>	<code>find /workspace -type f -amin +60</code>
<i>base64 decode aGVsbG8=</i>	<code>echo 'aGVsbG8='   openssl enc -base64 -d</code>
<i>Sort and print only group names from /etc/group</i>	<code>cut -d: -f1 /etc/group   sort</code>

## 1.4 Hypotheses

Given these research questions and tasks, we propose the following hypotheses.

1. If NL2SH datasets are manually verified by a human, then they will contain fewer errors, because human verification is more accurate than automated methods.
2. If a functional equivalence heuristic is provided with the outputs and side effects of command execution, then it will determine the functional equivalence of Bash commands more accurately, because runtime behavior provides additional information about equivalence.
3. If model size is increased, then accuracy will improve, because larger models can capture more complex relationships between natural language and Bash commands. However,

with a fixed model size, if training, decoding, and inference strategies are optimized, then accuracy will improve, because these strategies allow the model to specialize for the task of translation.

## 1.5 Contributions

Our contributions are summarized as follows.

1. We create a manually verified test dataset of 600 instruction-command pairs and a training dataset of 40,939 pairs, increasing the size of previous test [16] and training datasets [19] by 441% and 135%, respectively.
2. We present a novel functional equivalence heuristic that combines command execution with LLM evaluation of command outputs, capable of determining the functional equivalence of two Bash commands with 95% confidence, a 16% increase over previous heuristics [16].
3. We evaluate popular LLMs using our test data and heuristic and demonstrate that parsing, in-context learning, in-weight learning and constrained decoding can improve NL2SH accuracy by up to 32%.

## 1.6 Background

NL2SH translation falls under the broader domain of machine translation, where models automatically translate text or speech from one language to another. LLMs are well suited for this task, enabling translation that was impossible with previous methods [20]. Evaluating NL2SH models requires determining the functional correctness of generated commands. Functional correctness is defined as whether the code produces the correct output for each input, as specified, or as compared to ground truth [21]. Functional correctness does not consider the diversity of code generated, time complexity, or memory complexity [22].

Ensuring the functional correctness of code is difficult because validation methods are error-prone and take an impractical amount of time for large volumes of code [21]. There are two main validation techniques: static and dynamic analysis. Static analysis checks code without execution, using parsers, lexical analysis or control flow checking [23]. Dynamic analysis evaluates code outputs and runtime behavior using an execution environment [16]. Some frameworks combine static and dynamic analysis [18].

Determining the functional correctness of Bash commands translated from natural language is a sub-problem of validating code correctness. We assess the functional correctness of Bash commands by comparing the generated (model) command with a ground truth Bash command. We define the term "*functional equivalence heuristic*" (*FEH*) to describe a heuristic that performs this comparison and determines the functional correctness of a Bash command. Due to varying definitions in this field, Table 1.4 provides definitions of terms used in this thesis.

Table 1.4: Definition of terms.

<b>Term</b>	<b>Definition</b>
Natural Language Task	$t \in T$ English
Ground Truth Command	$b \in B$ Bash-5.2
Functionally Equivalent Command	$\hat{b} \in B$
Model Command	$b' \in B$
Docker Environment	$e \in E$ Linux
Command Output and Side Effects	$o \in O$ stdout and system state
Model Weights	$\theta \in \mathbb{R}$
Translation	$f : T \times \mathbb{R} \rightarrow B \quad f(t, \theta) = b'$
Execution	$g : B \times E \rightarrow O$ $g(b, e) = o \quad g(b', e) = o'$
Ideal Functional Equivalence Heuristic	$m : T \times B \times B \times O \times O \rightarrow \{0, 1\}$ $m(t, b, b', o, o') = \begin{cases} 1 : o \approx o' \\ 0 : o \neq o' \end{cases}$
Training Dataset	$D_T : \{(t_i, b_i) \mid i = 1, 2, \dots, x\}$
Test Dataset	$D_H : \{(t_i, b_i, \hat{b}_i) \mid i = 1, 2, \dots, y\}$ $m(t, b, \hat{b}, g(b, e), g(\hat{b}, e)) = 1 \quad \text{ideal } m$ $D_T \cap D_H = \emptyset$
Benchmark	$(D_H, m)$

# Chapter 2

## Related Work

### 2.1 Summary

NL2SH translation is a well-studied natural language processing (NLP) task. Table 2.1 summarizes the contributions of previous work by listing the names of datasets, functional equivalence heuristics (FEHs), and models used in this field. The table is sparsely populated because the majority of previous work focused on improving a NL2SH dataset, FEH, or model in isolation. Numerous NL2SH projects are simply wrappers for models in previous work, so we do not include them in our discussion [11–15, 24].

### 2.2 Related Work

The 2020 NeurIPS NLC2CMD Competition formalized the task of NL2SH translation by providing a human-curated NL2Bash dataset of 9,305 instruction-command pairs and the NL2CMD benchmark for evaluating submitted models [4, 25]. The competition resulted in numerous NL2SH models and showed that fine-tuning a pre-trained foundation model could outperform dedicated transformer [27], recurrent neural network [37], abstract syntax tree [29], and sequence to sequence [26] based models for the task of NL2SH translation [31].

The NL2CMD benchmarks’s FEH parses commands and assigns a similarity score based on the utilities used, order of utilities, and number of utility flags. This heuristic outperforms conventional string comparison techniques, such as edit distance, for measuring the functional equivalence of commands. However, Agarwal et al. [4] state that the NL2CMD FEH could be improved by executing commands and measuring the similarity of the outputs and side effects. Verification by execution is preferable because Bash is a Turing-complete language, so verifying the equivalence of two commands before execution is undecidable due to side

Table 2.1: Summary of NL2SH datasets, FEHs, and models created in previous work.

Citation	Datasets	FEHs	Models
Lin et al. [25]	NL2Bash	-	-
Gros [26]	-	-	AIInix
Agarwal et al. [4]	-	NL2CMD	Tellina
Fu et al. [27]	-	-	Magnum
Ramesh [28]	NL2CMD	-	-
Bharadwaj et al. [29]	-	-	AST
Jenson and Liu [30]	-	-	T5, GPT2
Shi et al. [31]	-	-	ShellGPT
Yang et al. [16]	InterCode-Bash	InterCode	-
Mali [32]	text_to_bash	-	-
Cassano et al. [33]	MultiPL-E	Unit Tests	-
Song et al. [17]	-	TSED	-
Aggarwal et al. [18]	CodeSift	CodeSift	-
Vo et al. [34]	IBM_Instana	Podman	-
Romit [19]	LinuxCmds	-	-
Chatterjee et al. [35]	-	-	ScriptSmith
Joshi [36]	-	-	CodeLlama
<b>Ours (2025)</b>	NL2SH-ALFA	IC-ALFA	Llama, Qwen, GPT

effects [38]. Despite this known shortcoming, the NL2CMD benchmark is widely used for model evaluations [39].

Yang et al. [16] address this shortcoming with the InterCode-Bash benchmark. The benchmark uses a subset of 224 instruction-command pairs from the NL2Bash dataset for testing. InterCode’s FEH executes the model command and ground truth command in identical Docker containers [40]. The results of command execution are then compared using three checks. First, the pre and post-execution states of each container are compared using git-diff. Second, the file contents of each container are compared using MD5 hashes. Third, the standard output of both commands are vectorized and compared using the term frequency, inverse document frequency (TFIDF) method [41]. If every check finds the execution results identical, the model and ground truth command are considered functionally equivalent [16]. Although this method is more accurate than previous heuristics, it will fail to identify a valid model command that has syntactically different output from the ground truth command.

Huang et al. [42] and Vijayaraghavan et al. [43] present similar execution-based frameworks for Jupyter Notebooks and VHDL code, respectively. Vo et al. [34] describe an execution-based framework similar to the InterCode benchmark using Podman containers.

Focusing on the FEH, Song et al. [17] present a novel benchmark that uses OpenAI’s GPT-4 model to determine functional equivalence. Their FEH passes the model command

and ground truth command to GPT-4 with the prompt, *"Given 2 Bash commands, please generate a similarity score from 0 to 1."* Song et al. [17] find this method fails to determine functional equivalence because current LLMs are unable to emulate command execution. Maveli et al. [45] and Naik [46] confirm this finding with broader evaluations of LLM's ability to determine semantically equivalent or different pairs of programs. They find LLMs show a lack of depth in understanding code semantics.

Aggarwal et al. [18] attempt to advance the work presented by Song et al. [17], by addressing the scalability constraints of execution-based frameworks. Their FEH, CodeSift, uses an LLM to convert the model command to a natural language description. Then they compare this natural language description with the original natural language task using an LLM. While they find CodeSift to be more effective than conventional FEHs, their work lacks comparison with execution based heuristics. Further, their FEH introduces uncertainty by requiring accurate Bash to natural language translation, which is equally as challenging as the task they aim to measure, natural language to Bash translation. Their follow on work concludes that *"executing [Bash] scripts within a controlled environment would offer more reliable assessments"* [35].

The MultiPL-E benchmark is widely used for evaluating code generation models, containing 540 Bash scripting tasks [33]. This benchmark uses unit tests to determine if a generated script produces the expected output for a given input. The use of unit tests is sufficient for this benchmark because its tasks, such as string manipulation and math calculations, are deterministic and result in simple outputs. This method fails to assess file manipulation, system administration and network management tasks because they produce more complex outputs than what can reasonably be assessed with unit tests.

Current SOTA NL2SH models use general purpose LLMs for translation. In practice, users can either accept, reject, or edit the model translation [11, 13]. A human-in-the-loop approach is necessary because the models may produce incorrect translations [47]. Efforts to improve model performance include fine-tuning and prompt engineering. Jenson and Liu [30] fine-tune the BART, T5, and GPT-2 models on the NL2Bash dataset and find model performance improves as measured by the NL2CMD benchmark. Joshi [36] conducts a similar study, fine-tuning the CodeLlama2 model on the NL2Bash dataset. Unfortunately, neither of these studies evaluate their models using a reliable execution-based benchmark.

A related area of research is Bash to natural language translation, the reverse of our task. Yu et al. [48] present BashExplainer and find that training their model with additional Bash context from internet forums improves the accuracy of translations. This finding is corroborated by Chen et al. [49] in their work on Bash comment generation. Bash to natural language translation is outside the scope of this thesis, but provides informative findings.

## 2.3 Advancements

Figure 2.1 summarizes the shortcomings of FEHs in previous work and our improvements. We begin by creating verified and expanded NL2SH datasets starting from multiple datasets presented in previous work. Next, we combine the InterCode execution FEH presented by Yang et al. [16] with the language model evaluation presented by Song et al. [17]. Using our new datasets and FEH, we evaluate methods for improving model performance in Section 3.3.

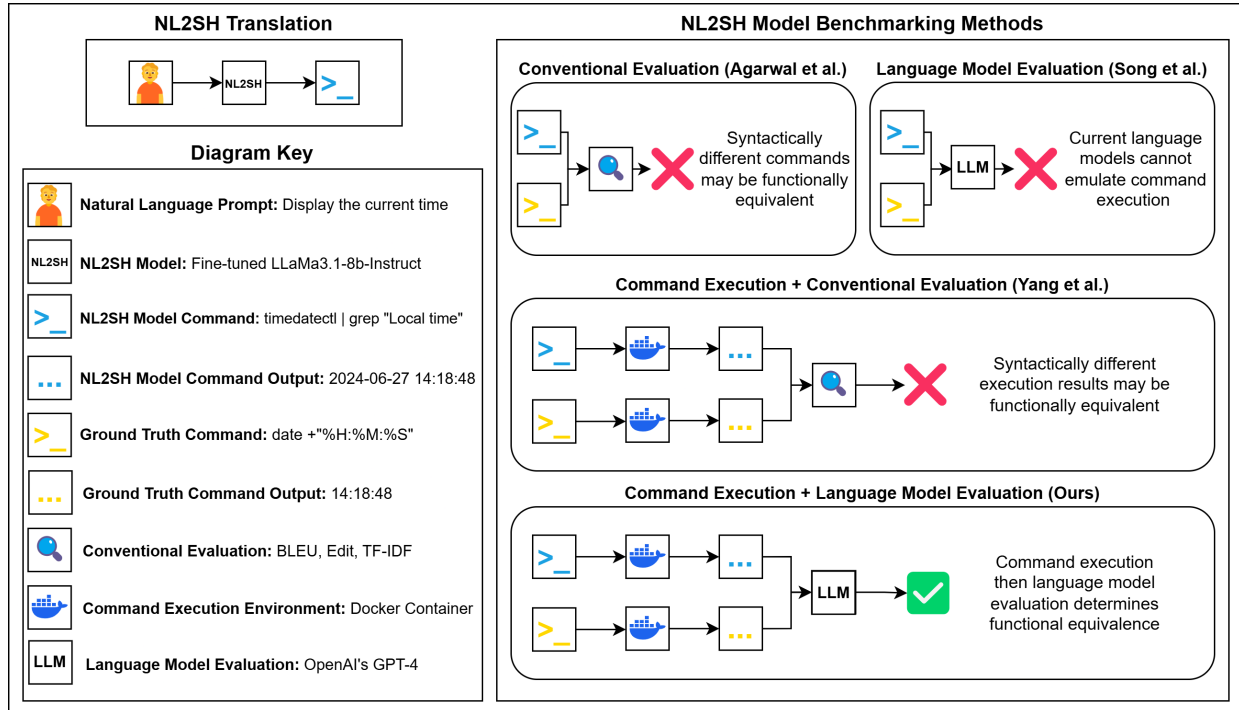


Figure 2.1: A diagram of NL2SH translation and a comparison of functional equivalence heuristics from previous work.

