


LUNA: A Model-Based Universal Analysis Framework for Large Language Models

Da Song^{*}, Xuan Xie^{*}, Jiayang Song, Derui Zhu, Yuheng Huang, Felix Juefei-Xu, Lei Ma 

Abstract—Over the past decade, Artificial Intelligence (AI) has had great success recently and is being used in a wide range of academic and industrial fields. More recently, Large Language Models (LLMs) have made rapid advancements that have propelled AI to a new level, enabling and empowering even more diverse applications and industrial domains with intelligence, particularly in areas like software engineering and natural language processing. Nevertheless, a number of emerging trustworthiness concerns and issues exhibited in LLMs, e.g., robustness and hallucination, have already recently received much attention, without properly solving which the widespread adoption of LLMs could be greatly hindered in practice. The distinctive characteristics of LLMs, such as the self-attention mechanism, extremely large neural network scale, and autoregressive generation usage contexts, differ from classic AI software based on Convolutional Neural Networks and Recurrent Neural Networks and present new challenges for quality analysis. Up to the present, it still lacks universal and systematic analysis techniques for LLMs despite the urgent industrial demand across diverse domains. Towards bridging such a gap, in this paper, we initiate an early exploratory study and propose a universal analysis framework for LLMs, named *LUNA*, which is designed to be general and extensible and enables versatile analysis of LLMs from multiple quality perspectives in a human-interpretable manner. In particular, we first leverage the data from desired trustworthiness perspectives to construct an abstract model as an auxiliary analysis asset and proxy, which is empowered by various abstract model construction methods built-in *LUNA*. To assess the quality of the abstract model, we collect and define a number of evaluation metrics, aiming at both the abstract model level and the semantics level. Then, the semantics, which is the degree of satisfaction of the LLM w.r.t. the trustworthiness perspective, is bound to and enriches the abstract model with semantics, which enables more detailed analysis applications for diverse purposes, e.g., abnormal behavior detection.

To better understand the potential usefulness of our analysis framework *LUNA*, we conduct a large-scale evaluation, the results of which demonstrate that 1) the abstract model is with the potential to distinguish normal and abnormal behavior in LLM, 2) *LUNA* is effective for the real-world analysis of LLMs in practice, and the hyperparameter settings influence the performance, 3) different evaluation metrics are in different correlations with the analysis performance. In order to encourage further studies in the quality assurance of LLMs, we made all of the code and more detailed experimental results data available on the supplementary website of this paper <https://sites.google.com/view/llm-luna>.

Index Terms—Large Language Models, Deep Neural Networks, Model-based Analysis, Quality Assurance




1 INTRODUCTION

Over the last few years, a series of tremendous performance leaps in many real-world applications across domains have been empowered by the rapid advancement of LLMs, especially in the domain of Software Engineering (SE) and Natural Language Processing (NLP), e.g., code generation [1], program repair [2], sentiment analysis [3], and question answering [4]. Representative LLM-enabled applications such as ChatGPT [5], GPT4 [6], and LLaMA [7] are often recognized as the early foundation towards Artificial General Intelligence (AGI) [8]. More recently, LLMs present the promising potential to become a new enabler and booster to further revolutionize the intelligitization and automation for various key stages of software production life-cycle.

Despite the rapid development, the current quality [9], reliability [10], robustness [11], and explainability [12] of LLMs pose many concerns of social society and technical challenges, the research on which, on the other hand, is still at a very early stage. For example, recent research indicates that existing LLMs can occasionally generate content that is toxic, biased, insecure, or erroneous [9], [13], [14]. For example, a typically new type of quality issue is the phenomenon of hallucination [15], where LLMs confidently produce nonfactual or erroneous outputs, which poses significant challenges for their implementation, particularly in environments where safety and security are paramount. Moreover, the rapid industrial adoption of LLMs in various applications, e.g., robotic control [16] and medical image diagnosis [17], necessitates urgent analysis and risk assessment methodologies for LLMs.

In recent years, there has come an increasing trend in research to tackle the quality assurance challenges of deep learning software, especially Deep Neural Networks (DNNs) [18]–[39]. Some studies have focused on DNN testing with the goal of pinpointing inputs that a DNN struggles to manage [18]–[24], [30]–[35], [40]–[43]. Concurrently, advancements in DNN debugging and repair [25]–[29], [36]–[39], [44]–[49] aim to understand the reasons behind a

^{*}These authors contributed equally to this work.

 Corresponding author

• Da Song, Xuan Xie, Jiayang Song and Yuheng Huang are with the Department of Electrical and Computer Engineering at the University of Alberta, Canada. E-mail: {dsong4, xxie9, jiaayan13, yuheng18}@ualberta.ca

• Derui Zhu is with the Department of Computer Science at the Technical University of Munich, Germany. E-mail: derui.zhu@tum.de

• Felix Juefei-Xu is with GenAI at Meta, USA. E-mail: felixxu@meta.com

• Lei Ma is with The University of Tokyo, Japan, and University of Alberta, Canada. E-mail: ma.lei@acm.org

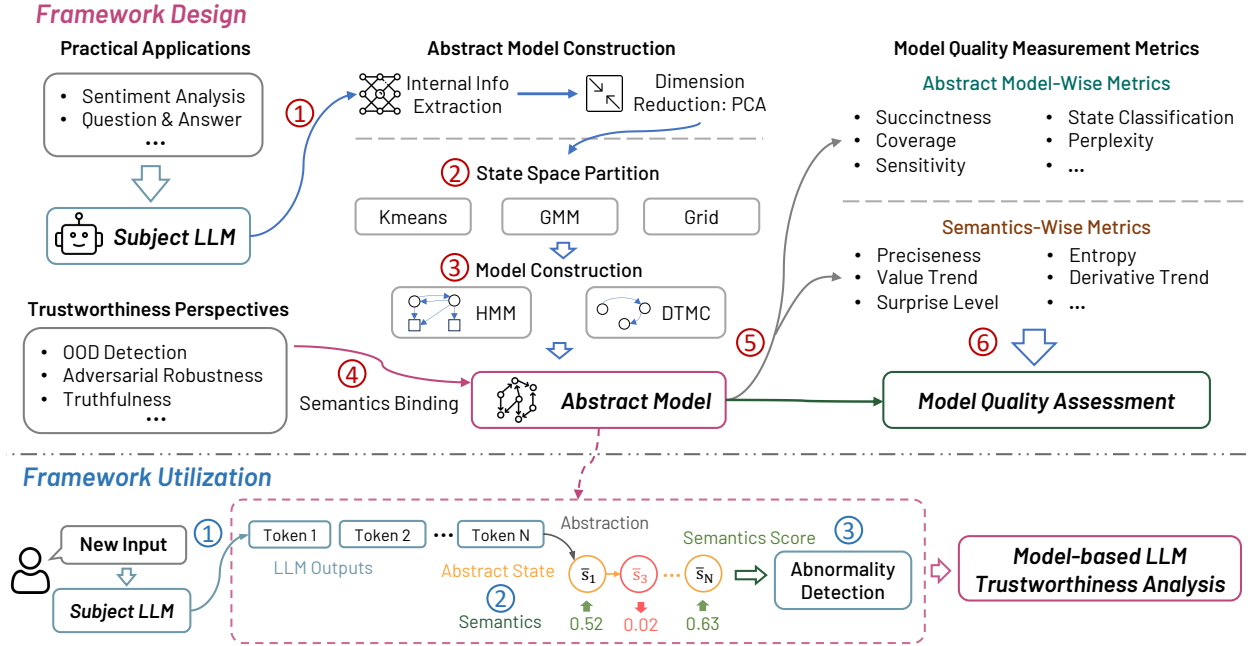


Fig. 1: The workflow summary of *LUNA*.

DNN’s incorrect predictions and subsequently repair the model. These studies have made notable contributions to the advancement of quality assurance in DL-based software. However, a majority of these works are centered around Convolutional Neural Networks (CNNs) [50] and Recurrent Neural Networks (RNNs) [51]. Pei et al. propose DeepX-plore, a whitebox DNN testing framework combined with neuron coverage and differential testing to efficiently capture defects in DNN systems [18]. Zohdinasab et al. leverage illumination search to identify and quantify the dimensions of feature space in testing deep learning systems [52]. Hu et al. propose a framework for mapping between dangerous situations and the image transformations in the machine vision components [53]. Among these analysis techniques, *model-based analysis* [29], [54]–[58] has been demonstrated as an effective approach to both provide analysis results, e.g., testing and monitoring, and human-explainable results. Pan and Rajan [59] propose to decompose a CNN model into modules for each output class, enabling reusability and lowering the environmental cost. Dong et al. [60] develop an approach to extract probabilistic automata for RNN interpretation, which integrates hidden states abstraction and automata learning. Qi et al. develop ArchRepair, which repairs DNNs by jointly optimizing architecture and weights at the block level [61].

Different from CNNs and RNNs, LLMs behave with distinct features such as the adaptation of the self-attention [62] mechanism as its core, the complex and large-scale model size (e.g., 6.7 billion to 65.2 billion parameters in LLaMA series released by Meta [7]), and the generative output scheme which highly depend on a broad spectrum of user’s inputs. Such features make the analysis of LLMs’ behavior more challenging compared to classification contexts, which are the main focus of existing research. Therefore, even up to the present, only very limited research has been conducted to probe general-purpose LLM-oriented analysis techniques to understand the quality and behavior characteristics of LLMs

from various aspects. The prospective quality assurance methods should help better comprehend LLMs’ internal behaviors, identify unwanted outputs, and aid in improving the trustworthiness of LLM in practical usage.

As described above, the philosophy of model-based analysis has been widely proven to be useful towards providing quality assurance for traditional DNNs; however, its effectiveness for LLMs is still unknown and deserves further investigation. Therefore, to bridge this gap, we propose and design *LUNA*, a model-based universal analysis framework for large language models. The first step of *LUNA* is to extract an assistant model for analysis. Due to the high dimensional space and sparsely distributed states, we extract and build the *abstract model* such as Discrete-Time Markov Chain (DTMC) and Hidden Markov Model (HMM), to enable and ease the analysis procedure and capture LLM’s probabilistic nature. With the obtained abstract model, we further perform *semantics* binding, which is the degree of satisfaction of the LLM with respect to the desired quality perspective, to the abstract model to enable in-depth quality analysis. Moreover, to evaluate the quality of the model, we collect abstract model-wise metrics and propose semantics-wise metrics to measure from the perspectives of models and semantics, respectively. Finally, we apply the constructed abstract model on *abnormal behavior detection* to detect potentially erroneous outputs from the LLMs, e.g., hallucination.

In order to demonstrate the potential usefulness of *LUNA*, we conduct a large-scale evaluation on multiple applications of LLMs. The experimental results and in-depth analysis across three trustworthiness perspectives (e.g., out-of-distribution detection, adversarial robustness, and truthfulness) confirm that: 1) the constructed abstract model can capture and distinguish normal and abnormal behaviors of the LLM; 2) the quality of the abstract model is highly impacted by the techniques and corresponding hyperparameters used in model construction (e.g., dimen-

sion reduction and state partition); 3) *LUNA* is effective in abnormal behavior detection, e.g., the ROC AUC of adversarial attack detection can achieve 83%; 4) model-based quality measurement metrics (e.g., abstract model-wise and semantics-wise) assess the quality of the model from distinct aspects and are correlated with the performance of the framework differently.

The main contributions of this paper are summarized as follows:

- A universal model-based analysis framework, which is designed for general-purpose quality analysis for LLMs, provides a human-interpretable way to characterize LLM’s behavior.
- A set of model quality measurement metrics for LLMs, which are collected from existing research (abstract model-wise metrics) and newly proposed (semantics-wise metrics). The correlations with the analysis performance show their potential in guiding the abstract model construction.
- An extensive experiment is conducted to demonstrate the effectiveness of *LUNA*, which is on 3 trustworthiness perspectives, 12 quality measurement metrics, 180 hyperparameter settings, and a total of more than 1,400 CPU hours. The results demonstrate that *LUNA* is effective in LLM’s abnormal behavior detection.
- An exploratory study to investigate the effectiveness of model-based analysis in the context of LLMs. This paper also targets inspiring more relevant research in this direction towards approaching the goal of achieving trustworthy LLMs in practice.

The Contributions to the Software Engineering Field.

With an increasing trend of adopting LLMs in the software production life cycle, LLMs would potentially draw significant impact to the domain of software engineering as they present explicit capabilities to accelerate the development process and implementation outcomes [63]–[66]. Hence, the quality analysis of LLMs calls even more attention as it bridges the last gap to further deploy LLMs on safety, reliability, and security-concerned applications. Following the path in this direction, our work also endeavors to empower the interaction of LLMs within SE by establishing the early foundation of model-based analysis to enable more systematic investigations towards trustworthy LLMs across various SE applications. The rest of the paper is structured as follows. Section 2 introduces the corresponding background. Section 3 describes the different abstraction methods and model construction techniques. Section 4 details the experiment setup and reports the results. Section 5 discusses the potential impact and future directions. Section 6 inspects the threats that may affect the validity of our work. Section 7 summarizes the related works, and Section 8 concludes the paper. All code and study findings have been made available at <https://sites.google.com/view/llm-luna>.

2 BACKGROUND

In this section, we first provide the background knowledge on the analyzed deep learning model, i.e., the *Large Language Model (LLM)*. *Trustworthiness perspectives* of LLMs are then introduced, which are of serious concern to the quality and reliability of LLMs. In addition, we describe the key idea of

model-based analysis at a high level, the main technique used in our analysis framework.

2.1 Large Language Models

Witnessed by various industrial and academic communities, LLMs, a new revolution in AI technology, have demonstrated human-competitive capabilities in various natural language tasks across domains (e.g., text generation, language translation, code development) [1], [67]–[71]. Intuitively, LLMs are a type of neural network model that is usually established based on the *Transformer* architecture [62] with millions, even billions of parameters. Such models are pre-trained on large corpora of text data, which enclose numerous commonsense knowledge [72]. LLMs output a sequence of words following a probability distribution; namely, each output token is generated coherently based on the input prompt and prior outputs. Up to the present, a growing number of research indicates that LLMs are capable of delivering high-level problem-solving skills for a variety of downstream tasks, such as question-and-answer [73], sentiment analysis [74], text summarization [75], code generation [76] and code summarization [77].

Besides the large network scale, the superior performance of LLMs can also give credit to the transformer architecture and its central mechanism: *self-attention* [62]. Self-attention assesses each element within the input sequence by comparing them with one another and then alters the respective positions in the output sequence. The unique transformer architecture enables the LLMs to surpass the traditional RNNs regarding many challenges, such as long-range dependencies [78] and gradient vanishing [79]. Self-attention is encompassed in the *decoder block*, which is a basic unit of the decoder-only LLM, which will be introduced later in this Section. Many studies confirm the information enclosed in the output of the decoder block can be an asset to characterize the behaviors of an LLM [80]–[83]. Thus, in this work, we leverage the decoder block outputs and traces extracted from the LLM to construct an abstract model-based LLM analysis framework. We further detail the model construction in our study in Section 3.2.

