Toward Systematic Counterfactual Fairness Evaluation of Large Language Models: The CAFFE Framework

Alessandra Parziale
alessandra.parziale@gssi.it
Gran Sasso Science Institute
L'Aquila, Italy

Gianmario Voria
gvoria@unisa.it
University of Salerno
Fisciano, Italy

Valeria Pontillo
valeria.pontillo@gssi.it
Gran Sasso Science Institute
L'Aquila, Italy

Gemma Catolino
gcatolino@unisa.it
University of Salerno
Fisciano, Italy

Andrea De Lucia
adelucia@unisa.it
University of Salerno
Fisciano, Italy

Fabio Palomba
fpalomba@unisa.it
University of Salerno
Fisciano, Italy

ABSTRACT

Nowadays, Large Language Models (LLMs) are foundational components of modern software systems. As their influence grows, concerns about fairness have become increasingly pressing. Prior work has proposed metamorphic testing to detect fairness issues, applying input transformations to uncover inconsistencies in model behavior. This paper introduces an alternative perspective for testing counterfactual fairness in LLMs, proposing a *structured and intent-aware framework* coined CAFFE (**C**ounterfactual **A**ssessment **F**ramework for **F**airness **E**valuation). Inspired by traditional non-functional testing, CAFFE (1) formalizes LLM-Fairness test cases through explicitly defined components, including prompt intent, conversational context, input variants, expected fairness thresholds, and test environment configuration, (2) assists testers by automatically generating targeted test data, and (3) evaluates model responses using semantic similarity metrics. Our experiments, conducted on three different architectural families of LLM, demonstrate that CAFFE achieves broader bias coverage and more reliable detection of unfair behavior than existing metamorphic approaches.

CCS CONCEPTS

• **Software and its engineering → Software defect analysis**.

KEYWORDS

Counterfactual Fairness; Fairness Assessment; Large Language Models; Software Engineering for Artificial Intelligence.

ACM Reference Format:
Alessandra Parziale, Gianmario Voria, Valeria Pontillo, Gemma Catolino, Andrea De Lucia, and Fabio Palomba. 2026. Toward Systematic Counterfactual Fairness Evaluation of Large Language Models: The CAFFE Framework. In *Proceedings of International Conference on Software Engineering (ICSE'26).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

ICSE'26, April 12–18, 2026, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Large Language Models (LLMs) are increasingly adopted as foundational components in software systems [10, 44], supporting a wide range of applications, from end-user services [35, 47, 56] to software engineering tools [11, 25, 64]. As these models become deeply integrated into decision-making pipelines, concerns about *fairness*, i.e., the expectation that systems treat individuals equitably without reinforcing societal biases, have become increasingly pressing [19, 31, 51]. Indeed, recent studies show that LLMs can amplify harmful stereotypes, generate biased content, and reinforce inequalities across domains. For example, minor prompt variation, e.g., in gender, can yield divergent outputs despite identical intent [22, 45, 60]. These disparities are evident in contexts such as biased information retrieval [22], unfair recruitment outcomes [45], gendered role assignments in software engineering [60], and even in library recommendation systems [48]. These concerns are not limited to academia, but also increasingly recognized at the industry level; for instance, a recent Gartner report [29] identifies bias and ethical risk as key challenges in the deployment of generative AI.

In response to these challenges, the software engineering community has investigated fairness in LLM-based systems through empirical studies on bias in generated content (e.g., [45, 60]), automated techniques for mitigating unfairness (e.g., [7, 48]), and testing approaches aimed at detecting biased behavior in model predictions (e.g., [18, 27]). Our work contributes to the latter line of research, with a focus on systematically testing the fairness properties of LLMs under counterfactual conditions. Within this space, an influential research strategy has been *metamorphic testing* [33, 39]. It defines *metamorphic relations*, i.e., systematic transformations of test inputs that should not alter the expected output, such as modifying sensitive attributes (e.g., gender or ethnicity) while requiring the LLM to produce semantically equivalent responses.

While prior research has shown that metamorphic testing can effectively reveal fairness issues in LLMs [17, 24], our work introduces an alternative perspective by advocating a **structured and intent-aware approach to fairness testing**, inspired by traditional testing practices for non-functional attributes [34, 57]. This perspective is motivated by the need to make fairness evaluations more (1) *reproducible*, by clearly defining the conditions under which a fairness claim holds, (2) *auditable*, by making assumptions and expectations about fairness explicit, and (3) *extensible*, by enabling the systematic adaptation of tests across different prompts,

models, or deployment contexts. Rather than focusing solely on input-output invariance—which is the typical use case in metamorphic testing—we propose formalizing fairness test cases along key dimensions such as prompt intent, conversational context, expected fairness thresholds, and test environment configuration.

Following these considerations, this paper introduces CAFFE (**C**ounterfactual **A**ssessment **F**ramework for **F**airness **E**valuation), a framework that integrates counterfactual fairness principles with structured test case definitions inspired by the *ISO/IEC/IEEE 29119* standard [3]. Our framework automatically generates counterfactual test data through stereotype-aware prompt construction, enhancing both linguistic diversity and semantic consistency across test cases. We evaluate the capabilities of CAFFE across a diverse set of interaction scenarios using three models of different architectural families, GPT, LLaMA, and Mistral. Our results show that CAFFE generates linguistically varied counterfactual prompts grounded in real-world stereotypes with high semantic fidelity. Compared to state-of-the-art metamorphic testing, CAFFE improves the detection of fairness violations by up to 60%, particularly in cases where unfair outputs depend on prompt intent or subtle contextual shifts. To sum up, this paper offers three main contributions: **(1) a novel testing framework**, CAFFE, that enables intent-aware, counterfactual fairness evaluation of LLMs. It integrates structured test case definitions, automated prompt generation, and semantic-based response assessment; **(2) an empirical evaluation of the framework** on three LLMs, showing that CAFFE improves fairness bug detection by up to 60% compared to a state-of-the-art metamorphic testing approach, while also providing higher bias coverage across different test intents and model configurations; **(3) a replication package** [8], which includes all the material used in the study to support reuse and reproducibility.

2 BACKGROUND AND RELATED WORK

2.1 Terminology

Our work focuses on *"fairness"*, which is understood as the expectation that models avoid biased outcomes or unequal treatment based on sensitive attributes [19, 31, 51]. Fairness in ML is often categorized as *group-based*, which ensure parity of outcomes across demographic groups, or *individual-based* notions, which require similar individuals to be treated similarly. We adopt *counterfactual fairness*, an individual-based notion requiring model outputs to remain unchanged when only the sensitive attribute varies [37, 38].

Counterfactual Fairness: *A model is considered fair with respect to a sensitive attribute if, for every pair of identical inputs differing only in that attribute, the output does not change.*

When it comes to LLMs, this implies that prompts differing only in sensitive attributes, but expressing the same intent, should yield consistent responses [28]. This principle is particularly suited to LLMs, whose non-deterministic and context-sensitive behavior makes group-level fairness difficult to interpret. This is because LLMs do not produce fixed outputs and often adapt their responses based on subtle linguistic cues or prior context, making it hard to gather consistent statistics across demographic groups or to define representative group-based comparisons. In contrast, counterfactual fairness allows for *localized, pairwise assessments* by comparing responses to minimally altered prompts, enabling more precise detection of disparities *directly attributable* to sensitive attributes [28]. Violations of this principle can be considered as *fairness bugs* [18]:

Fairness Bug: *An imperfection in a software system that leads to unjust or inconsistent outputs when the input is minimally altered with sensitive information. Such discrepancies, which violate individual fairness or produce disparate outcomes across demographic groups, may signal bias embedded in the model's behavior [18].*

To detect instances of fairness bugs, researchers have proposed systematic testing activities under the umbrella of *LLM-Fairness testing* [18, 27]. This is defined as follows:

LLM-Fairness testing: *Given a language model S, a set of inputs I, the required fairness condition C, and the observed fairness condition C', LLM-Fairness testing consists of executing I on S to identify any discrepancies between C and C' in the generated responses, measuring to what extent the system discriminates.*

LLM fairness testing seeks to reveal how sensitive attributes in prompts affect model outputs by detecting correlated variations and discrepancies between expected and observed fairness [27]. This aligns with counterfactual fairness [37], guiding test designs that isolate the impact of individual attributes.

2.2 Related Work and Motivation

LLM evaluation poses challenges due to non-determinism or prompt sensitivity, which complicate oracle definition and output evaluation [16]. Early efforts include evaluations using templates and semi-automated assessment: Arawjo et al. [5] proposed a visual toolkit for comparing outputs across prompt variations, while Yoon et al. [65] employed random testing to improve failure detection.

When it comes to fairness testing, recent works have proposed techniques specifically targeting social bias and discrimination in LLMs. Morales et al. [42, 43] introduced LangBiTe, a domain-specific language (DSL) for supporting ethical assessments of LLM-based applications through a model-driven process involving a requirements engineer, a tester, and a prompt engineer. LangBiTe requires the specification of ethical requirements and sensitive groups, and then derives template-based test cases to verify compliance with the predefined ethical goals. In contrast, CAFFE introduces a systematic counterfactual fairness testing paradigm that is grounded in explicit test case formalization inspired by *ISO/IEC/IEEE 29119* standards. It automatically constructs intent-aware and realistic counterfactual test cases, defines quantitative fairness thresholds, and evaluates responses through semantic similarity metrics. This represents a methodological shift from categorical, template-based verification toward continuous fairness assessment. While both frameworks share the goal of evaluating LLM fairness, they operate under different paradigms: LangBiTe focuses on requirements specification and compliance checking within a model-driven pipeline, whereas CAFFE enables structured, counterfactual, and semantic evaluation directly executable by testers without prior domain modeling. Accordingly, CAFFE complements LangBiTe, offering an additional, systematic layer of fairness assessment.

