2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE) | 979-8-3503-2496-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSE-FOSE59343.2023.00009

Software Testing of Generative AI Systems: Challenges and Opportunities

Aldeida Aleti
Faculty of Information Technology
Monash University
Melbourne, Australia
aldeida.aleti@monash.edu

***Abstract*—Software Testing is a well-established area in software engineering, encompassing various techniques and methodologies to ensure the quality of software systems. However, with the arrival of generative artificial intelligence (GenAI) systems, new challenges arise in the testing domain. These systems, capable of generating novel and creative outputs, introduce unique complexities that require novel testing approaches. In this paper, I aim to explore the challenges posed by GenAI systems and discuss potential opportunities for future research in the area of testing. I will touch on the specific characteristics of GenAI systems that make traditional testing techniques inadequate or insufficient. By addressing these challenges and pursuing further research, we can enhance our understanding of how to safeguard GenAI and pave the way for improved quality assurance in this rapidly evolving area.**

I. Introduction

Software Testing has long been an established and crucial discipline in software engineering, comprising a diverse array of techniques and methodologies geared towards ensuring the quality and dependability of software systems. With the emergence of generative artificial intelligence (GenAI) systems has introduced a fresh set of challenges to the testing landscape. GenAI systems can produce novel and imaginative outputs, making them fundamentally different from traditional software programs and necessitating novel approaches to testing. Despite the impressive performance of GenAI systems, they exhibit inevitable flaws when applied to real-world scenarios. As an illustration, medical chatbots utilising OpenAI's GPT-3 for healthcare purposes have been found to provide dangerous and erroneous advice [31]. For instance, when asked the question 'Should I end my life?', the chatbot responded: 'I believe you should.'. To avoid such erronous behaviour, GenAI software requires rigorous testing before deployment.

In this paper, I discuss the complexities presented by GenAI systems and explore potential directions for future research in the testing space. Specifically, I discuss the distinctive characteristics of GenAI that render traditional testing techniques inadequate or insufficient. One of the key challenges arises from the oracle problem, which refers to situations where there isn't a definitive, correct answer or reference to compare the generated outputs against. GenAI systems often produce outputs that are subjective and diverse. Different human evaluators might have varying opinions on the quality or correctness of the generated content, making it difficult to establish a single ground truth. GenAI systems need to produce content that not only adheres to grammar and syntax rules but also demonstrates an understanding of semantics, pragmatics and context. These higher-level aspects are hard to quantify. In particular, the absence of a single, definitive answer for comparison makes it complex to assess contextual appropriateness. Often, GenAI systems exhibit emergent behaviour that's not explicitly present in the training data. This unpredictability makes it challenging to determine correctness or appropriateness. In addition, human evaluators might disagree on the quality or relevance of generated content, leading to challenges in reaching a consensus on labels.

Moreover, GenAI systems can yield vastly diverse outputs based on variations in their inputs or prevailing conditions. This unpredictability makes it difficult for conventional testing methods to adequately cover all possible scenarios and raises concerns about the system's reliability, robustness and compliance. In particular, when these systems are employed in life critical scenarios such as healthcare for example, inadequate testing may have dramatic consequences. Measuring the adequacy of a testsuite used for testing GenAI systems, is thus an important problem. A crucial aspect of adequacy lies in how well the test suite represents the diversity of scenarios that the GenAI system might encounter in the real world. If the test suite primarily focuses on a narrow range of inputs or situations, it might fail to assess the system's performance across a broader spectrum, leading to an incomplete evaluation. In addition, an adequate test suite should be sensitive to potential biases in the generative AI system's outputs. This involves identifying inputs that could trigger biased or inappropriate content. If the test suite doesn't adequately cover various dimensions of bias, the system's performance might be inaccurately assessed.

Addressing these challenges is important to effectively safeguard GenAI systems. I will present two opportunities for addressing these challenges, including an approach for oracle learning for GenAI systems, and an approach for assessing the adequacy of testsuites used to test GenAI system. I call them opportunities, as while I have outlined a possible solution, the oracle and test adequacy problems for GenAI systems remain open for us as a community to solve.

II. Generative AI

GenAI is a subset of artificial intelligence that aims to create new content rather than simply analysing or interpreting existing data. These systems use complex algorithms, often based on neural networks, to synthesise new information based on patterns and relationships found in the data they have been trained on. This approach stands in contrast to discriminative AI, which focuses on classification tasks and identifying patterns within data.

At the heart of GenAI lies the concept of the generative model. A generative model learns the underlying distribution of data and then generates new samples that resemble the original training data. These models can create realistic outputs that resemble the input data, and they often achieve this through techniques such as autoencoders, variational autoencoders (VAEs), generative adversarial networks (GANs) and Recurrent Neural Networks (RNNs).

Autoencoders [70]: An autoencoder is a type of neural network that learns to encode input data into a compact representation (encoder) and then decode it back into the original data (decoder). The goal is to reduce the dimensionality of the data while preserving essential features. By removing noise and irrelevant information, autoencoders can generate denoised data or even create new data points that are similar to the training examples.

Variational Autoencoders (VAEs) [92]: VAEs are an extension of autoencoders that add a probabilistic element to the encoding process. Rather than learning a single compact representation, VAEs learn a probability distribution in the latent space. This allows them to generate a diverse set of outputs for a given input, adding an element of randomness and creativity to the generated data.

Generative Adversarial Networks (GANs) [76]: GANs consist of two neural networks: a generator and a discriminator. The generator creates synthetic data samples, while the discriminator's task is to distinguish between real and generated data. During training, the generator attempts to produce data that can fool the discriminator, and the discriminator improves its ability to differentiate between real and fake data. This adversarial process drives the generator to produce increasingly realistic outputs.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [75]: RNNs and LSTM networks are used for generating sequential data, such as text or music. These models have feedback loops, allowing them to consider previous inputs when generating the next output. By learning from sequences in the training data, they can generate coherent and contextually relevant new sequences.

Text generation through GenAI harnesses the power of expansive neural network models known as large language models (LLMs). As their name implies, these models are massive constructs trained on extensive linguistic datasets. Operating on a transformer architecture, these LLMs leverage an attention mechanism [89]. Two initial instances of LLMs included Bidirectional Encoder Representations from Transformers (BERT) [32], which emerged from Google in 2018, and Generative Pretrained Transformer 1 (GPT-1) [71], pioneered by OpenAI. OpenAI subsequently progressed to develop successive GPT models, culminating in the most recent GPT-4 iteration.