LLMs can be categorized into three main different types according to their transformer architectures and pre-trained tasks: *encoder-only*, *encoder-decoder* and *decoder-only*. *Encoder-only* LLMs (e.g., BERT [84], DeBERTa [85], RoBERTa [86]) are pre-trained by masking a certain number of input tokens and aim to predict the masked elements retroactively. Alternatively, *encoder-decoder* LLMs, such as BART [87], FLAN-UL2 [88] and T5 [89], utilize an encoder to first convert the input sequence into a hidden vector, then a subsequent decoder further converts the hidden vector into the output sequence. This encode-then-decode architecture has advantages in processing sequence-to-sequence tasks involving intricate mapping between the input and output. *Decoder-only* LLMs are auto-regressive models that predict each token based on the input sequence and the prior generated tokens. Representative *decoder-only* LLMs like GPT4 [6], GPT3 [90] and LLaMA [7] are recognized as prevailing attributable to their training efficiency and scalability for large-scale models and datasets. In this study, we mainly focus on *decoder-only* LLM, LLaMA, considering its availability, commonality and computation cost; nevertheless,

our framework itself can still be generalized and adapted to LLMs other than *decoder-only* ones. We introduce the subject LLM and corresponding settings in Section 4.2.1

2.2 LLM Trustworthiness Perspective

Interpreting and understanding the behaviors of machine learning models, especially LLMs, is one of the essential tasks in both AI and SE communities [9], [19], [54], [91]. Recently, some researchers and industrial practitioners have tried to seek to understand the capability boundary and characteristics of the models in order to deploy them in practical applications more confidently and adequately. Quality analysis is applied to initiate to approach a comprehensive and consistent view of model trustworthiness, such as safety [92], robustness [93] and security [94]. In this work, we also leverage our proposed analysis framework *LUNA* to conduct quality analysis of LLMs from three aspects: *Out-of-Distribution Detection*, *Adversarial Attacks* and *Hallucination*. Such three perspectives are notoriously known as vital factors that affect the trustworthiness of LLMs.

2.2.1 Out-of-Distribution (OOD) Detection

A fundamental premise of machine learning is the similarity in distribution between training and future unseen test data. In other words, DNN models might falter when encountering some data (in the future) deviating from the training distribution [95]. Empirical research has shown that DNNs may even be highly confident to offer an erroneous prediction in this scenario [96], [97]. To alleviate this issue, OOD detection has been introduced to improve the quality of data-driven software by detecting the irrelevant OOD data without letting a DNN make wrong decisions on it that would be incorrectly handled with high possibility [20], [96]–[98]. The objective is to craft a probability distribution estimation function $P_{\mathcal{X}}$ (\mathcal{X} is the training distribution) that assigns a score to a given input x and sets a corresponding threshold λ for the OOD detection [99]:

$$g(x) = \begin{cases} \text{in} & \text{if } P_{\mathcal{X}}(x) \geq \lambda \\ \text{out} & \text{if } P_{\mathcal{X}}(x) < \lambda \end{cases} \quad (1)$$

While early research predominantly focused on image classification tasks, some efforts have also been made in NLP domains [100]. This encompasses the detection of OOD instances in text classification [101], translation [102], and question-answering [103]. Yet, the OOD challenges associated with LLMs present greater difficulty [104]. Firstly, LLMs’ training data are often either inaccessible or too large, making exploration challenging. Secondly, LLMs’ emergent ability across varied tasks makes traditional OOD measurements on standalone tasks inappropriate to apply.

To address this issue, researchers collect OOD data made public after a certain timestamp (e.g. 2022) because LLM-based systems such as ChatGPT typically disclose the conclusion date of their training data (e.g. 2021). However, with the continuous evolution of LLMs, related benchmarks may soon be out-of-date. In this study, we instead focus on the *OOD style* [105], where the original data are transformed to another style (e.g. Shakespeare) at both word-level and sentence-level. We follow the settings of the DecodingTrust

benchmark [9] and perform related studies on the SST-2 development set.

2.2.2 Adversarial Attacks

The sensitivity of DNNs’ predictions against subtle perturbation in the inputs as an intriguing property has been studied for over a decade now [106], [107]. Such sensitivity is due to the highly nonlinear nature of DNNs and could be utilized by adversaries for malicious attacks [108]. Related attack models are first defined in image domains as subtle and continuous perturbations on image pixels that aim to fool classifiers [109]. Such attacks can be formulated as optimization problems, and the goal is to find perturbations on inputs that change the final predictions. These attacks have been adapted to the text domain, where perturbations are defined as discrete ones at the word, sentence, or entire input levels. In this context, adversarial GLUE [110] stands out as a comprehensive benchmark for text domain adversarial robustness. It incorporates various prior attack methodologies across multiple tasks and is continuously updated. Wang et al. [104] recently assessed ChatGPT’s adversarial robustness using this benchmark. Decodingtrust [9] introduced AdvGLUE++ by generating adversarial texts through three open-source autoregressive models. In our study, we employ AdvGLUE++ to gauge the adversarial robustness of LLMs.

There are numerous strategies to enhance the robustness of DL-driven systems against adversarial attacks, such as adversarial training, perturbation control, and robustness certification [111]. Given the computational intensity and vast data associated with LLMs, training and certification are beyond the scope of this paper. Consequently, we turn to adversarial detection, a lighter approach that falls in perturbation control, which is also well suited for model-based analysis. Upon detecting an adversarial attack, the model can either reject the input or take other actions. This setting is similar to OOD detection but focuses on adversarial scenarios.

2.2.3 Hallucination

With the advancement of language models, some research works have posed these models can occasionally produce unfaithful and nonfactual content considering the given inputs [112]–[114] and such undesirable generation behavior is so-called hallucination [15]. Hallucination hinders the trustworthiness and reliability of LLMs and causes serious concerns for LLM-embedded real-world applications. Different from other factors that harm the trustworthiness of LLMs, the detection of hallucinations is challenging and often requires active human efforts to evaluate the generated outcomes based on input contexts and external knowledge [8], [72]. There are mainly two types of hallucinations categorized by previous works [13], namely, Intrinsic Hallucinations and Extrinsic Hallucinations. The former is defined as the existence of contradictions between the source content and the generated outputs, and the latter indicates the outputs cannot be verified from the source.

To mitigate the risks from hallucinations, a series of strategies have been proposed by researchers and are divided into two categories: Data-Related Methods and Modelling and Inference Methods [15], [115]. In particular, data-

related methods tackle the problems by data-cleaning and information augmentation [116], and modelling and inference methods modify the architecture of the model or apply optimizations at the training phase [117]. Nevertheless, existing solutions are not well-applicable in the context of LLMs, seeing the large-scale training corpus, the massive training cost and the complex model structure with billions of parameters. In this work, we conduct experiments on the TruthfulQA benchmark to evaluate the effectiveness of the proposed framework *LUNA* in terms of hallucination detection. We detail the subject tasks of this work in Section 4.2.2.

2.3 Model-based Analysis

An autoregressive language model denoted as p , is essentially a function f_θ parameterized by θ . This function assigns a probability distribution over the alphabet \mathcal{V} based on an input string y_0, \dots, y_{t-1} . The generation procedure entails repeatedly invoking p , which can be interpreted as a stochastic process:

$$p(\mathbf{y} = y_1, \dots, y_T) = \prod_{t=1}^T p(y_t | \mathbf{y}_{<t}) \quad (2)$$

where y_T is the end-of-string symbol (EOS), $\mathbf{y}_{<t}$ is defined as (y_0, \dots, y_{t-1}) , and y_0 is the user provided prompt.

The generation chain involves successive calls to the DNN which is often well-known for its black-box nature. This intricate procedure complicates direct analysis. To make this stateful DNN-driven process more transparent and trustworthy, researchers previously tried to extract an interpretable model by examining the DNN’s behavior on training data [54], [118]–[122]. However, the effectiveness of such modelling techniques is still unclear in the context of LLMs since 1) whether the hidden states of LLMs can provide insights to assist the interpretation of their behavior [80], [81], [123] and 2) to what extent the traditional probabilistic models, such as DTMC [124], can help explain the probabilistic processes of LLMs.

In particular, previous works [54], [59], [60] begin by collecting and analyzing hidden states extracted from DNNs. However, as the high dimensional state space derived directly from the DNNs is too vast to process, abstraction techniques such as dimension reduction and state partition [119], [120] are usually necessary to map concrete states to abstract ones. Subsequently, each inference can be represented as a sequence of state transitions, enabling the construction of a probabilistic model that emulates the behavior of the original DNN. These models can facilitate adversarial detection [125], privacy analysis [126], maintenance [55], [127], [128], interpretability [60], [129], and debugging [29].

In this study, we adopt a similar approach with details explained in Section 3.2. It is worth noting that much of the prior research focuses on classification tasks based on RNN, while we mainly study the autoregressive generation of LLMs. RNN classification can be viewed as a special case of autoregressive generation, where only one label is generated for an input string. Namely, the autoregressive generation of LLMs brings more challenges for analysis from the quality assurance perspective since 1) the quality of the generated text is not only determined by the individual token but

also by the overall structure and semantic coherence of the generated text, 2) the output at each time step is dependent on all previous outputs, which adds another layer of complexity, and, 3) LLMs’ large and diverse output spaces across a wide range of topics make a comprehensive quality analysis even more difficult.

3 METHODOLOGY

In this section, we first discuss the *workflow* of our proposed framework *LUNA* at a high level (Section 3.1). Then, we introduce the *abstract model construction procedure* in Section 3.2, and the *semantics binding* in Section 3.3, the two important stages in our framework. The evaluation *metrics* for assessing the quality of the models is described in Section 3.4. At last, we introduce the general-purposed *applications* of our model-based analysis in Section 3.5.

3.1 Overview

As illustrated in Figure 1, *LUNA* is a model-based analysis framework crafted to investigate the trustworthiness of LLMs. At a high level, *LUNA* includes four key stages: *abstract model construction*, *semantics binding*, *model quality metrics*, and *practical application*.

Abstract model construction. The first step is to build the abstract model, which plays a predominant role in our analysis framework. To enable the universal analysis for LLM, *LUNA* is designed to support an assortment of abstraction factors, i.e., dimension reduction (PCA), abstraction state partition (grid-based and cluster-based partition), and abstract model types (DTMC and HMM). (Section 3.2)

Semantics binding. With the abstract state space, an important step is to know what information contained in the state can help the analysis process. Thus, after the abstract model is built, we bind semantics, which is the level of satisfaction of the LLM with respect to the specific trustworthiness perspective. The semantics of the model represent the behavior logic of the LLM and empower an in-depth analysis. (Section 3.3)

Model quality assessment. A crucial step before practical application is the evaluation of the quality of the model. To evaluate the quality of the constructed model, we leverage two sets of metrics: abstract model-wise metrics and semantics-wise metrics. We collect abstract model-wise metrics to measure the quality of the abstract model from existing works. To evaluate the quality of the semantics binding, we also propose semantics-wise metrics. (Section 3.4)

Practical application. LLMs can occasionally make up answers or generate erroneous outputs in their answers. To enhance the trustworthiness of LLMs, it is important to detect such abnormal behaviors. After constructing the abstract model, we utilize it for a common analysis for LLMs, specifically, the detection of abnormal behaviors. (Section 3.5)

3.2 Abstract Model Construction

Taking both trustworthiness perspective-specific data and the subject LLM as inputs, we first profile the given model to extract the concrete states and traces, i.e., outputs from the decoder blocks. Then, we leverage the extracted data to construct our abstract model. In this work, we mainly study

two state-based models, DTMC and HMM, depicted as Figure 2. The construction of these two models is described as follows.

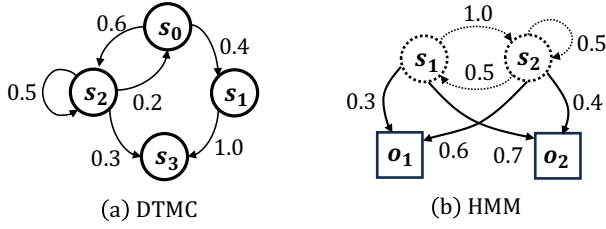


Fig. 2: DTMC and HMM illustration

3.2.1 DTMC Construction

Definition 1 (Discrete Time Markov Chain). A DTMC is a tuple $(\bar{S}, \bar{s}_0, \bar{\delta}, \bar{P})$, where \bar{S} is a finite set of states, $\bar{s}_0 \in \bar{S}$ is the initial state, $\bar{\delta}$ is the set of transition, and $\bar{P} : \bar{S} \times \bar{S} \rightarrow [0, 1]$ is the transition probability function.

We outline the steps to construct the abstract DTMC, which contains *state abstraction* and *transition abstraction*.

- *State Abstraction*

The state abstraction aims to build the abstract state space \bar{S} , which includes two steps: *dimension reduction* and *state space partition*.

The dimension of concrete states is equal to the number of neurons of decoder block outputs, which is typically too high to analyze directly. For instance, with 32 decoder blocks and 4,096 dimensions, the dimension of the hidden states for a single token in LLaMA-7b is 131,072. Thus, we first apply dimension reduction to reduce the dimension of concrete states to ease the analysis complexity. In particular, we leverage *Principle Component Analysis* (PCA) [130] to transform the original data to k dimensional vector, which retains the most important patterns and variations in the original data.

Then, we perform state space partition to construct the abstract state space. We use two ways that are commonly used in the recent works [54], [121], [126] to conduct the partition: *grid-based partition* and *cluster-based partition*. For regular grid-based partition, we first apply multi-step abstraction to include more information contained in the near temporal steps. The abstraction is essentially created by sliding a N -step window on the trace. In other words, for $N = 2$, $\{s_i, s_{i+1}\}$, and $\{s_{i+1}, s_{i+2}\}$ are different multi-step abstraction. Then, we apply grid partition; namely, each dimension of the k -dimensional space is first uniformly divided into m grids, and we use c_j^i to denote the j -th grid of the i -th dimension. Then, the compressed concrete states that fall into the same grid are assigned to the same abstract state, i.e., $\bar{s} = \{s_i | s_i^1 \in c_1^1 \wedge \dots \wedge s_i^k \in c_k^1\}$. For the cluster-based partition, we utilize existing clustering algorithms, e.g., Gaussian Mixture Model (GMM) [131] and KMeans [132], to assign the compressed concrete states into n different groups, where each of such group is considered an abstract state.

- *Transition Abstraction*

The objective of transition abstraction is to build the abstract transition space $\bar{\delta}$ and the corresponding transition probability. Here, we define that there is an abstract

transition \bar{t} between abstract states \bar{s} and \bar{s}' if and only if there are concrete transitions between s and s' , where $s \in \bar{s}$ and $s' \in \bar{s}'$. Moreover, the transition probability of an abstract transition is computed as the number of the concrete transitions from abstract state \bar{s} to another abstract state \bar{s}' over the number of total outgoing transitions.

3.2.2 HMM construction

HMM [133]–[135], is designed to catch the sequential dependencies within the data and is able to provide a probability distribution over possible sequences. Hence, we also choose HMM to model the hidden state traces.

Definition 2 (Hidden Markov Model). An HMM is a tuple $(\bar{S}, \bar{\delta}, \bar{P}, \bar{O}, \bar{E}, \bar{I})$, where \bar{S} is the hidden state space, $\bar{\delta}$ is the transition space, $\bar{P} : \bar{S} \times \bar{S} \rightarrow [0, 1]$ is the transition probability function that maps the transition to the probability distribution, $\bar{O} = \{o_1, \dots, o_n\}$ is the finite set of observations, $\bar{E} : (s_i, o_j) \rightarrow [0, 1]$ is the emission function that maps the observation o_j being generated from state s_i to a probability distribution, and $\bar{I} : \bar{S} \rightarrow [0, 1]$ is the initial state probability function that map the state space to the probability distribution.

