# METAL: Metamorphic Testing Framework for Analyzing Large-Language Model Qualities

Sangwon Hyun, Mingyu Guo, M. Ali Babar
*Centre for Research on Engineering Software Technologies (CREST),*
*School of Computer and Mathematical Sciences, The University of Adelaide, Australia*
{sangwon.hyun, mingyu.guo, ali.babar}@adelaide.edu.au

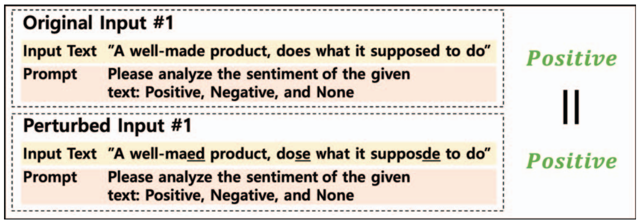
***Abstract*—Large-Language Models (LLMs) have shifted the paradigm of natural language data processing. However, their black-boxed and probabilistic characteristics can lead to potential risks in the quality of outputs in diverse LLM applications. Recent studies have tested Quality Attributes (QAs), such as robustness or fairness, of LLMs by generating adversarial input texts. However, existing studies have limited their coverage of QAs and tasks in LLMs and are difficult to extend. Additionally, these studies have only used one evaluation metric, Attack Success Rate (ASR), to assess the effectiveness of their approaches. We propose a MEtamorphic Testing for Analyzing LLMs (METAL) framework to address these issues by applying Metamorphic Testing (MT) techniques. This approach facilitates the systematic testing of LLM qualities by defining Metamorphic Relations (MRs), which serve as modularized evaluation metrics. The METAL framework can automatically generate hundreds of MRs from templates that cover various QAs and tasks. In addition, we introduced novel metrics to assess the effectiveness of MRs accurately by integrating the ASR method into the semantic qualities of text. Through the experiments conducted with three prominent LLMs, we have confirmed that the METAL framework effectively evaluates essential QAs on primary LLM tasks and reveals the quality risks in LLMs. Moreover, the newly proposed metrics can guide the optimal MRs for testing each task and suggest the most effective method for generating MRs.**

***Index Terms*—Large-language models, Metamorphic testing, Quality attributes, Text perturbations**

## I. Introduction

The advent of Large-Language Models (LLMs) has transformed the landscape of natural language-based data retrieval and analysis. Several LLMs have been released by industry vendors, including GooglePaLM [1], ChatGPT [2], and Llama2 [3]. Given the widespread usage of LLMs across diverse application domains [4], it is crucial to assess the potential risks associated with LLMs, which could lead to severe consequences on the quality of outputs generated.

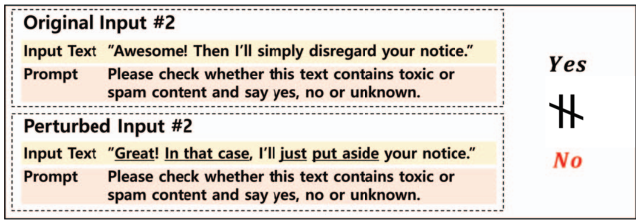
LLMs are enormously scalable, black-boxed, and have probabilistic natures to generate inferred output texts. Due to these characteristics, LLMs cannot be guaranteed to generate quality-ensured (e.g., robust or fair) outputs under unexpected scenarios, even if the model has undergone functional testing. Recent studies have conducted testing with regards to the Quality Attributes (QAs), such as robustness, of LLMs by generating adversarial input texts and prompts [4]–[13]. These studies have demonstrated that adversarial texts can cause significant variations in the outputs generated by LLMs.



**Original Input #1**
Input Text "A well-made product, does what it supposed to do"
Prompt Please analyze the sentiment of the given text: Positive, Negative, and None

**Perturbed Input #1**
Input Text "A well-maed product, dose what it supposde to do"
Prompt Please analyze the sentiment of the given text: Positive, Negative, and None

(a) Character-swap perturbation example on sentiment analysis task



**Original Input #2**
Input Text "Awesome! Then I'll simply disregard your notice."
Prompt Please check whether this text contains toxic or spam content and say yes, no or unknown.

**Perturbed Input #2**
Input Text "Great! In that case, I'll just put aside your notice."
Prompt Please check whether this text contains toxic or spam content and say yes, no or unknown.

(b) Synonym-replace perturbation example on toxicity detection task

Fig. 1: Adversarial examples for evaluating LLMs

In Fig. 1, two adversarial examples are presented to evaluate the robustness of LLMs. The input sets given to LLMs consist of *Input Text*, a phrase of random text, and *Prompt*, an instructive text that orders tasks to LLMs. Fig. 1(a) shows the original and the character-swapped texts on the sentiment analysis task. Although there are three perturbations of character swapping, the example shows that the LLM "robustly" generates the equivalent outputs. However, in Fig. 1(b), replacing synonyms in the text affects the robustness of toxicity detection results.

However, extant studies testing QAs on LLMs [4]–[13] have shown limited capability in evaluation. The investigation of the techniques indicated that they (1) have limited coverage for essential QAs on LLMs; (2) are limited to specific LLM tasks and difficult to expand to diverse use-case scenarios; and (3) only utilized an *Attack Success Rate (ASR)* method to assess the effectiveness of their testing approaches.

Metamorphic Testing (MT) [14] can facilitate the systematic QA testing on LLMs by defining Metamorphic Relations (MRs) that serve as evaluation measures. Each example depicted in Fig. 1 can be defined as follows:

- MR1: The outputs of LLMs given original and *character-swapped* input texts should be the same.
- MR2: The outputs of LLMs given original and *synonym-replaced* input texts should be the same.