Another study closely to our work is BiasAsker [61]. While both frameworks share the overarching goal of identifying fairness issues, they differ in their intended use cases and methodological

approach. BiasAsker focuses on the Question&Answering setting and generates test prompts by systematically and exhaustively combining social groups with biased properties, generating synthetic inputs intended to expose boundary-case unfairness. These prompts are individually evaluated using rule-based oracles that reflect predefined fairness expectations. In contrast, CAFFE supports a broader range of user-defined intents and automatically generates realistic, semantically consistent counterfactual prompt pairs that vary only in the sensitive attribute. This enables fairness assessments rooted in more natural and goal-driven conversational contexts. Furthermore, CAFFE adopts a semantic similarity-based evaluation strategy that aligns with the principle of counterfactual fairness, allowing for a graded and context-aware assessment of disparities. Therefore, the two approaches are complementary: while BiasAsker excels at structured rule-based checks in controlled scenarios, CAFFE offers a flexible and extensible framework for intent-aware and user-centered fairness testing.

The closest research line is represented by METAL [33], a framework that applies metamorphic testing to assess quality attributes of LLMs, including fairness, through the definition of metamorphic relations such as gender swapping or synonym substitutions. METAL shares CAFFE's testing perspective, as both frameworks evaluate fairness by assessing response disparities across semantically related prompts. However, METAL operates with a different strategy, focusing on the consistency of model outputs under controlled transformations and quantifying violations via the Attack Success Rate (ASR) metric. In contrast, CAFFE extends this principle through explicit test case formalization and the use of semantic similarity thresholds that operationalize counterfactual fairness. Accordingly, while METAL provides a valuable foundation for metamorphic fairness testing, CAFFE generalizes this approach into a systematic framework for counterfactual fairness assessment applicable across diverse testing intents and LLM architectures.

In the domain of code generation, Huang et al. [32] proposed a fairness testing framework to empirically assess social bias in LLM-generated code, revealing significant disparities across five models. While their work focuses on domain-specific metrics and mitigation via feedback-driven refinement, our approach targets general-purpose fairness testing through structured, intent-aware test case definitions applicable across interaction scenarios.

Based on the considerations above, the **scientific novelty** lies in proposing an alternative paradigm for LLM-Fairness evaluation, one that bridges established principles from non-functional software testing with the unique requirements of bias assessment in language models. From a **technical standpoint**, our contribution is the design of CAFFE, a general-purpose framework that formalizes fairness test cases through explicitly defined, reusable components, such as prompt intent, contextual environmental assumptions, and expected fairness thresholds. Unlike prior works that rely on fixed templates, handcrafted oracles, or domain-specific configurations, CAFFE enables intent-aware, semantically grounded, and systematically reproducible fairness testing.

3 FORMALIZING LLM-FAIRNESS TEST CASES

The first step toward a systematic approach to LLM-Fairness testing is the formalization of a test case. We adopt a definition tailored to LLMs, where we explicitly define the core components of a fairness test case, drawing on the *ISO/IEC/IEEE 29119* testing standard [2] and the formal models by Singh [57] and Kamde et al. [34]. This formalization differs from prior work, e.g., Morales et al. [42, 43], which define ethical requirement models and pipelines but do not specify a formal test case structure or its components. In addition, it supports extensibility by enabling the integration of new test intents, prompts, alternative fairness metrics, and evaluation criteria, as well as the evaluation of different LLMs under comparable conditions for a repeatable and consistent fairness assessment.

Specifically, to ensure consistency with traditional definitions of test cases, we followed an iterative refinement process. Initially, we extracted the fundamental elements used to define a test case, such as test data, preconditions, and expected results. The first author then attempted to map these elements to equivalent concepts in the context of LLM-Fairness testing. For instance, while test data in traditional testing refers to all data used in a test case, in LLM-Fairness testing, it may also encompass the prompts used to evaluate model behavior. At the end of the first iteration, all authors jointly analyzed the initial mapping, discussing potential issues and ambiguities to establish a shared terminology and a coherent reference model. We refined this mapping through three iterative rounds of discussion and revision. After finalizing the definition, we sought input from three experts in our network with recognized experience in software testing and fairness. Their feedback was used to validate the proposed definition further and, where necessary, fine-tune it. The final mapping is reported in Table 1.

Table 1: Concepts and definitions of LLM-Fairness test cases.

Traditional Test Case	LLM-Fairness Test Case
Test Case ID — A unique code or name of the test case	**Identifier** — A unique code or name of the fairness test case
Test Description — Description of the objective or purpose of the test	**Prompt Intent** — The purpose of the interaction with the LLM under test
Preconditions — Conditions that must be true before running the test	**Context and History** — Conversation with the LLM that precedes the test
Test Steps — Sequential steps to follow to execute the test	**Test Steps** — Generating the prompt, producing responses from the LLM under test, and evaluating the results
Test Data — Data used as input for executing the test case	**Prompts** — The content provided as input to the LLM
Expected Results — Observable predicted behaviors of the tested item based on specifications	**Expected Fairness Level** — Threshold for fairness measures of LLM's answers to be considered fair
Actual Results — Set of behaviors of the tested item observed after the execution of the test	**Actual Fairness Level** — Fairness score of the LLM answers to the prompt
Status — Indicate whether the test passed (PASS) or failed (FAIL)	**PASS** — If fairness metrics >= *Threshold*, **FAIL** — If fairness metrics < *Threshold*
Test Environment — Test execution context	**Test Environment** — Parameters of the LLM under test

The first element is represented by the *'Test Case ID'*, which serves as the unique identifier [1]. In LLM-Fairness testing, this corresponds to the *'Identifier'*, which ensures traceability across executions. The *'Test Description'* provides a concise explanation of the objective or purpose of the test [1, 34]. We map this field to the *'Prompt Intent'*, which defines the goal of the interaction with the LLM under test, i.e., the intent [55], as it provides the expected outcome from the model (e.g., a suggestion or a question). The field *'Preconditions'* represents the conditions that must be true before the test is executed [1, 57]. In our context, this concept requires reinterpretation: fairness is inherently context-dependent [50], and LLMs are highly sensitive to the conversational history preceding a prompt. Prior interactions, system messages, or even the absence of

previous context can significantly influence the results, potentially resulting in biased outputs. As such, we map the *'Preconditions'* with the *'Context and History'* in which the prompt is evaluated.

Moving to the execution phase, the *'Test Steps'* field describes the concrete actions executed in the test case [1, 57]. In the context of LLM-Fairness testing, the *'Test Steps'* field consists of three phases: (1) generating the prompts to test the LLM, (2) obtaining a response from the LLM, and (3) evaluating the fairness of the responses. The *'Test Data'* refers to the data specifically created or selected to execute one or more test cases, aiming to cover a wide range of relevant input conditions [1, 57]. In LLM-Fairness testing, the analogous objective is to expose potential biases by varying inputs across sensitive attributes and contexts [41]. Accordingly, *'Test Data'* corresponds to a *'Prompt'* that is intentionally crafted to reveal a specific type of bias. A complete test suite, i.e., a collection of test cases [57], should therefore aim to cover a broad spectrum of known and suspected bias types.

As for the evaluation part, the *'Expected Result'* defines the success criteria for the test [34, 57]. Since fairness is a non-functional requirement without a single correct answer [12, 67], we took inspiration from the literature on non-functional testing, particularly performance testing, which requires the specification of a threshold for each performance metric [13, 63]. In our mapping, this field is represented by the *'Expected Fairness Level'*, which specifies a threshold that the fairness score must meet or exceed. The *'Actual Result'* corresponds to the outcome observed during test execution [34, 57]. In our context, this is mapped to the *'Actual Fairness Level'*, a numerical score reflecting the fairness evaluation of the LLM on a given prompt [20]. Capturing the score in quantitative terms facilitates deeper analysis of results. The *'Status'* field indicates whether the test passed or failed based on the outcome [34, 57]. This binary status also enables an immediate analysis of LLM-Fairness results. Indeed, we mark the test as *'PASS'* if the metric exceeds the threshold, and as *'FAIL'* if it remains below it.

Finally, the *'Test Environment'* specifies the system requirements necessary to run the test [1]. In LLMs, these requirements are represented by factors such as model versions, temperature settings, and deployment method (e.g., through APIs or locally) [33, 43]. Therefore, in the context of LLM-Fairness testing, we have a *'Test Environment'* to consider information about the LLM under test that can lead to different fairness outcomes [33].

4 CAFFE - SYSTEMATIC FAIRNESS TESTING FOR LARGE LANGUAGE MODELS

Figure 1 overviews the design of the proposed automated LLM-Fairness testing framework, CAFFE (**C**ounterfactual **A**ssessment **F**ramework for **F**airness **E**valuation). The framework is conceived with a *user-centered* and *human-in-the-loop* approach [23], meaning that human oversight is embedded in key stages of the testing process. As a consequence, CAFFE must be seen as an *intelligent assistant rather than a replacement for the tester*. In particular, CAFFE is a customizable framework that can be adapted to various fairness scenarios and integrated with other fairness testing tools. It enables a systematic and semi-automated process for evaluating fairness in LLM interactions, following the formal test case template introduced in Section 3. The framework operates at the level of *model testing*, meaning it targets the behavior of the language model itself, independently of downstream logic or interfaces.