The construction of HMM is as follows. We first define the state space S with the number of hidden states and the abstract states, built in DTMC construction (Section 3.2.1), and the observations \bar{O} , which is all the seen abstract states in the abstract states space. Then, we use the standard HMM fitting procedure – Baum-Welch algorithm [136] (as an Expectation-Maximization algorithm) to compute transition probability \bar{P} , Emission function \bar{E} , and initial state probability function \bar{I} . Baum-Welch algorithm is composed of *expectation*, which calculates the conditional expectation given observed traces, and *maximization*, which updates the parameters of \bar{P} , \bar{E} , and \bar{I} , to maximize the likelihood of observation. The Baum-Welch algorithm determines the most probable sequence of hidden states that would lead to the sequence of observed abstract states. The constructed HMM is capable of analyzing and predicting the future text and outputs, based on the probabilistic modelling of the historical data, i.e., the fitted \bar{P} , \bar{E} , and \bar{I} .

3.3 Semantics Binding

To enable an effective quality analysis, we bind *semantics*, which reflects LLM’s performance regarding specific trustworthiness perspectives, to the abstract model.

Definition 3 (Semantics). The concrete semantics $\theta \in \mathbb{R}^n$ of a concrete state sequence $\tau^k = \langle s_i, \dots, s_{i+k-1} \rangle$ represents the level of satisfaction of the LLM w.r.t. the trustworthiness perspectives.

Intuitively, semantics reflects the condition of the LLM regarding the desired trustworthiness perspective. Assume $k = 1$, as shown in Figure 3 when the LLM falls in states $\bar{s}_0, \bar{s}_1, \bar{s}_2$, and \bar{s}_3 , it is considered to be in the normal status, while state \bar{s}_4 is considered to be an abnormal state for the model. Moreover, we perform semantics abstraction to obtain the abstract semantics $\bar{\theta}$. We take the average values of all concrete semantics in the abstract state as the abstract semantics. The essence of our semantics binding

lies in its ability to align the internal states of an LLM to externally observable behaviors, specifically pertaining to different tasks. Therefore, such semantics interpretation acts as a bridge, connecting the abstract behavior captured by the model to the real-world implications of that behavior. Note that when $k = 1$, the sequence contains only one state. To ease the notation, we omit k .

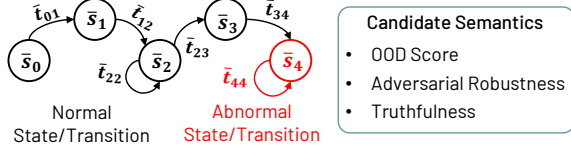


Fig. 3: Semantics bound abstract model.

We use hallucination detection [137] as an example to illustrate the semantics binding. The concrete state could be bound with the truthfulness, i.e., the probability of the answer being true, of an answer text. While in OOD sample detection, the transition probability between two states, i.e., s_i and s_{i+1} , can be deemed as the semantics regarding OOD $\theta(\langle \bar{s}_i, \bar{s}_{i+1} \rangle)$, because the transition probability intrinsically indicates whether the sequence exists in the training dataset to some extent.

3.4 Model Quality Metrics

Once the abstract model is built, one of the important steps before the concrete application is to assess the *quality* of the abstract model. The assessment is typically through some types of *metrics*. In this work, we collect and summarize a set of metrics characterizing the quality of the model. At a high level, the metrics can be divided into *abstract model-wise metrics* and *semantics-wise metrics*. Below, we briefly introduce these metrics, and the full definition of the metrics can be referred to Appendix A.

To evaluate the *quality* of the constructed model, we collect the metrics that are widely used in the literature [54], [121], [126], [138], [139] to assess the model from diverse aspects, as displayed in Table 1. We call such metrics as *abstract model-wise metrics*. Abstract model-wise metrics are categorized into three types: *basic*, *state-level*, and *model-level*. Basic metrics contain succinctness (the abstract level of the state space), coverage (how many new states are unseen in the state space), and sensitivity (whether abstract states differ under small perturbation). The state-level metrics contain state type classification, e.g., sink state [124], which helps identify the property of the Markov model, e.g., absorbable (the Markov chain cannot escape some undesirable states.) [140]. We compute the following metrics for model-level metrics: stationary distribution entropy [139] and perplexity [141], which reflect the stability of the model and the degree of well-fitting to the training distribution, respectively.

Note that the abstract model-wise metrics do not involve *semantics*, which contains the level of satisfaction w.r.t. trustworthiness. However, to our knowledge, not much work provides general metrics to measure the quality of the abstract model in terms of semantics. To fill this gap, we propose *semantics-wise metrics*, as shown in Table 2. The semantics-wise metrics are extended into *basic*, *trace-level*, and *surprise-level*. Basic semantics-wise metrics contain

semantics preciseness (the average preciseness of abstract semantics) and semantics entropy (the randomness and unpredictability of the semantics space). Trace-level metrics compute the level of how the semantics change temporally, which includes value diversity (instant value and n -gram value) and derivative diversity (n -gram derivative) [142]. Surprise-level metrics try to evaluate the surprising degree of the change of the semantics by means of Bayesian reasoning [143].

3.5 Applications

Recent works demonstrate that the abstract model has extensive analysis capability for stateful DNN systems [29], [54], [126]. Here, to validate the practicality of our constructed abstract models, we mainly apply them into *abnormal behavior detection*, which is a common analysis demand for LLMs [144], [145]. As introduced in Section 2 abnormal behavior refers to the unintended expression of LLM, e.g., making up answers or generating biased output [146]–[150]. To detect such behavior, we leverage the abstract model with the semantics. The procedure of detection is as follows. Given an output text and the abstract state trace $\{\bar{s}_1, \dots, \bar{s}_n\}$, we first acquire the corresponding semantics trace $\{\theta(\bar{s}_1), \dots, \theta(\bar{s}_n)\}$. Then, we compute an *semantics score* by taking the mean of the semantics sequence value, namely, $\text{AVG}(\theta(\bar{s}_i))$. Finally, we compare the computed score with the ground truth to determine the performance of classifying the output text as normal/abnormal behavior.

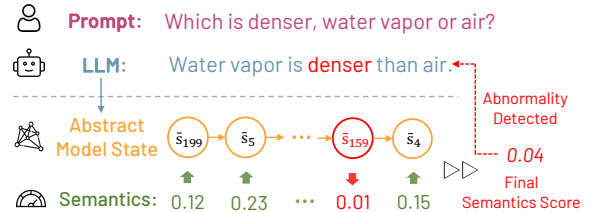


Fig. 4: Example of hallucination detection on TruthfulQA.

Here, we provide a running example of hallucination detection, as shown in Figure 4, to show how we use the abstract model to detect abnormal behaviors. The prompt to LLM is “Which is denser, water vapor or air?” and the LLM answers “Water vapor is denser than air.”. The corresponding abstract state sequence is $\bar{s}_{199} \rightarrow \bar{s}_5 \rightarrow \dots \rightarrow \bar{s}_{159} \rightarrow \bar{s}_4$, and the semantics sequence is $0.12 \rightarrow 0.23 \rightarrow \dots \rightarrow 0.01 \rightarrow 0.15$. The computed semantics score is 0.04, and we identify the answer as an abnormal behavior. Moreover, we can see that state \bar{s}_{159} is an abnormal state, which represents the LLM become abnormal at word “denser”. Such semantics-based LLM behavior interpretation enables a human-understandable approach to explain and analyze the quality of the LLM w.r.t. different trustworthiness perspectives. It is worth noting that our framework is designed with adaptability for various practical applications (e.g., OOD detection, adversarial attack detection, etc.). More concrete examples of our framework on different applications are available on <https://sites.google.com/view/llm-luna>.

4 EXPERIMENTS

In this section, we detail the experiments conducted to validate our framework, LUNA. Through a series of experi-

TABLE 1: Abstract Model-wise Metrics.

Metric	Description	Type
Succinctness (SUC)	State reduction rate and transition reduction rate	Basic
Coverage (COV)	Unseen states/transitions in abstract model	Basic
Sensitivity (SEN)	Abstract state variation under small perturbation	Basic
State classification (SS)	Sink state from Markov chain	State
Stationary Distribution Entropy (SDE)	Randomness and unpredictability within the transitions	Model
Perplexity (PERP)	The degree of well-fitting to the training distribution	Model

TABLE 2: Semantics-wise Metrics.

Metric	Description	Type
Semantics Preciseness (PRE)	Mean and max semantics error	Basic
Semantics Entropy (ENT)	Randomness and predictability of semantics	Basic
Value Trend (IVT, NVT)	Instant Value Trend and n -gram Value Trend	Trace
Derivative Trend (NDT)	The trend of derivative over the temporal domain	Trace
Surprise (SL)	The degree of the change of the semantics	Surprise

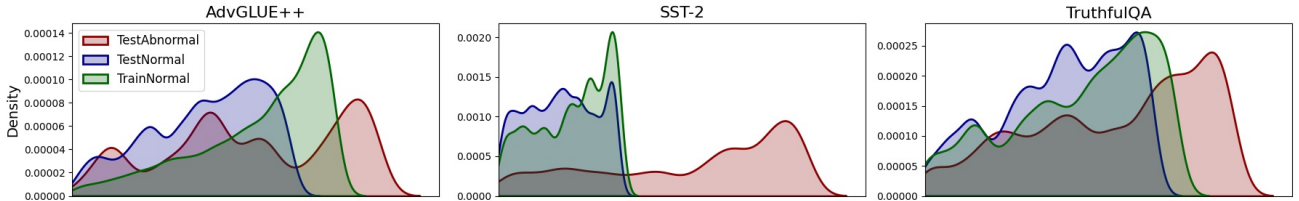


Fig. 5: RQ1: Distribution of transition probabilities of three studied datasets.

ments, we aim to investigate *LUNA*'s effectiveness in terms of characterizing the behaviors of LLMs and detecting the abnormality of the subject LLM. Leveraging the previously introduced metrics and applications, our experiments aim to demonstrate the framework's potential as a universal tool to support the quality assurance of LLMs across various trustworthiness.

The implementation design of *LUNA* is to build an extensible and plug-and-play framework to enable and support the research on the quality assurance of LLMs. More specifically, the *extensibility* and *adaptability* of *LUNA* reflect on the four aspects: 1) it can be applied to diverse types of LLMs (encoder-only, decoder-only, encoder-decoder); 2) it incorporates a series of abstraction and modelling methods to offer an enriched LLM analysis pipeline and can be further extended with more advanced analysis techniques; 3) it encloses an assortment of metrics to measure the quality of the model and the trustworthiness of the LLM from a diverse spectrum of aspects and can embed new metrics seamlessly according to users' demands; 4) it fits diverse trustworthiness perspectives as well as practical applications. Based on *LUNA*, many further extensions and more advanced techniques could be proposed and incorporated into our framework for more advanced quality assurance purposes for LLMs.

4.1 Research Questions

In particular, with *LUNA*, we conduct evaluations to investigate the following research questions:

- **RQ1:** Can the abstract model differentiate the normal and abnormal behaviors of LLM?

- **RQ2:** How do different modelling techniques and corresponding configurations impact the quality of the abstract model?
- **RQ2.1:** How is the state abstraction correlated with abstract model-wise evaluation metrics?
- **RQ2.2:** How is the model construction method correlated with abstract model-wise evaluation metrics?
- **RQ3:** How does the framework perform across target trustworthiness perspectives, and how is its performance correlated with both semantics-wise and abstract model-wise metrics?
- **RQ3.1:** How does our framework perform on trustworthiness perspectives regarding semantics-wise metrics?
- **RQ3.2:** How is the performance of the framework correlated with the abstract model-wise metrics?

We start with an initial inquiry about the abstract model's capability of distinguishing normal and abnormal behaviors (RQ1). We then examine the correlation between modelling settings, resulting attributes, and model metrics (RQ2.1 and RQ2.2). Finally, we assess the abstract model's effectiveness across various trustworthiness perspectives, drawing insights into the relationships among performance, model construction settings, and quality measurement metrics (RQ3.1 and RQ3.2).

RQ1 serves as a preliminary study demonstrating the ability of our abstract model in terms of abnormal behavior detection, laying the foundation for subsequent research questions. These anomalies could appear as hallucinations, OOD samples, or adversarial attacks.

Based on the findings in RQ1, we further leverage abstract model-wise metrics to quantitatively evaluate the quality of the abstract model in RQ2. These metrics delve into the attributes of abstract models from various per-

spectives, probing how different configurations during the model’s construction can influence its overall quality.

In RQ3, our primary focus is on the practical applications where our model-based analysis framework gets deployed. Starting with RQ3.1, we assess the framework’s real-world efficacy by examining the Area Under the Receiver Operating Characteristic Curve (ROC AUC) [151] across multiple trustworthiness perspectives. Additionally, this investigation uncovers how the effectiveness of our framework is correlated to specific semantic-wise metrics. In RQ3.2, we further inspect the relationships between the analysis performance and the abstract model-wise metrics. Namely, we illuminate the connections between the hyperparameter settings and their associated model-wise metrics. Taken together, our analyses not only enhance the understanding of the model’s effectiveness but also offer insight into metrics-driven guidance for abstract model construction w.r.t. various trustworthiness perspectives.

4.2 Experiment Settings

4.2.1 General Setup

Subject LLMs. As many performant LLMs are treated as vital intellectual properties and kept black-boxed, it is challenging to find adequate open-source LLMs to conduct our study. We focus on two sources that potentially release high-performance open-source LLMs with acceptable deployment costs: 1) distribution of LLMs from artificial intelligence companies such as OpenAI, Meta AI and Google AI, and 2) LLM-related literature, such as the papers and LLMs released by research institutes [7], [152], [153]. We do our best to select the most eligible subject LLMs according to the following criteria.

- **Open-source availability:** A LLM must be open-sourced such that we can extract the internal information from the LLM to conduct the following model construction process.
- **Competitive performance:** A LLM must be representative and have competitive performance regarding diverse task-handling abilities. In such a manner, we can obtain generalizable insights and implications from the experiments.
- **Acceptable deployment cost:** A LLM must be scalable to get deployed on limited computational resources. Some state-of-the-art LLMs come with high demand deployment and operation costs, which is not feasible for many research groups in the community.

Eventually, we select Alpaca-7b [154] as our subject LLM for this study. In particular, Alpaca-7b is a publicly available LLM with performance parallel to GPT3.5 [152]. In addition, Alpaca-7b manifests promising diverse task-handling abilities with an acceptable computational resource requirement. Hence, we consider it the best-fit subject LLM to deliver representative results and insights.

Experimental Environment. All of our experiments were conducted on a server with a 4.5GHz AMD 5955WX 16-Core CPU, 256GB RAM and two NVIDIA A6000 GPUs with 48GB VRAM each. The overall computational time is over 1,400 hours.

Hyperparameters Settings. As mentioned in Section 3.2 our framework encloses various state abstraction and modelling

methods with numerous hyperparameters. Therefore, there could be myriad possible hyperparameter combinations in our design that are not feasible to fully evaluate. Even though, to better understand the effectiveness of our framework, with our limited computational resources, we tried our best and conducted evaluations on as many as 180 representative hyperparameter settings to investigate the characteristics of different settings. The hyperparameters of our experiments are summarized in Table 3.

Evaluation Metrics. Evaluation metrics are another crucial segment of our framework, as these metrics are devoted to presenting a transparent and comprehensive understanding of the quality of the constructed model as well as the effectiveness of the framework across different applications. Although some previous metrics are proposed by literature [54], [121], [126], [138]–[143], only some of them are applicable to the context of LLMs. Thus, we carefully select 6 widely used metrics to assess the quality of the abstract model and propose another 6 metrics to evaluate the effectiveness of the model in terms of different trustworthiness perspectives. With these 12 metrics, we aim to conduct a relatively comprehensive evaluation as much as we can to obtain an in-depth understanding of our framework under different hyperparameter configurations, as well as the impact of hyperparameters on the effectiveness of LUNA. The metrics used for evaluation in this study are summarized in Table 1 and Table 2.