# Chapter 3

## Methodology

### 3.1 Dataset Validation

Bash is considered a low-resource programming language due to the limited availability of NL2SH data [50]. We aim to augment NL2SH datasets and begin with an evaluation of InterCode. All 224 commands in the InterCode dataset were manually curated from the NL2Bash dataset presented by Lin et al. [25], containing 9,305 instruction-command pairs. The InterCode dataset is significantly smaller than the NL2Bash dataset because a Docker environment is configured for each command, enabling execution. We manually verify all 224 instruction-command pairs and find that over half of the InterCode dataset is erroneous.

Table 3.1 shows the number of errors organized by type. We define three types of errors: invalid prompt, invalid command, and invalid environment. An invalid prompt error refers to a natural language instruction that describes an impossible task or does not give enough information to accomplish the task. An invalid command error refers to a Bash command that does not accomplish the task described in the prompt or does not execute. An invalid environment error refers to an incorrect Docker configuration such as a missing file, environment variable, or utility that prevents a valid command from accomplishing the task. Our manual verification reveals 102 instruction-command pairs with one or more errors and 11 duplicate pairs.

We fix 82 of these errors by correcting natural language prompts, Bash commands, and Docker configuration files. We remove 11 duplicate and 20 irreparable pairs from the dataset, resulting in 193 verified pairs. We create 117 more verified pairs by referencing Bash tutorials and books, such as *The Linux Command Line* by Shotts [1] and *The Linux Command Line and Shell Scripting Bible* by Blum and Bresnahan [51].

Additionally, for our 300 verified pairs, we create a second Bash command that accomplishes the task described in the prompt. Our final test dataset contains two functionally

Table 3.1: InterCode dataset errors.

<b>Error Type</b>	<b>Count</b>	<b>Percentage</b>
Duplicate	11	4.9%
Invalid Prompt	17	7.6%
Invalid Cmd	24	10.7%
Invalid Env	18	8.0%
Invalid Prompt and Cmd	29	12.9%
Invalid Prompt and Env	0	0.0%
Invalid Cmd and Env	3	1.3%
Invalid Prompt, Cmd and Env	11	4.9%
Invalid Total	113	50.4%
Valid Total	111	49.6%

equivalent, ground truth Bash commands for each natural language instruction, for a total of 600 instruction-command pairs. This is an increase of 441% over the 111 valid commands in the InterCode dataset. Our annotated corrections for the InterCode dataset errors can be found on HuggingFace<sup>1</sup>.

We collect training data by combining the NL2Bash dataset with three publicly available NL2SH datasets [19, 28, 32]. Further, we scrape the tldr-pages, a collection of example Bash commands, as a new data source [52]. We combine these data sources and deduplicate with exact matching. Finally, we use the bashlex parser to remove unparseable commands [53].

We de-conflict our training and test dataset using exact matching and semantic similarity, removing 917 rows from the training data. First, we remove rows from the training data that exactly match instructions or commands in the test data. Next, we remove pairs from the training data with a natural language prompt that is syntactically similar to a prompt in the test data using the mx-bai-embed-large-v1 embedding model and a cosine similarity threshold of 0.9 [54]. Our final training dataset contains 40,939 instruction-command pairs, an increase of 135% over the previous largest dataset. Figure 3.1 shows the relationships between the data sources used to create our datasets. Our final datasets can be found on HuggingFace<sup>2</sup>.

## 3.2 Functional Equivalence Heuristic

Our evaluation of related work in Section 2.2 reveals the InterCode benchmark is more accurate than previous NL2SH benchmarks because its FEH uses execution-based evaluation. However, its TFIDF method for comparing command outputs may fail to determine functional

<sup>1</sup>InterCode-Corrections HuggingFace

<sup>2</sup>NL2SH-ALFA HuggingFace

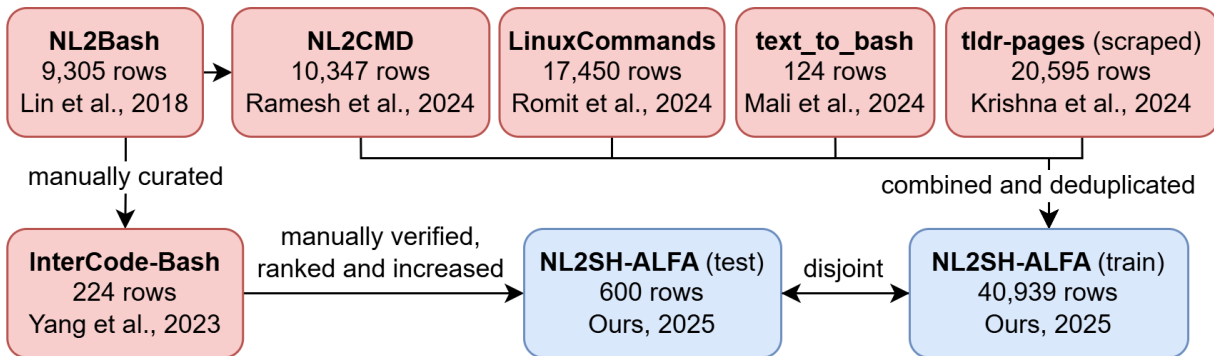


Figure 3.1: Relationships between NL2SH datasets.

equivalence because syntactically different outputs may convey the same information to the end user, conditioned on the user’s prompt.

For example, consider the prompt *"Print the disk usage of the current directory"*, a ground truth command of `"du -s ."` and a model command of `"du -d 0 -h"`. The first command outputs the number of bytes and the second command outputs the number of bytes in human-readable format. The two commands are functionally equivalent, conditioned on the prompt. However, their outputs contain different characters, resulting in a low similarity score using `TfidfVectorizer`. Similar issues arise when comparing commands that print hardware or system information in different formats, display text with line numbers or other delimiters, or when using non-deterministic utilities, such as those that interact with the network. The difficulty of determining functional equivalence is exacerbated by ambiguity in natural language prompts, which is an inherent problem with human inputs.

To address this problem, we replace the `TfidfVectorizer` method for comparing command outputs with an LLM. Our intuition is that an LLM can determine more complex cases of functional equivalence by evaluating the semantics of command outputs with relation to the prompt. Replacing `TfidfVectorizer` with an LLM increases the computational cost of the FEH. Additionally, since LLMs are stochastic, our FEH has inherent variability. We compare our FEH with previous heuristics in Section 4.2, finding it achieves superior performance.

We present our FEH and test dataset as a new version of the InterCode benchmark, InterCode-ALFA. Our benchmark and datasets are released under MIT licenses. In addition to the dataset and FEH modifications, we add error handling and update the Docker configuration files to use stable Linux releases. We also identify and fix an error in the benchmark’s Docker reset script that causes the filesystem structure of the two execution environments to diverge. We publish the benchmark source code on GitHub<sup>3</sup> and provide a

<sup>3</sup>InterCode-ALFA GitHub

Python package on PyPI<sup>4</sup> for ease of use. Our benchmark and dataset can be configured with 10 lines of code, simplifying the process for evaluating new models.

### 3.3 Translation Methods

Using our benchmark, we evaluate the NL2SH performance of the Llama, Qwen and GPT model families [44, 47, 55]. We find the models have poor baseline performance and identify three translation failure modes: incorrect output format, incorrect utility and syntactically incorrect Bash command.

Incorrect output format refers to a translation with extraneous information, such as an explanation of the translation, or additional text formatting, such as markdown code blocks. Incorrect utility refers to a translation with a utility that cannot accomplish the task described in the prompt. Syntactically incorrect Bash command refers to a translation that is not valid Bash syntax.

To address these failure modes, we evaluate four methods for improving model performance: markdown parsing 3.3.1, constrained decoding (CD) 3.3.2, in-context learning (ICL) 3.3.3, and in-weight learning (IWL) 3.3.4. Our results are presented in Section 4.3.

#### 3.3.1 Markdown Parser

Despite prompting models with *"You will not output markdown or other formatting"*, translations often include markdown formatting, likely due to instruct tuning. We implement a markdown parser to extract the Bash command from the first code block in model outputs, discarding additional text.

#### 3.3.2 Constrained Decoding

We inspect the token probabilities for each ground truth Bash command in our test dataset using the Llama3.1-8b-Instruct model. We find the average relative probability of the first token is four orders of magnitude smaller than the following tokens. In our case, the first token of each command is a Bash utility. This indicates the model is unlikely to select the correct utility as the first token. However, if it does select the correct utility, the following flags and arguments are correct with high probability. We address this by constraining the first tokens of the model output to a list of Bash utilities using grammar-constrained decoding Geng et al. [56].

---

<sup>4</sup>InterCode-ALFA PyPI

### 3.3.3 In-Context Learning

In-context learning can improve model performance for a variety of tasks [57, 58]. We select 50 representative instruction-command pairs from our training dataset as ICL examples. We create embeddings for the commands using the mxbai-embed-large-v1 model and cluster the embeddings using k-means clustering. The closest instruction-command pair to each centroid is selected as an ICL example. We append these pairs to our translation prompt as show in Figure A.5. We evaluate the performance of Llama3.1-8b-Instruct with the number of appended pairs ranging from 1-50 and find the optimal number to be 25, with performance saturating as more pairs are added. We use 25 example instruction-command pairs for all ICL evaluations.

### 3.3.4 In-Weight Learning

We use our training dataset to perform a LoRA fine-tune of the Llama and Qwen models [59]. Our previous work finds that fine-tuning models with additional context, such as manual pages, decreases downstream performance [60]. Based on this finding, we do not include additional data in our fine-tuning.

We experiment with common hyper-parameters within our hardware constraint of a single Nvidia RTX A6000. We find that training each model for 10 epochs with an adapter rank of 64, adapter alpha of 32, adapter dropout of 0.1, batch size of 32 and learning rate of  $1e - 5$  results in the best performance. We do not fine-tune the GPT models due to financial constraints and the inability to control training hyper-parameters.

# Chapter 4

## Experiments and Results

### 4.1 Dataset Validation

The impact of cleaning our training dataset and manually verifying our test dataset is encompassed in the results of our translation methods experiment in Section 4.3. First, if we improved the quality of test data, then baseline model performance on our InterCode-ALFA benchmark should be higher than on the original InterCode benchmark, holding all other variables equal. This is because models will no longer be penalized for failing to accomplish incorrect test cases. Second, if we improved the quality of training data, then in-weight learning will result in higher model performance. This is because model training improves with more accurate examples.

Table 4.1: The baseline accuracy of the GPT-3.5 and GPT-4 models on the InterCode and InterCode-ALFA benchmarks. The intersection of models tested on each benchmark only includes two models. We find the baseline accuracy of both models on the InterCode-ALFA benchmark is significantly higher than on the InterCode benchmark, indicating our verification process improved the quality of test data.

Model	Baseline Accuracy	
	InterCode	InterCode-ALFA
gpt-3.5-t-0125	0.34	0.58
gpt-4-0613	0.48	0.68

### 4.2 Functional Equivalence Heuristics

We compare our FEH with the heuristics presented by Sparck Jones [41], Papineni et al. [61], Agarwal et al. [4], Yang et al. [16] and Song et al. [17] in previous work using our test dataset. A FEH should return true given two functionally equivalent Bash commands, and false given

two non-equivalent Bash commands. We record the precision, recall, F1 score, accuracy and Runtime of each FEH and report our results in Table 4.2.

Our test dataset, described in Section 3.1, is structured  $\{\text{nl}, \text{bash}, \text{bash2}\}$ , providing 300 pairs of functionally equivalent commands. To create a set of non-equivalent commands, we arbitrarily rotate the third column of the dataset by ten positions. The result is 600 pairs of Bash commands. Explicitly, each FEH is tested using 300 functionally equivalent pairs  $m(t, g(b, e), g(\hat{b}, e)) = 1$  where  $g(b, e) \approx g(\hat{b}, e)$  and 300 non-equivalent pairs  $m(t, g(b, e), g(\hat{b}, e)) = 0$  where  $g(b, e) \neq g(\hat{b}, e)$ .

For the bleu and nl2cmd FEHs we use a threshold of 0.75 for functional equivalence. For the tfidf and mxbai-embed FEHs we calculate the cosine similarity of the resulting embeddings and use a threshold of 0.75 for functional equivalence. For the llama-3.1-8b-inst, gpt-3.5-t-0125 and gpt-4-0613 FEHs we pass the tasks and commands to each model using the prompt in Figure A.1. For the exec + tfidf and exec + mxbai-embed FEHs, we pass the stdout of command execution to the models, calculate the cosine similarity of the resulting embeddings and use a threshold of 0.75 for functional equivalence. Finally, for the exec + llama-3.1-8b-inst, exec + gpt-3.5-t-0125 and exec + gpt-4-0613 FEHs, we pass the tasks, commands and stdouts of command execution to each model using the prompt in Figure A.2. We use a temperature of zero and a static seed value of 123 for all LLMs, resulting in deterministic outputs. Table 4.2 lists our results. Figure 4.1 and Figure 4.2 are visualizations of the data from the table. We discuss these results in Chapter 5.

Table 4.2: Evaluation of Bash functional equivalence heuristics. Heuristics were tested on a dataset comprising 300 pairs of equivalent commands and 300 pairs of non-equivalent commands. We find execution paired with LLM evaluation significantly increases recall. Bold indicates the highest F1 score and accuracy as well as the lowest evaluation time.