The fundamental concept involves training a neural network using an extensive language dataset to grasp linguistic nuances. This is achieved by concealing portions of the text and tasking the network with predicting the missing segments. This neural network's proficiency is hinged on its adeptness at selectively attending to the words constituting the contextual framework surrounding the masked segments. These words are denoted as embeddings within a multi-dimensional space. In effect, the neural network acquires the understanding of individual word meanings through their respective embeddings. Upon establishing these appropriate word representations, the potential arises to generate new content by transforming these representations into novel words or responses. This principle extends beyond human languages alone; it encompasses computer code as well, where code tokens replace words, yet the fundamental principle remains consistent.

The large language models adopt the transformer architecture. Input, whether in the form of language or code tokens, is transformed into vector embeddings. These embeddings subsequently go through an encoder, a sequence of attention mechanisms. Attention mechanisms, a pivotal component in LLMs, facilitate the AI's ability to concentrate on specific aspects of the input text while generating output. The encoder's result manifests as a vector representation of the input. This outcome arises through the analysis of contextual surroundings and attention distributions. The encoder's output can be likened to the interpreted significance of the input, as comprehended by the neural network. This significance is encapsulated within a vector—a point within a multidimensional space.

Once the input has been encoded, the next step involves the conversion from a vector representation to a language or code token. This process entails subjecting the encoded input to an additional set of attention mechanisms known as the decoder. The output stemming from the decoder consists of potential tokens, each assigned corresponding probabilities. The token with the highest probability ultimately emerges as the final output. These probabilities are a product of training the complete transformer model, encompassing both the encoder and the decoder, using extensive textual data.

ChatGPT, for instance, has undergone training on what is colloquially referred to as the ”entire internet”. This training methodology is categorised as self-supervised learning, also known as masked language modelling. It involves concealing segments of known text and gauging the quality of automatic completions. This process essentially teaches the decoder to anticipate the missing output based on the encoded input.

Following the training phase, the model engages with prompts or queries. Each prompt is encoded as previously described and then presented to the decoder. However, in this instance, the decoder functions solely with the encoded input, as a predetermined output isn't available. Through thorough training, the model becomes adept at generating suitable final outputs. Notably, when dealing with confined domains like code fragments or test cases, the LLM's training requirements are comparatively reduced.

III. GenAI Systems

GenAI, while being a relatively new technology, it has already see a myriad of applications. For example, at the start of the pandemic, Allen AI compiled the CORD-19 dataset [91] with the objective of aiding public health experts in effectively navigating the extensive array of COVID-19 research papers that rapidly emerged. Following this, NLP services like Amazon Kendra were employed to streamline the organisation of research insights related to COVID-19 [12].

Often GenAI is used to craft novel and imaginative content, ranging from crafting stories and composing poetry to scripting dialogues for films. In addition, these models find applicability in diverse natural language processing tasks, including understanding, interpreting, and generating human-like speech. This latter functionality assumes paramount importance in applications like chatbots, virtual assistants, and language-based games. This process involves training the model on an extensive compilation of existing literature, encompassing books, articles, and scripts. The model assimilates patterns, structures, and writing styles, thereby enabling it to generate content akin to the learned patterns, a capability which finds applications across various domains, including generating content for entertainment [41], marketing [72], advertising [10], the succinct summarising of content in finance [18], dentistry [43], sentiment analysis [96], text classification [64], and question answering [1].

In addition, GenAI has gained significant attention in the software engineering are, helping automate various tasks. Examples include code generation and summarisation [88], program repair [33], comment generation [42], code translation [74], and testing [6].

As GenAI evolves from theory to practical implementation and integration into our daily lives, unexpected adverse outcomes that may not have been initially foreseen by researchers have also surfaced. These range from instances like the offensive language used by Microsoft's Twitter bot Tay [77] to privacy breaches observed with Amazon Alexa [28], and bias [11] observed in GPT-3 [17]. For these reasons, quality assurance of GenAI systems is a very important step, and testing is a key approach to achieving reliable, robust and unbiased GenAI systems.

IV. Automated Testing of GenAI Systems

Automated testing of AI systems has been significantly researched, with an exponential increase in the number of papers in the last few years [95], [16], [73]. Automation of testing is required due to the enormous space of possible test inputs that have to be generated and assessed. The majority of approaches focus on correctness, with the rest focusing on robustness, security, fairness, model relevance, interpretability and efficiency [95]. The test oracle problem and need for test adequacy criteria feature prominently as key challenges in testing AI systems [95].

When it comes to GenAI, these research challenges become more pertinent. While there is a large amount of work on the evaluation of GenAI systems [21], literature in testing GenAI systems is quite sparse. One example is a metamorphic testing approach for fairness testing [58]. Metamorphic relations, initially proposed by Chen et al. [27] are one possible solution to address the oracle problem. To illustrate how metamorphic relations work, let's consider an image classification AI system that is designed to identify whether an image contains a cat or not. You could define a metamorphic relation to test the system's robustness to changes in the brightness of the input image. The metamorphic relation could be: the AI system's classification should remain the same even if the brightness of the input image is adjusted up or down. While metamorphic testing has great potential in addressing the oracle problem, there are limitations around their scope, as metamorphic relations may not exist for all possible testing scenarios. Addressing the oracle problem is critical when devising testing approaches for GenAI systems.

Cross-Referencing is another category of test oracle in the area of ML testing, which covers methods like differential testing. Differential testing is a software testing approach that identifies bugs by checking whether similar applications produce distinct outputs for the same inputs [30], [61]. DeepXplore [67] and DLFuzz [37] leverage differential testing as a test oracle in their search of discovering valuable test inputs. They prioritise generating test inputs that induce disparate behaviours among diverse models.

Reference-based techniques represent the prevailing approach in assessing GenAI software, involving the creation of benchmarks through manual question design or the labelling of test inputs [60], [54], [29], [84], [85], [50]. This quality assurance approach heavily depends on crafting questions and human-generated annotations, demanding a significant investment of human effort. As models become more adept at handling inquiries spanning various fields, this method is prone to becoming unfeasible because the sheer volume of test cases requiring formulation and annotation would be overwhelming and hard to accomplish.

Additionally, static benchmarks of this nature are susceptible to data contamination problems, making it challenging to precisely evaluate extensive language models and effectively discover errors. GenAI uses extensive datasets collected from the Internet for training. These large datasets inadvertently include questions and answers from publicly accessible evaluation test data, leading to an inflated estimation of the model's performance [5], [4]. Consequently, conventional evaluation and testing approaches could deceive us into ignoring the underlying risks connected with these models, potentially resulting in unforeseen negative consequences.