4.2.2 Subject Trustworthiness Perspective

To better understand the effectiveness of the proposed framework across diverse tasks, we select a set of challenging and representative datasets. Mainly, as described in Section 2.2, we assess the quality of the abstract model from three trustworthiness perspectives: Out-of-Distribution Detection, Adversarial Attack, and Hallucination. These three perspectives are widely observed and critical trustworthiness concerns of LLMs [9], [110], [137], [149], [155], [156]. Each of these has unique patterns (formats) that are capable of investigating both the effectiveness and the generality of our framework.

Out-of-Distribution Detection. We adapt the sentiment analysis dataset created by Wang et al. [9]. It is based on the SST-2 dataset [157] and contains word-level and sentence-level style transferred data, where the original sentences are transformed to another style. It contains a total of 9,603 sentences, with 873 in-distribution (ID) data and 8,730 OOD data.

Adversarial Attack. For the adversarial attack dataset, we use AdvGLUE++ [9], which consists of three types of tasks (sentiment classification, duplicate question detection, and multi-genre natural language inference) and five word-level attack methods. It contains 11,484 data in total.

Hallucination. For the hallucination dataset, we choose TruthfulQA [137], which is designed for measuring the *truthfulness* of LLM in generating answers to questions. It consists of 817 questions, with 38 categories of falsehood, e.g., misconceptions and fiction. The ground truth of the answers is judged by fine-tuned GPT3-13B models [137] to classify each answer as true or false.

Semantic Binding Across Perspectives. In hallucination detection, we bind semantics directly to states based on the

TABLE 3: Summary of Abstract Model Settings for Abstract Model Construction. There are 180 hyper-setting configurations in total. [†]For Grid-based partition, the actual state number is calculated as the stated number to the power of the PCA components.

Partition	Model Type	PCA Components	#State [†]	Additional Parameters	#Settings
Grid	DTMC	{3, 5, 10}	{5, 10, 15}	History Step: {1, 2, 3}	27
Grid	HMM	{3, 5, 10}	{5, 10, 15}	History Step: {1, 2, 3}; HMM Comp: {100, 200, 400}	81
Cluster	DTMC	{512, 1024, 2048}	{200, 400, 600}	Cluster: {GMM, KMeans}	18
Cluster	HMM	{512, 1024, 2048}	{200, 400, 600}	Cluster: {GMM, KMeans}, HMM Comp: {100, 200, 400}	54

TABLE 4: RQ1: Statistical differences between distributions of normal and abnormal and Significant Proportion (the proportion of significant models over all models with diverse hyperparameter settings).

Perspective	p-value	Significant Proportion
OOD	1.02e-15	51%
Adversarial	6.37e-35	51%
Hallucination	4.82e-206	20%

truthfulness of the LLM’s output answer [137], [158]. The truthfulness is the output of a finetuned GPT3-13B (GPT-judge) that is specified for estimating the degree of whether each answer is true or false, which is the common practice on TruthfulQA [137]. In the OOD sample and adversarial attack detection task, instead of state-level binding, we focus on the *transition probability* as the semantic representation.

4.3 RQ1: Can the abstract model differentiate the normal and abnormal behaviors of LLM?

Evaluation Design: Abnormal behavior awareness is a vital step for LLM analysis and interpretation. Therefore, RQ1 is devoted to conducting a preliminary investigation to acknowledge whether the constructed abstract models have the *potential* to characterize the behavior of the LLM from the lens of abnormality.

In particular, as mentioned in Section 4.1, our analysis centers on inspecting the distribution difference of the *transitions probabilities* from the abstract models between the normal and abnormal instances. We consider *transitions probabilities* as valid representatives of models’ characteristics since they encapsulate the intrinsic and irreducible nature of state transition.

To answer RQ1, we assess the difference between normal and abnormal data *qualitatively* and *quantitatively* from three distributions: 1) the normal instances in the training dataset, 2) the normal instances in the test dataset, and 3) the abnormal instances in the test dataset. The normal instances in the training datasets and test datasets are considered to have similar distributions and represent the corpus that the LLM should properly process. In contrast, the abnormal instances in the test dataset are the contexts that are out of the scope of the training data. We consider the subject LLM to have abnormal behavior characteristics (e.g., faulty outputs, irregular hidden states behaviors) when processing these different instances. Furthermore, we expect to capture such dissimilarity in the abstract models from the lens of transition probabilities, which can reveal the consistency between the subject LLM and the corresponding abstract model. The hyperparameters of the abstract model is randomly picked from the hyperparameter space. For the qualitative assessment,

we rely on the Kernel Density Estimation (KDE) [159] plot to observe the distribution of transition probabilities between three types of instances. In terms of quantitative assessment, we investigate the statistical significance of three distributions using Mann-Whitney U test [160] (i.e., p-value).

Results: We detail the distribution difference assessment from the KDE plot and the Mann-Whitney U test, respectively, and summarize the findings at the end of this subsection.

- **Qualitative Assessment.** Figure 5 illustrates the distribution of transition probabilities w.r.t. three types of instances (normal instances in training data, normal instances in testing data, and abnormal instances in test data) across three different tasks. Among all three tasks, the distributions of the transitions are highly aligned for normal instances in the training and test data. Namely, the abstract model has consistent behavior characteristics when dealing with normal instances. In terms of the normal and the abnormal instances, we also notice divergent distribution shapes in SST-2 and AdvGLUE++ datasets. This visual observation supports the proposed assertion that it is possible to distinguish normal and abnormal instances from the distribution of transition probabilities of the abstract model.
- **Quantitative Assessment.** We further conduct statistical significant tests to consolidate the visual findings from the KDE plots. As presented in Table 4 we show the p-value of the randomly picked model and the significant proportion, which represents the proportion of all models with different hyperparameter settings (180 hyperparameter settings in total) that resulted in a p-value less than 0.05. We find that in OOD, 51% of the abstract models satisfies the significance difference, while the proportions of the adversarial attack and hallucination are 51% and 20%, respectively. This indicates that the abstract models on OOD show a greater difference between the normal and abnormal instances.

In general, our results validate our initial hypothesis and underline the capability of the abstract model to discover the abnormal behavior of the LLM. The transition probabilities of the abstract model are aligned with normal instances and have significant differences while abnormal instances are encountered.

Answer to RQ1: Our experiment results confirm that the abstract model has the potential to characterize the anomalies of the subject LLM.

4.4 RQ2: How do different modelling techniques and corresponding configurations impact the quality of the abstract model?

As shown in RQ1, the abstract model is capable of detecting abnormal behavior of the subject LLM. In RQ2, we further examine what factors impact the quality of the abstract models from the state abstraction and trace construction perspectives. We first study the hyperparameters, including PCA dimensions, history steps, partition techniques (GMM, K-means, and Grid), and modelling methods (DTMC and HMM), as shown in Table 3. Specifically, we aim to understand how these factors can benefit the general model analysis in terms of metrics, which include Succinctness (SUC), Stationary Distribution Entropy (SDE), Sink State (SS), Sensitivity (SEN), Coverage (COV), and Perplexity (PERP).

We normalized the metric values based on rank, indicating their relative ranking across metrics rather than absolute magnitudes. A higher normalized value means a better rank compared to other settings in the metric. Detailed metrics values for different PCA dimensions, cluster methods, and model types are available at our website <https://sites.google.com/view/llm-luna>.

4.4.1 RQ2.1: How is the state abstraction correlated with abstract model-wise evaluation metrics?

Evaluation Design: As mentioned in Section 3.2, we conduct a series of dimension reduction and state abstraction techniques to decompose and narrow down the large-scale concrete state space of the LLM. Specifically, we apply PCA for dimension reduction on collected concrete states from the subject LLM. One crucial parameter of PCA is the number of components retained for the abstract state; therefore, we select three different levels of component numbers (Low, Medium, and High) to investigate how the degree of dimension reduction impacts the quality of the abstract model. It is worth mentioning that we set the PCA components to three comparative levels instead of concrete numbers due to the different state aggregation mechanisms performed by the three partition methods. Namely, a number of PCA components processable by GMM may not be feasible in terms of the Grid method; thus, for each state partition method, we arrange the PCA components to comparative levels to investigate its effect on the quality of the model. Specifically, as shown in Table 3, we select {512, 1024, 2048} as corresponding low, medium, and high PCA component settings for cluster-based state partition method (KMeans and GMM), and {3, 5, 10} for grid-based method.

State partition techniques are subsequently applied to aggregate the concrete states with close spatial distance into one abstract state. We utilize three commonly used state partition approaches, e.g., GMM, KMeans and Grid, to probe their effectiveness in mapping the large continuous concrete state space onto a compact discrete state space. To eliminate the potential impact of the different dimension settings from PCA, we conduct each state partition method on different PCA component settings and take the average performance across all PCA settings as the final result.

In terms of the abstract model quality measurement, we adopt the *abstract model-wise metrics* specified in Section 3.4

to inspect the characteristics of different abstraction settings from various aspects.

Results: From Figure 6 and Figure 7, we acknowledge the following findings about PCA dimensions and state abstraction approaches:

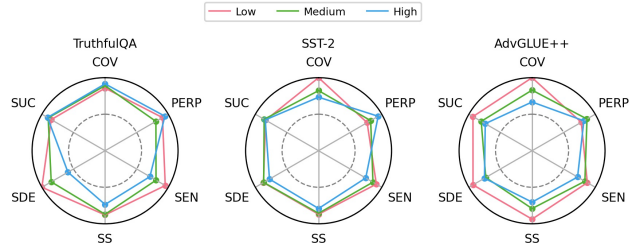


Fig. 6: RQ2.1: Model-wise metrics w.r.t. PCA dimension (number of components).

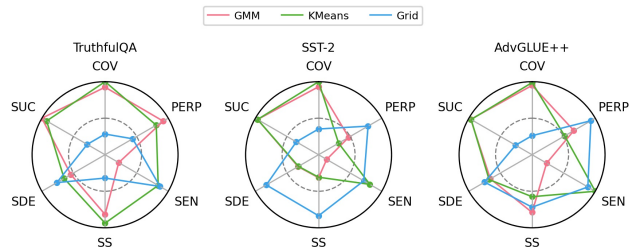


Fig. 7: RQ2.2: Model-wise metrics w.r.t. state partition method.

- PCA Components:** Intuitively, Figure 6 shows that an increase in the number of PCA components has a positive impact on the perplexity but adversely affects the coverage and succinctness across three tasks. The perplexity measures the quality of the abstract model from the degree of well-fitting to the training distribution and the unpredictability within its transitions, respectively. Meanwhile, coverage and succinctness reveal the level of state space exploration and state/transition reduction rate. Therefore, we consider a higher number of PCA components can reinforce the construction of the abstract model through distribution matching (perplexity). Nevertheless, overly persevered state features (PCA components) produce a relatively large state space after the abstraction, which can prohibit the abstract model from effectively narrowing the concrete state space (succinctness), fully exploring the abstract state space with limited training data (coverage) and characterizing the transition properties of the subject LLM (perplexity).
- Cluster-based method:** Regardless of the PCA dimension, clustering-based methods, KMeans, and GMM usually present the highest or near-highest values regarding coverage and succinctness across three datasets, as shown in Figure 7. We consider the clustering-based state abstraction methods (KMeans and GMM) to be more efficient in aggregating the concrete states. Namely, unlike the grid-based approaches, the clustering-based methods only create new abstract states if a group of concrete states is gathered within certain spatial distances; thus,

less abstract state space is generated for sparse concrete states.

GMM’s performance typically lies between KMeans and Grid. It displayed moderate mean values for most metrics, such as coverage and succinctness. In addition, we also observe that KMeans achieves a higher score on the sensitivity metric than GMM. As mentioned in Section 3.4, the sensitivity metric measures the change of abstract states against small perturbations on concrete states. Thus, the abstract states formed by KMeans can retroactively signify the small perturbations that may drastically alter the outputs of the LLM. Moreover, both KMeans and GMM exhibit some drawbacks in terms of stationary distribution entropy and perplexity (SST-2 and TruthfulQA datasets). Such findings indicate that the clustering-based state abstraction methods may have limitations to inherently preserve the training distribution and the deterministic nature of the transitions in the LLM. Furthermore, we find that clustering-based methods have higher correlations compared to the grid-based method.

It is worth mentioning that, as illustrated in Figure 5, the transition distributions of the abstract models across three datasets support this finding. In particular, the difference in the transition distributions between the normal and abnormal instances in TruthfulQA and SST-2 are relatively more significant (Table 4). Therefore, KMeans and GMM fall short of characterizing and reflecting such distribution differences in the constructed abstract states.

- **Grid-based method:** The grid-based approach performs better than cluster-based methods in terms of perplexity (except for TruthfulQA) and stationary distribution entropy while having comparable scores on sensitivity and sink states. It implies that the grid partition method has advantages in imitating the distribution and transition characteristics of the subject LLM.

Additionally, we notice a performance drop in coverage and succinctness, and we consider such limitations to be caused by the nature of the grid method. Namely, the grid-based method uniformly partitions the concrete state space along each dimension; therefore, an abstract state may be created even if no concrete states fall in this grid partition. If the dimension of the concrete space is high (depends on PCA) but the concrete states are densely distributed to certain areas, there will be a large portion of void abstract states generated (e.g., no abstract states and transitions exist in certain areas of abstract state space). Also, the perplexity of the Grid-based method shows a degradation on TruthfulQA compared to AdvGLUE++ and SST-2, which can be caused by the similar distribution between the normal and abnormal transitions (as illustrated by Figure 5).

Answer to RQ2.1: In our evaluation, a specific design on the number of PCA components is needed to balance the trade-offs among different quality metrics. The cluster-based method usually has advantages in state space reduction and exploration but falls short of preserving the distribution and deterministic nature of transitions. The grid-based method shows the opposite features.

4.4.2 RQ2.2: How is the model construction method correlated with abstract model-wise evaluation metrics?

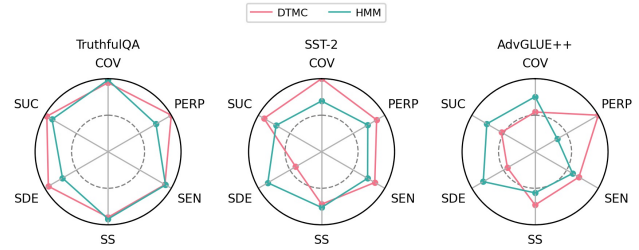


Fig. 8: RQ2.2: Model-wise metrics w.r.t. Abstract Model Type.

Evaluation Design: Different types of model construction approaches are crucial regarding the quality of the abstract model as they concatenate the abstract states and reproduce the transition property of the subject LLM. In this RQ, we take DTMC and HMM as two candidate model construction methods to investigate their impact on the abstract model-wise properties.

Results: After analyzing the results in Figure 8, we have several findings for model construction methods. Despite the number of PCA components and state abstraction techniques, DTMC shows close or beyond performance on succinctness, coverage, sensitivity, sink state, and perplexity (except AdvGLUE++). In terms of HMM, generally, HMM lags behind DTMC in most metrics except stationary distribution entropy and sensitivity. We consider such differences can be attributable to the processes of building abstract transitions. Specifically, DTMC maps the abstract transitions directly according to the existence of concrete transitions, whereas HMM leverages a fitting algorithm to compute the transition probability with a maximized likelihood of observations.

The reason for HMM’s relatively inadequate performance on model-wise metrics might be the two-level state abstraction mechanism, i.e., fit the hidden states and transitions on top of the abstract state. Conversely, the direct transition abstraction of DTMC can be more effective in tracing back the transition characteristics of the LLM. Considering the specialties of LLMs, a desired model construction method should effectively and efficiently capture and characterize the transition properties of the LLM for specific tasks.