Heuristic	Precision	Recall	F1	Accuracy	Time (s)
nl2cmd [4]	0.98	0.20	0.33	0.60	1.55
bleu [61]	0.99	0.39	0.56	0.69	<b>0.13</b>
tfidf [41]	0.99	0.46	0.63	0.73	0.36
exec + tfidf [16]	0.99	0.65	0.79	0.82	267.54
mxbai-embed [54]	0.84	0.88	0.86	0.85	8.31
exec + mxbai-embed ( <b>Ours</b> )	0.97	0.83	0.90	0.90	276.24
llama-3.1-8b-inst [55]	1.00	0.05	0.10	0.53	601.46
exec + llama-3.1-8b-inst ( <b>Ours</b> )	0.88	0.74	0.80	0.82	505.72
gpt-3.5-t-0125 [57]	1.00	0.37	0.54	0.69	272.06
exec + gpt-3.5-t-0125 ( <b>Ours</b> )	0.98	0.60	0.75	0.80	489.88
gpt-4-0613 [17]	1.00	0.51	0.68	0.76	417.92
exec + gpt-4-0613 ( <b>Ours</b> )	0.99	0.91	<b>0.95</b>	<b>0.95</b>	588.42

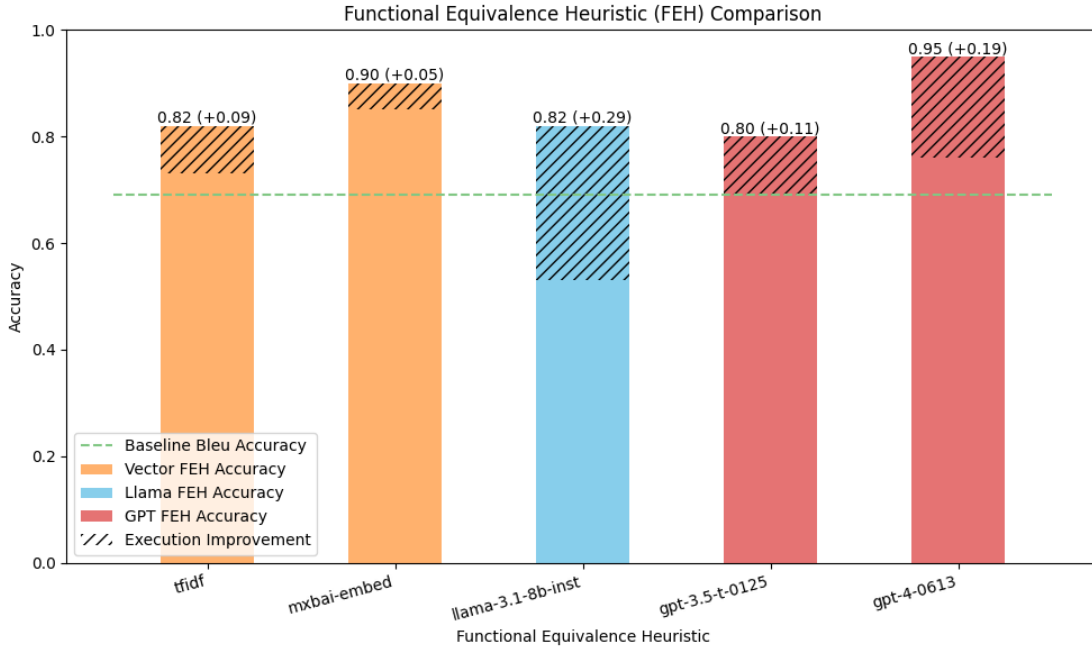


Figure 4.1: Evaluation of Bash functional equivalence heuristics. The hashed section of each bar indicates the accuracy improvement from passing command execution outputs to the heuristic, as opposed to just the commands. We find execution allows all of our heuristics to achieve better accuracy than the baseline Bleu accuracy.

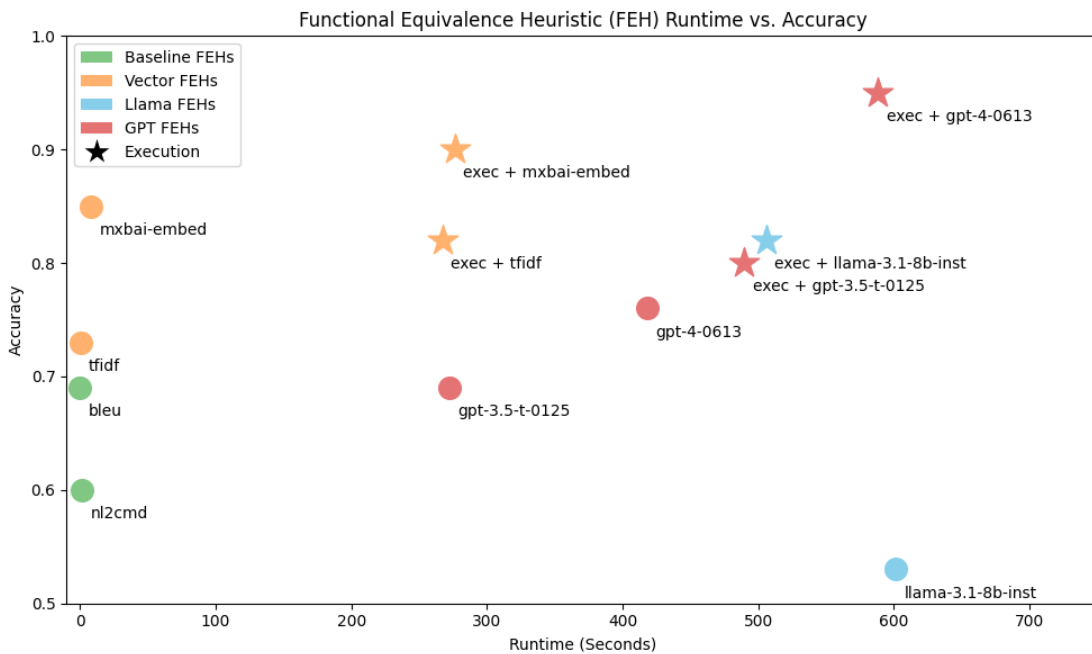


Figure 4.2: Bash functional equivalence heuristic runtime versus accuracy. The use of an execution environment and an LLM both increase the computational cost of a heuristic. We find the exec + mxbai-embed heuristic achieves a good balance of runtime and accuracy.

### 4.3 Translation Methods

We evaluate the impact of parsing, constrained decoding, in-context learning and in-weight learning on the NL2SH performance of the Llama, Qwen and GPT model families. Additionally, we include the performance of Anish Joshi’s Bash Scripting Assistant, a fine-tuned version of CodeLlama model [62], to demonstrate how our methods compare to previous work [36]. All models are evaluated using version 0.3.6 of the InterCode-ALFA benchmark with the execution + mxbai-embed FEH. Our results are summarized in Table 4.3.

We use the prompt in Figure A.3 for the baseline evaluation. For the constrained decoding evaluation, we use the prompt in Figure A.4 and constrain the first tokens of the model output to a list of Bash utilities, as described in Section 3.3.2. For the parser evaluation, we use the prompt in Figure A.4 and pass model outputs to a markdown parser, as described in Section 3.3.1. For the in-context learning evaluation, we use the prompt in Figure A.5. Finally, for the in-weight learning evaluation, we use the prompt in Figure A.4 and the fine-tuned models described in Section 3.3.4. We use a temperature of zero and a static seed value of 123 for all model evaluations, resulting in deterministic outputs. Table 4.3 lists our results. Figures 4.3, 4.4, 4.5, and 4.6 are visualizations of the data from the table. We discuss these results in Chapter 5.

Table 4.3: Impact of constrained decoding, parsing, in-context learning and in-weight learning on the NL2SH performance of Llama, Qwen and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Translation method can improve performance up to 32% over the baseline, but model size remains the dominant factor. The highest accuracy in each row and column is indicated with bold and an underline, respectively. The CodeLlama model is included as a baseline from previous work.

Model	Base	CD	Parse	ICL	IWL
llama-3.2-1b-instruct	0.12	0.19	0.32	0.34	<b>0.37</b>
llama-3.2-3b-instruct	0.17	0.39	<b>0.49</b>	0.47	0.47
llama-3.1-8b-instruct	0.46	<u>0.51</u>	0.53	<b>0.56</b>	0.40
qwen2.5-coder-0.5b-instruct	0.10	0.05	0.35	<b>0.36</b>	0.27
qwen2.5-coder-1.5b-instruct	0.21	0.06	<b>0.50</b>	0.44	0.19
qwen2.5-coder-3b-instruct	0.26	0.06	<b>0.58</b>	0.50	<u>0.51</u>
qwen2.5-coder-7b-instruct	0.61	0.08	<b>0.62</b>	0.62	<u>0.51</u>
gpt-3.5-turbo-0125	0.58	-	0.67	<b>0.69</b>	-
gpt-4o-mini-2024-07-18	0.71	-	<b>0.72</b>	0.71	-
gpt-4o-2024-08-06	<u>0.74</u>	-	<u>0.73</u>	<u>0.73</u>	-
gpt-4-0613	0.68	-	0.68	<b>0.73</b>	-
code-llama-7b [36]	<b>0.54</b>	-	-	-	-

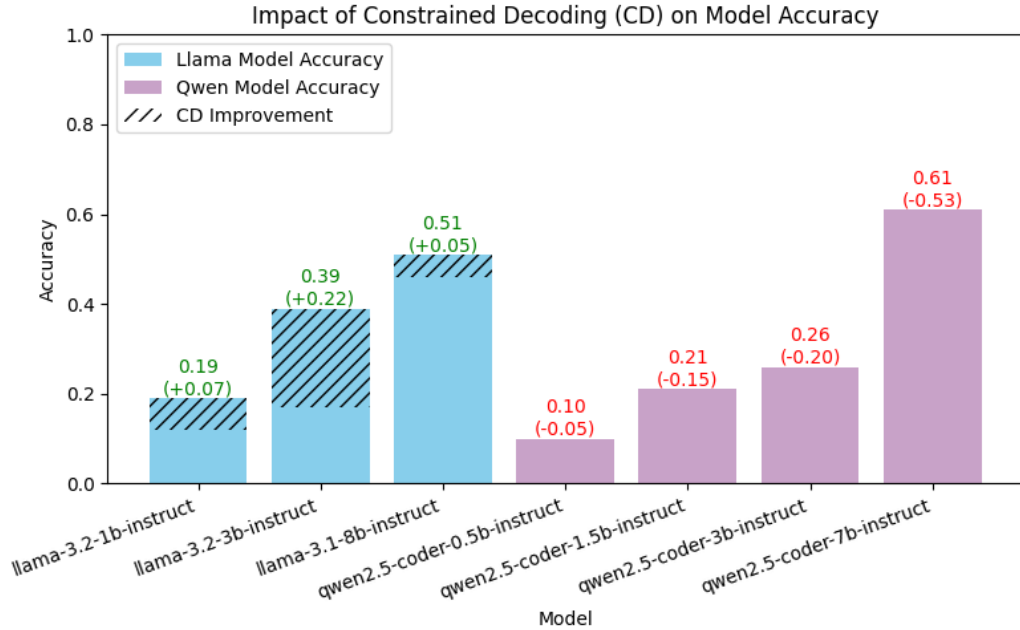


Figure 4.3: Impact of constrained decoding on the NL2SH performance of Llama and Qwen models. Accuracy is measured using the exec + mxbai-embed FEH. Constrained decoding improves the performance of the Llama models but significantly degrades the Qwen models.

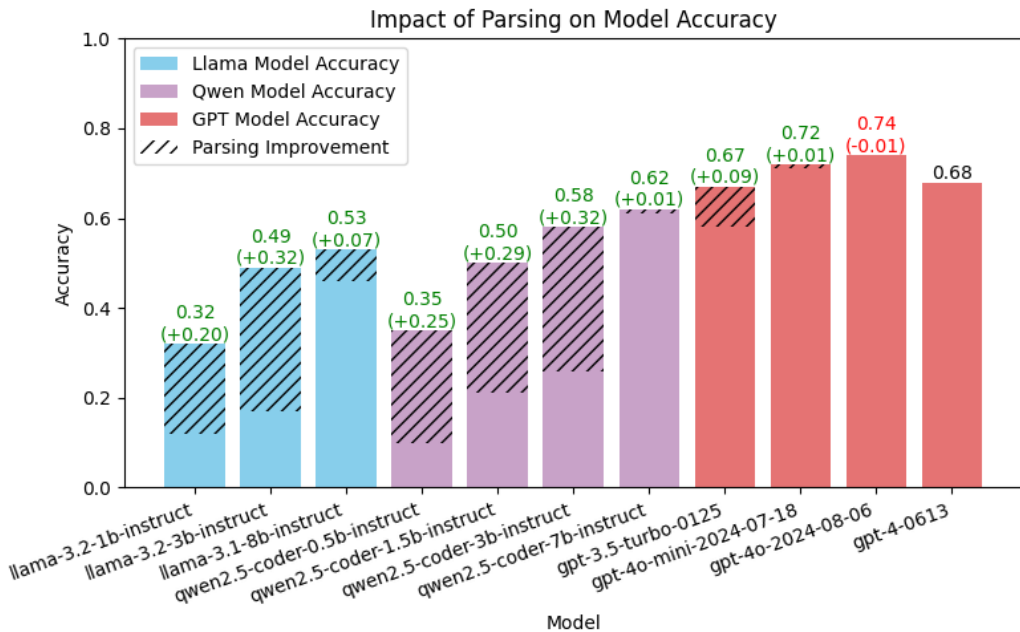


Figure 4.4: Impact of parsing on the NL2SH performance of Llama, Qwen, and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Parsing improves the performance of all models except for GPT-4 and GPT-4o. Parsing has the largest impact for models in the 0.5B to 3B parameter range.