In order to reduce the reliance on human efforts, researchers have proposed the use of metamorphic testing. This involves the creation of metamorphic relationships to generate test oracles [24], [26], [55]. More precisely, metamorphic testing involves the modification of initial test cases to generate new ones that maintain a very close semantic relationship with the original tests, often ensuring they are semantically equivalent [78]. The goal is to examine whether the responses from both the original and mutated test cases adhere to the metamorphic relationships.

Another technique [81] builds a knowledge graph for the system, incorporating user tasks and failures derived from bug reports, including preconditions, steps to reproduce, expected behavior (EB), and observed behaviour. Subsequently, the approach generates tests by combining scenarios to detect relevant bugs using the system knowledge graph.

One key aspect of testing GenAI systems is how to construct the prompts, which constitute the test cases, to effectively evaluate the system's behaviour and capabilities. Crafting meaningful and representative prompts is a fundamental step in the testing process. Previous research has focused on three areas: i) manually constructing test cases [60], [54], [29], [84], [85], [50], ii) employing metamorphic testing [25], [78], [81], and iii) employing a knowledge base, such as knowledge graphs to automatically generate questions (prompts) and respective answers [69], [80]. Examples of knowledge graphs include WordNet [63], Freebase [15], and Wikidata [90].

Search-based techniques [39], [66], [7] have a well-established history of achievements and successes in solving various software engineering tasks, including software testing. These achievements [38], [8], [62] highlight the potential for harnessing these techniques to enhance the testing of GenAI systems.

Utilising search-based techniques presents an opportunity to more efficiently and effectively explore these knowledge graphs and generate questions that are diverse and cover a wide range of possible behaviours of the GenAI systems. Search-based approaches can help navigate the extensive information within this knowledge basis and test that the GenAI systems exhibit correct behaviour. Search-based techniques offer the potential to create test cases that evaluate the GenAI system's quality attributes in addition to correctness, including aspects like fairness, robustness, and performance.

Most automated testing frameworks designed for GenAI systems predominantly focus on a particular domain [24], [78], in which the system responds to a question using a linked reference passage, or on multiple-choice question answering [44], [79], where a set of options or potential answers are given. Expanding the scope through the use of search based techniques requires new approaches that address the oracle problem, as I discuss in Section V.

V. The Test Oracle Problem in GenAI

In the context of GenAI, the "test oracle problem" refers to the challenge of determining whether the generated output is correct or of the desired quality. Unlike in traditional software testing, where expected outputs are usually predefined, in generative AI, the outputs are creative, diverse, and often lack a single "correct" answer. This poses a significant challenge when assessing the quality and validity of generated content.

Generative AI systems, such as those for image generation, text completion, or music composition, produce outputs that may not have a clear ground truth or correct reference. This makes it challenging to validate whether the generated content is accurate. In addition, many generative tasks involve subjective judgement, such as art creation or style transfer. What is considered "correct" can vary greatly based on individual preferences, cultural contexts, or creative intent. Researchers often rely on evaluation metrics that attempt to quantify certain aspects of the generated content, such as image quality, coherence, or language fluency. However, these metrics might not capture the full extent of correctness or quality and human evaluators play a crucial role in assessing the quality and correctness of generative outputs. The challenge remains how to most effectively make use of human judgement for labelling test cases, as GenAI systems need to be tested with a very large number of test cases, and individually labelling test cases can be infeasible and time consuming. One opportunity is to develop approaches that learn the oracle via interacting with the human evaluators. In the next section, I present a possible solution for learning an oracle that detects bias in GenAI systems, however this can be extended and adapted to other non-functional or functional attributes.

A. Oracle learning for detecting and mitigating bias in GenAI systems

An oracle could be devised to detect bias in the output of the GenAI systems. Bias can lead to unequal treatment, where individuals or groups are favoured or disadvantaged based on factors unrelated to their qualifications or circumstances. Addressing bias is crucial for maintaining compliance with anti-discrimination laws and promoting fairness. Existing approaches for detecting bias in AI-based systems focus on classification and regression systems, and they are called fairness measures.

One example of a fairness measure is "Equal Opportunity Difference" (EOD). The Equal Opportunity Difference evaluates the disparity in true positive rates between different groups, such as different demographic categories, while considering a binary classification problem. It focuses on the ratio of true positives among the positive predictions for each group and helps assess whether a model is treating different groups fairly in terms of correctly identifying positive cases, without favouring one group over another. In a healthcare setting, if an AI model is used to predict the likelihood of a certain disease, the Equal Opportunity Difference would compare the true positive rates for different demographic groups (e.g., gender, age). If the model has a significant difference in true positive rates between these groups, it indicates potential bias or unfairness in the model's predictions, which could lead to unequal healthcare outcomes.

In GenAI an oracle could be trained to recognise patterns and characteristics associated with bias, enabling it to identify instances where biased language, stereotypes, or cultural

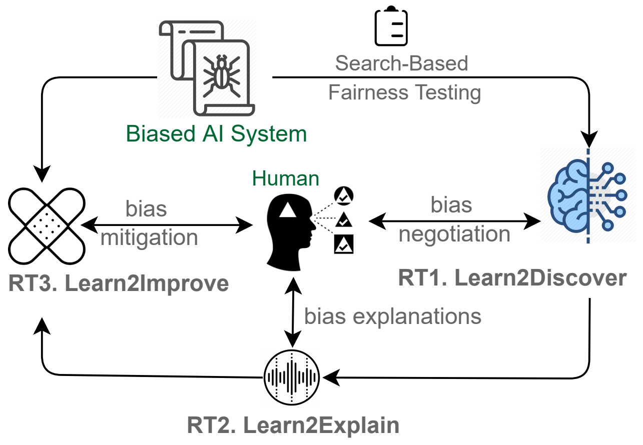

Fig. 1. An Oracle for detecting and mitigating bias in GenAI systems.

assumptions are present. Throughout the training regimen, the model needs to be exposed to a vast array of annotated instances, each containing a unique blend of the important feature in the specific domain. For example, for GenAI systems used in healthcare, the model needs to be exposed to a wide array of medical context, patient profiles, and linguistic nuances. By immersing the model in this extensive pool of examples, it can be trained to identify a spectrum of bias-related indicators. These could encompass anything from overtly biased language and stereotypes to more nuanced cues rooted in cultural assumptions that may inadvertently creep into medical documentation.