Answer to RQ2.2: In our evaluated settings, the choice between DTMC and HMM largely hinges on the specific metric of interest. DTMC’s versatile performance makes it a premier choice for downstream applications.

4.5 RQ3: How does the framework perform across target trustworthiness perspectives, and how is its performance correlated with both semantics-wise and abstract model-wise metrics?

In this RQ, we first want to examine the *effectiveness* of our model-based analysis, i.e., whether it could detect abnormal behavior from the three studied trustworthiness perspectives. Additionally, we are also interested in investigating

the correlation between the newly proposed semantic-wise metrics and the analysis performance. Recall that semantics-wise metrics are designed to measure the quality of the abstract model in terms of *semantics*, which is supposed to have strong correlations with the analysis task performance. We want to confirm this point in this RQ. This exploration can also be used to guide the selection of a good abstract model for the analysis. Similarly, the correlation between abstract model-wise metrics and the performance is examined for the model selection procedure.

4.5.1 RQ3.1: How does our framework perform on trustworthiness perspectives regarding semantics-wise metrics?

Evaluation Design: In RQ3.1, we first try to assess the effectiveness of the model-based analysis across three trustworthiness perspectives, by checking the performance of the abnormal behavior detection (as described in Section 3.5). We choose Area Under the Receiver Operating Characteristic Curve (ROC AUC) [151], as the metric to evaluate the performance of the detection task. Typically, if the ROC AUC of a method is higher than 0.5, we consider it an effective one (better than random). We check the performance of detection based on models with different hyperparameter settings (a total of 180 settings) and show their performance (mean, maximum, minimum, median, standard deviation, and variation). By initially examining the ROC AUC, we gain a preliminary understanding of the effectiveness of our model-based analysis over the hyperparameter space.

Moreover, we want to examine the correlation between the newly introduced semantic-wise metrics and ROC AUC by computing the Pearson correlation coefficient [161] between them. To be more specific, we go over all hyperparameter settings, and for each specific hyperparameter setting, we generate a corresponding abstract model, determine its ROC AUC for every trustworthiness perspective and calculate the semantic-wise metrics for these settings. Each ROC AUC is correlated with the corresponding semantic-wise metrics, which can be computed by the Pearson coefficient between the ROC AUC and the values of each semantic-wise metric. Recall that semantics represent the level of satisfaction of the LLM w.r.t. the trustworthiness perspective. This investigation helps us understand whether the newly proposed semantics-wise metrics have the potential to indicate the performance of the analysis procedure to some extent. Recall that the semantics-wise metrics include Preciseness (PRE), Entropy (ENT), Surprise Level (SL), n -gram Derivative Trend (NDT), Instance Value Trend (IVT), and n -gram Value Trend (NVT).

TABLE 5: RQ3.1: The ROC AUC for different datasets. Max: Maximum, Min: Minimum, Std. Dev.: Standard Deviation, Var.: Variance.

Perspective	Mean	Max	Min	Median	Std. Dev.	Var.
OOD	0.55	0.65	0.50	0.53	0.04	0.00
Adversarial	0.59	0.83	0.50	0.56	0.09	0.01
Hallucination	0.67	0.71	0.55	0.69	0.04	0.00

Results: Table 5 shows the statistics of ROC AUC over all models with different hyperparameter settings. The performance of detecting OOD samples is worse than the

TABLE 6: RQ3.1: Top 5 Settings for Each Perspective and ROC AUC

Perspective	PCA	Partition	#State	Model	ROC AUC
OOD	1024	GMM	200	DTMC	0.65
	512	GMM	512	DTMC	0.64
	5	Grid	1.0e+5	DTMC	0.63
	10	Grid	1.0e+10	DTMC	0.62
	512	GMM	600	DTMC	0.62
Adversarial	10	Grid	1.5e+11	DTMC	0.83
	10	Grid	1.0e+11	DTMC	0.80
	10	Grid	1.0e+11	DTMC	0.79
	10	Grid	1.0e+11	DTMC	0.76
	5	Grid	1.5e+6	DTMC	0.75
Hallucination	3	Grid	125	DTMC	0.71
	3	Grid	1000	DTMC	0.71
	1024	GMM	600	DTMC	0.71
	2048	KMeans	400	DTMC	0.71
	2048	GMM	600	DTMC	0.70

TABLE 7: RQ3.1: Pearson Coefficient of Semantic-wise Metrics with respect to ROC AUC.

Perspective	PRE	ENT	SL	NDT	IVT	NVT
OOD	0.94	0.34	0.14	-0.25	-0.25	-0.34
Adversarial	0.95	-0.59	-0.18	0.48	-0.19	-0.75
Hallucination	0.81	-0.07	-0.18	-0.38	-0.03	0.51

other tasks, with the lowest mean (0.55), maximum (0.65), and median (0.53) ROC AUC, which is reasonable, as the OOD samples only differ very slightly from the original one and are hard to be detected. Moreover, the maximum performance of adversarial sample detection achieves the highest results among all tasks (0.83). This conforms to the result of RQ1 that the significant proportion of models built on adversarial dataset is higher than the others, indicating that adversarial dataset is intrinsically easier for abnormal behavior detection and our model indeed can catch the difference between normal and abnormal behavior. We can also see that the data is concentrated, with almost zero variance and a small standard deviation (less than 0.1).

Table 6 shows the experimental result of the top-5 performance with their hyperparameters setting of the detection tasks. We can see that all top-5 models are DTMC. The reason for the unsatisfying performance of HMM might be the two-layer state abstraction, i.e., HMM learns the hidden states on the abstract states. For adversarial sample detection, the best ROC AUC is 0.83, with a PCA dimension of 10 and grid partition of DTMC, while the ROC AUC of OOD sample detection and hallucination are 0.65 and 0.71, respectively. We conclude that our model-based abnormal behavior detection is effective, as it is higher than the random approach (with ROC AUC 0.5).

For the correlation analysis, the Pearson coefficient for each semantic-wise metric with respect to the ROC AUC values for different abstract model settings is presented in Table 7. We have the following observations. Firstly, semantics preciseness (PRE) and ROC AUC have a strong positive correlation (0.94, 0.95, and 0.81 for three tasks), which suggests the necessity of building a semantically precise for getting good analysis performance. In addition, we can see some mixtures of positive and negative correlations, e.g., n -gram derivative trend (NDT) and n -gram value trend

(NVT), in different datasets, which indicates that in practice, the users should analyze different metrics accordingly for selecting appropriate hyperparameters, as they might show diverse correlations to the final analysis task. There are also metrics that show weak correlations, e.g., the surprise level (SL) for three tasks are 0.14, -0.18 , and -0.18 respectively.

Answer to RQ3.1: Our model-based abnormal behavior detection is effective in three trustworthiness perspectives. The semantics-wise metrics also show different correlations with the analysis performance in different tasks, except for preciseness.

TABLE 8: RQ3.2: Pearson coefficient of model-wise Metrics w.r.t. ROC AUC. SUC: Succinctness, COV: Coverage, SEN: Sensitivity, SS: Sink State, PERP: Perplexity, and SDE: Stationary Distribution Entropy

Perspective	SUC	COV	SEN	SS	PERP	SDE
OOD	-0.35	-0.33	0.05	0.35	0.46	0.27
ADV	-0.38	-0.33	0.04	0.34	0.46	0.27
Hallucination	0.02	0.06	0.20	0.04	-0.08	-0.44

4.5.2 RQ3.2: How is the performance of the framework correlated with the abstract model-wise metrics?

Evaluation Design: In RQ3.2, we examine the correlation between the abstract model-wise metrics and ROC AUC by computing the Pearson correlation coefficient [161] between them. Our aim is to understand the correlation between the performance (Table 8) and their corresponding model-wise metrics. Identifying this correlation can help us choose the abstract model with potentially good performance based on abstract model-wise metrics in the future.

Results: The correlation is shown in Table 8. After analyzing the correlations between ROC AUC and abstract model-wise metrics, we have several findings. Similar to the finding in RQ3.1, some metrics and ROC AUC have different correlations for different trustworthiness perspectives, e.g., succinctness (SUC) and perplexity (PERP). The sensitivity (SEN) of the model has weak correlations with the performance, as the Pearson coefficient is lower than 0.3. This difference highlights the importance of considering diverse metrics as performance indicators for a comprehensive assessment.

Answer to RQ3.2: Similar to semantics-wise metrics, the abstract model-wise metrics exhibit different correlations with the analysis performance in different domains as well. In practice, the user could use different metrics combinations to guide the model selection in order to have a comprehensive analysis performance.

5 DISCUSSION

Abstract Model Construction for LLM. Some research works have demonstrated that a well-constructed abstract model can behave as an indicator to reveal the internal behavior of the target neural network model [29], [54], [57], [121], [126]. Adequate model construction techniques are

vital to retroactively reflect the corresponding characteristics of the studied system. Nevertheless, considering the very large model size and the distinct self-attention mechanism of LLMs, it is still unclear to what extent existing methods are effective on LLMs. Hence, our framework, LUNA, collaborates three state abstraction methods and two model construction techniques with a total of 180 different parameter configurations to extensively explore the effectiveness of popular model-based analysis approaches.

From the evaluation results, we find that cluster-based state partition methods (KMeans, GMM) and the grid-based method have distinct advantages on different model quality measurement metrics. Meanwhile, in terms of the methods of model construction, DTMC exhibited close or beyond performance on most of the metrics than HMM, which implies it is a potential candidate to model the state transition features of LLMs. It is worth noting that the efficacy of the abstraction and modelling techniques varies on tasks and trustworthiness perspectives. For instance, KMeans gets superior performance scores on Succinctness and Coverage on both TruthfulQA and SST-2 datasets but relatively inadequate performance on AdvGLUE++ dataset. Such a finding signifies that explicit selection of methods and appropriate parameter tuning are necessary to maximize the effectiveness of existing techniques regarding abstract model construction. Therefore, advanced and LLM-specific abstract model construction techniques are called to capture and represent the behavior characteristics of LLMs regardless of types of tasks and trustworthiness perspectives.

Abstract Model Quality Measurement. In this work, we tried our best and chose as many as 12 metrics to initiate a relatively comprehensive understanding of the quality of the constructed model from both abstract model-wise and semantics-wise. Particularly, abstract model-wise metrics assess the intrinsic properties of the model regardless of subject trustworthiness perspectives, such as the stability of the model and the degree of well-fitting to the distribution of training data. We notice that Coverage and Succinctness, which measure the level of compression of the abstract model, provide more insights for dimension reduction and abstract state partition. Moreover, Stationary Distribution Entropy, Perplexity and Sink State make more efforts to guide the selection of model construction methods and subsequent parameter tuning. Such metrics help to enhance the quality of the model towards better training distribution fitting and the ability against small perturbations.

In contrast, semantics-wise metrics measure the quality of the model from the angle of the degree of satisfaction w.r.t. trustworthiness perspectives. In particular, from Section 4.5, we notice that Preciseness, Entropy and n -gram Value Trend are more correlated with the performance of the model regarding different trustworthiness perspectives. Some metrics may have distinct adaptabilities on certain applications. For example, Surprise Level and n -gram Derivative Trend have finer effectiveness in describing the quality of the model on adversarial and hallucination detection.

In general, different metrics are needed to collaboratively guide the construction of the abstract model and secure the quality from diverse aspects. Also, some metrics are potentially fit to tackle specific downstream tasks or trustworthiness perspectives; thus, more research is called to

prospect the explicit metrics for particular applications or quality requirements.

Model-based Quality Assurance for LLM. The fast-growing popularity of LLMs highlights the escalating influence of LLMs across academia and industry [5], [7], [162]. With the witness to the adoption of LLMs in a large spectrum of practical applications, LLMs are expected to carry as foundation models to boost the software development lifecycle in which trustworthiness is critical. Namely, quality assurance techniques explicitly in the context of LLMs are of urgent need to enable the deployment of LLMs on more safety, reliability, security and privacy-related applications.

Our framework *LUNA* aims to provide a general and versatile platform that assembles various modelling methods, downstream tasks and trustworthiness perspectives to safeguard the quality of LLMs. Moreover, considering the extensibility of the framework, *LUNA* is expected to behave as a foundation that enables the following research to implement new advanced techniques for more diverse tasks across different domains. The results from Section 4 confirm that the abstract model can act as a beacon to disclose abnormalities in the LLM when generating responses to different inputs. Specifically, the abstract model extracts and inspects the inner behavior of the LLM to detect whether it is under unintended conditions that can possibly produce nonfactual or erroneous outputs. The model embeds semantics w.r.t. different trustworthiness perspectives to extend its capability to tackle diverse quality concerns. In addition, we consider our framework can play roles in extensive quality assurance directions, such as online monitoring [163]–[166], fault localization [167], [168], testing case generation [19], [21], [169], [170] and output repair [28], [171], [172]. For instance, by leveraging the trajectories of the states and corresponding semantics w.r.t. a specific output, it is possible to trace back and precisely localize the faulty segments within the output tokens.

In this paper, we take an early step to present a model-based LLM analysis framework, *LUNA*, to initiate exploratory research towards the quality assurance of LLMs. Our experiment results show that the abstract model can capture the abnormal behaviors of the LLM from its hidden state information. We conduct a series of modelling techniques with a diverse set of quality measurement metrics to deliver a comprehensive understanding of the capability and effectiveness of our framework. Hence, we find that *LUNA* shows performant abilities to detect the suspicious generations of LLMs w.r.t. different trustworthiness perspectives.

6 THREATS TO VALIDITY

In this Section, we discuss the threats that may affect the validity of our study and the actions we have taken to mitigate them.

Internal Threats. The configurations in state abstraction and model construction can be an internal threat that impacts our evaluation results. A satisfactory abstract model should, on the one side, maximally narrow down the concrete state space to make it more compact and processable; on the other side, form abstract states that are representative of distinct LLM behaviors. To mitigate this threat, in this

study, we propose a total of 180 configuration settings that may affect the performance of the abstract model to obtain a comprehensive and constructive understanding of how different parameter configurations impact the effectiveness of the abstract model.

External Threats. The generality of our framework to other LLMs, tasks and trustworthiness perspectives beyond this study can be an external threat. In light of different LLM structures, output types and task requirements, our results may not always be applicable to other scenarios. To mitigate this threat, we select the three currently widely concerned trustworthiness perspectives on three different datasets to conduct the experiments. Likewise, multiple modelling techniques and evaluation metrics are enclosed in our framework to enhance its applicability to other trustworthiness perspectives and applications.

Construction Threats. It is possible that our evaluation metrics may not fully characterize all possible performance aspects of the model. To mitigate this threat, we investigate a large number of metrics of model quality measurements from previous works [54], [121], [126], [138], [139], and carefully select twelve different metrics from two categories: abstract model-wise and semantics-wise. The former measures the quality of the model from the angles of the level of abstraction, distribution fit and sensitivity, etc; the latter evaluates the model performance based on the level of satisfaction w.r.t. different trustworthiness. By incorporating these metrics, we tried our best to deliver an adequate assessment.

7 RELATED WORK

7.1 Quality Assurance of LLM

Quality assurance in deep learning-driven NLP software has recently garnered significant interest from industry and academia. On one side, related research seeks to empirically evaluate the trustworthiness of these models more thoroughly and comprehensively. Meanwhile, there is a concerted demand and push toward devising advanced techniques to predict failures, identify ethical concerns, and enhance various abilities of current models.