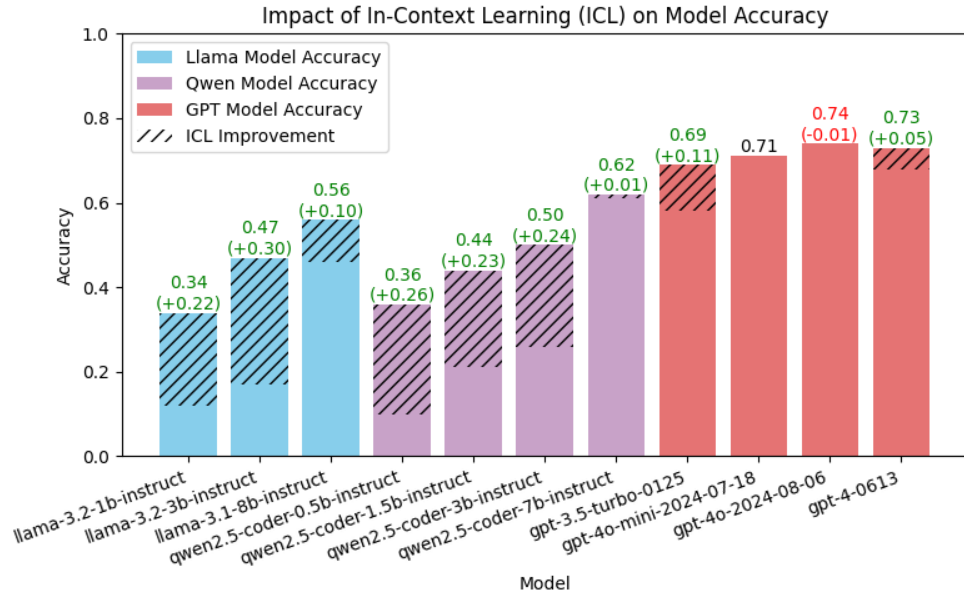


Figure 4.5: Impact of in-context learning on the NL2SH performance of Llama, Qwen, and GPT models. Accuracy is measured using the exec + mxbai-embed FEH. Parsing improves the performance of all models except for GPT-4o-mini and GPT-4o. ICL has the largest impact for models in the 0.5B to 3B parameter range.

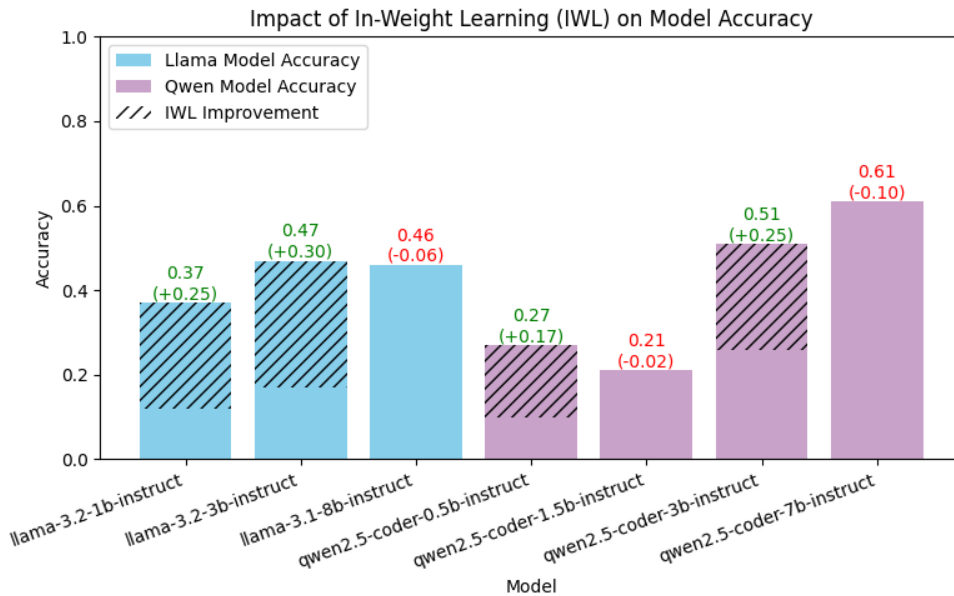


Figure 4.6: Impact of in-weight learning on the NL2SH performance of Llama and Qwen models. Accuracy is measured using the exec + mxbai-embed FEH. IWL significantly improves the performance of the 1B and 3B parameter Llama models, but otherwise has inconsistent results, likely due to our hardware constraints limiting the size of LoRA adapters.

# Chapter 5

## Discussion

### 5.1 Dataset Validation

Our first research question aims to validate NL2SH datasets to ensure models are evaluated using valid translations. Manual verification of the InterCode dataset identified over half of the instruction-command pairs as erroneous. Using our corrected version of the InterCode dataset, the gpt-3.5-t-0125 and gpt-4-0613 models achieve 22% and 20% higher accuracy, respectively, as shown in 4.1. Since we hold all other benchmark and model variables equivalent, this increase in performance can be attributed to our corrected test dataset. The models are able to achieve better performance because they are no longer penalized for failing incorrect test cases. We find that human verification of test data is important for reliable evaluations.

Since the InterCode dataset was sampled from the NL2Bash dataset, and the InterCode dataset contained a large percentage of errors, there is a risk the NL2Bash dataset also contains a significant percentage of erroneous data. This is concerning because the NL2Bash dataset is commonly used to train NL2SH models [27, 29, 31, 36, 37].

Manual creation and verification of our test dataset took over 100 hours, highlighting the need for more efficient means to verify larger datasets, such as our training dataset. One interesting method for automated verification suggested by Vendrow et al. [63] is the use of model consensus. For each case in a dataset, if a broad set of models arrive at the same translation, then the translation is more likely to be correct. If the models disagree, then the translation is either a difficult case, or it is incorrect. This method could be used to reduce the number of cases that require manual verification. We suggest this method as means to verify our training dataset in future work.

With regard to our training dataset, we are confident the data filtering process described in Section 3.1 removed invalid translations. Further, we postulate the tldr-pages is a large source of clean data due to its crowd sourced nature. Fine-tuning the Llama and Qwen

models using our training dataset results in an average performance increase of 11%. This is evidence that our training data contains valid translations. If the training data was of poor quality, then we would expect fine-tuning to degrade model performance on average. Further efforts to validate NL2SH training datasets would be a valuable contribution to the field.

## 5.2 Functional Equivalence Heuristics

Our second research question aims to design an FEH that accurately measures the quality of model translations. We find that command execution paired with LLM evaluation of command outputs can determine the functional equivalence of Bash commands with 95% accuracy. The ability of LLMs to condition command outputs on natural language inputs is an advancement that was not possible with previous heuristics. Further, command execution improves performance across methods, and the use of an LLM significantly increases recall. Broadly, LLM evaluation of execution outputs is a promising advancement for measuring the functional correctness of generated code and more investigation is warranted.

In accordance with Maveli et al. [45], we find that without execution, current LLMs are poor arbiters of functional equivalence, achieving similar performance when compared to conventional evaluation methods. This is because current LLMs fail to understand code semantics [64, 65]. Non-execution methods fail because two commands can be syntactically similar and yield different results when executed. For example, changing a single flag can result in vastly different command outputs. Further, two commands with no syntactic similarity can yield identical results when executed. For example, the `awk` and `sed` utilities can accomplish identical text processing tasks but use different domain-specific languages, requiring different syntax. Notably, the low recall of `bleu` and `nl2cmd` FEHs indicates these methods cannot identify cases where syntactically different commands are functionally equivalent.

Our improvements to FEH accuracy come at the cost of runtime, as shown in Figure 4.2. The use of an execution environment and an LLM both increase the computational cost of a heuristic. Command execution incurs a fixed cost, while LLM evaluation incurs a variable cost that increases as the size of the evaluation model increases. One bottleneck with our execution environment is that Docker containers must be run sequentially. This is because a container must be reset after each command to ensure that the state of the system is not altered by previous commands.

An improvement in future work could be to design an execution environment that allows for parallel execution of containers. This would reduce the time required to run our benchmark. We find the `exec + mxbai-embed` heuristic achieves a good balance of runtime and accuracy. Further, the `mxbai-embed` model can be run locally on our hardware and does not incur

additional monetary cost due to the use of an API. For these reasons, we use the `exec + mxbai-embed` heuristic for our model evaluations.

## 5.3 Translation Methods

Our third research question aims to improve the accuracy of NL2SH models as measured by a reliable benchmark. We find that constrained decoding, parsing, in-context learning and in-weight learning can improve model performance by up to 32%. Our evaluation shows that the number of model parameters has the largest impact on performance.

We find that constrained decoding is model dependent, with performance increases for Llama models and significant performance decreases for Qwen models. Parsing and ICL provide performance increases across Llama and Qwen models, with average increases of 21% and 19%, respectively. However, these methods have a decreasing impact as model size increases. This is evidence that incorrect output format is the dominant failure mode for small (less than 7 billion parameter) models.

With IWL, `llama-3.2-3b-instruct` and `qwen2.5-coder-0.5b-instruct` achieve the baseline performance of `llama-3.1-8b-instruct` and `qwen2.5-coder-3b-instruct`, respectively. Despite performance gains for small models, fine-tuning decreases the performance of `llama-3.1-8b-instruct` and `qwen2.5-coder-7b-instruct`. This is likely due to computational constraints on the size of our LoRA adapters, which we are unable to scale with model size.

Due to the strong correlation between model size and performance, our most significant results are ones where a translation method allows a model to outperform another model in a larger size class. Notably, with in-weight learning, the accuracy of `qwen2.5-coder-0.5b-instruct` is higher than the baseline accuracy of `qwen2.5-coder-3b-instruct`, a model six times larger in size. Another notable result is that the accuracy of `qwen2.5-coder-3b-instruct` with parsing is higher than the baseline accuracy of `code-llama-7b` and equal to the baseline accuracy of `gpt-3.5-turbo-0125`, which is rumored to be much larger in size.

We find that `gpt-4o-2024-08-06` achieves SOTA performance on our benchmark, correctly translating 74% of test cases. From the total of our experiences, we find that NL2SH translation is a difficult task for current models. A SOTA accuracy of 74% means that roughly one in four commands is incorrect. In jobs where Bash commands are used frequently, such as software engineering and research, this is unusably low performance. This low performance is exacerbated by the risk of incorrect commands, as detailed in Section 5.4. Further improvements to model accuracy are necessary before models can be used in practice.

## 5.4 Ethical Considerations

Due to the low performance of current NL2SH models, using these models in practice could result in invalid commands that have unintended effects on a system. We recommend that model-generated commands are never used without prior human verification. Further, we recommend that users test commands using a sand-boxed environment, such as `try` [66], before running them on personal systems. Figure 5.1 shows an example of a dangerous command observed during model testing. The hallucinated `rm -f /dev/null` command corrupted our benchmark’s Docker container. While our benchmark automatically creates a new Docker container to handle this type of error, the command could have corrupted a user’s system.

---

**Natural Language Prompt:**

*Recursively remove all empty folders from the /system/temp folder.*

---

**Ground Truth Command:**

```
find /system/temp -type d -empty -delete
```

---

**Llama3.1-8b-Instruct Command:**

```
find /system/temp -type d -empty -delete -print; rm -f /dev/null 2>&1
```

---

Figure 5.1: Dangerous translation observed in testing.

Natural language to Bash translation aims to increase computer accessibility by simplifying interactions with the command-line interface. Unfortunately, good and bad actors can benefit from increased accessibility. Models could be used to generate malicious Bash commands. This risk is difficult to mitigate because malicious use depends on the intent of the user. For example, a command to delete files could be used for a legitimate or harmful purpose. We do not believe current NL2SH models pose any risks beyond those of other readily available Bash resources.

## 5.5 Limitations and Future Work

This work presents a verified and expanded NL2SH test dataset. However, due to the time and effort required to manually configure an execution environment for each command, the dataset remains small, with only 600 test cases. Future work could combine data from other sources to increase the size of our test dataset [7, 34].

In contrast to the small size of our test dataset, our training dataset is too large for manual verification, and we are unable to guarantee its correctness. Future work could consider automated methods for verifying the correctness of training data, such as the model

consensus method described in Section 5.1. Further, future work could explore the feasibility of using synthetic training data.

Our datasets are limited to English prompts and one-line Bash commands. We do not consider other natural languages or scripting languages. Future work could explore the translation of other scripting languages, such as PowerShell. They could also consider multi-line scripts by treating this as a more general natural language to code translation task. Further, by translating our English prompts into other natural languages, future work could expand the accessibility of our datasets.

Due to the structure of datasets and the formulation of our translation task, our models do not output an explanation of the produced Bash command, nor support conversational interactions. In practice, a conversational model could enable a more natural experience. For example, the LLM could ask a clarifying question or the user could ask the LLM to modify a translation. Future work could explore conversational interactions with NL2SH models.

Although improved over previous methods, our functional equivalence heuristic has inherent variability due to the use of an LLM, requiring multiple runs to assess model performance or seeding to make the model deterministic. The use of an LLM also increases the computational cost of running our heuristic compared to conventional methods. Future work could explore methods for increasing the speed and scalability of the execution environment or LLM used by the heuristic, as discussed in Section 5.2. Future work could also apply our execution + LLM evaluation heuristic design to other machine translation tasks.

A different approach for natural language to code translation uses a benchmark that provides feedback to a model [67]. We only evaluate the first command generated by a model without retries or feedback because this is how a translation model is used in practice. However, future work could explore the impact of a feedback-based benchmark.

Despite improved model performance with constrained decoding, parsing, ICL and IWL, the accuracy of LLMs for NL2SH translation remains low. As discussed in Section 5.3 performance improvements are needed before models can be safely used in practice. Future work could evaluate the performance of reasoning models, such as OpenAI’s o1 and DeepSeek’s r1 [68, 69], on the task of NL2SH translation. Further, reasoning models could be trained specifically for both our translation and functional equivalence tasks.

Lastly, model accuracy is not the only consideration for NL2SH translation, efficiency is also important. Future work could for evaluate methods for improving model memory use and translation speed, such as quantization [70].