The MRs can be defined by the combination of target LLMs, input texts, prompts, relational operators (i.e., $=$, $\neq$), and perturbation functions (i.e., character-swap). Using MRs to evaluate LLMs can provide several benefits. First, there is no need to rely on inputs from test databases, which are finite and limited [15], because the inputs for MRs do not need to be labeled. Second, the MR-based evaluation is highly extendable to various tasks in LLMs because MRs can be executed as modules for evaluating corresponding QAs and tasks.

Our study aimed to build a MEtamorphic Testing for Analyzing LLMs (METAL) framework. The METAL framework addressed the above issues. We first defined MR templates covering essential QAs, such as `Robustness`, `Fairness`, `Non-determinisms`, and `Efficiency` and six main tasks for LLMs. Next, we developed an automated MR generation process based on the templates, which use multiple types of textual perturbations. Furthermore, we have explored the use of LLMs and our created functions in the MR generation process from the perspective of the self/cross-examination of LLMs. Finally, we introduced novel metrics that integrate *ASR* methods to the semantic and structural similarities of text data to assess the effectiveness of MRs precisely. We defined the following Research Questions (RQs) for our framework:

- RQ1. Can the framework evaluate LLMs and reveal the risks of achieving various QAs on each task?
- RQ2. Can the framework guide which MRs are the most effective for evaluating LLMs on specific tasks?
  - RQ2-1. Which MR shows high effectiveness in evaluating tasks in LLMs?
  - RQ2-2. Which MR presents the most optimized contribution to evaluating tasks in LLMs?
- RQ3. Which MR generation method performs best in the self/cross-examination of LLMs?

RQ1 aims to represent the evaluation results using MRs covering diverse QAs and tasks for LLMs. In RQ2, we aim to assess the efficacy of generated MRs using multiple measures proposed in this study. Finally, RQ3 identifies the most effective MR generation methods for each LLM.

We conducted an experiment in which we applied the proposed framework to three LLMs: GooglePaLM [1], ChatGPT [2], and Llama2 [3]. The results obtained verified that (1) GooglePaLM generally achieved the highest quality for most of the QAs and tasks; (2) the newly proposed MR effectiveness metrics were able to guide the most effective MRs in evaluating each task; (3) the feasibility of self/cross-examination of LLMs was empirically validated, and ChatGPT was revealed to generate MRs with high effectiveness.

The remainder of this paper is organized as follows: Section 2 presents works related to this study. Section 3 elucidates the METAL framework. Section 4 describes the experiment and empirical analysis results. Section 5 recommends directions for future work and concludes the study.

## II. Related Works

We investigated diverse studies evaluating the qualities of the Natural Language Processing (NLP) models. We classified them into three categories: Adversarial test datasets, Adversarial attack generators, and QA analysis on NLP models.

**Adversarial test datasets.** We examined adversarial test datasets and benchmarks for evaluating NLP models, such as AdvGLUE [16], AdversarialSQuAD [17], ANLI [18], SNLI [19], SQuAD 2.0 [20], and HELLASWAG [21]. The datasets are designed to test NLP models' `Accuracy`. AdvGLUE provides adversarial datasets for evaluating the `Accuracy` and `Robustness` of multiple tasks, including sentiment analysis and Natural Language Inference (NLI) [16]. AdvGLUE dataset involves human-crafted adversarial data samples (e.g., typos or stress-testing keywords in sentences). AdversarialSQuAD, ANLI, and SQuAD 2.0 provide adversarial attack and performance evaluation data for Q&A and NLI tasks. Adversarial SQuAD is extended from SQuAD by adding a distracting sentence to the end of the context paragraph [17]. ANLI provides manually crafted complex sentences for NLI task [18]. The SQuAD 2.0 involves unanswerable questions similar to the original answerable ones from SQuAD 1.0 [20]. Other datasets, SNLI and HELLASWAG, provide `Accuracy` testing data for NLI and Q&A tasks.

We found that these datasets can only cover limited tasks among many application scenarios of LLMs. Additionally, the datasets are designed to evaluate the specific functional requirement (i.e., `Accuracy`) of NLP models except for AdvGLUE, which covers the `Robustness` evaluation. Furthermore, the datasets only provide a limited number of test cases due to the cost of analyzing and attaching labels to the data, which is difficult to be compatible with the significantly varied usability of LLMs in different domains.

**Adversarial attack generators.** In order to address the limited diversity and transferability of finite test datasets [15], recent studies proposed adversarial attack generation models based on pre-trained NLP models [4]–[13]. The attack generator models transform the original texts to perturbed texts to validate `Robustness` of LLM tasks. For example, Liu et al. [6] proposed a pre-trained model to append embedding space for the original texts. Jin et al. [7] and Chiang et al. [11] suggested synonym substitution attack models, generating semantically identical but syntactically different sentences. Additionally, Wang et al. [8] provided semantic perturbation functions for the sentiment analysis task.

There exist other studies concentrated on generating modified prompts to evaluate qualities of LLMs [9], [22]–[26]. Guo et al. [22] presented several data integrity and privacy attack methods throughout the pipeline of building and fine-tuning pre-trained NLP models. Perez et al. [9] and Zhou et al. [25] proposed a prompt attack method, Goal hijacking and Prompt leaking, by adding specific commands and suffixes on prompts. Li et al. [23] and Shen et al. [24] presented the jailbreak method to well-known LLMs by giving specific roles to them inducing privacy and security leaking.