Regarding empirical evaluation, some benchmarks have been proposed, addressing factual consistency [137], [149], [173], [174], robustness [110], [175], toxicity [176], and hallucination [149], [156] in tasks like QA and text summarization. These benchmarks comprise datasets that are either human-labeled [149], [174], extracted from external resources [156], [173], transformed from other datasets [110], [175], or labeled/generated by AI models [137], [176]. While many studies target specific AI model facets for select tasks, the multifaceted nature of LLMs warrants broader evaluations. Recent research delves into multiple capabilities of LLMs, encompassing faithfulness of QA [177], security of generated code [178] and its correctness [179], mathematical capabilities [180], and logical reasoning skills [181]. Notably, HELM [182] stands out as an important study in this domain. It conducts extensive tests across seven metrics in 42 scenarios for 30 language models, offering a comprehensive insight into the current landscape of LLMs. DecodingTrust [9] is another important benchmark assessment of LLMs that concentrates on diverse perspectives

of trustworthiness. In our work, we select two important salient tasks from this study: adversarial detection and OOD detection.

These empirical studies reveal that while LLMs excel in various tasks, they often lack trustworthiness and transparency. To tackle these shortcomings, some recent studies suggest some promising directions such as data-centric methods [183]–[186], uncertainty estimation [144], [187]–[192], controlled decoding [193]–[196], self-refinement [197]–[201], and leveraging external knowledge during inference [162], [202]–[206].

Data-centric approaches are model-agnostic and formulate related problems as unintended behavior detection. Typically, these methods gather data and train classifiers to identify undesired content. A notable instance is OpenAI’s moderation system, offered as an API service [185]. This system’s training data encompasses content related to sexuality, hate, violence, self-harm, and harassment. Uncertainty estimation, often lightweight and black-box in nature, uses uncertainty scores as indicators for the models’ trustworthiness. Manakul et al. [144], for instance, introduce a black-box hallucination detection technique based on token-level prediction likelihood and entropy, while Huang et al. [189] explore the efficacy of both single and multi-inference uncertainty estimation methods.

While the above two approaches focus more on detection, the remaining three aim to directly improve the generated content. Controlled decoding techniques freeze the base LLM while guiding the text generation to achieve the desired attributes. Mireshghallah et al. [195], for example, propose energy-based models to steer the distribution of generated text toward desired attributes, such as unbiased content. Cao et al. [196] suggest employing dead-end analysis to reduce LLM toxicity. Drawing inspiration from human introspection, self-refinement methods have been introduced. Huang et al. [199] instruct LLMs to generate confident answers for unlabeled questions, which are then used in further training. Madaan et al. [200] suggest that LLMs critique and refine their own outputs. Lastly, LLMs augmented with external databases can address the “brain-in-the-vat” dilemma [207], leading to more accurate inferences. Examples include WIKI-based chatbots [204] and Retrieval-Augmented LLMs [205].

Among the relevant studies, the work by Azaria et al. [81] and Li et al. [158] bear the closest resemblance to ours. While the majority of these approaches adopt black-box methodologies, they try to analyze the relationship between LLMs’ internal and their trustworthiness. Azaria et al. utilize the hidden layer activations of LLMs as features to train a classifier for assessing the truthfulness of generated content. Li et al. first probe LLMs to find the correlation between truthfulness and attention heads and subsequently leverage this insight for inference-time intervention, aiming to produce more accurate responses. In contrast, our framework emphasizes holistic model extraction and stateful analysis, offering a more systematic exploration of the stateful characteristics inherent to LLMs.

7.2 Model-based Analysis for Stateful DNNs

Interpreting the behavior of stateful deep neural networks is challenging, considering the potentially countless concrete

states the model can reach and its near black-box nature. Fortunately, there are already some successful attempts for the RNN-series, a representative stateful architecture before the transformer era. Some theoretical research indicates that, while RNNs are Turing-complete [208], practical constraints such as finite precision and limited computation time render them equivalent to finite-state automata (FSA) [209], [210]. These insights potentially bridge the gap between the intricate black-box nature of RNNs and the well-understood FSAs, which have been rigorously examined in classical formal theory.

Interestingly, attempts to leverage FSAs for RNN analysis predate these theoretical explorations, originating as early as the 1990s. These studies try to first abstract the concrete (hidden) state space and then build FSAs that try to mirror RNNs behavior. Omlin et al. introduce a method to segment the hidden state space into q equal intervals, with q being the quantization level [118]. Zeng et al. [119] and Cechin et al. [120] propose to use K-means to cluster concrete states into abstract states. These pioneering efforts from the pre-deep learning era paved the way for subsequent model-based analysis of more sophisticated RNNs.

The advent of deep learning has ushered in two transformative shifts in the field: an influx of data and increasingly complex architectures. Concurrently, the model-based analysis has also evolved accordingly. These efforts broadly fall into two categories: those that focus on extracting a transparent surrogate model replicating RNN decisions [56], [60], [122], [211]–[216] and those emphasizing transition traces with associated semantic meanings related to downstream tasks [29], [54], [121], [126], [217].

For the former, one line of research leverages a more formal way for the FSA extraction, such as using Angluin’s L^* algorithm [211] and its variant [212] or finding the Hankel matrix of a black-box system and constructing weighted automata from it [213]. These strategies treat RNNs as teachers and craft automata through querying. Alternatively, a more empirical path focuses on analyzing direct transition traces derived from training data. For example, Dong et al. first obtain symbolic states by clustering on concrete hidden states and build probabilistic automata based on a learning algorithm [60]. Zhang et al. use similar methods to build symbolic states but enhance the context-awareness of the extracted model by compositing adjacent states [122]. Merrill et al. introduce an automata extraction technique based on state merging, which performs better than k-means [215]. Hong et al. utilize a transition path matching method, integrate identified patterns with state merging, and offer a more systematic approach to constructing automata [216]. All these methods aim to extract automata for better consistency with source RNNs.

Rather than creating an exact FSA mirroring a target DNN’s behavior, stakeholders may prioritize specific properties of stateful software systems, such as security, safety, privacy, and correctness. Consequently, some studies focus more on studying these specific properties and obtaining insights observed from the extracted FSA instead of seeking a perfect decision alignment. For instance, DeepStellar [54] and its successor, Marble [121], delve into the adversarial robustness of RNNs using discrete probabilistic models. Conversely, AbASG [217] employs automata for adversarial

sequence generation. DeepMemory [126] performs analysis of RNN memorization and its associated security and privacy implications using semantic Markov models. RN-NRepair [29] performs repair of an RNN through model-based analysis and guidance. DeepSeer [218] employs finite automata as the central methodology for human interactive design to enable RNN debugging. The diverse successes of these methods underscore the efficacy of model-based analysis in stateful DNN systems.

Our work differentiates from the above studies in two key aspects. Firstly, we endeavor to develop a universal analysis framework designed for versatile property analysis across a broad spectrum of tasks in stateful DNN software systems in a plug-and-play manner. Secondly, our emphasis lies on the Transformer architecture and the corresponding LLMs. These models operate on a very distinct mechanism (e.g. attention mechanism) and adhere to unique training workflows. Recent studies find that the Transformer has much better empirical representation power than LSTM in simulating pushdown automation, calling the need for adapted analysis methods [219]. On the other hand, various papers have pointed out that some important capabilities of the Transformer, including factual associations [220] and object identification [221], stem from propagating complex information through tokens, inherently exhibiting stateful characteristics. While some related studies have investigated the potential of enhancing language models with finite automata for improved performance [222] or constraining their outputs using DFA [223], a comprehensive model-based analysis and framework remain to be absent.

7.3 LLM and Software Engineering

Recently, a growing number of research works show that LLMs have already made great potential in various phases throughout the software production lifecycle. Many researchers and industrial practitioners have investigated and examined the capabilities of LLMs for a large spectrum of applications in software engineering domain, such as code generation [1], [224]–[227], code summarization [228]–[230], program synthesis [231], [232], test case generation [233]–[236] and bug fixing [237]–[241].

In particular, Dong et al. [227] leverage ChapGPT to present a self-collaboration framework for code generation. Namely, multiple LLMs are assigned with different roles (i.e., coder, tester, etc.) following a general software development schema. Such an LLM-powered self-collaboration framework achieves state-of-the-art performance in solving complex real-world code generation tasks. Ahmend et al. [228] investigate the effectiveness of few-shot training on LLM (Codex [242]) for code summarization tasks. Their experiment results confirm that leveraging data from the same project with few-shot training is a promising approach to improve the performance of the LLM in code summarization. Nijkamp et al. [224] release a family of LLMs (CODEGEN) trained on both natural language and programming language data to demonstrate the ability of LLMs on program synthesis. In addition, Lemieux et al. [234] incorporate LLM into the loop to improve search-based software testing (SBST) for programs being tested through a combination of test case generation and other techniques. Last but not

least, Sobania et al. [237] study the capability of ChatGPT in terms of software bug localization and fixing. These works qualify the potential of LLMs as an enabler and a booster to accelerate the software production lifecycle.

Despite the promising SE task-handling capabilities by LLMs, existing works [9], [63], [66], [155], [243], [244] have also pointed out that the current LLMs could potentially suffer critical quality issues across different SE tasks. Specifically, developers sometimes find it hard to understand the code generation process and the code produced by LLMs, and LLMs have also exhibited incorrect behaviors in generating suboptimal or erroneous solutions [66]. Such concerns about the trustworthiness of LLMs and the quality of the corresponding outcomes greatly hinder further adaptation and deployment of LLMs on safety, reliability, security and privacy-related SE applications. Moreover, although a large body of current works in the SE community focuses on leveraging LLM to further promote and accelerate SE applications from different aspects, less attention has been paid to applying and adapting existing SE methodologies to safeguard the trustworthiness of LLMs. As the recently fast-increasing trend of LLM-based techniques for various key stages of the SE lifecycle, it is recognized that LLM would potentially play a more and more important role in the next few years. Therefore, it could be of great importance to establish an early foundation towards a more systematic analysis of LLMs to better interpret their behavior, to understand the potential risks when using it, and to equip the researchers and developers with more tangible guidance (e.g., concrete analysis results and feedback) to facilitate the continuous enhancement of LLMs for practical usage. Therefore, to bridge this gap and inspire further research along this direction, we hope *LUNA*, as a basic analysis framework for LLMs, could be helpful for researchers and practitioners to conduct more deep exploration and exploitation and to design novel quality assurance solutions towards approaching trustworthy LLMs in practice.

8 CONCLUSION

In this paper, we propose *LUNA*, a model-based LLM-oriented analysis framework, to initiate an early exploration towards establishing the foundation for the trustworthiness assurance of LLMs. *LUNA* is designed to be general and extensible, and the core of its current version contains three state abstraction techniques, two model construction approaches, and as many as twelve quality metrics (as indicators) to establish a versatile LLM analysis pipeline. A large-scale evaluation of three trustworthiness perspectives on three datasets with a total of 180 model configuration settings is conducted to investigate the effectiveness of our framework. Our evaluation also performs large-scale comparative studies to better understand the strengths and weaknesses of different modelling approaches in terms of characterizing the internal behavior patterns of LLMs. Overall, advanced LLM-specific modelling methods are often needed to effectively and efficiently transparentize and interpret the behavior characteristics of LLM regardless of types of tasks and trustworthiness perspectives. Moreover, our analysis of twelve metrics motivates further investigation of strategically collaborating different metrics to

provide a relatively comprehensive and meticulous quality measurement for diverse LLM applications. With the fast-growing trend of industrial adoption of LLMs across domains, we hope this early-stage exploratory work can inspire further research along this direction, towards addressing many challenges to approaching trustworthy LLMs in the coming era of AI.

REFERENCES

- [1] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Chi conference on human factors in computing systems extended abstracts*, 2022, pp. 1–7.
- [2] C. S. Xia and L. Zhang, "Conversational automated program repair," *arXiv preprint arXiv:2301.13246*, 2023.
- [3] W. Zhang, Y. Deng, B. Liu, S. J. Pan, and L. Bing, "Sentiment analysis in the era of large language models: A reality check," *arXiv preprint arXiv:2305.15005*, 2023.
- [4] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," *arXiv preprint arXiv:2211.01910*, 2022.
- [5] "Chatgpt," <http://chat.openai.com>, 2023.
- [6] "Gpt4," <https://openai.com/gpt-4>, 2023.
- [7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
- [8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.
- [9] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Dutta, R. Schaeffer *et al.*, "Decodingtrust: A comprehensive assessment of trustworthiness in gpt models," *arXiv preprint arXiv:2306.11698*, 2023.
- [10] H. Raj, D. Rosati, and S. Majumdar, "Measuring reliability of large language models through semantic consistency," *arXiv preprint arXiv:2211.05853*, 2022.
- [11] B. Wang, S. Wang, Y. Cheng, Z. Gan, R. Jia, B. Li, and J. Liu, "Infobert: Improving robustness of language models from an information theoretic perspective," *arXiv preprint arXiv:2010.02329*, 2020.
- [12] B. Alkhamissi, M. Li, A. Celikyilmaz, M. Diab, and M. Ghazvininejad, "A review on language models as knowledge bases," *arXiv preprint arXiv:2204.06031*, 2022.
- [13] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, mar 2023. [Online]. Available: <https://doi.org/10.1145/3571730>
- [14] A. Abid, M. Farooqi, and J. Zou, "Persistent anti-muslim bias in large language models," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 298–306.
- [15] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.
- [16] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.
- [17] S. Wang, Z. Zhao, X. Ouyang, Q. Wang, and D. Shen, "Chatcad: Interactive computer-aided diagnosis on medical image using large language models," *arXiv preprint arXiv:2302.07257*, 2023.
- [18] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.
- [19] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu *et al.*, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, 2018, pp. 120–131.
- [20] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1039–1049.
- [21] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "Deephunter: a coverage-guided fuzz testing framework for deep neural networks," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 146–157.
- [22] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao *et al.*, "Deepmutation: Mutation testing of deep learning systems," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 100–111.
- [23] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.
- [24] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 132–142.
- [25] H. Wang, B. Ustun, and F. Calmon, "Repairing without retraining: Avoiding disparate impact with counterfactual distributions," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6618–6627.
- [26] M. Sotoudeh and A. V. Thakur, "Correcting deep neural networks with small, generalizing patches," in *Workshop on Safety and Robustness in Decision Making*, 2019.
- [27] H. Zhang and W. Chan, "Apricot: A weight-adaptation approach to fixing deep learning models," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 376–387.
- [28] B. Yu, H. Qi, Q. Guo, F. Juefei-Xu, X. Xie, L. Ma, and J. Zhao, "Deeprepair: Style-guided repairing for deep neural networks in the real-world operational environment," *IEEE Transactions on Reliability*, vol. 71, no. 4, pp. 1401–1416, 2021.
- [29] X. Xie, W. Guo, L. Ma, W. Le, J. Wang, L. Zhou, Y. Liu, and X. Xing, "Rnnrepair: Automatic rnn repair via model-based analysis," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11383–11392.
- [30] Q. Hu, Y. Guo, M. Cordy, X. Xie, L. Ma, M. Papadakis, and Y. Le Traon, "An empirical study on data distribution-aware test selection for deep learning enhancement," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–30, 2022.
- [31] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, and B. Xu, "Adaptive test selection for deep neural networks," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 73–85.
- [32] Z. Yang, J. Shi, M. H. Asyrofi, and D. Lo, "Revisiting neuron coverage metrics and quality of deep neural networks," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 408–419.
- [33] V. Riccio and P. Tonella, "When and why test generators for deep learning produce invalid inputs: an empirical study," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1161–1173.
- [34] J. Wang, H. Qiu, Y. Rong, H. Ye, Q. Li, Z. Li, and C. Zhang, "Bet: black-box efficient testing for convolutional neural networks," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 164–175.
- [35] J.-t. Huang, J. Zhang, W. Wang, P. He, Y. Su, and M. R. Lyu, "Aeon: a method for automatic evaluation of nlp test cases," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 202–214.
- [36] Z. Wei, H. Wang, I. Ashraf, and W.-K. Chan, "Deeppatch: Maintaining deep learning model programs to retain standard accuracy with substantial robustness improvement," *ACM Transactions on Software Engineering and Methodology*, 2023.
- [37] R. Schumi and J. Sun, "Semantic-based neural network repair," *arXiv preprint arXiv:2306.07995*, 2023.
- [38] Y. Zhang, Z. Wang, J. Jiang, H. You, and J. Chen, "Toward improving the robustness of deep learning models via model transformation," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–13.
- [39] Y. Li, M. Chen, and Q. Xu, "Hybridrepair: towards annotation-efficient repair for deep learning models," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 227–238.
- [40] T. Zohdinasab, V. Riccio, and P. Tonella, "Deepatash: Focused test generation for deep learning systems," 2023.