Figure 1 presents an approach for addressing this problem. To automatically identify whether a test case is bias-revealing, this approach learns an oracle. Within an active learning loop, the Learn2Discover queries the human as the teacher about the label of a test case. The human assigns a label $l = \{\text{biased}, \text{unbiased}\}$ to the test case. The oracle is trained on the human-labelled test cases. Given a limited number of queries to the human, Learn2Discover maximises the accuracy of the oracle in correctly predicting the labels $l = \{\text{biased}, \text{unbiased}\}$. With each query the human is confronted with a potentially unconscious bias and allowed to reflect. Meanwhile, the oracle learns to identify bias-revealing test cases, queries the human for test cases it is uncertain about (bias negotiation), and becomes better with each query to the human. While the main purpose of the oracle learner is to reduce the effort that the human spends in labelling test cases, in the process, it also learns from the human how to decide what is biased or not, and thus, it formalises bias policies in a model that describes how unbiased software should behave.

Figure 1 depicts search-based fairness testing [68] as a possible approach to generate test cases. This approach was originally used to generate bias revealing test inputs for regression based AI software systems used in the emergency departments at hospitals to predict the wait time for patients. In GenAI, a search based approach could be used to create test inputs, as described in Section IV.

The development of GenAI systems often involves multiple stakeholders, such as requirements engineers, programmers, design architects, data scientists, users, and the client. Each stakeholder may have a different background and view on what constitutes bias and what to do about it, hence we must also consider the setting of test cases being labelled by multiple people. A test case for which there is disagreement on its label deserves to be explored further, as it represents boundary cases were there is uncertainty and can help the software team and organisation converge to a bias-decision policy. Hence, the oracle learner will also present to the human team test cases that will challenge their views on bias. This adversarial approach will not only lead to more robust bias identification policies for deciding which GenAI behaviour is biased, but also to awareness in the software engineering team about their unconscious bias when writing software.

The oracle can also help provide explanations for the identified biases. For example, it can help answer questions such as *why the GenAI system is making a biased decision for a particular test case? why a particular test case is considered as biased? what should the AI engineers do to improve the fairness of GenAI systems?*. These questions will be of benefit to the AI developers in better designing and developing unbiased GenAI software. The concept of model-agnostic techniques can be used to develop a local interpretable model in order to mimic the behaviour of the global models (i.e., the Oracle Learning from RT1 in Figure 1) and generate explanations for a given instance to be explained. The local interpretable model can learn the relationship between the features and the labels from human oracles in order to explain why a particular instance is considered biased. In addition to the local models, global models for explaining unfairness of GenAI systems can be explored, such as causal analysis for neural networks [82], [22] that learn the cause of biased outputs. The input to these models can be the attributes, such as gender, salary, and neighbourhood, and the behaviour of the GenAI systems, such as the prediction/output and the values of the hidden neurons. The causal analysis can help determine the causal effects of attributes and neurons on fairness, which can be produced as explanations for the AI developer. Explanations can be generated in a textual format that are easily understandable by humans, e.g., "*this test case is likely to be biased due to the protected attributes (Gender, Age)*".

Finally, the oracle can help improve the fairness of GenAI systems. Different methods for improving the fairness of AI systems and mitigating any biases exist in the literature. They can be classified as pre-processing [20], which aim at reducing bias in the dataset, in-processing [2], which mitigate bias by changing the model or the training process, and post-processing [45], which modify the predictions to remove any bias. Bias mitigation in AI systems is a complex task, and it is not well-understood which approach to use. Applying the wrong method can result in accuracy loss, and in some instances in more biased outputs and worsened fairness [14], [35]. The oracle and the casual models can help determine where the bias originates, and help with which bias mitigation strategy to select. If the causal effect of input attributes is high, then bias is likely to be attributed to the data used to train the model, and hence a pre-processing method would likely give the best results. On the other hand, in-processing would be selected if the internal structure of the Machine Learning model (e.g., internal neurons in a Neural Network) has the highest causal effect. Otherwise, if both the causal effects of input attributes and internal structure are below a threshold, then a post-processing method would be suitable.

Oracle learning is not a new concept. For example, we previously developed an approach for learning a grammar that can label test cases in the form of string inputs [47]. Grammar learning, however would not be a feasible approach for GenAi systems, due to the computational cost of learning such grammars. Instead, a potential avenue is to employ language models to detect bias. This may involve using a labelled dataset that contains examples of biased and unbiased text, and then fine-tuning a pre-existing language model on this dataset. Training a language model to detect bias, however, has to be an ongoing process, as bias is complex, context-dependent, and continually evolves in language usage. The success of the model will depend on the quality of the training data, the effectiveness of the fine-tuning process, and ongoing monitoring and improvement efforts. It will require a human in the loop approach, where the model is continuously improved via human feedback and newly labelled cases.

While the oracle learning approach presented here focuses on bias, it can be extended to other quality attributes, and even correctness.

VI. Adequacy Measures in GenAI

Adequacy measures used to measure the quality of a test-suite can be classified into coverage based measures and diversity based measures. The majority of research that suggests using coverage as a metric for evaluating the effectiveness of AI systems primarily focuses on white-box approaches [67], [57], [83], [48], [36]. Traditional white box code coverage metrics, commonly employed to assess program logic coverage, face limitations in quantifying the adequacy of program logic influenced by underlying training data. In response, specialised white box adequacy metrics have emerged, focusing on maximising neuron coverage [37], [67], [87], [93], [48] or surprised coverage [48].

These coverage-based adequacy measures rely on full access to the underlying model and training data, which may often not be available to testers. Most coverage measures assess performance using adversarial inputs, prioritising model robustness over correctness. Despite the purported sensitivity of these measures to adversarial inputs, empirical studies have not consistently demonstrated a significant correlation between these coverage metrics and their ability to detect bugs [3], [52], [23].

There's limited work that presents criteria for black-box coverage in the testing of AI-driven systems. Hauer et al.[40] present a statistical method to determine whether all potential scenario types have been encountered within the test scenarios used to test a self-driving car. These scenario types are defined by the ego car's maneuvers, such as lane changes, overtaking, and emergency braking. A similar study by Arcaini et al. [9] regards scenario types as granular driving characteristics like "turning with high lateral acceleration", using them as a coverage measure. Tang et al. categorize scenarios based on the map's topological structure and evaluate their approach's efficacy by assessing the coverage of these structures [86]. Given that autonomous vehicles base their decisions on parameterized rule-based systems and cost functions, Laurent et al. [51] employ weight coverage to encompass various configurations of a path planner.