Existing attack models are designed to generate specific types of perturbation on text data, presenting low extendability to cover various QAs and tasks. Prompt attack methods are needed to analyze the overall ML pipeline to mitigate security and privacy issues. Moreover, all the studies assessed the efficacy of approaches based on the Attack Success Rate (ASR) method not considering the quality of perturbations.

**QA analysis on NLP models.** We examined general QA analysis studies on NLP models [27]–[29]. HELM [27] proposed by Liang et al. suggested the holistic process of evaluating general NLP models. HELM also provides diverse options for test datasets and evaluation measures on several QAs. However, HELM is designed to provide a holistic survey of appropriately evaluating several LLM tasks based on existing test benchmarks instead of providing an executable framework. Qiu et al. [28] investigated diverse adversarial attacks and categorized them into four levels, followed by analyzing adversarial training methods and overall defense strategies for the adversarial attacks. Likewise HELM, their study did not provide an executable framework for evaluating QAs on LLMs but focused on providing several alternatives to defense against adversarial attacks. Finally, Wang et al. [29] recently proposed a Metamorphic Testing (MT)-based evaluation on text spam detection. They categorized 11 types of perturbations and defined Metamorphic Relations (MRs) by examining thousands of real-world spam texts in English and Chinese. The defined MRs can be utilized to evaluate `Robustness` of spam detection tasks in LLMs.

The METAL framework facilitates a one-shot evaluation of LLMs on four QAs and six tasks. The framework can handle input texts from unlimited sources by applying MT techniques. We systematically defined five MR templates for achieving high coverage of QAs and tasks in LLMs. We have also developed an automated MR generation process that can apply 13 types of perturbations to input texts. In addition, the framework consists of a modularized structure enabling users to extend MR modules based on their target QAs and tasks. Finally, our study suggested using combined *ASR* and text similarity metrics to assess the evaluation results precisely.

## III. Approach: METAL Framework

The METAL framework is designed to effectively evaluate several tasks in LLMs based on various QAs. We first explain the essential QAs needed to be evaluated for main tasks in LLMs with the correlations between them. Next, we formalize the MR templates that can comprehensively cover the QAs and tasks. Third, we provide function-based and LLM-based MR generation methods using the MR templates. Finally, we outline the structure of our framework.

### A. Quality Attributes and Tasks in LLMs

The MT framework requires a set of MRs that serve as evaluation metrics and testing oracles for target systems. One of the main challenges in establishing an MT framework is defining an appropriate set of MRs [14]. In our study, we determined the appropriateness of the MR set as the coverage of QAs and tasks in LLMs. To ensure the validity of MRs in our framework, we followed a top-down approach to define MR templates covering essential QAs and tasks. Then, we generated MRs by functions and LLMs using the templates.

The first step to defining appropriate MRs is identifying the essential QAs and tasks that should be evaluated for LLMs. Based on surveys of quality evaluation of general ML models and generative AIs [15], [22], [27], [30]–[38], we have identified the following four QAs as critical to evaluate LLMs: `Robustness`, `Fairness`, `Non-determinism`, and `Efficiency` as described in Table I.

Several other QAs, such as `Explainability`, `Security`, and `Privacy`, have been identified as necessary for evaluating generative AIs in extant studies [30], [34], [35]. However, evaluating them on LLMs requires external data or considering the ML pipeline processes in addition to LLM outputs. For example, evaluating `Security` and `Privacy` not only involves evaluating the models but also requires data poisoning [30] and human-involved security analysis [39] throughout the entire process of training and deploying ML models. Similarly, evaluating `Explainability` may require additional data or background knowledge beyond what the LLMs themselves generate. Therefore, we focused on analyzing the four QAs that require only LLM outputs for evaluation. Additionally, we did not consider `Correctness` as a critical QA in this study as it is generally known to be the primary performance measure of the functionalities (e.g., prediction accuracy) of ML models [27], [38], [40].

In our study, `Robustness` refers to the ability of LLMs to react appropriately to abnormal conditions [41]. `Fairness` in LLMs means that the results produced by LLMs should not vary significantly based on the user's demographic characteristics [27], [42]–[44]. LLMs are known to exhibit `Non-determinism`, which means that they can produce different outputs given the same input [45]–[47]. Finally, `Efficiency` refers to the time LLMs take to generate outputs after receiving inputs. The following requirements present examples of each QA in LLM evaluation by comparing the outputs generated from original and perturbed inputs:

- `Robustness`: "The outputs of LLMs given original and character-swapped input texts should be identical".
- `Fairness`: "The outputs of LLMs given the same inputs by different demographic users should be identical".
- `Non-determinism`: "The outputs of LLMs given the same inputs repeatedly should be consistent".
- `Efficiency`: "The time difference between original and perturbed inputs should be less than a $threshold\_value$".

Additionally, the coverage of tasks is a crucial factor in representing the effectiveness of the testing framework on LLMs. In recent survey papers [27], [28], [48], six state-of-the-art application scenarios (i.e., tasks) of LLMs were introduced, including information retrieval, text summarization, Q&A, sentiment analysis, toxicity detection, and news classification as an example of text classification. While numerous tasks can be performed using NLP models, we focused on the six most commonly utilized tasks of LLMs.