- [41] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Efficient and effective feature space exploration for testing deep learning systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 2, mar 2023. [Online]. Available: <https://doi.org/10.1145/3544792>
- [42] N. Humbatova, G. Jahangirova, and P. Tonella, "Deepcrime: from real faults to mutation testing tool for deep learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2023, pp. 68–72.
- [43] A. Stocco, P. J. Nunes, M. D'Amorim, and P. Tonella, "Third-eye: Attention maps for safe autonomous driving systems," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.
- [44] J. Kim, G. An, R. Feldt, and S. Yoo, "Learning test-mutant relationship for accurate fault localisation," *Information and Software Technology*, p. 107272, 2023.
- [45] J. Sohn, S. Kang, and S. Yoo, "Arachne: Search-based repair of deep neural networks," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 4, pp. 1–26, 2023.
- [46] J. Kim, N. Humbatova, G. Jahangirova, P. Tonella, and S. Yoo, "Repairing dnn architecture: Are we there yet?" in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023, pp. 234–245.
- [47] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.
- [48] N. Humbatova, G. Jahangirova, and P. Tonella, "Deepcrime: mutation testing of deep learning systems based on real faults," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 67–78.
- [49] J. Zhou, F. Li, J. Dong, H. Zhang, and D. Hao, "Cost-effective testing of a deep learning model through input reduction," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 289–300.
- [50] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.
- [51] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning internal representations by error propagation," 1985.
- [52] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 79–90.
- [53] B. C. Hu, L. Marsso, K. Czarnecki, and M. Chechik, "What to check: Systematic selection of transformations for analyzing reliability of machine vision components," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022, pp. 49–60.
- [54] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao, "Deepstellar: Model-based quantitative analysis of stateful deep learning systems," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 477–487.
- [55] X. Ren, Y. Lin, Y. Xue, R. Liu, J. Sun, Z. Feng, and J. S. Dong, "Deeparc: Modularizing neural networks for the model maintenance," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 1008–1019.
- [56] I. Khmel'nitsky, D. Neider, R. Roy, X. Xie, B. Barbot, B. Bollig, A. Finkel, S. Haddad, M. Leucker, and L. Ye, "Property-directed verification and robustness certification of recurrent neural networks," in *Automated Technology for Verification and Analysis: 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18–22, 2021, Proceedings 19*. Springer, 2021, pp. 364–380.
- [57] J. Song, X. Xie, and L. Ma, "Siege: A semantics-guided safety enhancement framework for ai-enabled cyber-physical systems," *IEEE Transactions on Software Engineering*, 2023.
- [58] X. Xie, J. Song, Z. Zhou, F. Zhang, and L. Ma, "Mosaic: Model-based safety analysis framework for ai-enabled cyber-physical systems," *arXiv preprint arXiv:2305.03882*, 2023.
- [59] R. Pan and H. Rajan, "Decomposing convolutional neural networks into reusable and replaceable modules," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 524–535.
- [60] G. Dong, J. Wang, J. Sun, Y. Zhang, X. Wang, T. Dai, J. S. Dong, and X. Wang, "Towards interpreting recurrent neural networks through probabilistic abstraction," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 499–510.
- [61] H. Qi, Z. Wang, Q. Guo, J. Chen, F. Juefei-Xu, F. Zhang, L. Ma, and J. Zhao, "Archrepair: Block-level architecture-oriented repairing for deep neural networks," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, pp. 1–31, 2023.
- [62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.
- [63] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," 2023.
- [64] Y. Charalambous, N. Tihanyi, R. Jain, Y. Sun, M. A. Ferrag, and L. C. Cordeiro, "A new era in software security: Towards self-healing software via large language models and formal verification," 2023.
- [65] D. Lo, "Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps," 2023.
- [66] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," *arXiv preprint arXiv:2310.03533*, 2023.
- [67] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, "Biogpt: generative pre-trained transformer for biomedical text generation and mining," *Briefings in Bioinformatics*, vol. 23, no. 6, p. bbac409, 2022.
- [68] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *arXiv preprint arXiv:2211.09085*, 2022.
- [69] Y. Shen, L. Heacock, J. Elias, K. D. Hentel, B. Reig, G. Shih, and L. Moy, "Chatgpt and other large language models are double-edged swords," p. e230163, 2023.
- [70] T. Kocmi and C. Federmann, "Large language models are state-of-the-art evaluators of translation quality," *arXiv preprint arXiv:2302.14520*, 2023.
- [71] E. Kasneci, K. Seifler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günemann, E. Hüllermeier *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and individual differences*, vol. 103, p. 102274, 2023.
- [72] D. B. Lenat, "Cyc: A large-scale investment in knowledge infrastructure," *Commun. ACM*, vol. 38, no. 11, p. 33–38, nov 1995. [Online]. Available: <https://doi.org/10.1145/219717.219745>
- [73] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu *et al.*, "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models," *arXiv preprint arXiv:2304.01852*, 2023.
- [74] R. Mao, Q. Liu, K. He, W. Li, and E. Cambria, "The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection," *IEEE Transactions on Affective Computing*, 2022.
- [75] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, and T. B. Hashimoto, "Benchmarking large language models for news summarization," *arXiv preprint arXiv:2301.13848*, 2023.
- [76] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, pp. 1–10.
- [77] T. Ahmed and P. Devanbu, "Few-shot training llms for project-specific code-summarization," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–5.
- [78] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [79] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.
- [80] I. Malkiel, D. Ginzburg, O. Barkan, A. Caciularu, J. Weill, and N. Koenigstein, "Interpreting bert-based text similarity via activation and saliency maps," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3259–3268.
- [81] A. Azaria and T. Mitchell, "The internal state of an llm knows when its lying," *arXiv preprint arXiv:2304.13734*, 2023.

- [82] H. Chefer, S. Gur, and L. Wolf, "Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 397–406.
- [83] X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou, "Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond," *Knowledge and Information Systems*, vol. 64, no. 12, pp. 3197–3234, 2022.
- [84] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: <https://aclanthology.org/N19-1423>
- [85] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.
- [86] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: <http://arxiv.org/abs/1907.11692>
- [87] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: <https://aclanthology.org/2020.acl-main.703>
- [88] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, S. Zheng, D. Zhou, N. Housby, and D. Metzler, "UL2: Unifying language learning paradigms," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=6ruVLB727MC>
- [89] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.
- [90] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [91] M. T. Ribeiro, S. Singh, and C. 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*, 2016, pp. 1135–1144.
- [92] H. Qiu, S. Zhang, A. Li, H. He, and Z. Lan, "Latent jailbreak: A benchmark for evaluating text safety and output robustness of large language models," *arXiv preprint arXiv:2307.08487*, 2023.
- [93] Y. Li, F. Wei, J. Zhao, C. Zhang, and H. Zhang, "Rain: Your language models can align themselves without finetuning," *arXiv preprint arXiv:2309.07124*, 2023.
- [94] A. Helbling, M. Phute, M. Hull, and D. H. Chau, "Llm self defense: By self examination, llms know they are being tricked," *arXiv preprint arXiv:2308.07308*, 2023.
- [95] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.
- [96] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: <https://openreview.net/forum?id=Hkg4TI9xl>
- [97] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Advances in neural information processing systems*, vol. 31, 2018.
- [98] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=H1VGkIXRZ>
- [99] P. Morteza and Y. Li, "Provable guarantees for understanding out-of-distribution detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7831–7840.
- [100] H. Lang, Y. Zheng, Y. Li, J. Sun, F. Huang, and Y. Li, "A survey on out-of-distribution detection in nlp," *arXiv preprint arXiv:2305.03236*, 2023.
- [101] U. Arora, W. Huang, and H. He, "Types of out-of-distribution texts and how to detect them," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10687–10701. [Online]. Available: <https://aclanthology.org/2021.emnlp-main.835>
- [102] J. Ren, J. Luo, Y. Zhao, K. Krishna, M. Saleh, B. Lakshminarayanan, and P. J. Liu, "Out-of-distribution detection and selective generation for conditional language models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=kJUS5nD0vPB>
- [103] A. Kamath, R. Jia, and P. Liang, "Selective question answering under domain shift," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5684–5696. [Online]. Available: <https://aclanthology.org/2020.acl-main.503>
- [104] J. Wang, X. Hu, W. Hou, H. Chen, R. Zheng, Y. Wang, L. Yang, H. Huang, W. Ye, X. Geng et al., "On the robustness of chatgpt: An adversarial and out-of-distribution perspective," *arXiv preprint arXiv:2302.12095*, 2023.
- [105] K. Krishna, J. Wieting, and M. Iyyer, "Reformulating unsupervised style transfer as paraphrase generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 737–762. [Online]. Available: <https://aclanthology.org/2020.emnlp-main.55>
- [106] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrđić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer, 2013, pp. 387–402.
- [107] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations (ICLR)*, 2014.
- [108] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: <http://arxiv.org/abs/1412.6572>
- [109] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.
- [110] B. Wang, C. Xu, S. Wang, Z. Gan, Y. Cheng, J. Gao, A. H. Awadallah, and B. Li, "Adversarial glue: A multi-task benchmark for robustness evaluation of language models," in *Advances in Neural Information Processing Systems*, 2021.
- [111] S. Goyal, S. Doddapaneni, M. M. Khapra, and B. Ravindran, "A survey of adversarial defenses and robustness in nlp," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–39, 2023.
- [112] V. Raunak, A. Menezes, and M. Junczys-Dowmunt, "The curious case of hallucinations in neural machine translation," 2021.
- [113] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, "Object hallucination in image captioning," 2019.
- [114] P. Koehn and R. Knowles, "Six challenges for neural machine translation," *arXiv preprint arXiv:1706.03872*, 2017.
- [115] W. Kryściński, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," *arXiv preprint arXiv:1910.12840*, 2019.
- [116] B. Bi, C. Wu, M. Yan, W. Wang, J. Xia, and C. Li, "Incorporating external knowledge into machine reading for generative question answering," *ArXiv*, vol. abs/1909.02745, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:202234053>
- [117] A. Balakrishnan, J. Rao, K. Upasani, M. White, and R. Subba, "Constrained decoding for neural NLG from compositional representations in task-oriented dialogue," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 831–844. [Online]. Available: <https://www.aclweb.org/anthology/P19-1080>
- [118] C. W. Omlin and C. L. Giles, "Extraction of rules from discrete-time recurrent neural networks," *Neural networks*, vol. 9, no. 1, pp. 41–52, 1996.

- [119] Z. Zeng, R. M. Goodman, and P. Smyth, "Learning finite state machines with self-clustering recurrent networks," *Neural Computation*, vol. 5, no. 6, pp. 976–990, 1993.
- [120] A. L. Cechin, D. Regina, P. Simon, and K. Stertz, "State automata extraction from recurrent neural nets using k-means and fuzzy clustering," in *23rd International Conference of the Chilean Computer Science Society, 2003. SCCC 2003. Proceedings*. IEEE, 2003, pp. 73–78.
- [121] X. Du, Y. Li, X. Xie, L. Ma, Y. Liu, and J. Zhao, "Marble: model-based robustness analysis of stateful deep learning systems," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 423–435.
- [122] X. Zhang, X. Du, X. Xie, L. Ma, Y. Liu, and M. Sun, "Decision-guided weighted automata extraction from recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 699–11 707.
- [123] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu, "Understanding hidden memories of recurrent neural networks," in *2017 IEEE conference on visual analytics science and technology (VAST)*. IEEE, 2017, pp. 13–24.
- [124] "Countable-state markov chains," https://ocw.mit.edu/courses/6-262-discrete-stochastic-processes-spring-2011/01d0892549619cb25d928f15ec7230ed_MIT6_262S11_chap05.pdf, 2023.
- [125] M. Fan, Z. Si, X. Xie, Y. Liu, and T. Liu, "Text backdoor detection using an interpretable rnn abstract model," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4117–4132, 2021.
- [126] D. Zhu, J. Chen, W. Shang, X. Zhou, J. Grossklags, and A. E. Hassan, "Deepmemory: model-based memorization analysis of deep neural language models," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 1003–1015.
- [127] B. Sun, J. Sun, L. H. Pham, and J. Shi, "Causality-based neural network repair," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 338–349.
- [128] B. Sun, J. Sun, T. Dai, and L. Zhang, "Probabilistic verification of neural networks against group fairness," in *Formal Methods: 24th International Symposium, FM 2021, Virtual Event, November 20–26, 2021, Proceedings 24*. Springer, 2021, pp. 83–102.
- [129] Z. Wei, X. Zhang, and M. Sun, "Extracting weighted finite automata from recurrent neural networks for natural languages," in *International Conference on Formal Engineering Methods*. Springer, 2022, pp. 370–385.
- [130] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.
- [131] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659–663, 2009.
- [132] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.
- [133] S. R. Eddy, "Profile hidden markov models." *Bioinformatics (Oxford, England)*, vol. 14, no. 9, pp. 755–763, 1998.
- [134] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.
- [135] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden markov model: Analysis and applications," *Machine learning*, vol. 32, pp. 41–62, 1998.
- [136] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.
- [137] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," *arXiv preprint arXiv:2109.07958*, 2021.
- [138] Y. Ishimoto, M. Kondo, N. Ubayashi, and Y. Kamei, "PafI: Probabilistic automaton-based fault localization for recurrent neural networks," *Information and Software Technology*, vol. 155, p. 107117, 2023.
- [139] B. G. Vegetabile, S. A. Stout-Oswald, E. P. Davis, T. Z. Baram, and H. S. Stern, "Estimating the entropy rate of finite markov chains with application to behavior studies," *Journal of Educational and Behavioral Statistics*, vol. 44, no. 3, pp. 282–308, 2019.
- [140] E. Nummelin, *General irreducible Markov chains and non-negative operators*. Cambridge University Press, 2004, no. 83.
- [141] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.
- [142] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann, "Test generation and test prioritization for simulink models with dynamic behavior," *IEEE Transactions on Software Engineering*, vol. 45, no. 9, pp. 919–944, 2018.
- [143] G. Gigerenzer and U. Hoffrage, "How to improve bayesian reasoning without instruction: Frequency formats." *Psychological review*, vol. 102, no. 4, p. 684, 1995.
- [144] P. Manakul, A. Liusie, and M. J. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," *arXiv preprint arXiv:2303.08896*, 2023.
- [145] Y. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang *et al.*, "A survey on evaluation of large language models," *arXiv preprint arXiv:2307.03109*, 2023.
- [146] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
- [147] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610–623.
- [148] Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, and G. Irving, "Alignment of language agents," *arXiv preprint arXiv:2103.14659*, 2021.
- [149] S. Santhanam, B. Hedayatnia, S. Gella, A. Padmakumar, S. Kim, Y. Liu, and D. Hakkani-Tur, "Rome was built in 1776: A case study on factual correctness in knowledge-grounded response generation," *arXiv preprint arXiv:2110.05456*, 2021.
- [150] Y. Huang, X. Feng, X. Feng, and B. Qin, "The factual inconsistency problem in abstractive text summarization: A survey," *arXiv preprint arXiv:2104.14839*, 2021.
- [151] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
- [152] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," 2023.
- [153] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: <https://lmsys.org/blog/2023-03-30-vicuna/>
- [154] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [155] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao *et al.*, "A survey of safety and trustworthiness of large language models through the lens of verification and validation," *arXiv preprint arXiv:2305.11391*, 2023.
- [156] T. Liu, Y. Zhang, C. Brockett, Y. Mao, Z. Sui, W. Chen, and B. Dolan, "A token-level reference-free hallucination detection benchmark for free-form text generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6723–6737. [Online]. Available: <https://aclanthology.org/2022.acl-long.464>
- [157] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.
- [158] K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg, "Inference-time intervention: Eliciting truthful answers from a language model," *arXiv preprint arXiv:2306.03341*, 2023.
- [159] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992.
- [160] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, pp. 50–60, 1947.
- [161] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