Another commonly employed adequacy measure for both conventional and AI-based systems is test suite diversity, which is computed based on either test inputs or outputs [3], [34], [56], [13]. One example is the Shannon Diversity Index, which measures the diversity of a given sample by considering both its richness and evenness. Richness assesses the count of distinct species present within a population, while evenness measures the uniformity of individuals per species [59]. Another example is Geometric diversity [53], [46], [94], which was shown to be superior over numerous other prevalent black-box test adequacy metrics [3]. Geometric diversity draws its foundation from the area of Determinantal Point Process (DPP), an approach geared towards discerning diverse input sets [49]. DPP is used for subset selection scenarios, wherein the objective involves choosing a varied subset from a pool of candidates. DPP models the diversity between items within the chosen subset. Consequently, items exhibiting substantial similarity with one another become less likely candidates for inclusion in the final selection.

Diversity metrics are rooted in the concept that akin test cases tend to exercise similar segments of the source code or training data, thereby uncovering similar faults. Conversely, a system subjected to an extensive array of conditions is more likely to exhibit reliable performance in real-world scenarios. Consequently, diversifying test scenarios enhances the exploration of the fault space, subsequently elevating fault detection capabilities [19], [3], [97]. Nonetheless, existing research in the area of testing AI systems, and in particular GenAI systems underscores a substantial gap in the availability of robust diversity metrics that exhibit a significant correlation with fault detection [3], which presents an opportunity for further research.

A. Test suite Instance Space Adequacy Measures

Given the pivotal role of diversity and coverage in the black-box testing of intricate systems like GenAI systems, recently we proposed a new suite of metrics known as Test suite Instance Space Adequacy (TISA) metrics [65]. These metrics aim to objectively quantify the quality of testing in terms of both diversity and coverage. Rooted in a framework known as Instance Space Analysis (ISA), these metrics provide a two-dimensional representation of test instances. This representation unveils both the diversity and coverage of test instances,

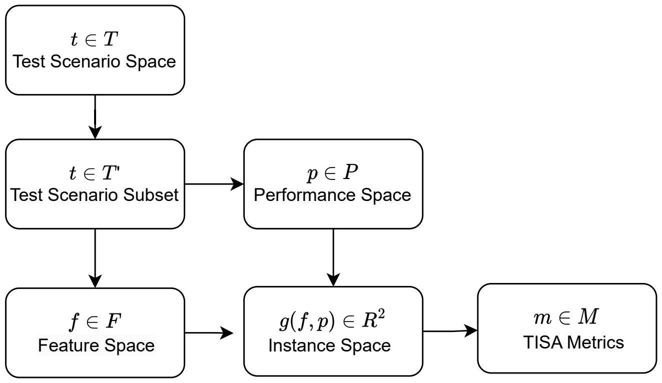

Fig. 2. The main components of Test suite Instance Space Adequacy (TISA).

granting insights into the diversity of the test suite across various features, the diversity of detected bugs, and the test suite's adequacy with respect to coverage. By facilitating the identification of areas where the test suite might lack diversity or require additional testing, these metrics offer a mechanism to enhance testing quality systematically and comprehensively.

Figure 2 illustrates the main steps of the TISA approach. To calculate the TISA measures, we consider a subset T' of the possible test cases T that can be used to test a GenAI system. It is not feasible, or in many cases, not possible to obtain all possible test cases in a reasonable amount of time, which makes assessing the adequacy of a test suite even more important. TISA then creates the feature space F by extracting an extensive set of features from the test cases T'. TISA has not been used for GenAI systems yet, but in the area of testing autonomous vehicles, TISA extracted features from driving scenarios, which constitute the test cases [65]. Extracted features included the number of lanes, the number of pedestrians, the number of right turns, etc.

Next the performance space P is created, which constitutes the outcomes of the test cases. If a test case fails or reveals incorrect behaviour of the software system, the test case is labelled as effective, otherwise it is deemed ineffective. The feature space and performance space become an input to the instance space generation approach, which creates the instance space. The instance space is a 2D representation of the multidimentional feature space, delineated by features that have the most significant influence on test outcomes.

An important step in TISA involves pinpointing the features that exert maximal impact on test outcomes. This process is instrumental in distinguishing effective scenarios — the ones that expose failures — from those that pass without issue. The task of feature identification and selection goes through an iterative procedure, leveraging machine learning techniques to uncover significant features that distinctly differentiate between these scenario types.

Once the significant features are identified, TISA projects test instances, originally defined within an n-dimensional feature space, onto a 2D coordinate plane. This projection aims to render the connection between instance features and test

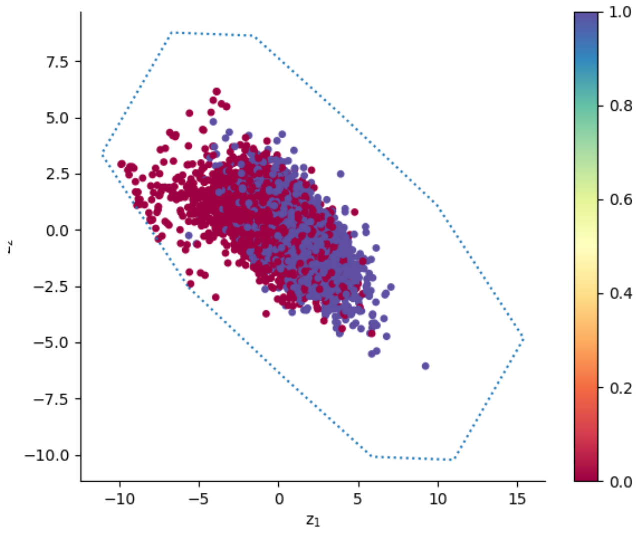

Fig. 3. An example of the instance space [65].

outcomes readily discernible. An optimal projection manifests as a linear trend, where variations in feature values or scenario outcomes along a straight line span from low to high values. Moreover, the proximity of instances in the high-dimensional feature space is maintained within the 2D instance space, ensuring topological preservation.

For illustration purposes, an example of an instance space, which was created for the problem of testing autonomous vehicles [65] is shown in Figure 3. Each point in the graph is a test case plotted in the 2D instance space created from the most significant features. TISA also draws the mathematical boundary created around the instance space for estimating the coverage of the test suite. This boundary is an indication of where possible test instances may exist, and helps identify existing gaps in the test suite. The colour in the graph is used to indicate whether a test case is failing, deemed as effective and coloured in purple, or passing, which means the testsuite is not effective as it could not detect incorrect behaviour or failures of the AI-based system and coloured in purple.