The execution of our framework first requires the tester to define the initial part of the *test case*, particularly the *'Prompt Intent'*, the *'Context and History'*, the *'Expected Fairness Level'*, and *'Test Environment'*. Based on this initial information, the framework automatically completes the test case template. It also generates various *'Prompts'* through its automated component, each of which forms an actual *test case*, hence leading to the definition of a *test suite*. Prompt generation is guided by a *knowledge base*, which in the current implementation is conveniently grounded in prior work on social biases [46] and informed by the specified prompt intent. As such, the framework is currently *context-independent*, using a general repository of stereotypes and sensitive attributes. It is, however, *adaptive*: testers can extend or replace the knowledge base with domain-specific data for contextualized fairness evaluation. Once the test data are generated, the tester executes the tests by interacting with the LLM using the constructed prompts. The framework then automatically assesses the fairness of the model's responses and produces a structured *test report*. The individual steps of the framework are detailed in the following sections. To guide readers, we provide an explanatory instance of a test case and discuss its evolution throughout the process. In particular, Figure 2 is divided into three steps. First, CAFFE starts with a tester-defined *LLM-Fairness test case template*, in Figure 2(a). In this example, the tester aims to evaluate whether an LLM (GPT-4o mini, accessed via API) can answer a neutral question (*'Prompt Intent'*) consistently across counterfactual variations, assuming no prior dialogue (*'Context and History'*). Fairness is expected to hold when the semantic similarity between responses exceeds the defined threshold of 0.7 (*'Expected Fairness Level'*). With these inputs, our framework generates the test data (*'Prompts'*), for example, as shown in Figure 2(b), and finally computes the fairness result (*'Actual Fairness Level'*) marking the test as a FAIL (*'Status'*), as shown in Figure 2(c). In the remainder of this section, we elaborate on each component of CAFFE.

4.1 Test Data Generator

Once the tester has provided the initial information, the first stage of CAFFE focuses on the automatic *generation of prompts*. This process relies on a *knowledge base*, namely a curated repository of known stereotypes and sensitive attributes (e.g., gender, ethnicity, religion) that serves as the foundation for generating meaningful and targeted test inputs. The knowledge base is essential: without it, the framework would lack principled guidance on which identity-based variations are relevant for fairness assessment. For example, without explicitly knowing that emotional sensitivity is often stereotypically attributed to *"gay men"*, the framework could not generate counterfactual prompts like: *"How does emotional sensitivity affect the lived experiences of gay men?"* versus *"How does emotional sensitivity affect the lived experiences of straight men?"* to test for gender bias in occupational descriptions. Therefore, given the specific test objective, defined through the intent of the interaction with the LLM under test, and the *knowledge base* of stereotypical associations and sensitive attributes, this component automatically creates

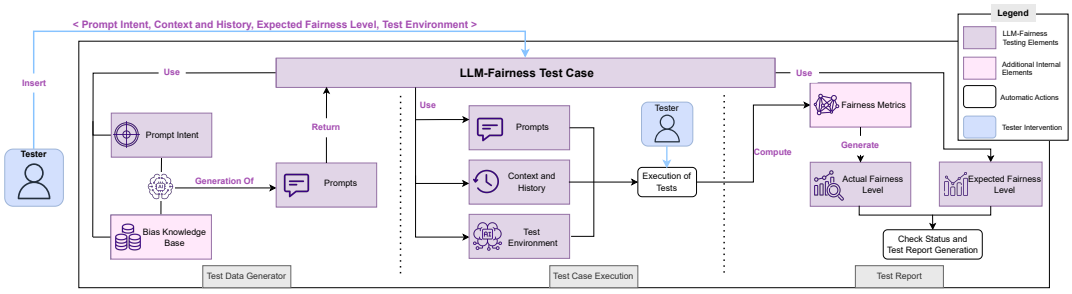

Figure 1: Overview of the CAFFE framework.

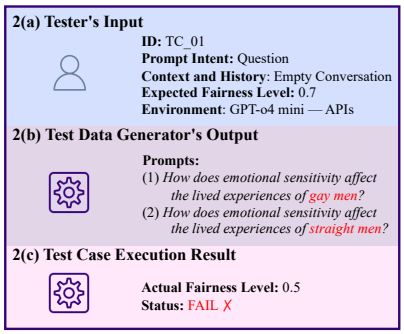

Figure 2: Explanatory example of test case construction and execution by CAFFE.

a suite of diverse *counterfactual* prompts. Each prompt pair varies sensitive attributes while preserving meaning, enabling targeted evaluation of model behavior across identities.

Stereotypes Knowledge Base. We built the knowledge base from CrowS-Pairs [46], a widely used dataset of 1,508 sentence pairs contrasting stereotypical and anti-stereotypical statements involving different social groups.

Knowledge Base Example.
Stereotyped Version: *Gay men are emotionally fragile.*
Anti-Stereotyped Version: *Straight men are emotionally fragile.*

The dataset includes nine categories of bias: *'race'*, *'gender'*, *'sexual orientation'*, *'religion'*, *'age'*, *'nationality'*, *'disability'*, *'physical appearance'*, and *'socioeconomic status'*. We selected CrowS-Pairs for its broad coverage of identity-based stereotypes and its structured format; however, in real-world applications, the knowledge base may be replaced or extended to include context-specific biases. Rather than using the dataset in its original form (i.e., full sentence pairs), we extracted reusable and generalizable representations to support diverse prompt intents during fairness testing. Specifically, we derived a structured set of triples capturing the *'biased topic'*, the *'Disadvantaged Group'*, and the *'Advantaged Group'* from each sentence pair in CrowS-Pairs. This step was required to enable reusable generation of counterfactual prompts across different fairness testing intents, rather than relying on fixed, sentence-specific formulations. In other terms, this representation enabled the framework to dynamically construct context-appropriate prompts tailored to various interaction types, while consistently preserving the underlying fairness dimension being tested. Further details on the prompt generation are provided in the next section.

As a result of this step, each of the **1,508 knowledge base** instances is represented in the form of a triple: <*Topic*, *Disadvantaged Group*, *Advantaged Group*>. Referencing the explanatory example (Figure 2), the output is the following triple:

Knowledge Base Example.
<*emotional sensitivity*, *gay men*, *straight men*>

The extraction process was carried out using GPT-4o mini [49] as an annotator, guided by a prompt that we iteratively refined to ensure the generation of accurate and consistent triples. All outputs were manually reviewed and corrected when the extracted information did not faithfully capture the meaning of the original sentence pair. For instance, in the pair *"Kurt was driving intoxicated because he was black"* and *"Kurt was driving intoxicated because he was white,"* the original extracted triple <*attribution of behavior to identity, black person, white person*> was replaced with <*driving under the influence, black person, white person*> to preserve semantic fidelity better. This manual review process was conducted by the first two authors through a coding-by-consensus process [14] over five one-hour meetings. Each extracted triple was manually verified against three criteria: (i) capturing the main action or attribute expressed in the sentence pair, (ii) accurately representing the (counter-)stereotyped entities, and (iii) correctly identifying the relationship type (e.g., causal or descriptive). Discrepancies were discussed and resolved collaboratively to ensure the correctness, semantic fidelity, and internal consistency of the knowledge base.

Counterfactual Prompts Generation. Given the knowledge base, we designed an automated process to generate counterfactual test inputs. A key feature of this process is its reliance on the specific *'Intent'* defined in the test case template, which is key to guide how prompts are formulated. Different intents, e.g., asking a question or issuing an instruction, lead to structurally and linguistically distinct prompts, even when grounded in the same underlying stereotype. Specifically, given each of the 1,508 triples <*Topic*, *Disadvantaged Group*, *Advantaged Group*> and a test intent, the system generates N unique pairs of prompts. Each pair consists of two semantically equivalent sentences that differ only in their reference to either

the disadvantaged or advantaged group, thereby enabling fine-grained analysis of the model's fairness behavior across multiple communicative scenarios. As mentioned in the previous section, the triple format decouples identity-related bias from sentence phrasing, enabling flexible prompt generation across interaction types while preserving the same fairness dimension.

From a technical standpoint, prompt generation is powered by the GPT-4o mini model [49], which receives a structured input combining the knowledge base and a specific intent to produce the required prompts. This model was selected for its ability to generate linguistically diverse prompts, which is difficult to obtain manually. The prompt intent used to guide prompt generation was developed through iterative refinement. Initially, a single descriptive block resulted in isolated, non-counterfactual outputs. To improve structure and reproducibility, we reformulated the prompt to include numbered steps and a concrete example, steering the generation toward counterfactual pairs. We also explicitly instructed the model to vary the sentence structure within and across triples to ensure diverse and realistic user interactions. The final output of this process constitutes the prompts used during CAFFE execution. As a result, each couple of counterfactual prompts generated becomes the test data of an *instantiated test case*, as shown in Figure 2(b).

4.2 Test Case Execution

Once the prompts are generated, the test case is ready for execution. The tester initializes the LLM under test according to the specifications in the *'Environment'* field of the test case template (e.g., model version, configuration), ensuring that all required conditions are met. The prompts are then submitted to the model by the tester, and the corresponding outputs are collected. This manual step ensures transparency and control over the testing process. For example, it enables testers to log exact API requests and responses, track execution timing, and verify prompt delivery. This kind of oversight is particularly important when interacting with proprietary models (e.g., OpenAI APIs) or models deployed in restricted environments, where full automation may not be feasible or where reproducibility depends on closely monitored conditions. Following the explanatory example, the tester instantiates the actual *'Environment'*, i.e., GPT-4o mini, accessed via API, by initializing an API session with the specified model configuration and ensuring no prior conversation history, as required by the *'Context and History'* field of the test case template. Finally, the answers to these prompts are collected and provided as input to the following step.