# Chapter 6

## Documentation Policy

### 6.1 Background

One application for NL2SH translation models is military cyber operations. Within the Department of Defense (DoD), members of the cyberspace operations community use Bash for tasks including network administration, cyber capabilities development, penetration testing, vulnerability assessment, incident response, and forensics [71, 72]. Broadly, Large Language Models (LLMs) are valuable tools for a wide range of military tasks [73]. Safe use of these models, and the datasets used to create them, can be promoted by providing documentation with user guidance [74]. Despite providing general guidance for the safe use of LLMs, at this time the DoD does not have a documentation policy for models or datasets [75–78].

Language model documentation falls under the broader domain of AI safety. In the DoD, the use of AI in cyberspace is governed by a hierarchy of documents. The overarching policy is Executive Order 14110 on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence [79]. The publication of Executive Order 14110 resulted in the development of the Data, Analytics, and Artificial Intelligence Adoption Strategy as well as the Defense Industrial Base Adoption of Artificial Intelligence for Defense Applications [80, 81]. These three documents provide broad guidance for the safe development and use of AI tools in military applications. Based on their guidance, the Chief Digital and Artificial Intelligence Office developed the Responsible Artificial Intelligence Strategy and Implementation Pathway, which identifies specific lines of effort (LOEs) to meet the DoD’s AI safety goals [75]. Figure 6.1 displays the hierarchy of AI policy documents [79–82].

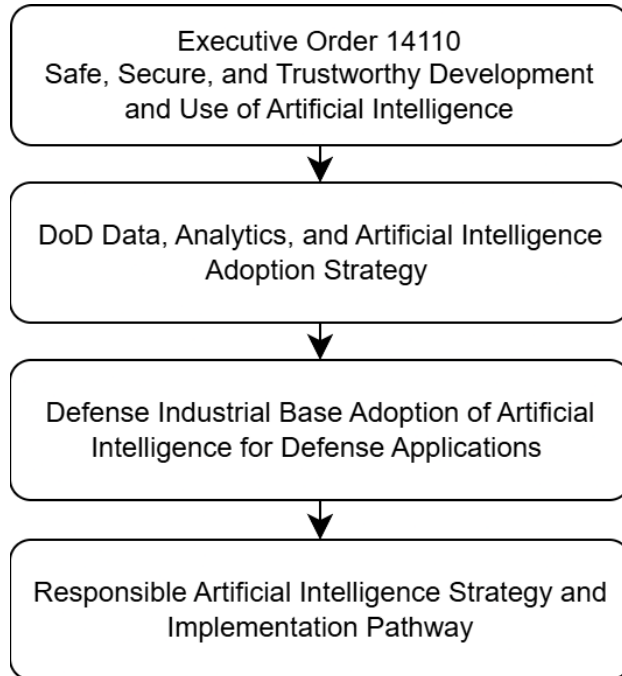


Figure 6.1: Hierarchy of DoD AI policy documents.

Civilian models and datasets provide documentation using the standardized framework of model and dataset cards. These cards (also referred to as system cards, datasheets, factsheets, and AI nutrition labels) provide information including the developers, license, size, modality, data source, benchmark results, intended use, out-of-scope use, biases, risks, limitations, and environmental impact of datasets and models [83, 84]. The DoD recognizes that civilian model and dataset cards can help address limitations of its AI policy. Line of effort 3.2.3 of the Responsible AI Strategy and Implementation Pathway states *"use AI Data and Model Cards to publish AI data assets"* [82]. Currently, the DoD relies on AI documentation frameworks provided by the Aether datasheet template and the HuggingFace model and dataset card templates [83–85]. However, these civilian frameworks do not address the increased consequences of errors when using AI in high-risk military environments, such as the use of LLMs in cyber warfare.

The challenge of developing documentation guidelines for datasets and models is a well studied problem. Creating documentation standards is challenging due to the broad range of uses for AI models and datasets [86]. Despite this challenge, adopting standards is necessary to promote accessibility and to clarify the intended use of models and datasets [87]. Bender et al. find that dataset documentation can alleviate issues related to exclusion and bias [88]. Gebru et al. find that dataset documentation facilitates communication and prioritizes transparency and accountability [89]. Arnold et al. find that model documentation with fairness and explainability sections can promote safe model use [90]. Pushkarna et al. highlight

the necessity of documentation on the origins, development, intent, and ethical considerations of data and models in high-risk domains [91].

A number of documentation templates are available [83, 84, 92], as well as toolkits to simplify the process of populating these templates [93, 94]. Currently, the most popular documentation frameworks for models and datasets are the HuggingFace model card and dataset card [86]. Despite the number of approaches for model and dataset documentation, no previous work provides a documentation framework that can be directly adopted by the DoD due to heightened military safety considerations.

Concisely, the open issues of DoD policy regarding language models and datasets are the following. (1) The DoD does not have policy for documentation of AI models or datasets. (2) Current civilian standards for model and dataset documentation do not address unique military safety considerations. Based on these problems, we aim to answer the following research question. How can the Department of Defense create a documentation policy for language models and datasets that promotes safe use in military applications? The following section discusses one potential path the Department of Defense could take for developing a military-specific documentation policy for language models and datasets.

## 6.2 Military Documentation Policy

### 6.2.1 Overview

The DoD uses LLMs for a variety of information processing tasks. Under the DoD’s Data, Analytics, and Artificial Intelligence Adoption Strategy, model documentation must be shared in a federated data catalog to promote safe use. The scientific community disagrees on the level of technical information required to determine if a model is safe for a task. Under-sharing technical information limits interpretability, while over-sharing poses a risk of exposing sensitive information. This technical tension has led to a policy impasse regarding model documentation standards. The DoD’s interim use of industry documentation standards deepens this controversy due to differing incentives for industry and military information sharing. We advise the DoD to partner with the federally funded research and development centers. This trusted research community can review models, advise on documentation, and find a pareto optimal tradeoff between between data sharing and risk. We recommend DoD documentation policy require justification of sanitized model information, balancing information assurance with the necessary level of transparency for the safe use of LLMs in military applications. Finally, we present documentation for our NL2SH models and datasets to illustrate how our recommendations can be applied in practice.

## 6.2.2 Technical Knowledge: Language Model Safety

LLMs are powerful tools for information processing tasks, such as summarization, code completion, and question answering [95]. The DoD harnesses these models for a variety of national security applications, including intelligence analysis, information retrieval, document summarization and language translation [96]. The DoD's use of AI for tasks other than information processing, such as autonomous systems, is outside the scope of this thesis.

Due to the sensitive nature of certain DoD tasks, the safety of language models is of heightened concern in military environments [79]. For example, LLMs have a tendency to 'hallucinate' and generate false information, which could compromise the integrity of a summary or translation [97]. The potential harm of a hallucination is greater when the information is acted upon in a military environment, which could result in loss of life, equipment, or strategic advantage. Due to increased potential for harm, the DoD is working to ensure a high standard of safety for LLM use, as outlined in the Responsible Artificial Intelligence Strategy and Implementation Pathway [82].

The question of LLM safety is technical in nature because it requires an understanding of a model's capabilities and limitations. The current industry standard for showcasing model capabilities is to demonstrate high performance on downstream tasks [55]. For example, OpenAI demonstrates the multilingual capabilities of the GPT-4 model by reporting performance on the Multilingual Massive Multitask Language Understanding benchmark [44, 98]. Broadly, this definition of safety relies on model accuracy on a benchmark within some threshold.

However, members of the scientific community argue that a low failure rate on test cases is insufficient for determining model safety. They contend that model interpretability is necessary to understand why a model performs well on certain tasks and poorly on others. Further, they show that interpretability can identify potential biases and edge cases that remain hidden in benchmarks [99, 100]. This definition of safety relies on understanding the model's internal mechanisms and training data. Moreover, this definition requires greater depth in documentation than the performance-based definition of safety.

An interpretability-based safety standard is best aligned with the DoD's goals for responsible AI use. The Political Declaration on Responsible Military Use of Artificial Intelligence and Autonomy commits to models that are "*developed with methodologies, data sources, design procedures, and documentation that are transparent to and auditable by their relevant defense personnel*" [101]. This commitment aims to avoid 'black box' models, where model inner-workings are opaque to users [102]. Despite general consensus on an interpretability-based definition of safety in the military, the depth of documentation needed to meet this standard is a point of contention.

### 6.2.3 Technical Tension: Interpretability and Security

To understand how LLM documentation varies, we examine practices outside the military community. OpenAI makes their current models accessible to the public via an application programming interface (API) and releases an explanation of model capabilities with examples, performance on benchmarks and external evaluations [103]. Meta releases similar information for their llama model family, and additionally, releases the training code, architecture and weights of their models [55]. Exhaustive documentation is common practice for the academic community, but far fewer models are created by academia than industry [104, 105].

Based on these practices, we define the following documentation continuum, ranging from the least to the most complete documentation: examples of capabilities → performance on benchmarks → external evaluation reports → black box model access → training code → model architecture and parameters → training data. Figure 6.2 illustrates this continuum, using the NIPR-GPT [106], ReALM [107], Sparrow [108], GPT-4 [44], AlphaFold [109], Llama3 [55], and OLMo2 [105] models as examples. Typically, documentation is cumulative.

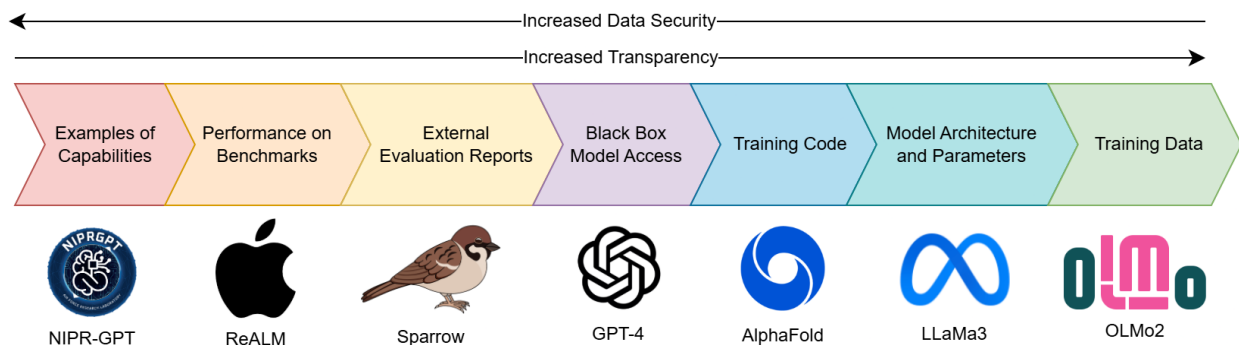


Figure 6.2: Model documentation continuum.

The technical question of model safety can be restated as: which point on this continuum is necessary to adequately characterize a model used for a particular task? Returning to the military context, under-sharing and over-sharing information at the extremities of this continuum both pose risks. Under-sharing, such as only releasing claims of model capabilities without evidential examples or benchmark performance, may lead to misapplications and fail to meet the goal of safe use. Over-sharing information, such as releasing training data and model weights, could expose sensitive information. The problem of exposing sensitive information is notable because DoD information handling is nuanced.

In the DoD, information exists within organizational boundaries, also known as *"silos"*. Personnel within an organization can access information in their silo, but not in others. Generally, information is restricted to those who require it for their job [80]. Training language models requires massive amounts of data [57]. In current DoD use cases, this

requires collecting data across organizational boundaries [106]. The resulting model is accessible to many users, but each user is only authorized to access a small subset of the training data, if any at all. Conceptually, this is similar to a multi-organization report, where each organization contributes findings from their data to a final report. The final report is accessible to all organizations, but the data remains siloed.

Strict data access controls prevent the release of training data without sanitization. This is unfortunate because direct evaluation of training data can provide insights into model biases [88]. Without access to training data, language model interpretability frameworks, such as Neuronpedia, TransformerLens and Lime, allow researchers to probe a model for biases, examine its inner-workings and elicit some information about why a model produces certain outputs [110–112]. These frameworks are essential for interpretability-based safety, but require access to a model’s weights. Given access to a model, information extraction techniques can be used to reverse-engineer the training data, potentially exposing sensitive information [113]. Additional adversarial attacks are possible with access to model weights, allowing for model misuse [114–117].

We can generally assume the users and partners accessing DoD models are trusted entities. However, regulations controlling classified information and the risk of model misuse remain valid concerns for over-sharing documentation. Unfortunately, sharing more model information gives a stronger assurance of safety, but also poses a greater risk of exposing protected information. The tension between model interpretability and information assurance is the driving force behind the political impasse over DoD documentation standards.

## 6.2.4 Policy Impasse: Documentation Standards

DoD model documentation standards must balance transparency with the protection of sensitive information. Documentation policy is motivated by line of effort (LOE) 3.2.3 of the Responsible AI Strategy and Implementation Pathway. LOE 3.2.3 proposes regulation where documentation templates are used when publishing DoD AI assets. Specifically, LOE 3.2.3 states *"Use AI Data and Model Cards to publish AI data assets in the DoD federated data catalog"* [82]. Policy addressing this LOE was planned for FY24, but is still in development.

The ambiguity of LOE 3.2.3 raises a number of questions, especially for those unfamiliar with the DoD or those lacking background on the Responsible Artificial Intelligence Strategy and Implementation Pathway. We aim to address these questions, succinctly.

1. *What are AI Data and Model Cards?* - Data and model cards (also referred to as system cards, datasheets, factsheets, and AI nutrition labels) provide information such as the developers, license, size, modality, data source, benchmark results, intended use,

out-of-scope use, biases, risks and limitations of datasets and models [83, 84]. This is how the information on our documentation continuum is published.