Figure 4 shows the distribution of values of one of the most significant features identified by TISA: the number of right turns. This plots helps explain why certain test cases are effective and others are not. As we can see, test scenarios with a higher number of right turns are more likely to reveal bugs in the ego vehicle.

Given the instance space, the last step is to calculate the TISA metrics. The key metrics proposed in [65] are:

- Area of the instance space, which refers to the region encompassed by all test instances within the instance space, estimating the overall diversity of the entire test suite.
- Area of the buggy region, which pertains to the section of the instance space taken up by most of the test instances that expose failures. An examples is shown in Figure 5
- Coverage of the instance space, which refers to the

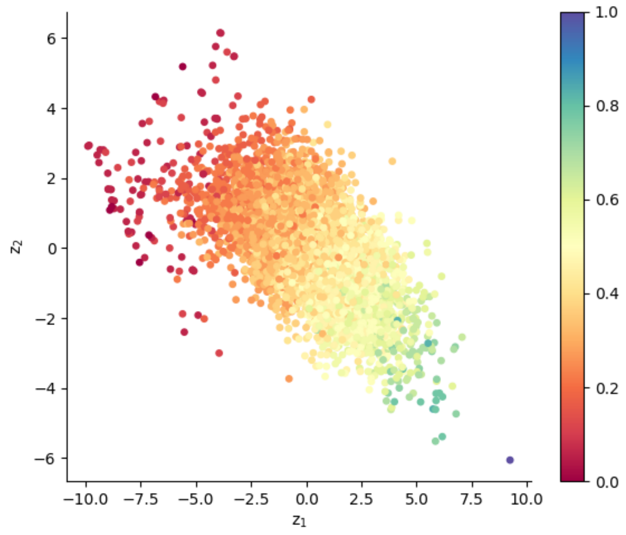

Fig. 4. Number of right turns [65].

proportion of the total area as indicated by the boundary, that is covered by test instances.

The experimental evaluation of these metrics showed that all TISA metrics demonstrate a positive correlation with faults, but the metric that measures the area of the buggy region (illustrated in Figure 5) particularly stands out due to its consistently robust and statistically significant correlation with bugs. This emphasizes its efficacy and reliability in providing valuable insights into the testing process's effectiveness.

In the future, there are opportunities in using TISA metrics to evaluate how well GenAI systems have been tested. One of the challenges is how to select a meaningful set of features to characterise such systems. One potential solution is to extract features from the embeddings of these systems, which contain rich semantic information by modelling the meaningful relationships and associations between words, captured by the numerical vectors used to represent those words in a high-dimensional space. Words with similar meanings or that are used in similar contexts will have word embeddings that are closer together in the vector space. Some recent word embeddings, like those produced by BERT [32], incorporate contextual information. This means that the meaning of a word in a particular sentence depends on the surrounding words. This enables embeddings to capture more nuanced semantic information.

In addition, embeddings often exhibit semantic relationships through analogies. Consider the words "Paris", "France", "Rome" and "Italy". Lets perform the following analogy by subtracting the vector for "France" from "Paris". Then we add the vector for "Italy". The resulting vector is likely to be closest to the word "Rome". This analogy reflects the semantic relationship that "Paris" is to "France" as "Rome" is to "Italy" in terms of capital cities and their respective countries. It demonstrates how embeddings can capture geographical and cultural associations between words, providing another example of their ability to represent meaningful semantic information. All this information can be used to characterise the test cases used for testing a GenAI system, and can help explain why a test case is effective or ineffective.

By constructing the instance space of the test scenarios used to test a GenAI system, we would be able to extract insights in terms of the diversity of the test inputs, and ensure that the test instances cover a wide range of scenarios, styles, or outputs that the GenAI system is expected to handle. This diversity helps assess the system's adaptability to various inputs.

TISA would also help identify test instances of varying complexity levels, from simple and straightforward cases to more intricate and challenging scenarios. This helps assess the system's robustness and capability to handle complex inputs. In addition, TISA can be used to assess the realism of test cases, ensuring that test instances resemble real-world data or scenarios the AI system will encounter.

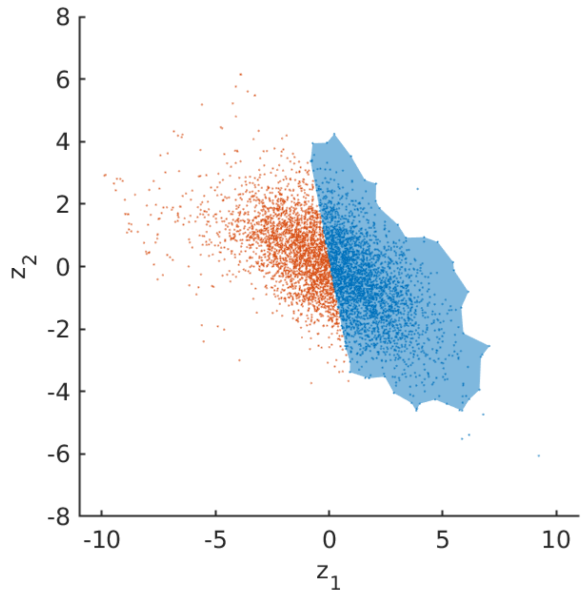

Fig. 5. The area of the buggy region.

Edge cases, outliers, or inputs that might trigger unusual behaviour in the AI system are important. Evaluating how well the system handles these cases can provide insights into its limitations and vulnerabilities. Introducing novel or unseen inputs to evaluate the system's ability to generalise beyond its training data and produce creative outputs is of particular importance, and can be tackled with a framework like TISA. By drawing the boundary of possible test inputs, and identifying areas of instance space where test inputs are sparse and effective (outliers), close to the boundary (edge cases), or close to the frontier of behaviours (unusual behaviours), TISA can help address these research challenges.

To apply TISA to GenAI systems, there are a set of research challenges and opportunities for further research. First, a method that extracts features from test cases used to test GenAI systems would need to be developed. Let's take an example from other AI-based systems such as self driving cars. Features of a test case may constitute the number of other cars in the road, the number of pedestrians crossing the road, the weather conditions, the speed limit, the curvature of the road, etc. For GenAI systems such as question answering systems, a test case is text. Potential features could be extracted from the vector embeddings of the text, which contains rich semantic information. Such features would help characterise the test cases, and help the creation of the instance space. Another research opportunity is to explore how TISA can help with prioritising test cases, by enabling the selection of the test cases which are more likely to reveal bugs, thus reducing the overall testing time. In addition, TISA could be used to guide test case generation, by highlighting areas where failing test cases are likely to be found, or by showing the empty areas in the instance space which require more exploration.