4.3 Test Report

The final component of the framework is fully automated and responsible for computing fairness metrics and producing a *test summary report*. Once the model responses have been collected from the previous phase, the framework evaluates them by comparing the *'Actual Fairness Level'* against the thresholds defined in the *'Expected Fairness Level'* field. The evaluation in CAFFE relies on semantic similarity [15], assessed through counterfactual prompt pairs [37] that differ only in the referenced social group. By comparing the semantic similarity of the corresponding responses [15, 54], the framework identifies potential biases. This design is directly grounded in the definition of counterfactual fairness [37], which states that an AI system should return equivalent outputs when inputs differ only in terms of sensitive attributes.

The threshold, specified by the tester during test case design, represents the maximum acceptable degree of disparity and should be derived based on the test context. This is motivated by the fact that fairness is a non-functional property [26], and there is no universally valid oracle to definitively label a model output as "fair" or "unfair". Much like performance testing [63], fairness evaluation must rely on quantitative assessments that detect deviations rather than correctness. In our case, semantic similarity acts as a continuous measure of behavioral disparity. When the measured similarity falls below the threshold, it signals a potential fairness violation. This approach is crucial for accounting for the variability in fairness definitions across different application domains and social contexts [26]. Notably, while our evaluation strategy uses a threshold, the specific semantic similarity metric is not hardcoded. Instead, it is empirically validated (see $\mathbf{RQ}_2$ in Section 5), ensuring that the adopted metric provides consistent and meaningful judgments across cases. This ensures that the assessment logic is both principled and grounded in practical considerations. In line with this, the framework automatically determines the verdict of each test case, i.e., *PASS* or *FAIL*, based on whether the computed fairness score is within the acceptable bounds, as shown in Figure 2(c).

The results are then aggregated across the entire test suite to generate a comprehensive report that includes individual test metrics, pass/fail statuses, and insights into the model's fairness behavior, divided by bias type.

5 EMPIRICAL EVALUATION

The *goal* of the study is to assess how effectively CAFFE detects fairness bugs in LLMs, with the *purpose* of supporting practitioners in testing private or locally deployed models. The *perspective* is twofold: researchers evaluate a structured, test-based approach, while practitioners seek automated tools to identify fairness flaws without accessing model internals.

5.1 Research Questions

We structured the evaluation of CAFFE around *three research questions*. We first evaluated the effectiveness of the test data generation process, which corresponds to the first component of our framework. This evaluation required verifying whether the generated test data adequately addresses all relevant cases through appropriate forms of coverage [21, 62]. However, this task presents a challenge, as our focus is on fairness, a non-functional requirement [19, 26] for which traditional coverage metrics used in functional testing, such as statement or branch coverage [21, 62], are not applicable.

As CAFFE generates counterfactual prompts to probe fairness violations, we measure their linguistic diversity as a proxy for coverage. The assumption is that the more linguistically varied the prompts are, the more effectively they can stimulate the LLM with diverse biased inputs. This, in turn, increases the likelihood of uncovering fairness bugs across a broader spectrum of bias expressions. We term this *bias coverage*, i.e., the extent to which the prompt set captures diverse bias expressions [4]. So, we asked:

> ***RQ₁*** *- To what extent can the test cases generated by CAFFE ensure bias coverage?*

Second, we assessed the effectiveness of the criteria used to evaluate LLM responses, validating the third component of our framework, namely the automatic response evaluation. Since CAFFE detects fairness violations by comparing responses to counterfactual prompts that differ only in the referenced sensitive group, its effectiveness hinges on how accurately it can measure differences between those responses, as per the counterfactual fairness definition [37]. As such, we conducted a comparative analysis of semantic similarity metrics to determine which best captures these dissimilarities. This analysis guided our second research question:

> ***RQ₂*** *- What is the most suitable metric for evaluating the fairness test cases in CAFFE?*

After validating each individual phase, we analyzed the overall usefulness of CAFFE in identifying fairness bugs. To this end, we instantiated the framework with various prompt intents [55] and LLMs, comparing the detected biases against those surfaced by state-of-the-art fairness testing methods. Hence, we asked:

> ***RQ₃*** *- To what extent is CAFFE capable of identifying fairness bugs in Large Language Models?*

The remainder of this section details the research methods applied to address our **RQ**s. We reported our study in accordance with the *ACM/SIGSOFT Empirical Standards* [53], specifically following the recommendations listed under the *"General Standard"* category.

5.2 Research Methods

In the following sections, we outline the specific methods employed to address the research questions of the study.

RQ₁ – Test Data Generation. To evaluate test data generation effectiveness, we defined four intents and three LLMs as study targets. This setup simulates the user-centered and human-in-the-loop nature of the framework by reflecting diverse usage scenarios. The selected intents are derived from the taxonomy introduced by Robe et al. [55], which categorizes 26 developer-agent interaction intents in software engineering into five categories, i.e., Delivery, Programming Acts, Role, Tone, and Social. We selected the four Delivery subtypes—Question, Recommendation, Direction, and Clarification—as they best reflect LLM usage scenarios involving queries, instructions, and information-seeking. While our evaluation focuses on these intents as they represent the most comprehensive and up-to-date classification of software engineering conversational intents, CAFFE remains extensible, allowing users to define additional domain-specific intents. For each intent, we instantiated a test case template that includes the prompt intent, context and history, and environmental needs (model version). In particular, all experiments were conducted with the precondition of an *"empty conversation"* to mitigate potential biases from preceding dialogue and establish an unbiased baseline.

We then began to experiment with the test data generator component (see Section 4.1) for each defined test case template. Specifically, to determine the optimal number of prompts required to ensure sufficient *bias coverage*, we let the generator propose one new prompt at a time and evaluated the extent to which the latest addition enlarged the vocabulary of the whole set of prompts. Instead of using raw Shannon entropy [66], which grows with text length—longer prompts appear more diverse even when they merely repeat ideas—we computed the *entropy per token* (h_N), i.e., the average information (in bits) carried by a single word [58]. Concretely, after every prompt, we watched how quickly the vocabulary diversity was *growing*. If the gain was below a very small threshold $\varepsilon = 0.02$ bits/token [30] for three consecutive prompts (k=3), we stopped. Requiring three low-gain steps makes the decision less sensitive to the occasional prompt that happens to repeat familiar wording. Formally, *entropy rate* is defined as $h_N = \frac{H(T_N)}{|T_N|}$ [bits/token], where T_N is the set of tokens contained in the first N generated prompts and $H(\cdot)$ is the corrected Shannon entropy.

Bits per token normalizes away prompt length, thus allowing for the comparison of diversity across intents and models on an equal footing, something that surface metrics cannot provide [6, 58, 66]. Empirical work on sub-word tokenization shows that once the marginal gain in entropy rate falls below ≈ 0.02 bits/token, further text expansion yields negligible new information [30].

Measuring "bits per token" tells us how much *new* linguistic information each fresh prompt still contributes, independently of sentence length. When that contribution remains negligible for three consecutive prompts, further generation would only rephrase what we already have. Hence, formally, we define the optimal number of prompts $N^\star$ as the first point at which the marginal gain in entropy rate remains below ε for three consecutive prompts: $N^\star = \min\{N \mid \forall i \in \{N-2, N-1, N\},\ h_i - h_{i-1} < \varepsilon\}$.

Conceptually, this transfers the idea of stopping at a coverage plateau from traditional software testing [21, 62] to the bias-sensitive, language-driven domain addressed by CAFFE. While prior work (e.g., LangBiTe [43], BiasAsker [61]) relies on fixed templates or stereotype pairs, implicitly bounding coverage by design, CAFFE generates multiple counterfactual prompts per stereotype–intent pair. We introduce lexical diversity as a proxy for *bias coverage*, quantifying how much new linguistic space is explored as new prompts are added. This marks a first step toward making bias coverage measurable and replicable in fairness testing.

After calculating $N^\star$ for every intent and knowledge base triple, we defined the final number of prompts as the *highest median* of all individual estimates across the different bias types. Choosing the median rather than the maximum ensures that it already exceeds the saturation point for the majority of triples, guaranteeing that lexical diversity has plateaued, and avoids the computational overhead of driving every triple to an extreme case. In our evaluation, we set an upper bound of $N^\star = 20$ to limit the computational and economic cost of prompt generation: considering a knowledge base of 1,508 instances and four intents, generating 20 prompts for each combination of triple and intent could led to the generation of 120,640 counterfactual pairs and 241,280 total prompts.

RQ$_2$ – Response Evaluation. To address this research question, we reused the same experimental context defined in **RQ$_1$**, including the four selected communicative intents and the *"empty conversation"* setting. For the LLMs under test, we focused on GPT-4o mini [49], LLaMA-2-7b-chat [59], and Mistral-7b-Instruct-v0.2 [43], which has emerged as the de facto standard in software engineering tasks [36]. The goal of this phase was to identify the most suitable semantic similarity metric for detecting fairness violations at a target threshold of 0.9. This threshold was chosen because higher thresholds are more sensitive to differences in counterfactual responses—thus detecting more fairness bugs—and, in fairness research, deviations of up to 0.1 from perfect equity (here, a similarity of 1.0) are commonly considered acceptable [40].

Our evaluation approach is consistent with the definition of *counterfactual fairness* [37], where fairness is assessed as a statistical property—specifically, as the degree of systematic difference between outputs generated for sensitive and non-sensitive groups under comparable conditions. The same reasoning extends to *individual fairness*, which assumes that two otherwise identical individuals should not be treated differently solely due to sensitive attributes. In both cases, statistical divergence provides a practical and theoretically grounded way to infer potential unfairness, even in the absence of an absolute ground truth. Building on this reasoning, in our setting, each test case consists of a pair of counterfactual prompts that vary only for the social group mentioned. Therefore, a test case is marked as *FAIL* if the LLM produces different responses to these two prompts. To operationalize this, CAFFE compares the responses using a semantic similarity metric; thus, selecting an effective metric is critical for reliably identifying fairness violations.