We categorized the six tasks into two types: Classification and Generative. Classification tasks produce a specific categorization output for input text, including sentiment analysis, toxicity detection, and news classification. The first two

TABLE I: The correlation table of QAs and tasks in LLMs with the MR template coverage

| Tasks in LLMs \ Quality Attributes | | Robustness | Fairness | Non-determinisms | Efficiency |
|---|---|---|---|---|---|
| Classification | Sentiment analysis | *Equivalence_MRT* *Discrepancy_MRT* | *Set_Equivalence_MRT* | *Set_Equivalence_MRT* | *Distance_MRT* |
| | News classification | | - | *Set_Equivalence_MRT* | |
| | Toxicity detection | | *Set_Equivalence_MRT* | *Set_Equivalence_MRT* | |
| Generative | Question & Answering | *Distance_MRT* | *Set_Distance_MRT* | *Set_Distance_MRT* | |
| | Text summarization | | - | *Set_Distance_MRT* | |
| | Information retrieval | | - | | |

Classification tasks are single-label tasks that classify input texts as positive or negative and toxic or non-toxic. The news classification is a multi-label task that classifies given texts into several news categories, such as business, sports, or science. The Generative tasks of LLMs generate inferred text based on input texts, such as text summarization, Q&A, and information retrieval. The information retrieval task returns a top-10 list of information from the given texts, while text summarization generates a paragraph summarizing the input texts.

Finally, we investigated the correlations between the QAs and tasks based on extant studies [27], [28], [35]. As described in Table I, for `Robustness`, `Non-determinism`, and `Efficiency`, the six tasks are highly correlated because the QAs are commonly used to evaluate those tasks.

However, for `Fairness`, we only identified the correlations with sentiment analysis, toxicity detection, and Q&A tasks based on the existing survey [27]. Confirming significant relationships between `Fairness` and news classification, text summarization, and information retrieval tasks is challenging. For example, output changes between news classification labels (from business to social) based on the user's regional group may not have as significant an impact as output distinctions in toxicity detection, where the exact text from different regional groups may be identified as different results. Therefore, this study focused on evaluating `Fairness` in sentiment analysis, toxicity detection, and Q&A tasks based on the impacts of failures. We defined a set of MR templates to cover all identified correlations between QAs and tasks.

## B. Metamorphic Relation Templates for LLM Evaluation

MR Templates (MRTs) are used to systematically define MRs based on corresponding functional requirements or QAs [49], [50]. In Table I, we have defined five MR templates that cover all the identified correlations of QAs and tasks of LLMs in this study. An MR template comprises a combination of $LLMS$, $Input$, $Prompt$, $REL\_OP$, $Perturb$, and $Dist$ instances. Given $Text \ni text$, where a $text$ indicates a phrase, the main components of MR templates are defined as follows:

$$LLMS \triangleq \{M : Text \times Text \rightarrow Text \,|\, M \text{ is an executable LLM API or model}\}, \quad (1)$$

$$Input \triangleq \{Text \ni input \,|\, input \text{ is a sample } text\} \quad (2)$$

$$Prompt \triangleq \{Text \ni prompt \,|\, prompt \text{ is an instructive } text, \text{ identifying specific task execution in LLM}\} \quad (3)$$

$$Relation\_OP \triangleq \{=, \leq, <, \geq, >, \neq\}, \quad (4)$$

$$Perturb \triangleq \{P : Text \rightarrow Text \,|\, P \text{ is a } text \text{ perturbation function for adversarial transformation}\}, \quad (5)$$

$$Dist \triangleq \{D : Text \times Text \rightarrow \mathbb{R} \,|\, D \text{ is a distance function for two given } texts \text{ with a specific purpose}\}. \quad (6)$$

Equation (1) describes a set of LLMs that take input and prompt texts and return an output text. Equation (2) and (3) specify the input and prompt texts for LLMs, respectively. The $input$ is a sample paragraph or sentence provided to LLMs, while the $prompt$ determines the task that LLMs execute, such as "Please analyze the sentiment of the following text" for sentiment analysis or "Please summarize the following text in five sentences" for text summarization task. Equation (4) defines the set of relational operators used in MRs, including equivalence ($=$) and inequivalence operators ($\leq, <, \geq, >$), and discrepancy operator ($\neq$). Equation (5) and (6) define the set of perturbation and distance calculation functions utilized in MRs. The set $Dist$ includes various distance calculation functions for text data, such as semantic text similarity and ranking distances. Section III-C and IV-B will explain the perturbation and distance functions used in our framework.

***Equivalence_MRT*** defines the most commonly used form of MRs [14], which compares the output equivalence given the original input and perturbed input for `Robustness` evaluation. We defined *Equivalence_MRT* as follows:

$$Equivalence\_MRT \implies M(input, prompt) = M(P(input), prompt). \quad (7)$$

This template is widely utilized for short-answer outputs, including Classification tasks in LLMs [8], [9], [13]. Based on Equation (7), we can define MRs by applying different semantic-preserving perturbations, such as Character-swap, Add-space, or Synonym-substitution functions.

On the contrary, to ensure the `Robustness` of LLMs against semantic-altering perturbations [27], the following ***Discrepancy_MRT*** is needed:

$$Discrepancy\_MRT \implies M(inpt, prompt) \neq M(P(input), prompt). \quad (8)$$

TABLE II: Perturbation table for METAL framework

| Level of Perturbations \Semantic Impact | Semantic-preserving | Semantic-altering | Etc |
|---|---|---|---|
| Character-level | ReplaceCharacters(), DeleteCharacters() ConvertTol33tFormat(), AddRandomCharacters() AddSpaces(), SwapCharacters(), ShuffleCharacters() | - | - |
| Word-level | ReplaceSynonyms(), AddRandomWords() | ReplaceAntonyms() | - |
| Sentence-level | RemoveSentences() | ReplaceSentences() | AssignDemographicGroup() |

Equation (8) depicts that the model's outputs on the original and the perturbed input should differ. This can be achieved by generating attacks such as Antonym-substitution or Pronoun-substitution, which change the meaning while preserving its grammatical structure (e.g., changing 'he' to 'she' in gender identification questions). By evaluating LLMs against this MR template, we can ensure they can robustly handle such attacks.