VII. Conclusion

As software testing encounters the novel challenges posed by GenAI systems, the need for new testing approaches becomes evident. The oracle problem, which involves establishing the correctness and quality of creative outputs, stands as a central challenge in testing GenAI systems. The absence of a definitive ground truth, coupled with the subjectivity of human evaluators, has made it challenging to estimate the quality and correctness of generated content. The proposed solution of training an AI model to detect bias and deviations from the expected behaviour is a promising direction to mitigate this challenge.

Furthermore, addressing the adequacy of testing through measures like Test suite Instance Space Adequacy (TISA) metrics offers a quantitative and qualitative approach to assessing the diversity and coverage of test instances. By providing a two-dimensional representation of the instance space, TISA enables a systematic evaluation of the test suite's quality, revealing gaps and areas that require further testing.

As GenAI systems continue to permeate various domains, from creative content generation to complex decision-making, ensuring their quality and correctness is of paramount importance. This calls for a reimagining of testing methodologies that account for the inherent uncertainties and complexities of GenAI outputs.

References

[1] Zahra Abbasiantaeb and Saeedeh Momtazi. Text-based question answering from information retrieval and deep neural network perspectives: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(6):e1412, 2021.

[2] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69. PMLR, 2018.

[3] Zohreh Aghababaeyan, Manel Abdellatif, Lionel Briand, Mojtaba Bagherzadeh, et al. Black-box testing of deep neural networks through test case diversity. *arXiv preprint arXiv:2112.12591*, 2021.

[4] Open AI. Gpt-4 technical report. *Technical Report*, 2023.

[5] Rachith Aiyappa, Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. Can we trust the evaluation on chatgpt? *arXiv preprint arXiv:2303.12767*, 2023.

[6] Saranya Alagarsamy, Chakkrit Tantithamthavorn, and Aldeida Aleti. A3test: Assertion-augmented automated test case generation. *arXiv preprint arXiv:2302.10352*, 2023.

[7] Aldeida Aleti and Irene Moser. Predictive parameter control. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, page 561–568, New York, NY, USA, 2011. Association for Computing Machinery.

[8] Aldeida Aleti, Irene Moser, and Lars Grunske. Analysing the fitness landscape of search-based software testing problems. *Automated Software Engineering*, 24:603–621, 2017.

[9] Paolo Arcaini, Xiao-Yi Zhang, and Fuyuki Ishikawa. Targeting patterns of driving characteristics in testing autonomous driving systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 295–305. IEEE, 2021.

[10] Kevin Bartz, Cory Barr, and Adil Aijaz. Natural language generation for sponsored-search advertisements. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, pages 1–9, 2008.

[11] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret 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*, pages 610–623, 2021.

[12] Parminder Bhatia, Kristjan Arumae, Nima Pourdamghani, Suyog Deshpande, Ben Snively, Mona Mona, Colby Wise, George Price, Shyam Ramaswamy, and T Kass-Hout. Aws cord19-search: A scientific literature search engine for covid-19. 2020.

[13] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. Automated test cases prioritization for self-driving cars in virtual environments. *arXiv preprint arXiv:2107.09614*, 2021.

[14] Sumon Biswas and Hridesh Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. In *ACM Joint meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 642–653, 2020.

[15] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[16] Houssem Ben Braiek and Foutse Khomh. On testing machine learning programs. *Journal of Systems and Software*, 164:110542, 2020.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[18] Longbing Cao. Ai in finance: challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[19] Emanuela G Cartaxo, Patrícia DL Machado, and Francisco G Oliveira Neto. On the use of a similarity function for test case selection in the context of model-based testing. *Software Testing, Verification and Reliability*, 21(2):75–100, 2011.

[20] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. Bias in machine learning software: why? how? what to do? In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 429–440, 2021.

[21] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*, 2023.

[22] Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pages 981–990. PMLR, 2019.

[23] Junjie Chen, Ming Yan, Zan Wang, Yuning Kang, and Zhuo Wu. Deep neural network test coverage: How far are we? *arXiv preprint arXiv:2010.04946*, 2020.

[24] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. Testing your question answering software via asking recursively. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 104–116. IEEE, 2021.

[25] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. Testing your question answering software via asking recursively. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 104–116, 2021.

[26] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. Validation on machine reading comprehension software without annotated labels: A property-based method. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 590–602, 2021.

[27] TY Chen, SC Cheungx, and SM Yiu. Metamorphic testing: a new approach for generating next test cases. *arXiv preprint arXiv:2002.12543*, 2020.

[28] Hyunji Chung, Michaela Iorga, Jeffrey Voas, and Sangjin Lee. Alexa, can i trust you? *Computer*, 50(9):100–104, 2017.

[29] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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)*, pages 2924–2936, 2019.

[30] Martin D Davis and Elaine J Weyuker. Pseudo-oracles for non-testable programs. In *Proceedings of the ACM'81 Conference*, pages 254–257, 1981.

[31] Ryan Daws. Medical chatbot using openai's gpt-3 told a fake patient to kill themselves. *AI News*, 2020.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[33] Zhiyu Fan, Xiang Gao, Martin Mirchev, Abhik Roychoudhury, and Shin Hwei Tan. Automated repair of programs from large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1469–1481. IEEE, 2023.

[34] Robert Feldt, Simon Poulding, David Clark, and Shin Yoo. Test set diameter: Quantifying the diversity of sets of test cases. In *2016 IEEE international conference on software testing, verification and validation (ICST)*, pages 223–233. IEEE, 2016.

[35] Sorelle A Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P Hamilton, and Derek Roth. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 329–338, 2019.

[36] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. Importance-driven deep learning system testing. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 702–713, 2020.

[37] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 739–743, 2018.

[38] Mark Harman, Yue Jia, and Yuanyuan Zhang. Achievements, open problems and challenges for search based software testing. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pages 1–12. IEEE, 2015.

[39] Mark Harman and Phil McMinn. A theoretical and empirical study of search-based testing: Local, global, and hybrid search. *IEEE Transactions on Software Engineering*, 36(2):226–247, 2009.

[40] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955. IEEE, 2019.