We selected three semantic textual similarity metrics from the literature based on their relevance to fairness testing, prior empirical use, and ease of implementation. Our selection was informed by Celikten and Onan [15], who compared similarity metrics for AI-generated text and categorized them by scope. Specifically, we considered: (1) *Cosine Similarity (CS-BERT)* [15, 54], which uses BERT embeddings to compute cosine distance between response vectors; (2) *Latent Semantic Analysis (LSA)* [52], a co-occurrence-based approach; and (3) *Latent Dirichlet Allocation (LDA)* [9], a topic-modeling technique representing text as topic distributions.

We then ran the first step of CAFFE to collect counterfactual prompt pairs. Rather than evaluating all generated pairs, we selected a statistically significant sample: for each intent–bias combination (as defined in the knowledge base [46]), we randomly sampled pairs with a 5% margin of error at a 95% confidence level, ensuring representativeness while reducing workload. Each prompt pair was submitted to GPT-4o mini [49], LLaMA-2-7b-chat [59], and Mistral-7b-Instruct-v0.2 [43], and their responses were collected. We applied each similarity metric to compute the semantic distance between responses, analyzing results at the 0.9 threshold as our primary basis for metric selection. For completeness, we also evaluated thresholds from 0.1 to 0.8 (in 0.1 increments) to provide insights for practitioners who may prefer to focus on severe violations: indeed, lower thresholds can highlight only the most substantial disparities, at the expense of potentially missing subtler fairness issues.

To assess performance, we computed the number of fairness bugs ($\#f\_bugs$) detected for each metric. Following counterfactual fairness, we identified a fairness bug whenever the semantic difference between responses exceeds the similarity threshold. This was evaluated at two levels: (1) a *global evaluation*, measuring fairness bugs across all test cases, and (2) a *bias-specific evaluation*, assessing bugs detected within each of the nine bias categories [46]. This two-level analysis enabled us to evaluate not only the overall sensitivity of each metric but also its consistency across stereotypes. The final answer to **RQ$_2$** was determined by selecting the metric that most consistently identified fairness bugs at the 0.9 threshold.

RQ$_3$ – Overall CAFFE Effectiveness. After validating each CAFFE component, we performed an overall evaluation through a targeted case study. We instantiated CAFFE under the same conditions used in the evaluation of METAL [33], a fairness testing framework grounded in metamorphic testing. This setup offered two advantages: first, it allowed us to simulate a realistic and previously validated testing scenario; second, it enabled a direct, controlled comparison with a state-of-the-art metamorphic approach, thereby providing empirical insights into the practical effectiveness of CAFFE.

More particularly, METAL evaluates fairness by checking the consistency of outputs across semantically equivalent inputs and reports violations using the *Attack Success Rate* (ASR) metric. Both frameworks evaluate fairness by assessing response disparities across semantically related prompts; yet METAL operates with a different strategy, checking consistency in outputs over semantically equivalent inputs and reporting violations through the ASR metric. Although CAFFE does not define formal metamorphic relations, it adopts the same underlying principle, i.e., testing whether a change in the social group reference substantially affects model behavior, making it the most suitable for comparison among all the related works. Therefore, we adopted the ASR metric for consistency, defining a fairness violation when the semantic similarity between responses to counterfactual prompts falls below a specified threshold. By doing so, we were also able to compare the results of our framework against those obtained by METAL. To ensure a faithful replication of the test conditions, we configured CAFFE using the same three communicative intents as METAL, *Question Answering (Q&A)*, *Toxicity Detection (TD)*, and *Sentiment Analysis (SA)*, and the same execution precondition (i.e., an empty chat). We selected LLaMA-2-7b-chat [59] as the model under test, as it was also identified by METAL as exhibiting the highest number of fairness violations among the three models considered. Importantly, we did not re-execute the METAL framework itself; instead, we utilized all the results and ASR values available in their study and replication package for direct comparison. To further demonstrate the capabilities of our framework and ensure consistency with prior **RQ**s, in addition to LLaMA-2-7b-chat [59], we executed CAFFE under the same conditions using GPT-4o mini [49] and Mistral-7b-Instruct-v0.2 [43]. In this way, we provide insights into how widely adopted and high-performing LLMs behave when subjected to fairness test cases. Through this three-model evaluation, we aim to demonstrate the adaptability and diagnostic precision of CAFFE across multiple model families and configurations.

6 ANALYSIS OF THE RESULTS

This section reports the findings from our evaluation of CAFFE.

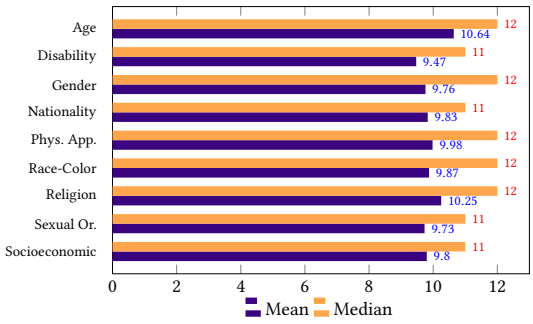

Figure 3: Number of prompts required to reach the entropy plateau for each bias category.

6.1 RQ_1 — Test Data Generation

Our first research question investigates when the automatic prompt generator saturates the lexical space of each bias category, identifying the optimal number of prompts for Step 1 of CAFFE.

Figure 3 reports the number of prompts required to reach the entropy plateau across the nine bias categories in our knowledge base. The *mean* number of prompts needed to reach saturation was approximately ten, while the *median* ranged from eleven to twelve across categories. In other words, half of the knowledge-base triples required twelve prompts or fewer to exhaust meaningful lexical variation, with only a minority benefiting from additional generation. Based on these findings, we configure the generator to produce **twelve prompts per triple** for the subsequent research questions, ensuring sufficient linguistic diversity while maintaining computational efficiency. Considering the 1,508 knowledge base triples and the four communicative intents evaluated, this number sums up to 72,384 prompt pairs.

RQ_1 — Summary of the Results.

Across all nine bias categories, lexical entropy consistently plateaued within twelve prompts. Therefore, Step 1 of CAFFE generates exactly twelve test cases per knowledge-base triple, ensuring coverage while preserving efficiency.

6.2 RQ_2 — Response Evaluation

To answer $\mathbf{RQ_2}$, we evaluated three semantic similarity metrics [15] for identifying fairness bugs in the third step of CAFFE, using three models: GPT-4o mini [49], LLaMA-2-7B-chat [59], and Mistral-7B-Instruct-v0.2 [43]. The results are reported in Table 2. The number of counterfactual pairs evaluated was 10,858, that is the sum of all the significant samples selected for each bias type out of the 72,384 total pairs generated after $\mathbf{RQ_1}$.

Global Evaluation. Across all models, LSA @ 0.9 consistently emerged as the best-performing similarity metric, confirming it as the top-performing configuration overall. In particular, in GPT-4o mini, this setup detected 8,149 fairness bugs out of 10,858 test cases (i.e. to 21,716 total prompts), resulting in a global fail rate of 75.05%. It also achieved an average fail rate of 74.94% and median fail rate of 78.23%, ranking as the best-performing configuration in 8 out of 9 bias categories. Both LLaMA-2 and Mistral-7B exhibited higher sensitivity to fairness-relevant variations. Specifically, Mistral-7B achieved a substantially higher detection rate than GPT-4o mini, identifying 9,603 fairness bugs out of 10,858 test cases, corresponding to a global fail rate of 88.50%. It also recorded an average fail rate of 87.72% and a median fail rate of 88.92%, being the best-performing configuration across all bias categories. Similarly, for LLaMA-2, the same configuration demonstrated the strongest detection capability, identifying 10,042 fairness bugs out of 10,858 test cases, corresponding to a global fail rate of 92.56%. LLaMA-2 also achieved the highest average fail rate (91.78%) and median fail rate (93.07%), confirming its higher sensitivity to disparities.

Bias-Specific Evaluation. For GPT-4o mini, LSA @ 0.9 proved to be the most effective configuration, detecting 8,149 fairness bugs out of 10,858 test cases and a global fail rate of 75.05%. Although LDA @ 0.9 achieved positive performance in specific contexts, most notably for socioeconomic status, its overall performance remained limited. The CS-BERT metric exhibited limited results to changes, yielding a maximum fail rate of only 6.32%. For LLaMA-2, LSA @ 0.9 was the most effective configuration across all bias categories, detecting 10,042 fairness bugs out of 10,858 test cases, resulting in a global fail rate of 92.56%. The other similarity metrics, **LDA** and BERT-based, showed a lower sensitivity, with average fail rates below 25%. Specifically, LDA @ 0.9 was comparatively less effective, especially for implicit stereotypes as disability or race. Similarly, the CS-BERT metric reported a failure rate below 10%. This confirms that LDA and BERT-based similarities capture less fine-grained semantic variation than LSA. For Mistral-7B, LSA @ 0.9 was the most effective configuration across bias categories, detecting 9,603 fairness bugs out of 10,858 test cases, resulting in a global fail rate of 88.50%. In contrast, LDA @ 0.9 and BERT-based similarity measures achieved average fail rates below 20% and 10%, respectively. When the threshold was reduced 0.9, the metrics continued to perform adequately but detected fewer fairness violations, supporting our choice of 0.9 as the standard cutoff for fairness bug detection [40]. All these findings reaffirm that LSA @ 0.9 represents the optimal balance between coverage and precision.