***Set_Equivalence_MRT*** is another extension of *Equivalence_MRT* that compares the original output with a set of outputs generated by LLMs. Let $O \ni o$ be the set of outputs generated by using a subset of perturbation functions, $P$, and the $MRT$ can be defined as follows:

$$Set\_Equivalence\_MRT \implies \forall o \in O : M(input, prompt) = o. \quad (9)$$

Equation (9) covers the `Fairness` and `Non-determinism` evaluation in LLMs. For example, a set of perturbation functions that change the demographic population group can be used to evaluate whether LLMs return the same outputs given the same input texts from the users with different demographic groups (E.g., gender, region, and orientation). Additionally, the template can also be used to check the `Non-determinism` of LLMs by applying a $\phi$ instance in $P$, which indicates no perturbation, to Equation (9) to calculate the output variances given the same inputs repeatedly.

***Distance_MRT*** facilitates a quantified comparison of original and perturbed outputs [14] from LLMs as follows:

$$Distance\_MRT \implies D(M(input, prompt), M(P(input), prompt)) \leq \alpha \quad (10)$$

Where $\alpha$ indicates a threshold value for deciding the satisfaction of MRs. We utilized several distance functions, $D \in Dist$, to generate MR instances from the ***Distance_MRT***, such as semantic contextual similarities for `Robustness` evaluation of Generative tasks, ranking distances for measuring the outputs from the information retrieval task, and time measuring functions to evaluate the inference efficiency of different inputs in LLMs. Section IV-B will explain the functions in detail.

***Set_Distance_MRT*** extends the Equation (10) to estimate the differences in output variances, specifically addressing the `Non-determinism` attribute in Generative tasks. The template is defined as follows:

$$Set\_Distance\_MRT \implies \forall o \in O : D(M(input, prompt), o) \leq \alpha \quad (11)$$

Equation (11) covers the comparison of the set of outputs generated by repeatedly giving the same input to LLMs. Therefore, this template can be used to evaluate the non-deterministic nature of Generative tasks by checking if multiple runs of the same input result in similar outputs within a certain threshold of variance $\alpha$.

The set of MRTs proposed in our study is designed to evaluate principal QAs on main tasks in LLMs comprehensively. We believe that these MRTs, from Equations (7) to (11), represent the first attempt to formally define evaluation measures for LLMs with high coverage of QAs and tasks.

### *C. Generating Metamorphic Relations using Templates*

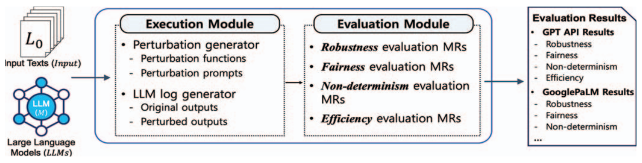
The MR Templates define the scope of evaluation according to the evaluation goals, QAs, and the target tasks. The last step to automatically generate the MRs inherited from MRTs is to specifically determine perturbation functions $P \in Perturb$ and to assign them to each template. The MR example in Fig. 1(a) is instantiated by assigning SwapCharacters() perturbation function to *Equivalence_MRT* template.

Table II explains 13 perturbation functions, $P$s, developed in our framework. Existing studies have applied various perturbation functions using character, word, and sentence-level categorizations [26]–[28]. In our framework, we have added a dimension of semantic impact to this categorization, which characterizes the effect of the perturbation functions on the semantic consistency of the original text.

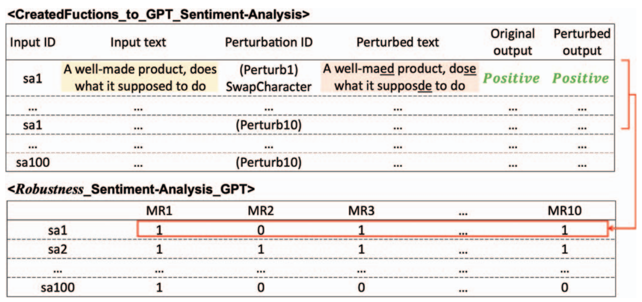
We have devised perturbation functions to fully cover each category in Table II. First, we developed seven semantic-preserving attacks at the character level because there is no semantic context at that level. The functions replace, delete, add, swap, and shuffle random characters in the given *input*. Additionally, we have included the ConvertTol33tFormat() function, which converts the text to l33t format (e.g., 'apple' to '4ppl3'). Finally, the AddSpaces() function randomly assigns spaces in the text, such as 'years' to 'y_ear__s'.

We have built five perturbation functions for word and sentence levels, including semantic-preserving and altering perturbations. We have categorized the ReplaceAntonyms() function as semantic-altering because it changes the context of words in sentences. Similarly, we have assigned the ReplaceSentences() function to the semantic-altering class because the random replacement of dummy sentences (e.g., 'Lorem ipsum dolor sit amet.') would have a more significant impact on the context of the overall text than removing random sentences.