2. *What is the DoD federated data catalog?* - The DoD federated data catalog is a centralized database, accessible to all DoD personnel as well as authorized government agencies, defense contractors and partners [118].
3. *Why are model cards appropriate for publishing assets?* - Model cards are widely used in the academic community, with a number of studies finding that their use promotes accessibility, clarifies the intended use of models, and supports AI safety efforts [74, 87–91].
4. *Who creates the model card template?* - This is the responsibility of the Chief Digital and Artificial Intelligence Office (CDAO) within the DoD [82].
5. *Who fills in the template to create the model card?* - Technical experts, likely the model creators, fill in the model card template [82].
6. *Who reads the model card?* - DoD personnel read the model card to understand how to use the model, similar to a set of instructions. DoD partners read the model card for safety auditing or for potential adoption for their own applications [80].
7. *What information is on the model card template?* - This question is the crux of the policy controversy because the model card template will become a defacto mandate for the depth of model documentation [82].

An important note is that this policy controversy does not take the form of two separate actors, with different incentives, fighting for a policy that best suits their interests. Instead, this controversy is driven by two differing incentives within a single organization, the DoD. This could be solved by the DoD changing its AI safety standards to allow for a lower level of transparency or by changing classification standards to allow for relaxed information sharing. However, we find this unlikely due to the deeply entrenched nature of these standards and the lack of response in the form of change with other similarly disruptive technologies [79].

Determining the correct amount of information to share is difficult, with risks for both under-sharing and over-sharing. Notably, the Harvard Kennedy School’s Documentation Framework for AI Transparency lists this as an open research question, specifically asking, “*What information should be included in the documentation and what level of detail should be provided?*” [119]. Currently, the DoD recommends using the Aether template from Microsoft [85] or the HuggingFace model card template [83] to meet LOE 3.2.3 [75]. The interim use of two differing industry documentation templates raises a number of concerns.

## 6.2.5 Interim Policy Concerns

### Fragmented Standard-Setting

Our first concern with the application of industry documentation templates to DoD AI assets is the fragmentation of documentation standards. Standard setting in this domain is defined by market-based setting with a private locus of regulation, resulting in multiple firms creating different documentation standards. Figure 6.1 shows a variety of documentation templates proposed by industry and academia. Due to the varying levels of information provided in each template, the DoD risks incomplete documentation by selecting a single industry template. Further, selecting multiple industry templates would result in non-standard documentation across assets.

Table 6.1: Documentation templates proposed by industry and academia.

Citation	Model Documentation	Dataset Documentation
Mitchell et al.[87]	Model Cards	-
Sanseviero et al.[83]	HuggingFace Model Cards	-
Arnold et al.[90]	Fact Sheets	Fact Sheets
Google AI[92]	Google Model Cards	-
OpenAI[103]	System Cards	-
Wadhvani et al.[93]	TensorFlow Model Card Toolkit	-
Tagliabue et al.[94]	DAG Model Card Toolkit	-
Bender et al.[88]	-	Data Statements
Geburu et al.[89]	-	Datasheets
HuggingFace[84]	-	HuggingFace Dataset Cards
Microsoft[85]	-	Aether Datasheet
Pushkarna et al.[91]	-	Data Cards
Holland et al.[120]	-	Nutrition Labels

### Market Incentives

Our second concern with the application of industry documentation templates to DoD AI assets is that industry documentation is driven by competitive market factors. AI model makers, such as OpenAI, have incentives to leave out information in model documentation that could be valuable to competitors. This is reflected in their documentation templates, which lack information on training processes and data sources[121]. Further, model makers have incentives to leave out information that reveals shortcomings of their models because this information could discourage consumers from using their product[122]. These information asymmetries are expediently justified by the need to protect intellectual property. However, inclusion of data sources, training techniques, and model limitations is essential for the safety

evaluation of AI models in military applications. The DoD should not adopt a documentation standard that is censored by market incentives.

### **Heightened Risk**

Our third concern with the application of industry documentation templates to DoD AI assets is that these templates do not address unique military safety considerations. Civilian models are not intended to be used in military applications due to heightened risk in military environments[123]. Industry documentation lacks the thorough evaluation of failure modes required for the use of models in military applications. Due to this shortcoming, the use of industry documentation standards could result in unsafe use of DoD AI assets.

### **Military-Specific Documentation**

The incomplete nature of industry documentation templates and their lack of safety considerations for high risk applications make them unsuitable for DoD use. While the interim use of industry templates is acceptable, we emphasize the need for a military-specific documentation template to meet LOE 3.2.3. The impasse over LOE 3.2.3 should be resolved so that industry templates can be replaced in a timely fashion. The creation of military-specific documentation requires the technical expertise to examine models using interpretability frameworks and the ability to handle sensitive training information, such as classified data. Luckily, trusted entities with deep technical expertise already exist, the Federally Funded Research and Development Centers (FFRDCs). We advise the DoD to partner with the FFRDCs to create military-specific documentation for AI models.

### **6.2.6 Third Party Institutions**

Writing documentation that explains how to safely use a model requires a deep understanding of how the model was created, what it was trained on, and how it works [119]. To perform this technically-based policy analysis, we suggest the FFRDCs because they are analysis groups whose specific mission is to support government [124]. We believe the FFRDCs are well suited for this task because they have technical expertise in the field of language models, maintain security clearances, operate independently from market pressure, and have established relationships with the DoD [125, 126]. We find that the DoD needs assistance to create model documentation because they have not demonstrated the technical expertise to do so independently, as discussed in Section 6.2.7.

The use of the FFRDCs shifts the technical responsibility of creating a military-specific documentation template from the DoD. The DoD can task the FFRDCs to create a complete

documentation template, and fill out the template on a per-model basis. Specifically, the DoD can send exhaustive information about a model to the FFRDCs, who then evaluate the model and data using interpretability frameworks, saving all their work. The FFRDCs can then create a model card, sanitizing the documentation to meet necessary standards for publication in the DoD federated data catalog. The creation and archiving of complete documentation is important even if the released documentation is sanitized because full documentation preserves knowledge for internal use and can be used for auditing purposes. In cases where the FFRDCs deem that model documentation needs to be sanitized, their external evaluation still provides an additional safety assurance. This means use of the FFRDCs will always increase the level of documentation on our continuum.

In cases where sanitization is required to protect sensitive information, we recommend that the DoD mandates justification for removed information. This serves two purposes, to show that information was purposely omitted and to describe the rationale. For example, if a model's training data contains classified information, a model card could state *"Training data examples and sources are not included in this model card due to a risk of exposing classified information"*. Although inclusion of this justification may seem obvious, it is not common practice for current models [106].

Concretely, our recommended solution to the policy impasse is as follows. The content of the DoD model card template must balance transparency with the protection of sensitive information. Further, filling out the template requires a deep technical understanding of a model and its training data.

1. The DoD tasks the FFRDCs to create an exhaustive, military-specific documentation template for AI models.
2. The DoD sends models and training data to the FFRDCs for evaluation.
3. The FFRDCs evaluate the models and data using interpretability frameworks to gain a deep understanding of their capabilities and limitations, archiving their complete findings.
4. The FFRDCs fill out the DoD documentation template on a per-model basis.
5. The FFRDCs sanitize model documentation on a per-model basis, adding justification for removed information.
6. The DoD publishes the sanitized model documentation in the DoD federated data catalog and respond to special requests that require consulting the complete findings.

### 6.2.7 NIPR-GPT Evaluation

We evaluate the US Air Force’s (USAF) NIPR-GPT model to illustrate how our recommendations can be applied in practice. NIPR stands for Non-Secure Internet Protocol Router Network and is a USAF private network for exchanging unclassified information. GPT stands for Generative Pre-trained Transformers, a class of language models [57]. The current NIPR-GPT model is a fine-tuned version of the Llama-3.3-70B-instruct model from Meta AI [106]. The model was released in 2023 and is designed for processing unclassified Air Force documents and is mainly used for summarization, drafting and coding assistance [127]. The NIPR-GPT team provides some guidance for using the model, but their documentation lacks any information about training datasets, benchmark results or model biases. Further, their documentation is not in a standard model card format.

On our continuum, the NIPR-GPT model documentation is at the first point, only giving a brief examples of capabilities. This does not meet DoD safety standards. In fact, it does not even meet the interim industry standards for documentation. We find the lack of documentation surprising and limiting for the purposes of our evaluation. We recommend the NIPR-GPT team partners with an FFRDC, sharing their code, model and training data. The FFRDC can provide a safety analysis of the model and write a model card for the DoD federated catalog, promoting responsible model use.

## 6.3 Model and Dataset Documentation

In lieu of a DoD documentation template, we write documentation for our NL2SH models and datasets using the HuggingFace model and dataset card templates [83, 84]. Additionally, we include a section in our documentation titled "*Considerations for use in high-risk environments*" to address safety considerations that could apply to a military context. Notably, none of our models achieve a usable level of accuracy, especially in high-risk environments, as discussed in Section 5.3. We make this clear in our documentation to prevent misuse. Documentation for our NL2SH models and datasets can be found at the links in Table 6.2.

Table 6.2: Model and dataset documentation links.

<b>Model/Dataset Name</b>	<b>Documentation Link</b>
NL2SH-ALFA Dataset	<a href="https://huggingface.co/datasets/westenfelder/NL2SH-ALFA">huggingface.co/datasets/westenfelder/NL2SH-ALFA</a>
InterCode-Corrections Dataset	<a href="https://huggingface.co/datasets/westenfelder/InterCode-Corrections">huggingface.co/datasets/westenfelder/InterCode-Corrections</a>
Qwen2.5-Coder-0.5B-Instruct	<a href="https://huggingface.co/westenfelder/Qwen2.5-Coder-0.5B-Instruct-NL2SH">huggingface.co/westenfelder/Qwen2.5-Coder-0.5B-Instruct-NL2SH</a>
Qwen2.5-Coder-1.5B-Instruct	<a href="https://huggingface.co/westenfelder/Qwen2.5-Coder-1.5B-Instruct-NL2SH">huggingface.co/westenfelder/Qwen2.5-Coder-1.5B-Instruct-NL2SH</a>
Qwen2.5-Coder-3B-Instruct	<a href="https://huggingface.co/westenfelder/Qwen2.5-Coder-3B-Instruct-NL2SH">huggingface.co/westenfelder/Qwen2.5-Coder-3B-Instruct-NL2SH</a>
Qwen2.5-Coder-7B-Instruct	<a href="https://huggingface.co/westenfelder/Qwen2.5-Coder-7B-Instruct-NL2SH">huggingface.co/westenfelder/Qwen2.5-Coder-7B-Instruct-NL2SH</a>
Llama-3.2-1B-Instruct	<a href="https://huggingface.co/westenfelder/Llama-3.2-1B-Instruct-NL2SH">huggingface.co/westenfelder/Llama-3.2-1B-Instruct-NL2SH</a>
Llama-3.2-3B-Instruct	<a href="https://huggingface.co/westenfelder/Llama-3.2-3B-Instruct-NL2SH">huggingface.co/westenfelder/Llama-3.2-3B-Instruct-NL2SH</a>
Llama-3.1-8B-Instruct	<a href="https://huggingface.co/westenfelder/Llama-3.1-8B-Instruct-NL2SH">huggingface.co/westenfelder/Llama-3.1-8B-Instruct-NL2SH</a>

# Chapter 7

## Conclusion

In this thesis, we explore applications for LLMs in NL2SH translation and benchmarking. We identify issues with current benchmarks, including inaccurate datasets and unreliable functional equivalence heuristics. To address these problems, we correct and expand NL2SH datasets and create a new heuristic to determine the functional equivalence of Bash commands. We assess our heuristic and find that Bash command execution paired with language model evaluation of command outputs can determine the functional equivalence of commands more accurately than previous heuristics. Using our dataset and heuristic, we evaluate how constrained decoding, parsing, in-context learning and in-weight learning impact the performance of Llama, Qwen and GPT models. We find that parsing and in-context learning reliably improve the performance of open and closed-source LLMs for the task of NL2SH translation. Additionally, we consider the applications of NL2SH models in military contexts and reason that proper documentation promotes safe model use. We find that writing documentation requires a balance between transparency and security, which are often in conflict. We advocate for the DoD to leverage the FFRDCs to create military-specific documentation for language models on a per-model basis, prioritizing openness and providing justification for sanitization when necessary. Ultimately, we find that NL2SH translation remains a difficult task for LLMs, and improvements are needed in this field before models can be safely used in practice.

# Appendix A

## Translation Prompts

---

**Functional Equivalence Heuristic Prompt: LLM**

---

*You will be given a task and two Bash commands. The first command is the ground truth. If the second command accomplishes the task, return true. Otherwise, return false. Only output 'true' or 'false'. Task: `natural_language_prompt`, Ground Truth Command: `ground_truth_command`, Model Command: `model_command`.*

---

Figure A.1: Prompt for evaluating the functional equivalence of Bash commands.

---

**Functional Equivalence Heuristic Prompt: Execution + LLM**

---

*You will be given a task, two Bash commands, and the output of the two Bash commands. The first command is the ground truth. If the second command accomplishes the task, return true. Otherwise, return false. Only output 'true' or 'false'. Task: `natural_language_prompt`, Ground Truth Command: `ground_truth_command`, Model Command: `model_command`, Ground Truth Command Output: `ground_truth_command_output`, Model Command Output: `model_command_output`.*

---

Figure A.2: Prompt for evaluating the functional equivalence of Bash commands after execution. Note the addition of command outputs compared to the prompt in Figure A.1.

---

**Translation Prompt: Baseline**

---

*Your task is to translate a natural language instruction to a Bash command. You will receive an instruction in English and output a Bash command that can be run in a Linux terminal. You will not output markdown or other formatting. You will not include additional information.*

natural\_language\_prompt

---

Figure A.3: NL2SH translation prompt used in the baseline evaluation.