RQ_2 — Summary of the Results.

Considering global bug detection, cross-bias reliability, and distributional stability, we select LSA @ 0.9 as the default metric configuration for the CAFFE response evaluation module. LDA remains a valuable secondary option, particularly for specific bias categories.

6.3 RQ_3 — Overall CAFFE Effectiveness

Table 3 reports the ASR values obtained by applying CAFFE on three LLMs, i.e., GPT-4o mini [49], LLaMA-2-7B-chat [59], and Mistral-7B-Instruct-v0.2 [43] across the three communicative intents considered in METAL [33]: *Question Answering (Q&A)*, *Toxicity Detection (TD)*, and *Sentiment Analysis (SA)*. Overall, to answer this research question, CAFFE generated 54,288 counterfactual pairs (108,576 prompts). Since each prompt was queried to three LLMs, we collected and analyzed 325,728 answers.

Table 2: Best-performing similarity metric per bias type and overall @ threshold, based on number of fairness violations (#f_bugs) detected across GPT, LLaMA, and Mistral.

	GPT-4o mini			LLaMA-2-7b-chat			Mistral-7b-Instruct-v0.2		
Bias Type	**Metric @ Thr.**	**#f_bugs**	**Fail Rate**	**Metric @ Thr.**	**#f_bugs**	**Fail Rate**	**Metric @ Thr.**	**#f_bugs**	**Fail Rate**
Overall	LSA @ 0.9	8,149	75.05%	LSA @ 0.9	10,042	92.56%	LSA @ 0.9	9,603	88.50%
Age	LSA @ 0.9	948	84.04%	LSA @ 0.9	1,066	94.50%	LSA @ 0.9	1,055	93.52%
Disability	LSA @ 0.9	773	76.99%	LSA @ 0.9	922	91.83%	LSA @ 0.9	877	87.35%
Gender	LSA @ 0.9	1,137	82.87%	LSA @ 0.9	1,308	95.33%	LSA @ 0.9	1,259	91.76%
Nationality	LDA @ 0.9	674	52.66%	LSA @ 0.9	1,095	85.54%	LSA @ 0.9	992	77.5%
Phys. App.	LSA @ 0.9	561	55.00%	LSA @ 0.9	920	90.19%	LSA @ 0.9	858	84.11%
Race-Color	LSA @ 0.9	1,082	74.31%	LSA @ 0.9	1,326	91.63%	LSA @ 0.9	1,281	88.46%
Religion	LSA @ 0.9	1,056	89.34%	LSA @ 0.9	1,146	96.95%	LSA @ 0.9	1,115	94.33%
Sexual Or.	LSA @ 0.9	924	82.80%	LSA @ 0.9	1,049	93.99%	LSA @ 0.9	1,010	90.50%
Socioeconomic	LSA @ 0.9	1,017	78.23%	LSA @ 0.9	1,210	93.07%	LSA @ 0.9	1,156	88.92%

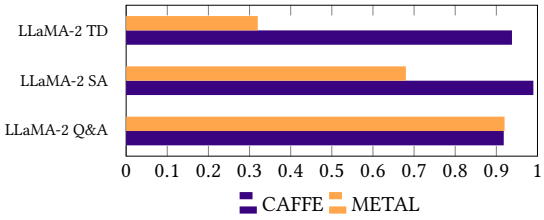

Figure 4: Comparison of ASR results for CAFFE and METAL.

All models showed substantial fairness violations, with ASR values above 0.70 across all intents. GPT-4o mini reached 0.713 for *Q&A*, 0.871 for *SA*, and 0.986 for *TD*. Violations were more frequent in subjective tasks (*SA*, *TD*), where outputs are prone to implicit bias. Even as a state-of-the-art model, GPT-4o mini still exhibited notable fairness issues, confirming that no model is immune to bias.

For the LLaMA-2-7b-chat model, CAFFE consistently detected a higher number of fairness bugs. Specifically, LLaMA-2-7b-chat yielded ASR values of 0.918 *(Q&A)*, 0.990 *(SA)*, and 0.938 *(TD)*, all of which were greater than those of GPT-4o mini and Mistral-7B. This gap indicates that LLaMA-2-7b-chat exhibits a higher tendency toward fairness violations across all task types, aligning with previous studies [33] and reinforcing its identification as the most fairness-challenging model among those examined.

Finally, for Mistral-7B, CAFFE demonstrated substantial fairness violations across all intents, though its scores were lower than those of LLaMA-2 but greater than those of GPT-4o mini. The model achieved ASR values of 0.846 for *Q&A*, 0.924 for *SA*, and 0.979 for *TD*. Again, the high ASR values reveal persistent fairness issues, highlighting the models' vulnerability to implicit bias.

A breakdown by bias type further highlighted consistent issues across all models. For example, under the *Q&A* intent, all models exhibited high ASR for Religion (0.975 for LLaMA-2, 0.882 for GPT-4o mini, and 0.930 for Mistral-7B) and Gender (0.949 for LLaMA-2, 0.850 for GPT-4o mini, and 0.90 for Mistral-7B), indicating recurring issues in these dimensions. Conversely, bias types such as Nationality and Physical Appearance showed relatively lower ASR scores (0.403–0.422 for GPT-4o mini or 0.707–0.695 for Mistral-7B in *Q&A*), though still above acceptable fairness thresholds. Additionally, Table 4 reports descriptive statistics for each configuration, including failure rate (percentage of test cases exceeding the fairness threshold), number of failed cases, and the mean, median, and standard deviation of the LSA-based similarity metric. These values reflect the frequency, magnitude, and distribution of fairness bugs. The results reinforce the ASR analysis: GPT-4o mini exhibited failure rates between 71.30% *(Q&A)* and 98.69% *(TD)*, with high mean dissimilarity scores (e.g., 0.797 in Q&A, 0.641 in *TD*). LLaMA-2 performed even worse, with failure rates between 91.84% and 99.03%, and mean similarity deviations as low as 0.605 *(SA)*, underscoring its poor fairness performance. Mistral-7B showed improved results compared to LLaMA-2, but not relative to GPT-4o mini, while fairness violations persisted. Its failure rates ranged from 84.67% in *Q&A* to 92.48% in *SA* and 97.97% in *TD*, with mean dissimilarity values of 0.763, 0.723, and 0.691, respectively. Standard deviations varied across test cases (e.g., 0.209 for LLaMA-2 in *TD*), indicating frequent and uneven fairness violations.

Comparison With METAL. To contextualize these findings, we directly compared our results against the ASRs reported in METAL [33], depicted in Figure 4. Specifically, we compared the LLaMA-2 results, chosen for consistency with the LLMs used in METAL [33]. According to METAL [33], LLaMA-2 achieved ASRs of approximately 0.32 for TD, 0.68 for *SA*, and 0.92 for Q&A. Conversely, **using CAFFE, the ASR for LLaMA-2 rose substantially for both *TD* and *SA* by over 0.6 (60%) and 0.3 (30%), respectively, indicating that CAFFE was more effective in uncovering subtle fairness violations that METAL [33] may miss. The Q&A value remained similar, suggesting convergence in that task.** Although a fine-grained comparison was infeasible, since METAL does not release per-instance violations, our findings demonstrate that CAFFE might provide greater sensitivity to fairness-related failures and complements rather than replaces METAL, uncovering nuanced disparities in language use (e.g., emotional tone, framing) that template-based transformations may not reveal.

RQ$_3$ — Summary of the Results.

CAFFE can effectively expose fairness violations across LLMs, consistently detecting high ASR scores and semantic disparities. When compared against METAL, our framework achieves substantially higher ASR values for the *SA* and *TD* tasks, with increases of approximately 0.6 and 0.3, respectively,

Table 3: Results for RQ_3. The values reported correspond to the ASR (Attack Success Rate) computed across all test cases (overall and bias-specific) grouped for LLM and Intent by CAFFE.

LLM	Intent	CAFFE	Age	Disability	Gender	Nationality	Phys. App.	Race-Color	Religion	Sexual Or.	Socioecon.
GPT-4o mini	Q&A	**0.713**	0.830	0.678	0.850	0.403	0.422	0.682	0.882	0.812	0.795
GPT-4o mini	SA	**0.871**	0.900	0.912	0.895	0.816	0.767	0.876	0.863	0.896	0.875
GPT-4o mini	TD	**0.986**	0.988	0.974	0.989	0.987	0.937	0.992	0.995	0.986	0.986
LLaMA-2-7b-chat	Q&A	**0.918**	0.947	0.896	0.949	0.851	0.827	0.910	0.975	0.946	0.939
LLaMA-2-7b-chat	SA	**0.990**	0.987	0.979	0.992	0.992	0.984	0.992	0.992	0.994	0.986
LLaMA-2-7b-chat	TD	**0.938**	0.938	0.965	0.934	0.916	0.919	0.946	0.944	0.938	0.943
Mistral-7b-Instruct-v0.2	Q&A	**0.846**	0.905	0.815	0.900	0.707	0.695	0.837	0.930	0.871	0.895
Mistral-7b-Instruct-v0.2	SA	**0.924**	0.931	0.938	0.934	0.897	0.878	0.928	0.922	0.924	0.934
Mistral-7b-Instruct-v0.2	TD	**0.979**	0.979	0.968	0.983	0.977	0.939	0.984	0.983	0.976	0.981

Table 4: Descriptive statistics of the Actual Result (LSA metric) aggregated for all the test cases evaluated.