Finally, we have included a function for assigning a demographic group, which adds a sentence to provide background knowledge of the user's demographic group for `Fairness` evaluation. For example, the sentence 'The following text is asked by [Gender] or [Age] or [Race] or [Orientation] person.' is appended to the task prompt. We have used 21 options to



(a) Overall structure of the proposed framework



<CreatedFuctions_to_GPT_Sentiment-Analysis>

| Input ID | Input text | Perturbation ID | Perturbed text | Original output | Perturbed output |
|---|---|---|---|---|---|
| sa1 | A well-made product, does what it supposed to do | (Perturb1) SwapCharacter | A well-maed product, dose what it supposde to do | *Positive* | *Positive* |
| ... | ... | ... | ... | ... | ... |
| sa1 | ... | (Perturb10) | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |
| sa100 | ... | (Perturb10) | ... | ... | ... |

<*Robustness*_Sentiment-Analysis_GPT>

| | MR1 | MR2 | MR3 | ... | MR10 |
|---|---|---|---|---|---|
| sa1 | 1 | 0 | 1 | ... | 1 |
| sa2 | 1 | 1 | 1 | ... | 1 |
| ... | ... | ... | ... | ... | ... |
| sa100 | 1 | 0 | 0 | ... | 0 |

(b) Example outputs generated by the framework

Fig. 2: Overall structure and outputs of METAL framework

represent demographic groups, including 3 gender groups, 3 age groups, 10 race groups, and 5 orientation groups.

In addition to the 13 perturbation functions we created, we utilized LLMs to generate perturbed text from the perspective of self and cross-examination. To achieve this, we engineered specific prompts for each perturbation, such as "Please randomly replace a maximum of three synonyms for each sentence in the above text." The reason for using LLMs in perturbation is that specific perturbations require contextual analysis of each token in given sentences, which is highly similar to the sentence analysis process in LLMs. However, in experiments, we observed several faulty applications of perturbations, such as a perturbed text that cannot be understood by humans or a perturbed text replaced by word-level synonyms that are entirely inappropriate in context, for instance:

- "Atlan Jayne will leave Qantas tomorrow."
  $\rightarrow$ "Atlhn Jaycezwqil jqave Qnotas aorytou."
- "What's really bad for the body but people keep doing it?"
  $\rightarrow$ "What's truly regretful for the dead body simply people go along set it?"

To address the low-quality and incorrectly generated perturbation issues, we newly designed a quality metric for the text perturbation explained in detail in Section IV-B.

### D. Framework Implementation

The METAL framework is designed to return the evaluation results for four main QAs on six tasks in LLMs. Given a set of input texts ($Input$) and a set of LLMs ($LLMS$), the framework operates in two main modules: Execution and Evaluation. Fig. 2 provides an overview of the structure and example outputs of the framework.

The Execution module consists of a perturbation generator, which generates perturbations using functions and LLM-based prompts, and an LLM log generator, which collects the outputs generated by the LLMs. An example output generated by the Execution module is shown at the top of Fig. 2(b). The output includes the Input ID and text, the Perturbation ID and text generated by the module, and the outputs generated by the LLMs given the original and perturbed input texts.

TABLE III: Overall Experimental Settings for METAL

| Experiment Setting | |
|---|---|
| Target LLMs | PaLM-text-bison-001, GPT-3.5-Turbo, Llama-2-7b-chat.Q4 on Llama-CPP |
| Quality Attributes | Robustness (R), Fairness (F), Non-determinism (ND), Efficiency (E) |
| Covered Tasks | 6 tasks for R, ND, E<br>3 tasks for F |
| # of MRs generated | R: 240 MRs based on 3 MRTs<br>F: 21 MRs by 2 MRTs<br>ND: 6 MRs by 2 MRTs<br>E: 6 MRs by 1 MRT |
| # of input texts | 600 and 300 input texts for R and F<br>ND and E share the 900 outputs |
| Range of tokens | 15 to 4K tokens for each input text |
| # of estimated requests | 67,800 requests for each LLM |
| # of estimated tokens requested | 33,340,000 tokens for each LLM |
| Python version | 3.11.4 |
| Virtual environment | Conda 23.7.2 |
| Memory | 16 GB |

The Evaluation module uses MRs generated from MRTs to validate the execution results. The evaluation results are represented using binary values, with 1 indicating the satisfaction of MRs and 0 showing the opposite. For each original input in the Execution module, 10 types of perturbations are applied, resulting in a row with 10 MR columns in the results.

The METAL framework enables the use of input data from various sources, detouring the high cost of the labeling process. The MR generation structure is template-based and modularized, allowing users to extend and apply various MRs and tasks based on their analytic goals. Additionally, the framework offers API-based, model-based LLM execution functions to evaluate various LLMs dynamically. The detailed manual, implementation, and MR-based evaluation examples are available on a GitHub repository[1]. Section Appendix also describes a visualized example of MR-based evaluation.

## IV. Experiment

We conducted experiments to demonstrate the effectiveness of the proposed METAL framework and the constituent MRs by addressing four research questions outlined in Section I.

### A. Experiment Design

Table III depicts the overall statistics of the experiment conducted. We utilized LLMs from the leading industry vendors: Google, OpenAI, and Meta. We evaluated 21 combinations of QA and tasks on the LLMs by following the correlation of four QAs and six tasks: Toxicity Detection (TD), Sentiment Analysis (SA), News Classification (NC), Question&Answering

[1]https://github.com/abalon1210/METAL-Framework

(Q&A), Text summarization (TS), and Information Retrieval (IR), identified in Table I. Our framework generated a total of 273 Metamorphic Relations (MRs). This included 240 MRs for `Robustness` evaluation, indicating ten perturbations generated by four methods for six tasks. Out of the 13 perturbations defined in Table II, we selected 10 for each task based on their characteristics. For example, in the Classification and Q&A tasks, the input texts typically consist of one or two sentences, which are difficult to remove or replace. Similarly, for the other tasks, which include 10 to 20 sentences of input text, character-level perturbations such as deleting characters would not be expected to have a significant impact. In our `Robustness` evaluation, we set 10 perturbations for each task referring to existing studies [28], [29]. For `Fairness`, 21 MRs are generated by using 21 demographic group options based on 2 MR Templates. 6 MRs for `Non-determinism` and `Efficiency` are executed by each task.