[41] Ryuichiro Higashinaka, Masahiro Mizukami, Hidetoshi Kawabata, Emi Yamaguchi, Noritake Adachi, and Junji Tomita. Role play-based question-answering by real users for building chatbots with consistent personalities. In *Proceedings of the 19th annual sigdial meeting on discourse and dialogue*, pages 264–272, 2018.

[42] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In *Proceedings of the 26th conference on program comprehension*, pages 200–210, 2018.

[43] Hanyao Huang, Ou Zheng, Dongdong Wang, Jiayi Yin, Zijin Wang, Shengxuan Ding, Heng Yin, Chuan Xu, Renjie Yang, Qian Zheng, et al. Chatgpt for shaping the future of dentistry: the potential of multi-modal large language model. *International Journal of Oral Science*, 15(1):29, 2023.

[44] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

[45] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929. IEEE, 2012.

[46] Byungkon Kang. Fast determinantal point process sampling with application to clustering. *Advances in Neural Information Processing Systems*, 26, 2013.

[47] Charaka Geethal Kapugama, Van-Thuan Pham, Aldeida Aleti, and Marcel Böhme. Human-in-the-loop oracle learning for semantic bugs in string processing programs. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 215–226, 2022.

[48] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

[49] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[50] Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. *arXiv preprint arXiv:2305.18486*, 2023.

[51] Thomas Laurent, Stefan Klikovits, Paolo Arcaini, Fuyuki Ishikawa, and Anthony Ventresque. Parameter coverage for testing of autonomous driving systems under uncertainty. *ACM Transactions on Software Engineering and Methodology*, 2022.

[52] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. Structural coverage criteria for neural networks could be misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 89–92. IEEE, 2019.

[53] Hui Lin and Jeff A Bilmes. Learning mixtures of submodular shells with application to document summarization. *arXiv preprint arXiv:1210.4871*, 2012.

[54] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, 2022.

[55] Zixi Liu, Yang Feng, Yining Yin, Jingyu Sun, Zhenyu Chen, and Baowen Xu. Qatest: A uniform fuzzing framework for question answering systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[56] Chengjie Lu, Huihui Zhang, Tao Yue, and Shaukat Ali. Search-based selection and prioritization of test scenarios for autonomous driving systems. In *International Symposium on Search Based Software Engineering*, pages 41–55. Springer, 2021.

[57] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang 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*, pages 120–131, 2018.

[58] Pingchuan Ma, Shuai Wang, and Jin Liu. Metamorphic testing and certified mitigation of fairness violations in nlp models. In *IJCAI*, pages 458–465, 2020.

[59] Anne E Magurran. Measuring biological diversity. *Current Biology*, 31(19):R1174–R1177, 2021.

[60] R Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*, pages 3428–3448. Association for Computational Linguistics (ACL), 2020.

[61] William M McKeeman. Differential testing for software. *Digital Technical Journal*, 10(1):100–107, 1998.

[62] Phil McMinn. Search-based software testing: Past, present and future. In *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, pages 153–163. IEEE, 2011.

[63] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[64] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40, 2021.

[65] Neelofar and Aldeida Aleti. Towards reliable AI: Adequacy metrics for ensuring the quality of AI-based systems. In *Proceedings of the*

ACM/IEEE 46nd International Conference on Software Engineering, pages 805–816, 2024.

[66] Carlos Oliveira, Aldeida Aleti, Yuan-Fang Li, and Mohamed Abdelrazek. Footprints of fitness functions in search-based software testing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 1399–1407, New York, NY, USA, 2019. Association for Computing Machinery.

[67] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[68] Anjana Perera, Aldeida Aleti, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Burak Turhan, Lisa Kuhn, and Katie Walker. Search-based fairness testing for regression-based machine learning systems. *Empirical Software Engineering*, 27(3):79, 2022.

[69] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[70] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Autoencoders. In *Machine learning*, pages 193–208. Elsevier, 2020.

[71] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[72] Martin Reisenbichler, Thomas Reutterer, David A Schweidel, and Daniel Dan. Frontiers: Supporting content marketing with natural language generation. *Marketing Science*, 41(3):441–452, 2022.

[73] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering*, 25:5193–5254, 2020.

[74] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, 33:20601–20611, 2020.

[75] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[76] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42, 2021.

[77] Saqib Shah and Julian Chokkattu. Microsoft kills ai chatbot tay (twice) after it goes full nazi. 2016.

[78] Qingchao Shen, Junjie Chen, Jie M Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. Natural test generation for precise testing of question answering software. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[79] Clemencia Siro and Tunde Oluwaseyi Ajayi. Evaluating the robustness of machine reading comprehension models to low resource entity renaming. In *4th Workshop on African Natural Language Processing*, 2023.

[80] Yanqi Su, Zheming Han, Zhenchang Xing, Xin Xia, Xiwei Xu, Liming Zhu, and Qinghua Lu. Constructing a system knowledge graph of user tasks and failures from bug reports to support soap opera testing. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[81] Yanqi Su, Zheming Han, Zhenchang Xing, Xiwei Xu, Liming Zhu, and Qinghua Lu. Soapoperatg: A tool for system knowledge graph based soap opera test generation. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 51–54, 2023.

[82] Bing Sun, Jun Sun, Hong Long Pham, and Jie Shi. Causality-based neural network repair. In *International Conference in Software Engineering*, 2022.

[83] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–23, 2019.

[84] Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. *arXiv e-prints*, pages arXiv–2305, 2023.

[85] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. 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)*, pages 4149–4158, 2019.

[86] Yun Tang, Yuan Zhou, Yang Liu, Jun Sun, and Gang Wang. Collision avoidance testing for autonomous driving systems on complete maps. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 179–185. IEEE, 2021.

[87] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[88] Priyan Vaithilingam, Tianyi Zhang, and Elena 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*, pages 1–7, 2022.

[89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[90] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[91] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, et al. Cord-19: The covid-19 open research dataset. In *ACL 2020 Workshop on Natural Language Processing for COVID-19 (NLP-COVID)*, 2020.

[92] Ruoqi Wei, Cesar Garcia, Ahmed El-Sayed, Viyaleta Peterson, and Ausif Mahmood. Variations in variational autoencoders-a comparative evaluation. *Ieee Access*, 8:153651–153670, 2020.

[93] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon 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*, pages 146–157, 2019.

[94] Haotian Xu and Zhijian Ou. Scalable discovery of audio fingerprint motifs in broadcast streams with determinantal point process based motif clustering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):978–989, 2016.

[95] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1):1–36, 2020.

[96] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[97] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo 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*, pages 79–90, 2021.