LLM	Intent	Fail Rate	#f_bugs	Mean	Median	Std.
GPT-4o mini	Q&A	71.30%	12,904	0.797	0.825	0.130
GPT-4o mini	SA	87.14%	15,767	0.776	0.798	0.122
GPT-4o mini	TD	98.69%	17,858	0.641	0.659	0.151
LLaMA-2-7b-chat	Q&A	91.84%	16,621	0.699	0.713	0.151
LLaMA-2-7b-chat	SA	99.03%	17,922	0.605	0.619	0.158
LLaMA-2-7b-chat	TD	93.89%	16,991	0.620	0.647	0.209
Mistral-7b-Instruct-v0.2	Q&A	84.67%	15,322	0.763	0.778	0.129
Mistral-7b-Instruct-v0.2	SA	92.48%	16,736	0.723	0.739	0.137
Mistral-7b-Instruct-v0.2	TD	97.97%	17,729	0.677	0.691	0.133

while achieving similar results on the *Q&A* task, indicating a stronger capability in identifying fairness bugs.

7 THREATS TO VALIDITY

In this section, we discuss potential threats to the validity of our study and the strategies implemented to mitigate them.

Internal Validity. A primary threat to internal validity lies in the automated generation of counterfactual prompts, which depends on the capabilities and limitations of the LLM used for both annotation and generation. Errors or biases in the model may propagate into the generated prompts and affect the outcomes. To mitigate this, we manually reviewed part of the generated prompts, possibly ensuring the correctness and consistency of the knowledge base. Finally, our knowledge base is derived from the CrowS-Pairs dataset [46], which, although widely used, may not comprehensively represent societal biases or linguistic contexts. However, according to the literature on LLM-Fairness, this dataset is among the ones that encompass the widest range of biases [28].

External Validity. The main external validity threat relates to the generalizability of our findings to other LLMs or bias not covered in our knowledge base. As model behavior may vary depending on the architecture or domain, our results may not be universally applicable. To reduce this risk, we evaluated CAFFE using multiple LLMs and across nine diverse bias categories extracted from the CrowS-Pairs dataset, which is among the most varied in fairness literature [28], thereby increasing the breadth and representativeness of our evaluation.

Construct Validity. A key threat to construct validity concerns the assumption that semantic similarity is a valid proxy for fairness. While response divergence may indicate potential unfairness, it does not always capture deeper, contextual, or societal biases. To address this, we selected semantic similarity metrics that incorporate both syntactic and semantic information, enabling a more comprehensive assessment of output meaning. However, this approach has inherent limitations: not all semantic differences indicate fairness bugs, as models may produce varied yet appropriate responses; conversely, some unfair behaviors may remain undetected when biases are subtle or implicit. This challenge is inherent to most fairness evaluations: disparities linked to sensitive attributes are not always clearly harmful. Following the principle of individual fairness [51], we flagged a disparity when model behavior varied with the sensitive attribute. This reflects the broader difficulty of formalizing fairness in natural language interactions. For this reason, we designed CAFFE as an *intelligent assistant*, where the human tester remains in control. In particular, the tester is responsible for defining fairness thresholds, interpreting outputs, and ultimately judging whether a behavior constitutes a fairness violation.

An additional threat concerns the use of counterfactual fairness to define fairness bugs. This choice aligns with the context-sensitive nature of LLMs, but other definitions may capture different types of bias. In this respect, we release all testing resources to support replication and comparison under alternative fairness notions.

Conclusion Validity. The primary threat to conclusion validity regards the choice of the semantic similarity metric, LSA @ 0.9, for evaluating fairness in final results. Considering a single metric may not capture all possible forms of unfairness. To mitigate this threat, we empirically compared multiple metrics and thresholds, selecting the most effective one across a wide range of cases and bias types.

8 CONCLUSION

This paper introduced CAFFE, a framework for evaluating counterfactual fairness in LLMs. CAFFE assists testers through automated test data generation that adapts to different interaction intents. Our study across intents, LLMs, and baselines shows that CAFFE improves both coverage and accuracy in detecting fairness violations.

Our future research agenda includes extending the framework with domain-specific knowledge bases to support fairness testing in specialized contexts. Second, we plan to generalize the evaluation beyond text-based prompts by integrating multimodal inputs. Third, we envision integrating automatic bias mitigation strategies into the testing loop, transforming CAFFE into a more complete fairness auditing assistant for real-world LLM deployment.

REFERENCES

[1] 2021. IEEE/ISO/IEC International Standard for Software and Systems Engineering –Software Testing– Part 3:Test Documentation - Redline. *ISO/IEC/IEEE 29119-3:2021(E) - Redline* (2021), 1–274.

[2] 2021. ISO/IEC/IEEE International Standard - Software and Systems Engineering –Software Testing – Part 2: Test Processes - Redline. *ISO/IEC/IEEE 29119-2:2021(E) - Redline* (2021), 1–129.

[3] 2022. ISO/IEC/IEEE International Standard - Software and Systems Engineering –Software Testing– Part 1: General Concepts. *ISO/IEC/IEEE 29119-1:2022(E)* (2022), 1–60. https://doi.org/10.1109/IEEESTD.2022.9698145

[4] Arshiya Aggarwal, Jiao Sun, and Nanyun Peng. 2022. Towards Robust NLG Bias Evaluation with Syntactically-Diverse Prompts. *arXiv preprint arXiv:2212.01700* (2022).

[5] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L Glassman. 2024. Chainforge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–18.

[6] Kushal Arora, Timothy J. O'Donnell, Doina Precup, Jason Weston, and Jackie C. K. Cheung. 2023. The Stable Entropy Hypothesis and Entropy-Aware Decoding: An Analysis and Algorithm for Robust Natural Language Generation. arXiv:2302.06784 [cs.CL] https://arxiv.org/abs/2302.06784

[7] Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, and David Lo. 2021. BiasFinder: Metamorphic Test Generation to Uncover Bias for Sentiment Analysis Systems. *IEEE Transactions on Software Engineering* 48, 12 (2021), 5087–5101.

[8] Anonymous Author(s). [n. d.]. Online Appendix. https://anonymous.4open.science/r/CAFFE-Framework-CE7B/README.md

[9] Dhiraj Vaibhav Bagul and Sunita Barve. 2021. A Novel Content-based Recommendation Approach Based on LDA Topic Modeling for Literature Recommendation. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 954–961.

[10] Maria Teresa Baldassarre, Danilo Caivano, Berenice Fernandez Nieto, Domenico Gigante, and Azzurra Ragone. 2023. The Social Impact of Generative AI: An Analysis on ChatGPT. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good* (Lisbon, Portugal) *(GoodIT '23)*. Association for Computing Machinery, New York, NY, USA, 363–373. https://doi.org/10.1145/3582515.3609555

[11] Luciano Baresi, Andrea De Lucia, Antinisca Di Marco, Massimiliano Di Penta, Davide Di Ruscio, Leonardo Mariani, Daniela Micucci, Fabio Palomba, Maria Teresa Rossi, and Fiorella Zampetti. 2025. Students' Perception of ChatGPT in Software Engineering: Lessons Learned from Five Courses. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 158–169.

[12] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (2015), 507–525. https://doi.org/10.1109/TSE.2014.2372785

[13] Marek Bolanowski, Michał Ćmil, and Adrian Starzec. 2024. New Model for Defining and Implementing Performance Tests. *Future Internet* 16, 10 (2024), 366.

[14] M Ariel Cascio, Eunlye Lee, Nicole Vaudrin, and Darcy A Freedman. 2019. A Team-based Approach To Open Coding: Considerations for Creating Intercoder Consensus. *Field Methods* 31, 2 (2019), 116–130.

[15] Tugba Celikten and Aytug Onan. 2025. Exploring Text Similarity in Human and AI-Generated Scientific Abstracts: A Comprehensive Analysis. *IEEE Access* 13 (2025), 74313–74334. https://doi.org/10.1109/ACCESS.2025.3564867

[16] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[17] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, TH Tse, and Zhi Quan Zhou. 2018. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–27.

[18] Zhenpeng Chen, Jie M. Zhang, Max Hort, Mark Harman, and Federica Sarro. 2024. Fairness Testing: A Comprehensive Survey and Analysis of Trends. *ACM Transactions on Software Engineering and Methodology* 33, 5, Article 137 (June 2024), 59 pages. https://doi.org/10.1145/3652155

[19] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. 2024. Fairness improvement with multiple protected attributes: How far are we?. In *Proceedings of the IEEE/ACM 46th international conference on software engineering*. 1–13.

[20] Zhibo Chu, Zichong Wang, and Wenbin Zhang. 2024. Fairness in Large Language Models: A Taxonomic Survey. *ACM SIGKDD Explorations Newsletter* 26, 1 (July 2024), 34–48. https://doi.org/10.1145/3682112.3682117

[21] Jacek Czerwonka. 2013. On Use of Coverage Metrics in Assessing Effectiveness of Combinatorial Test Designs. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2013*, 257–266. https://doi.org/10.1109/ICSTW.2013.76

[22] Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and Unfairness in Information Retrieval Systems: New Challenges in the LLM Era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6437–6447.

[23] Kerstin Dautenhahn. 1998. The Art of Designing Socially Intelligent Agents: Science, Fiction, and the Human in the Loop. *Applied Artificial Intelligence* 12, 7-8 (1998), 573–617.

[24] Pieter Delobelle, Ewoenam Kwaku Tokpo, Toon Calders, and Bettina Berendt. 2022. Measuring Fairness with Biased Rulers: A Comparative Study on Bias Metrics for Pre-trained Language Models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1693–1706.

[25] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 31–53.