This experiment randomly sampled 900 input texts from various web sources, such as Amazon reviews for the SA task and News articles with headings for TS, IR, and NC tasks. We opened all the input texts and prompts used for this experiment in our repository. The input texts include various sets of tokens ranging from 15 to 4K in length. We estimated that each LLM received approximately 67,800 requests, each being requested and returning approximately 33,340,000 tokens for this experiment. This includes the five repetitions of all the executions for the LLMs. Further details on the number of iterations will be explained in Section IV-C.

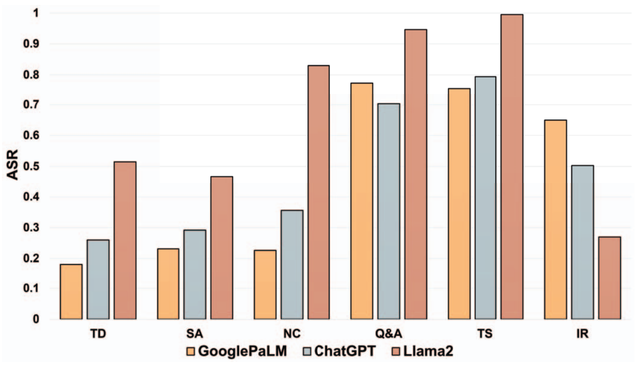
Finally, we executed the METAL framework using Python version 3.11.4 in a Conda virtual environment with version 23.7.2. The framework was executed with 16GB of RAM.

## B. Experiment Results

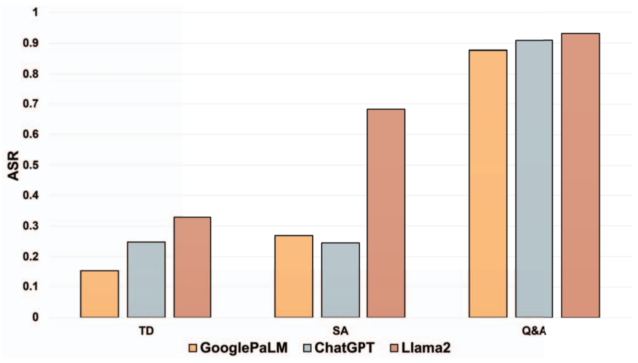
**RQ1. Quality evaluation results on LLMs.** In order to evaluate LLMs and reveal potential risks, we first analyzed the MR satisfaction results by using *Attack Success Rate (ASR)* [27] with text similarity metrics. The *ASR* is calculated by dividing the number of times when MRs are unsatisfied with input texts by the number of all executions. For example, assuming 10 MRs are executed on 100 inputs and MRs are unsatisfied 200 times, the *ASR* would be 0.2 (200/1,000).

The satisfaction of MRs is decided through the MR Templates, which are applied differently depending on tasks as described in Table I. For Classification tasks, comparing the output differences generated by the original and perturbed inputs is self-explanatory because the outputs are categorized as positive/negative, and news categories (e.g., business, sports).

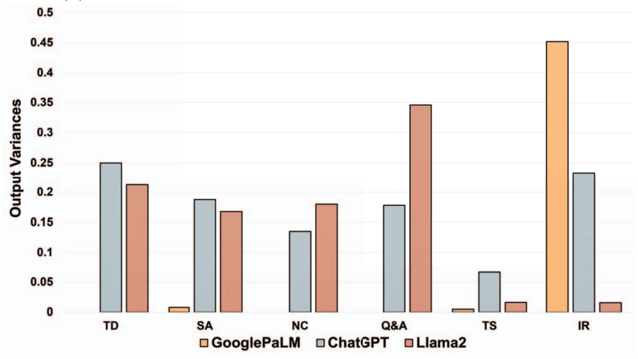
In Generative tasks, we applied several distance functions and threshold values to validate the MR satisfaction. First, we applied the Universal Sentence Encoder (USE) model, provided by Google [51], to calculate the *Semantic Textual Similarity (STS)* of the Q&A generated sentences. In the TA tasks, we applied the *STS* metric to calculate the contextual similarity average for each summarized sentence. We set the threshold value as 0.6 to determine the contextual equivalence of the two tasks [51]. Finally, we defined *Maximum STS Ranking Distance (MSRD)* metric for the IR task that returns a top-10 ranked information retrieved from the given paragraph. By extending the Kendall tau ranking distance [52] to the text data, the *MSRD* metric calculates the average ranking differences between each original text and the most contextually similar text in the perturbed ranking list. For example, if the 1st-rank text in the original output presents the highest *STS* value with the 3rd-rank text in the perturbed output, the ranking difference is 2. This way, we calculate the average ranking distances and determine the equivalence of the original and perturbed IR-task outputs with the threshold value of 2.