---

**Translation Prompt: Parser, Constrained Decoding and In-Weight Learning**

---

*Your task is to translate a natural language instruction to a Bash command. You will receive an instruction in English and output a Bash command that can be run in a Linux terminal.*

natural\_language\_prompt

---

Figure A.4: NL2SH translation prompt used in the parsing, constrained decoding and in-weight learning evaluations.

---

## Translation Prompt: In-Context Learning

---

Your task is to translate a natural language instruction to a Bash command. You will receive an instruction in English and output a Bash command that can be run in a Linux terminal.

Show logged-in users info

```
w
```

Print the contents of "xx.sh"

```
cat xx.sh
```

Change owner to "root" and group to "www-data" of "/foobar/test\_file"

```
chown root:www-data /foobar/test_file
```

delete all the text files in the current folder

```
find . -type f -name "*.txt" -delete
```

find all the files in the /path folder and delete them

```
find /path -type f -delete
```

Print the exit status of the last executed command

```
echo $?
```

Display a tree of processes

```
ps tree
```

Display information about all CPUs

```
lscpu
```

Make an HTTPS GET request to example.com and dump the contents in 'stdout'

```
curl https://example.com
```

Display system memory

```
free
```

List all files, including hidden files

```
ls -a
```

Print a sequence from 1 to 10

```
seq 10
```

Get the properties of all the user limits

```
ulimit -a
```

List the name and status of all services

```
service --status-all
```

...

---

---

**Translation Prompt: In-Context Learning, Continued**

---

*Display a calendar for the current month*  
`cal`

*Show the environment*  
`env`

*create directory TestProject*  
`mkdir TestProject`

*Query the default name server for the IP address of example.com*  
`nslookup example.com`

*Print Hello World*  
`echo "Hello World"`

*List all bound commands and their hotkeys*  
`bind -p`

*Display the openssl version*  
`openssl version`

*Print current time, uptime, number of logged-in users*  
`uptime`

*Print file system disk space usage*  
`df`

*List all configuration values available*  
`getconf -a`

*Delete empty folder 'nonsense\_dir'.*  
`rmdir nonsense_dir`

`natural_language_prompt`

---

Figure A.5: NL2SH translation prompt used in the in-context learning evaluation.

# Bibliography

- [1] William E Shotts. *The Linux Command Line: A Complete Introduction*. No Starch Press, 2019.
- [2] Chet Ramey and Brian Fox. *Bash: GNU Project’s Shell*, 2024. Accessed: 2024-09-09.
- [3] Michael Kerrisk. Linux manual pages. <https://man7.org/linux/man-pages/index.html>, 2024. Accessed: 2024-09-09.
- [4] Mayank Agarwal, Tathagata Chakraborti, Quchen Fu, David Gros, Xi Victoria Lin, Jaron Maene, Kartik Talamadupula, Zhongwei Teng, and Jules White. Neurips 2020 nlc2cmd competition: Translating natural language to bash commands. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 302–324. PMLR, 06–12 Dec 2021. URL <https://proceedings.mlr.press/v133/agarwal21b.html>.
- [5] Jean E. Sammet. The use of english as a programming language. *Commun. ACM*, 9(3):228–230, mar 1966. ISSN 0001-0782. doi:10.1145/365230.365274. URL <https://doi.org/10.1145/365230.365274>.
- [6] Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.
- [7] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=zAdUB0aCTQ>.
- [8] Jiacen Xu, Jack W. Stokes, Geoff McDonald, Xuesong Bai, David Marshall, Siyue Wang, Adith Swaminathan, and Zhou Li. Autoattacker: A large language model guided system to implement automatic cyber-attacks, 2024.
- [9] Stephen Moskal, Sam Laney, Erik Hemberg, and Una-May O’Reilly. Llms killed the script kiddie: How agents supported by large language models change the landscape of network threat testing, 2023.

- [10] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt: An llm-empowered automatic penetration testing tool, 2023.
- [11] Zach Lloyd, Michelle Lim, and Alope Desai. Warp: Your terminal, reimaged. <https://www.warp.dev/>, 2024. Accessed: 2024-09-09.
- [12] Farkhod Sadykov. Shellgpt: A command-line productivity tool powered by ai large language models like gpt-4. [https://github.com/TheR1D/shell\\_gpt](https://github.com/TheR1D/shell_gpt), 2024. Accessed: 2024-09-09.
- [13] Johan Rosenkilde, Matt Rothenberg, and Andy Feller. Github copilot for cli. <https://githubnext.com/projects/copilot-cli/>, 2024.
- [14] Microsoft. Microsoft ai shell. <https://learn.microsoft.com/en-us/powershell/utility-modules/aishell/overview?view=ps-modules>, 2024.
- [15] Amazon Web Services. Amazon codewhisperer natural language to bash translation. <https://docs.aws.amazon.com/codewhisperer/latest/userguide/command-line-conversation.html>, 2024.
- [16] John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: standardizing and benchmarking interactive coding with execution feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA, 2023. Curran Associates Inc.
- [17] Yewei Song, Cedric Lothritz, Xunzhu Tang, Tegawendé Bissyandé, and Jacques Klein. Revisiting code similarity evaluation with abstract syntax tree edit distance. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–46, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.acl-short.3. URL <https://aclanthology.org/2024.acl-short.3/>.
- [18] Pooja Aggarwal, Oishik Chatterjee, Ting Dai, Prateeti Mohapatra, Brent Paulovicks, Brad Blancett, and Arthur De Magalhaes. CodeSift: An LLM-Based Reference-Less Framework for Automatic Code Validation . In *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, pages 404–410, Los Alamitos, CA, USA, July 2024. IEEE Computer Society. doi:10.1109/CLOUD62652.2024.00052. URL <https://doi.ieeecomputersociety.org/10.1109/CLOUD62652.2024.00052>.
- [19] Romit. Linuxcommands, 2024. URL <https://huggingface.co/datasets/Romit2004/LinuxCommands>. Accessed: 2024-08-20.
- [20] Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. Multilingual machine translation with large language models: Empirical results and analysis. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*,

- pages 2765–2781, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.findings-naacl.176. URL <https://aclanthology.org/2024.findings-naacl.176>.
- [21] Liguó Chen, Qi Guo, Hongrui Jia, Zhengran Zeng, Xin Wang, Yijiang Xu, Jian Wu, Yidong Wang, Qing Gao, Jindong Wang, Wei Ye, and Shikun Zhang. A survey on evaluating large language models in code generation tasks, 2024. URL <https://arxiv.org/abs/2408.16498>.
- [22] Heejae Chon, Seonghyeon Lee, Jinyoung Yeo, and Dongha Lee. Is functional correctness enough to evaluate code language models? exploring diversity of generated codes, 2024. URL <https://arxiv.org/abs/2408.14504>.
- [23] Danil Shaikhelislamov, Mikhail Drobyshevskiy, and Andrey Belevantsev. Codepatchllm: Configuring code generation using a static analyzer, 2024. URL [https://genai-evaluation-kdd2024.github.io/genai-evaluation-kdd2024/assets/papers/GenAI\\_Evaluation\\_KDD2024\\_paper\\_25.pdf](https://genai-evaluation-kdd2024.github.io/genai-evaluation-kdd2024/assets/papers/GenAI_Evaluation_KDD2024_paper_25.pdf).
- [24] Daniel Copley. Shelloracle, 2023. URL <https://github.com/djcopley/ShellOracle>.
- [25] Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, H el ene Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL <https://aclanthology.org/L18-1491/>.
- [26] David Gros. Ainix: An open platform for natural language interfaces to shell commands, May 2019. URL <http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127814>. Undergraduate Honors Thesis, Computer Science Department, University of Texas at Austin.
- [27] Quchen Fu, Zhongwei Teng, Jules White, and Douglas C. Schmidt. A transformer-based approach for translating natural language to bash commands. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1245–1248, 2021. doi:10.1109/ICMLA52953.2021.00202.
- [28] T. Ramesh. Nl2cmd, 2022. URL <https://huggingface.co/datasets/TRamesh2/NL2CMD>. Accessed: 2024-08-20.
- [29] Shikhar Bharadwaj, Shirish Shevade, and Marine Carpuat. Efficient constituency tree based encoding for natural language to bash translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3159–3168, Seattle, United States, July 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.naacl-main.230. URL <https://aclanthology.org/2022.naacl-main.230>.

- [30] Daniel Jenson and Yingxiao Liu. Translating natural language to bash commands using deep neural networks. [https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1224/reports/custom\\_116997097.pdf](https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1224/reports/custom_116997097.pdf), 2022. Accessed: 2024-10-09.
- [31] Jie Shi, Sihang Jiang, Bo Xu, Jiaqing Liang, Yanghua Xiao, and Wei Wang. Shellgpt: Generative pre-trained transformer model for shell language understanding. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 671–682, 2023. doi:[10.1109/ISSRE59848.2023.00082](https://doi.org/10.1109/ISSRE59848.2023.00082).
- [32] Yogesh Mali. text\_to\_bash, 2023. URL [https://huggingface.co/datasets/yogeshm/text\\_to\\_bash](https://huggingface.co/datasets/yogeshm/text_to_bash).
- [33] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation. *IEEE Transactions on Software Engineering*, 49(07):3675–3691, July 2023. ISSN 1939-3520. doi:[10.1109/TSE.2023.3267446](https://doi.org/10.1109/TSE.2023.3267446). URL <https://doi.ieeecomputersociety.org/10.1109/TSE.2023.3267446>.
- [34] Ngoc Phuoc An Vo, Brent Paulovicks, and Vadim Sheinin. Execution-based evaluation of natural language to bash and powershell for incident remediation, 2024. URL <https://arxiv.org/abs/2405.06807>.
- [35] Oishik Chatterjee, Pooja Aggarwal, Suranjana Samanta, Ting Dai, Prateeti Mohapatra, Debanjana Kar, Ruchi Mahindru, Steve Barbieri, Eugen Postea, Brad Blancett, and Arthur De Magalhaes. Scriptsmith: A unified llm framework for enhancing it operations via automated bash script generation, assessment, and refinement, 2024. URL <https://arxiv.org/abs/2409.17166>.
- [36] Anish Joshi. Bash scripting assistant. <https://github.com/AnishJoshi13/Bash-Scripting-Assistant>, 2024. Accessed: 2024-10-09.
- [37] Xi Victoria Lin. Program synthesis from natural language using recurrent neural networks, 2017. URL <https://api.semanticscholar.org/CorpusID:3809743>.
- [38] Berkeley Churchill, Oded Padon, Rahul Sharma, and Alex Aiken. Semantic program alignment for equivalence checking. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019*, pages 1027–1040, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367127. doi:[10.1145/3314221.3314596](https://doi.org/10.1145/3314221.3314596). URL <https://doi.org/10.1145/3314221.3314596>.
- [39] Quchen Fu, Zhongwei Teng, Marco Georgaklis, Jules White, and Douglas Schmidt. Nl2cmd: An updated workflow for natural language to bash commands translation. *Journal of Machine Learning Theory, Applications and Practice*, 1, 02/2023 2023. doi:<https://doi.org/10.13052/jmltapissn.2023.002>. URL <https://www.journal.riverpublishers.com/index.php/JMLTAP/article/view/8>.

- [40] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.
- [41] Karen Sparck Jones. *A statistical interpretation of term specificity and its application in retrieval*, pages 132–142. Taylor Graham Publishing, GBR, 1988. ISBN 0947568212.
- [42] Junjie Huang, Chenglong Wang, Jipeng Zhang, Cong Yan, Haotian Cui, Jeevana Priya Inala, Colin Clement, and Nan Duan. Execution-based evaluation for data science code generation models. In Eduard Dragut, Yunyao Li, Lucian Popa, Slobodan Vucetic, and Shashank Srivastava, editors, *Proceedings of the Fourth Workshop on Data Science with Human-in-the-Loop (Language Advances)*, pages 28–36, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.dash-1.5/>.
- [43] Prashanth Vijayaraghavan, Luyao Shi, Stefano Ambrogio, Charles Mackin, Apoorva Nitsure, David Beymer, and Ehsan Degan. Vhdl-eval: A framework for evaluating large language models in vhdl code generation, 2024. URL <https://arxiv.org/abs/2406.04379>.
- [44] OpenAI. Gpt-4 technical report, 2024.
- [45] Nickil Maveli, Antonio Vergari, and Shay B. Cohen. What can large language models capture about code functional equivalence?, 2024. URL <https://arxiv.org/abs/2408.11081>.
- [46] Atharva Naik. On the limitations of embedding based methods for measuring functional correctness for code generation, 2024.
- [47] Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.
- [48] Chi Yu, Guang Yang, Xiang Chen, Ke Liu, and Yanlin Zhou. Bashexplainer: Retrieval-augmented bash code comment generation based on fine-tuned codebert. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 82–93, 2022. doi:[10.1109/ICSME55016.2022.00016](https://doi.org/10.1109/ICSME55016.2022.00016).
- [49] Zhongqi Chen, Neng Zhang, Pengyue Si, Qinde Chen, Chao Liu, and Zibin Zheng. Shellfusion: An answer generator for shell programming tasks via knowledge fusion. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 93–97, 2023. doi:[10.1109/ICSE-Companion58688.2023.00032](https://doi.org/10.1109/ICSE-Companion58688.2023.00032).
- [50] Sathvik Joel, Jie JW Wu, and Fatemeh H. Fard. A survey on llm-based code generation for low-resource and domain-specific programming languages, 2024. URL <https://arxiv.org/abs/2410.03981>.
- [51] Richard Blum and Christine Bresnahan. *Linux Command Line and Shell Scripting Bible*. Wiley, 2021.