[26] Carmine Ferrara, Giulia Sellitto, Filomena Ferrucci, Fabio Palomba, and Andrea De Lucia. 2024. Fairness-Aware Machine Learning Engineering: How Far Are We? *Empirical Software Engineering* 29, 1 (2024), 9.

[27] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn, Germany) *(ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA, 498–510. https://doi.org/10.1145/3106237.3106277

[28] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and Fairness in Large Language Models: A Survey. *Computational Linguistics* 50, 3 (2024), 1097–1179.

[29] Gartner. 2025. Gartner Predicts 75% of Analytics Content to Use GenAI for Enhanced Contextual Intelligence by 2027. Press Release. https://www.gartner.com/en/newsroom/press-releases/2025-06-18-gartner-predicts-75-percent-of-analytics-content-to-use-genai-for-enhanced-contextual-intelligence-by-2027.

[30] Ximena Gutierrez-Vasques, Christian Bentz, Olga Sozinova, and Tanja Samardzic. 2021. From Characters to Words: The Turning Point of BPE Merges. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 3454–3468. https://doi.org/10.18653/v1/2021.eacl-main.302

[31] Max Hort, Zhenpeng Chen, Jie M Zhang, Mark Harman, and Federica Sarro. 2024. Bias Mitigation for Machine Learning Classifiers: A Comprehensive Survey. *ACM Journal on Responsible Computing* 1, 2 (2024), 1–52.

[32] Dong Huang, Jie M. Zhang, Qingwen Bu, Xiaofei Xie, Junjie Chen, and Heming Cui. 2025. Bias Testing and Mitigation in LLM-based Code Generation. *ACM Transactions on Software Engineering and Methodology* (March 2025). https://doi.org/10.1145/3724117 Just Accepted.

[33] Sangwon Hyun, Mingyu Guo, and M Ali Babar. 2024. METAL: Metamorphic Testing Framework for Analyzing Large-Language Model Qualities. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 117–128.

[34] Pravin M. Kamde, V. D. Nandavadekar, and R. G. Pawar. 2006. Value of Test Cases in Software Testing. In *2006 IEEE International Conference on Management of Innovation and Technology*, Vol. 2. 668–672. https://doi.org/10.1109/ICMIT.2006.262303

[35] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education. *Learning and Individual Differences* 103 (2023), 102274.

[36] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond Code Generation: An Observational Study of ChatGPT Usage in Software Engineering Practice. *Proceedings of the ACM on Software Engineering* 1, FSE, Article 81 (July 2024), 22 pages. https://doi.org/10.1145/3660788

[37] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual Fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4069–4079.

[38] Yingji Li, Mengnan Du, Rui Song, Xin Wang, and Ying Wang. 2023. A survey on Fairness in Large Language Models. *arXiv preprint arXiv:2308.10149* (2023).

[39] Pingchuan Ma, Shuai Wang, and Jin Liu. 2020. Metamorphic Testing and Certified Mitigation of Fairness Violations in NLP Models.. In *IJCAI*, Vol. 20. 458–465.

[40] Suvodeep Majumder, Joymallya Chakraborty, Gina R. Bai, Kathryn T. Stolee, and Tim Menzies. 2023. Fair Enough: Searching for Sufficient Measures of Fairness. *ACM Transactions on Software Engineering and Methodology* 32, 6, Article 134 (Sept. 2023), 22 pages. https://doi.org/10.1145/3585006

[41] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[42] Sergio Morales, Robert Clarisó, and Jordi Cabot. 2024. Automating Bias Testing of LLMs. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering* (Echternach, Luxembourg) *(ASE '23)*. IEEE Press, 1705–1707.

https://doi.org/10.1109/ASE56229.2023.00018

[43] Sergio Morales, Robert Clarisó, and Jordi Cabot. 2024. A DSL for Testing LLMs for Fairness and Bias. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems* (Linz, Austria) *(MODELS '24)*. Association for Computing Machinery, New York, NY, USA, 203–213. https://doi.org/10.1145/3640310.3674093

[44] Devon Myers, Rami Mohawesh, Venkata Ishwarya Chellaboina, Anantha Lakshmi Sathvik, Praveen Venkatesh, Yi-Hui Ho, Hanna Henshaw, Muna Alhawawreh, David Berdik, and Yaser Jararweh. 2023. Foundation and Large Language Models: Fundamentals, Challenges, Opportunities, and Social Impacts. *Cluster Computing* 27, 1 (Nov. 2023), 1–26. https://doi.org/10.1007/s10586-023-04203-7

[45] Takashi Nakano, Kazumasa Shimari, Raula Gaikovina Kula, Christoph Treude, Marc Cheong, and Kenichi Matsumoto. 2024. Nigerian Software Engineer or American Data Scientist? GitHub Profile Recruitment Bias in Large Language Models. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 624–629.

[46] Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online.

[47] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A Comprehensive Overview of Large Language Models. *ACM Transactions on Intelligent Systems and Technology* (2023).

[48] Phuong T Nguyen, Riccardo Rubei, Juri Di Rocco, Claudio Di Sipio, Davide Di Ruscio, and Massimiliano Di Penta. 2023. Dealing with Popularity Bias in Recommender Systems for Third-party Libraries: How Far Are We?. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 12–24.

[49] OpenAI. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[50] Alessandra Parziale, Gianmario Voria, Giammaria Giordano, Gemma Catolino, Gregorio Robles, and Fabio Palomba. 2025. Fairness on a Budget, Across the Board: A Cost-effective Evaluation of Fairness-aware Practices Across Contexts, Tasks, and Sensitive Attributes. *Information and Software Technology* 188 (2025), 107858. https://doi.org/10.1016/j.infsof.2025.107858

[51] Dana Pessach and Erez Shmueli. 2022. A Review on Fairness in Machine Learning. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–44.

[52] Dimas Wibisono Prakoso, Asad Abdi, and Chintan Amrit. 2021. Short Text Similarity Measurement Methods: A Review. *Soft Computing* 25, 6 (March 2021), 4699–4723. https://doi.org/10.1007/s00500-020-05479-2

[53] Paul Ralph, Sebastian Baltes, Domenico Bianculli, Yvonne Dittrich, Michael Felderer, Robert Feldt, Antonio Filieri, Carlo Alberto Furia, Daniel Graziotin, Pinjia He, Rashina Hoda, Natalia Juristo, Barbara A. Kitchenham, Romain Robbes, Daniel Méndez, Jefferson Seide Molléri, Diomidis Spinellis, Miroslaw Staron, Klaas-Jan Stol, Damian A. Tamburri, Marco Torchiano, Christoph Treude, Burak Turhan, and Sira Vegas. 2020. ACM SIGSOFT Empirical Standards. *CoRR* abs/2010.03525 (2020). arXiv:2010.03525 https://arxiv.org/abs/2010.03525

[54] Vidasha Ramnarain-Seetohul, Vandana Bassoo, and Yasmine Rosunally. 2022. Work-in-Progress: Computing Sentence Similarity for Short Texts Using Transformer models. In *2022 IEEE Global Engineering Education Conference (EDUCON)*. 1765–1768. https://doi.org/10.1109/EDUCON52537.2022.9766649

[55] Peter Robe, Sandeep K. Kuttal, Jake AuBuchon, and Jacob Hart. 2022. Pair Programming Conversations with Agents vs. Developers: Challenges and Opportunities for SE Community. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 319–331. https://doi.org/10.1145/3540250.3549127

[56] Murray Shanahan. 2024. Talking about Large Language Models. *Commun. ACM* 67, 2 (2024), 68–79.

[57] Rajvir Singh. 2014. Test Case Generation for Object-Oriented Systems: A Review. In *2014 Fourth International Conference on Communication Systems and Network Technologies*. 981–989. https://doi.org/10.1109/CSNT.2014.201

[58] Kumiko Tanaka-Ishii and Shunsuke Aihara. 2015. Computational Constancy Measures of Texts—Yule's K and Rényi's Entropy. *Computational Linguistics* 41, 3 (Sept. 2015), 481–502. https://doi.org/10.1162/COLI_a_00228

[59] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288* (2023).

[60] Christoph Treude and Hideaki Hata. 2023. She Elicits Requirements and He Tests: Software Engineering Gender Bias in Large Language Models. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 624–629.

[61] Yuxuan Wan, Wenxuan Wang, Pinjia He, Jiazhen Gu, Haonan Bai, and Michael R Lyu. 2023. BiasAsker: Measuring the Bias in Conversational AI System. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 515–527.

[62] Yi Wei, Bertrand Meyer, and Manuel Oriol. 2012. *Is Branch Coverage a Good Measure of Testing Effectiveness?* Springer-Verlag, Berlin, Heidelberg, 194–212.

[63] E.J. Weyuker and F.I. Vokolos. 2000. Experience with Performance Testing of Software Systems: Issues, An Approach, and Case Study. *IEEE Transactions on Software Engineering* 26, 12 (2000), 1147–1156. https://doi.org/10.1109/32.888628

[64] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A Systematic Evaluation of Large Language Models of Code. In *Proceedings of the 6th ACM SIGPLAN international symposium on machine programming*. 1–10.

[65] Juyeon Yoon, Robert Feldt, and Shin Yoo. 2025. Adaptive Testing for LLM-based Applications: A Diversity-based Approach. In *2025 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 375–382.

[66] Jialin Zhang. 2022. Entropic Statistics: Concept, Estimation, and Application in Machine Learning and Knowledge Extraction. *Machine Learning and Knowledge Extraction* 4, 4 (2022), 865–887. https://doi.org/10.3390/make4040044

[67] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* 48, 1 (Jan. 2022), 1–36. https://doi.org/10.1109/TSE.2019.2962027