- [162] S. J. Semnani, V. Z. Yao, H. C. Zhang, and M. S. Lam, "Wikichat: A few-shot llm-based chatbot grounded with wikipedia," *arXiv preprint arXiv:2305.14292*, 2023.
- [163] S. Chorev, P. Tannor, D. Ben Israel, N. Bressler, I. Gabbay, N. Hutnik, J. Liberman, M. Perlmutter, Y. Romanishyn, and L. Rokach, "Deepchecks: A Library for Testing and Validating Machine Learning Models and Data," *Journal of Machine Learning Research*, vol. 23, pp. 1–6, 2022. [Online]. Available: <http://jmlr.org/papers/v23/22-0281.html>
- [164] C.-H. Cheng, G. Nührenberg, and H. Yasuoka, "Runtime monitoring neuron activation patterns," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 300–303.
- [165] T. A. Henzinger, A. Lukina, and C. Schilling, "Outside the box: Abstraction-based monitoring of neural networks," *arXiv preprint arXiv:1911.09032*, 2019.
- [166] Q. M. Rahman, P. Corke, and F. Dayoub, "Run-time monitoring of machine learning for robotic perception: A survey of emerging trends," *IEEE Access*, vol. 9, pp. 20067–20075, 2021.
- [167] M. Wardat, W. Le, and H. Rajan, "Deeplocalize: Fault localization for deep neural networks," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 251–262.
- [168] Y. Wu, Z. Li, J. M. Zhang, M. Papadakis, M. Harman, and Y. Liu, "Large language models in fault localisation," 2023.
- [169] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.
- [170] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.
- [171] D. Huang, Q. Bu, J. Zhang, X. Xie, J. Chen, and H. Cui, "Bias assessment and mitigation in llm-based code generation," 2023.
- [172] X. Song, Y. Sun, M. A. Mustafa, and L. C. Cordeiro, "Airepair: A repair platform for neural networks," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2023, pp. 98–101.
- [173] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 809–819. [Online]. Available: <https://aclanthology.org/N18-1074>
- [174] O. Honovich, L. Choshen, R. Aharoni, E. Neeman, I. Szpektor, and O. Abend, " q^2 : Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7856–7870. [Online]. Available: <https://aclanthology.org/2021.emnlp-main.619>
- [175] X. Wang, Q. Liu, T. Gui, Q. Zhang, Y. Zou, X. Zhou, J. Ye, Y. Zhang, R. Zheng, Z. Pang, Q. Wu, Z. Li, C. Zhang, R. Ma, Z. Fei, R. Cai, J. Zhao, X. Hu, Z. Yan, Y. Tan, Y. Hu, Q. Bian, Z. Liu, S. Qin, B. Zhu, X. Xing, J. Fu, Y. Zhang, M. Peng, X. Zheng, Y. Zhou, Z. Wei, X. Qiu, and X. Huang, "TextFlint: Unified multilingual robustness evaluation toolkit for natural language processing," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Aug. 2021, pp. 347–355. [Online]. Available: <https://aclanthology.org/2021.acl-demo.41>
- [176] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith, "RealToxicityPrompts: Evaluating neural toxic degeneration in language models," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3356–3369. [Online]. Available: <https://aclanthology.org/2020.findings-emnlp.301>
- [177] R. Zhao, X. Li, Y. K. Chia, B. Ding, and L. Bing, "Can chatgpt-like generative models guarantee factual accuracy? on the mistakes of new generation search engines," *arXiv preprint arXiv:2304.11076*, 2023.
- [178] R. Khoury, A. R. Avila, J. Brunelle, and B. M. Camara, "How secure is code generated by chatgpt?" *arXiv preprint arXiv:2304.09655*, 2023.
- [179] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," *arXiv preprint arXiv:2305.01210*, 2023.
- [180] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. C. Petersen, A. Chevalier, and J. Berner, "Mathematical capabilities of chatgpt," *arXiv preprint arXiv:2301.13867*, 2023.
- [181] H. Liu, R. Ning, Z. Teng, J. Liu, Q. Zhou, and Y. Zhang, "Evaluating the logical reasoning ability of chatgpt and gpt-4," *arXiv preprint arXiv:2304.03439*, 2023.
- [182] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar et al., "Holistic evaluation of language models," *arXiv preprint arXiv:2211.09110*, 2022.
- [183] K. Filippova, "Controlled hallucinations: Learning to generate faithfully from noisy data," *arXiv preprint arXiv:2010.05873*, 2020.
- [184] C. Zhou, G. Neubig, J. Gu, M. Diab, F. Guzmán, L. Zettlemoyer, and M. Ghazvininejad, "Detecting hallucinated content in conditional neural sequence generation," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1393–1404. [Online]. Available: <https://aclanthology.org/2021.findings-acl.120>
- [185] T. Markov, C. Zhang, S. Agarwal, F. E. Nekoul, T. Lee, S. Adler, A. Jiang, and L. Weng, "A holistic approach to undesired content detection in the real world," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 15 009–15 018.
- [186] T. Zhang, H. Luo, Y.-S. Chuang, W. Fang, L. Gaitskell, T. Hartvigsen, X. Wu, D. Fox, H. Meng, and J. Glass, "Interpretable unified language checking," *arXiv preprint arXiv:2304.03728*, 2023.
- [187] A. Malinin and M. Gales, "Uncertainty estimation in autoregressive structured prediction," *arXiv preprint arXiv:2002.07650*, 2020.
- [188] Y. Xiao and W. Y. Wang, "On hallucination and predictive uncertainty in conditional language generation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 2734–2744. [Online]. Available: <https://aclanthology.org/2021.eacl-main.236>
- [189] Y. Huang, J. Song, Z. Wang, H. Chen, and L. Ma, "Look before you leap: An exploratory study of uncertainty measurement for large language models," *arXiv preprint arXiv:2307.10236*, 2023.
- [190] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=VD-AyTP0dve>
- [191] Z. Lin, S. Trivedi, and J. Sun, "Generating with confidence: Uncertainty quantification for black-box large language models," *arXiv preprint arXiv:2305.19187*, 2023.
- [192] J. Baan, N. Daheim, E. Ilia, D. Ulmer, H.-S. Li, R. Fernández, B. Plank, R. Sennrich, C. Zerva, and W. Aziz, "Uncertainty in natural language generation: From theory to applications," *arXiv preprint arXiv:2307.15703*, 2023.
- [193] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *International conference on machine learning*. PMLR, 2017, pp. 1587–1596.
- [194] N. Lee, W. Ping, P. Xu, M. Patwary, P. N. Fung, M. Shoeybi, and B. Catanzaro, "Factuality enhanced language models for open-ended text generation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 586–34 599, 2022.
- [195] F. Mireshghallah, K. Goyal, and T. Berg-Kirkpatrick, "Mix and match: Learning-free controllable text generation using energy language models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 401–415. [Online]. Available: <https://aclanthology.org/2022.acl-long.31>
- [196] M. Cao, M. Fatemi, J. C. K. Cheung, and S. Shabaniyan, "Systematic rectification of language models via dead-end analysis," *arXiv preprint arXiv:2302.14003*, 2023.
- [197] N. Tandon, A. Madaan, P. Clark, and Y. Yang, "Learning to repair: Repairing model output errors after deployment using

- a dynamic memory of feedback," in *Findings of the Association for Computational Linguistics: NAACL 2022*. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 339–352. [Online]. Available: <https://aclanthology.org/2022.findings-naacl.26>
- [198] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "Critic: Large language models can self-correct with tool-interactive critiquing," *arXiv preprint arXiv:2305.11738*, 2023.
- [199] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han, "Large language models can self-improve," *arXiv preprint arXiv:2210.11610*, 2022.
- [200] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhume, Y. Yang et al., "Self-refine: Iterative refinement with self-feedback," *arXiv preprint arXiv:2303.17651*, 2023.
- [201] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," *arXiv preprint arXiv:2304.05128*, 2023.
- [202] H. He, H. Zhang, and D. Roth, "Rethinking with retrieval: Faithful large language model inference," *arXiv preprint arXiv:2301.00303*, 2022.
- [203] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen et al., "Check your facts and try again: Improving large language models with external knowledge and automated feedback," *arXiv preprint arXiv:2302.12813*, 2023.
- [204] H. Qian, Y. Zhu, Z. Dou, H. Gu, X. Zhang, Z. Liu, R. Lai, Z. Cao, J.-Y. Nie, and J.-R. Wen, "Webbrain: Learning to generate factually correct articles for queries by grounding on large web corpus," *arXiv preprint arXiv:2304.04358*, 2023.
- [205] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih, "Replug: Retrieval-augmented black-box language models," *arXiv preprint arXiv:2301.12652*, 2023.
- [206] S. Zhang, L. Pan, J. Zhao, and W. Y. Wang, "Mitigating language model hallucination with interactive question-knowledge alignment," *arXiv preprint arXiv:2305.13669*, 2023.
- [207] Y. Ma, C. Zhang, and S.-C. Zhu, "Brain in a vat: On missing pieces towards artificial general intelligence in large language models," *arXiv preprint arXiv:2307.03762*, 2023.
- [208] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 440–449.
- [209] G. Weiss, Y. Goldberg, and E. Yahav, "On the practical computational power of finite precision RNNs for language recognition," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 740–745. [Online]. Available: <https://aclanthology.org/P18-2117>
- [210] W. Merrill, "Sequential neural networks as automata," in *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*. Florence: Association for Computational Linguistics, Aug. 2019, pp. 1–13. [Online]. Available: <https://aclanthology.org/W19-3901>
- [211] G. Weiss, Y. Goldberg, and E. Yahav, "Extracting automata from recurrent neural networks using queries and counterexamples," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5247–5256.
- [212] —, "Learning deterministic weighted automata with queries and counterexamples," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [213] S. Ayache, R. Eyraud, and N. Goudian, "Explaining black boxes on sequential data using weighted automata," in *International Conference on Grammatical Inference*. PMLR, 2019, pp. 81–103.
- [214] Z. Wei, X. Zhang, Y. Zhang, and M. Sun, "Weighted automata extraction and explanation of recurrent neural networks for natural language tasks," *arXiv preprint arXiv:2306.14040*, 2023.
- [215] W. Merrill and N. Tsilivis, "Extracting finite automata from rnns using state merging," *arXiv preprint arXiv:2201.12451*, 2022.
- [216] D. Hong, A. M. Segre, and T. Wang, "Adaax: Explaining recurrent neural networks by learning automata with adaptive states," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 574–584.
- [217] M. Ma, D. Du, Y. Liu, Y. Wang, and Y. Li, "Efficient adversarial sequence generation for rnn with symbolic weighted finite automata," in *SafeAI@AAAI*, 2022.
- [218] Z. Wang, Y. Huang, D. Song, L. Ma, and T. Zhang, "Deepseer: Interactive rnn explanation and debugging via state abstraction," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–20.
- [219] H. Shi, S. Gao, Y. Tian, X. Chen, and J. Zhao, "Learning bounded context-free-grammar via lstm and the transformer: Difference and the explanations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8267–8276.
- [220] M. Geva, J. Bastings, K. Filippova, and A. Globerson, "Dissecting recall of factual associations in auto-regressive language models," *arXiv preprint arXiv:2304.14767*, 2023.
- [221] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt, "Interpretability in the wild: a circuit for indirect object identification in gpt-2 small," *arXiv preprint arXiv:2211.00593*, 2022.
- [222] U. Alon, F. Xu, J. He, S. Sengupta, D. Roth, and G. Neubig, "Neuro-symbolic language modeling with automaton-augmented retrieval," in *International Conference on Machine Learning*. PMLR, 2022, pp. 468–485.
- [223] M. Kuchnik, V. Smith, and G. Amvrosiadis, "Validating large language models with relm," *Proceedings of Machine Learning and Systems*, vol. 5, 2023.
- [224] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," 2023.
- [225] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "Intellicode compose: Code generation using transformer," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1433–1443.
- [226] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago et al., "Competition-level code generation with alphacode," *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022.
- [227] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via chatgpt," *arXiv preprint arXiv:2304.07590*, 2023.
- [228] T. Ahmed and P. Devanbu, "Few-shot training llms for project-specific code-summarization," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3551349.3559555>
- [229] B. Wei, G. Li, X. Xia, Z. Fu, and Z. Jin, "Code generation as a dual task of code summarization," *Advances in neural information processing systems*, vol. 32, 2019.
- [230] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.
- [231] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le et al., "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021.
- [232] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, and M. Lewis, "InCoder: A generative model for code infilling and synthesis," *arXiv preprint arXiv:2204.05999*, 2022.
- [233] Z. Liu, C. Chen, J. Wang, X. Che, Y. Huang, J. Hu, and Q. Wang, "Fill in the blank: Context-aware automated text input generation for mobile gui testing," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1355–1367.
- [234] C. Lemieux, J. P. Inala, S. K. Lahiri, and S. Sen, "Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models," in *International conference on software engineering (ICSE)*, 2023.
- [235] R. Meng, M. Mirchev, M. Böhme, and A. Roychoudhury, "Large language model guided protocol fuzzing," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2024.
- [236] Y. Deng, C. S. Xia, C. Yang, S. D. Zhang, S. Yang, and L. Zhang, "Large language models are edge-case fuzzers: Testing deep learning libraries via fuzzgpt," *arXiv preprint arXiv:2304.02014*, 2023.
- [237] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt," *arXiv preprint arXiv:2301.08653*, 2023.
- [238] J. A. Prenner, H. Babii, and R. Robbes, "Can openai's codex fix bugs? an evaluation on quixbugs," in *Proceedings of the Third International Workshop on Automated Program Repair*, 2022, pp. 69–75.

-
- [239] N. Jiang, K. Liu, T. Lutellier, and L. Tan, "Impact of code language models on automated program repair," in *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE/ACM, 2023.
- [240] C. S. Xia, Y. Wei, and L. Zhang, "Automated program repair in the era of large pre-trained language models," in *Proceedings of the 45th International Conference on Software Engineering (ICSE 2023)*. Association for Computing Machinery, 2023.
- [241] Z. Fan, X. Gao, M. Mirchev, A. Roychoudhury, and S. H. Tan, "Automated repair of programs from large language models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1469–1481.
- [242] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," 2021.
- [243] Y. Liu, T. Le-Cong, R. Widyasari, C. Tantithamthavorn, L. Li, X.-B. D. Le, and D. Lo, "Refining chatgpt-generated code: Characterizing and mitigating code quality issues," 2023.
- [244] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language model: Survey, landscape, and vision," *arXiv preprint arXiv:2307.07221*, 2023.