- [52] K.B.Dharun Krishna, Sebastiaan Speck, Owen Voke, and Darío Herenu. Tldr: Collaborative cheatsheets for console commands. <https://github.com/tldr-pages/tldr>, 2024. Accessed: 2024-10-09.
- [53] Idan Kamara. bashlex: Python parser for bash. <https://github.com/idank/bashlex>, 2016. Accessed: 2024-09-09.
- [54] Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. Open source strikes bread - new fluffy embeddings model, 2024. URL <https://www.mixedbread.ai/blog/mxbai-embed-large-v1>.
- [55] Meta Llama. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- [56] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore, December 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.emnlp-main.674. URL <https://aclanthology.org/2023.emnlp-main.674/>.
- [57] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf).
- [58] Duarte Alves, Nuno Guerreiro, João Alves, José Pombal, Ricardo Rei, José de Souza, Pierre Colombo, and Andre Martins. Steering large language models for machine translation with finetuning and in-context learning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11127–11148, Singapore, December 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.findings-emnlp.744. URL <https://aclanthology.org/2023.findings-emnlp.744>.
- [59] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL <https://arxiv.org/abs/2106.09685>.
- [60] Anonymous. Evaluating natural language to bash command translation on consumer-grade hardware, 2024. URL <https://openreview.net/forum?id=BLUN4PuWjE>.
- [61] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual*

- Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, USA, 2002. Association for Computational Linguistics. doi:[10.3115/1073083.1073135](https://doi.org/10.3115/1073083.1073135). URL <https://doi.org/10.3115/1073083.1073135>.
- [62] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL <https://arxiv.org/abs/2308.12950>.
- [63] Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. Do large language model benchmarks test reliability?, 2025. URL <https://arxiv.org/abs/2502.03461>.
- [64] Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=AjXkRZiVjB>.
- [65] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. Fully autonomous programming with large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pages 1146–1155, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi:[10.1145/3583131.3590481](https://doi.org/10.1145/3583131.3590481). URL <https://doi.org/10.1145/3583131.3590481>.
- [66] Ezri Zhu, Georgios Liargkovas, Michael Greenberg, and Konstantinos Kallas. try utility. <https://github.com/binpash/try>, 2024. Accessed: 2024-09-25.
- [67] Jane Pan, Ryan Shar, Jacob Pfau, Ameet Talwalkar, He He, and Valerie Chen. When benchmarks talk: Re-evaluating code llms with interactive feedback, 2025. URL <https://arxiv.org/abs/2502.18413>.
- [68] OpenAI. o1 System Card, 2024. URL <https://cdn.openai.com/o1-system-card-20241205.pdf>.
- [69] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,

Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanxia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

- [70] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL <https://arxiv.org/abs/1510.00149>.
- [71] U.S. Air Force. Cyber systems operations, 2024. URL <https://www.airforce.com/careers/intelligence/cyber-systems-operations>.
- [72] DoD Cyber Exchange. Dod cyber workforce, 2024. URL <https://public.cyber.mil/wid/dcwf/work-roles-2/>. Accessed: 2024-09-15.
- [73] William N. Caballero and Phillip R. Jenkins. On large language models in national security applications, 2024. URL <https://arxiv.org/abs/2407.03453>.
- [74] Weixin Liang, Nazneen Rajani, Xinyu Yang, Ezinwanne Ozoani, Eric Wu, Yiqun Chen, Daniel Scott Smith, and James Zou. What’s documented in ai? systematic analysis of 32k ai model cards, 2024. URL <https://arxiv.org/abs/2402.05160>.
- [75] Chief Digital and Artificial Intelligence Office. Responsible artificial intelligence (rai) toolkit. <https://rai.tradewindai.com/>, 2024.
- [76] Venice Goodwine. Department of the air force interim guidance on responsible adoption of generative artificial intelligence (genai) and large language models, December 2023.
- [77] Craig Martell. Chief digital and artificial intelligence office interim guidance on use of generative artificial intelligence, October 2023.

- [78] Winston Beauchamp. Department of the air force memorandum on use of large language models, June 2023.
- [79] Joseph Biden. Executive order 14110 - executive order on the safe, secure, and trustworthy development and use of artificial intelligence. <https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure>, October 2023.
- [80] Department of Defense. Department of defense data, analytics, and artificial intelligence adoption strategy. [https://media.defense.gov/2023/Nov/02/2003333300/-1/-1/1/DOD\\_DATA\\_ANALYTICS\\_AI\\_ADOPTION\\_STRATEGY.PDF](https://media.defense.gov/2023/Nov/02/2003333300/-1/-1/1/DOD_DATA_ANALYTICS_AI_ADOPTION_STRATEGY.PDF), November 2023.
- [81] Department of Defense. Defense industrial base adoption of artificial intelligence for defense applications, May 2024. URL <https://www.federalregister.gov/documents/2024/05/22/2024-11195/defense-industrial-base-adoption-of-artificial-intelligence-for-defense-applications-notice-of>.
- [82] Department of Defense. Responsible artificial intelligence strategy and implementation pathway. [https://www.ai.mil/docs/RAI\\_Strategy\\_and\\_Implementation\\_Pathway\\_6-21-22.pdf](https://www.ai.mil/docs/RAI_Strategy_and_Implementation_Pathway_6-21-22.pdf), June 2022.
- [83] HuggingFace. Hugging face model cards documentation. <https://huggingface.co/docs/hub/en/model-cards>, 2024.
- [84] HuggingFace. Hugging face dataset cards documentation. <https://huggingface.co/docs/hub/en/datasets-cards>, 2024.
- [85] Microsoft Aether Transparency Working Group. Aether data documentation template. <https://www.microsoft.com/en-us/research/uploads/prod/2022/07/aether-datadoc-082522.pdf>, 2022.
- [86] Angelina McMillan-Major, Salomey Osei, Juan Diego Rodriguez, Pawan Sasanka Ammanamanchi, Sebastian Gehrmann, and Yacine Jernite. Reusable templates and guides for documenting datasets and models for natural language processing and generation: A case study of the huggingface and gem data and model cards. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*. Association for Computational Linguistics, 2021. doi:10.18653/v1/2021.gem-1.11. URL <http://dx.doi.org/10.18653/v1/2021.gem-1.11>.
- [87] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 2019. doi:10.1145/3287560.3287596. URL <http://dx.doi.org/10.1145/3287560.3287596>.
- [88] Emily M. Bender and Batya Friedman. Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 12 2018. ISSN 2307-387X.

- doi:10.1162/tacl\_a\_00041. URL [https://direct.mit.edu/tacl/article/doi/10.1162/tacl\\_a\\_00041/43452/Data-Statements-for-Natural-Language-Processing](https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00041/43452/Data-Statements-for-Natural-Language-Processing).
- [89] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III au2, and Kate Crawford. Datasheets for datasets, 2021. URL <https://arxiv.org/abs/1803.09010>.
- [90] Matthew Arnold, Rachel K. E. Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilovic, Ravi Nair, Karthikeyan Natesan Ramamurthy, Darrell Reimer, Alexandra Olteanu, David Piorkowski, Jason Tsay, and Kush R. Varshney. Factsheets: Increasing trust in ai services through supplier’s declarations of conformity, 2019. URL <https://arxiv.org/abs/1808.07261>.
- [91] Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. Data cards: Purposeful and transparent dataset documentation for responsible ai, 2022. URL <https://arxiv.org/abs/2204.01075>.
- [92] Google. About google model cards. <https://modelcards.withgoogle.com/about>, 2024.
- [93] Abhishek Wadhvani and Priyank Jain. Machine learning model cards transparency review : Using model card toolkit. In *2020 IEEE Pune Section International Conference (PuneCon)*, pages 133–137, 2020. doi:10.1109/PuneCon50868.2020.9362382. URL <https://ieeexplore.ieee.org/document/9362382>.
- [94] Jacopo Tagliabue. Template-based generation of dag cards. <https://github.com/jacopotagliabue/dag-card-is-the-new-model-card>, 2021.
- [95] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024. URL <https://arxiv.org/abs/2307.06435>.
- [96] William N. Caballero and Phillip R. Jenkins. On large language models in national security applications, 2024. URL <https://arxiv.org/abs/2407.03453>.
- [97] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, November 2024. ISSN 1558-2868. doi:10.1145/3703155. URL <http://dx.doi.org/10.1145/3703155>.
- [98] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL <https://arxiv.org/abs/2009.03300>.
- [99] Ashkan Golgoon, Khashayar Filom, and Arjun Ravi Kannan. Mechanistic interpretability of large language models with applications to the financial services industry. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 660–668. ACM, November 2024. doi:10.1145/3677052.3698612. URL <http://dx.doi.org/10.1145/3677052.3698612>.

- [100] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Margaret Mitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi:[10.1145/3442188.3445922](https://doi.org/10.1145/3442188.3445922). URL <https://doi.org/10.1145/3442188.3445922>.
- [101] US Department of State. Political declaration on responsible military use of artificial intelligence and autonomy. <https://www.state.gov/political-declaration-on-responsible-military-use-of-artificial-intelligence-and-autonomy-2/>, 2023.
- [102] Scott Sullivan. Targeting in the black box: The need to reprioritize ai explainability. <https://lieber.westpoint.edu/targeting-black-box-need-reprioritize-ai-explainability/>, 2024.
- [103] OpenAI. Openai o1 system card. <https://cdn.openai.com/o1-system-card-20240917.pdf>, 2024.
- [104] BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth

Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobel, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéal, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Uldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Perinán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrmann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sängler, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata

- Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aaroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model, 2023. URL <https://arxiv.org/abs/2211.05100>.
- [105] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious, 2024. URL <https://arxiv.org/abs/2501.00656>.
- [106] Air Force Research Laboratory. Nipr-gpt guidance for use, September 2024. URL <https://niprgpt.mil/guidebook>. Accessed: 2024-09-09.
- [107] Joel Ruben Antony Moniz, Soundarya Krishnan, Melis Ozyildirim, Prathamesh Saraf, Halim Cagri Ates, Yuan Zhang, and Hong Yu. Realm: Reference resolution as language modeling, 2024. URL <https://arxiv.org/abs/2403.20329>.
- [108] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements, 2022. URL <https://arxiv.org/abs/2209.14375>.
- [109] Ewen Callaway. AI protein-prediction tool AlphaFold3 is now more open. *Nature*, 635 (8039):531–532, nov 2024. ISSN 0028-0836. doi:10.1038/d41586-024-03708-4. URL <https://www.nature.com/articles/d41586-024-03708-4>.
- [110] Johnny Lin. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL <https://www.neuronpedia.org>. Software available from neuronpedia.org.
- [111] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2023. URL <https://arxiv.org/abs/2303.08112>.
- [112] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD*

*International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

- [113] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021. URL <https://arxiv.org/abs/2012.07805>.
- [114] Eric Hartford. Uncensored models. <https://erichartford.com/uncensored-models>, 2023.
- [115] Seanie Lee, Minsu Kim, Lynn Cherif, David Dobre, Juho Lee, Sung Ju Hwang, Kenji Kawaguchi, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Moksh Jain. Learning diverse attacks on large language models for robust red-teaming and safety tuning, 2024. URL <https://arxiv.org/abs/2405.18540>.
- [116] Tingchen Fu, Mrinank Sharma, Philip Torr, Shay B. Cohen, David Krueger, and Fazl Barez. Poisonbench: Assessing large language model vulnerability to data poisoning, 2024. URL <https://arxiv.org/abs/2410.08811>.
- [117] Andy Arditi, Oscar Obeso, Aaqib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL <https://arxiv.org/abs/2406.11717>.
- [118] David Spirk. Federated data catalog - minimum metadata requirements. <https://www.ai.mil/Portals/137/Documents/Resources%20Page/Federated%20Data%20Catalog%20-%20Minimum%20Metadata%20Requirements.pdf>, 2021.
- [119] Kasia Chmielinski, Sarah Newman, Chris Kranzinger, Michael Hind, Jennifer Wortman Vaughan, and Margaret Mitchell. The clear documentation framework for ai transparency: Recommendations for practitioners & context for policymakers. [https://shorensteincenter.org/wp-content/uploads/2024/05/CleAR\\_KChmielinski\\_FINAL.pdf](https://shorensteincenter.org/wp-content/uploads/2024/05/CleAR_KChmielinski_FINAL.pdf), 2024.
- [120] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards, 2018. URL <https://arxiv.org/abs/1805.03677>.
- [121] James Gallifant, Andrew Fiske, Yuliya A Levites Strelakova, Jorge S Osorio-Valencia, Rachel Parke, Richard Mwavu, Nicole Martinez, Joseph W Gichoya, Marzyeh Ghassemi, Dina Demner-Fushman, L Griffin McCoy, Leo Anthony Celi, and Richard Pierce. Peer review of GPT-4 technical report and systems card. *PLOS Digital Health*, 3(1):e0000417, 2024.
- [122] Chris Stokel-Walker. Critics denounce a lack of transparency around gpt-4’s tech. <https://www.fastcompany.com/90866190/critics-denounce-a-lack-of-transparency-around-gpt-4s-tech>, March 2023.
- [123] Meta. Meta llama 2 acceptable use policy. <https://ai.meta.com/llama/use-policy/>, 2023.

- [124] Granger Morgan. Technically focused policy analysis. In *The Science of Science Policy: A Handbook*, pages 120–130. Stanford University Press, 2011.
- [125] Institute for Defense Analyses. Capabilities and expertise - federally funded research and development centers. <https://www.ida.org/en/ida-ffrdcs/systems-and-analyses-center/s3d/capabilities-expertise>, 2024.
- [126] Eric Horvitz, Jessica Young, Rama Elluru, and Chuck Howell. Key considerations for the responsible development and fielding of artificial intelligence. <https://arxiv.org/pdf/2108.12289>, 2021.
- [127] Jon Harper. How things are going with the air force’s experimental niprgpt chatbot. <https://defensescoop.com/2024/11/07/air-force-niprgpt-experimental-chatbot-how-things-are-going/>, 2024.