(a) `Robustness` evaluation results on all tasks and LLMs



(b) `Fairness` evaluation results on 3 tasks and LLMs



(c) `Non-determinism` evaluation results on all tasks and LLMs

Fig. 3: Essential QA evaluation results on LLMs

Fig. 3 illustrates the `Robustness`, `Fairness`, and `Non-determinism` evaluation results by using the metrics. In Fig. 3(a), the higher the *ASR* values, the more vulnerable

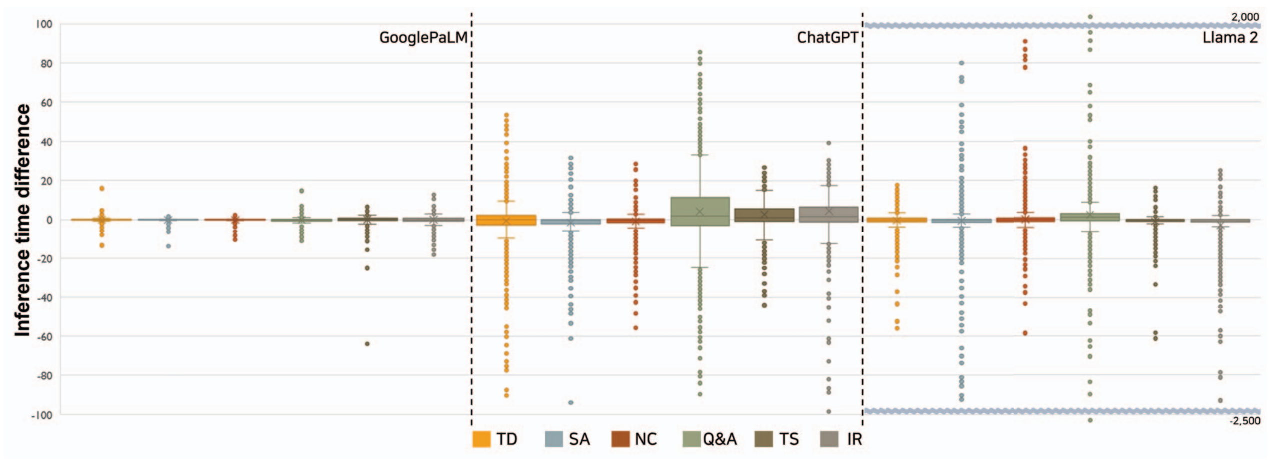


Fig. 4: `Efficiency` evaluation results on all tasks and LLMs

the LLM is to the adversarial perturbations. LLMs generally achieved lower *ASR* values in Classification tasks than in Generative tasks. While GooglePaLM and ChatGPT presented similar *ASR* results across all tasks, Llama2 had the highest *ASR* for all tasks except IR. Even in the TS task, Llma2 exhibits 0.99 *ASR*, indicating that almost all perturbations produced different outputs from Llama. We will discuss the Llama2 results in the IR task in conjunction with Fig. 3(c).

The results of the `Fairness` evaluation, which included 21 perturbations on the LLMs, are presented in Fig. 3(b). Among the three LLMs, GooglePaLM achieved the lowest *ASR* value on the TD task, while ChatGPT achieved the lowest *ASR* in the SA task. On the other hand, Llama2 presented the highest *ASR* value for all tasks in the `Fairness` evaluation. In the Q&A task, the LLMs produced similar outputs, with an average value of approximately 0.9 *ASR*.

Fig. 3(c) depicts the output variance results on each task and LLM to evaluate the *Non-determinism*. The output variance is calculated as the average of 1-*STS* values to represent the differences between the outputs generated from the same inputs. To achieve this, we utilized 20 iteration sets of results by giving the same inputs to each LLM. The higher the output variances, the more different the outputs are. GooglePaLM achieved shallow output variance values for all tasks except the IR task. GooglePaLM showed almost 0 output variances when given the same input texts repeatedly. In most of the tasks, ChatGPT and Llama2 produced similar outputs. However, in the IR task, GooglePaLM exhibited the highest output variances among the LLMs, while Llama2 presented the lowest output variance. Our investigation revealed that Llama2 frequently returned several sets of recommendations and manual texts when the model could not understand the given inputs or prompts. This explains its performance in the IR task and the results of the `Robustness` evaluation.

Fig. 4 illustrates the differences in model inference time for the LLMs. We measured the inference time difference by subtracting the perturbed time from the original time in seconds. Outputs closer to 0 indicate lower differences. Negative values indicate the perturbed time is larger than the original time. GooglePaLM showed the most stable efficiency differences among the LLMs when given original and perturbed inputs. ChatGPT exhibited varied results in the time differences, particularly in the generative tasks, where the original inference time was generally more significant than the inference time given perturbed inputs. Llama2 presented results similar to ChatGPT but with more significant variances, ranging from -2,500 to 2,000 differences in seconds.

> We observed that GooglePaLM generally achieved the highest quality on most of the QAs and tasks. ChatGPT presented similar results to GooglePaLM. However, we noticed that Llama2 showed the most fluctuating results across all evaluations.

**RQ2. The effectiveness of MRs.** We delved into the `Robustness` evaluation results from RQ1 to determine the most appropriate MR for each task. To measure the effectiveness of MRs, we defined the *Effectiveness of MRs, (EFM)* metric for an MR in LLM evaluation as follows:

$$EFM = M\text{-}ASR * PerturbQuality, \tag{12}$$

*M-ASR* represents the *ASR* metric for an MR. It is computed by dividing the number of unsatisfied executions by the total executions of the MR. The *PerturbQuality* metric is used to measure the quality of perturbed texts generated from original texts by each MR. If the effectiveness of MRs is evaluated without considering the *PerturbQuality*, then MRs that generate unreadable perturbed texts are considered the most effective because they produce higher *ASR*. To the best of our knowledge, we tried the first attempt to calculate the MR effectiveness by combining the two contrasting measures.

Algorithm 1 describes the calculation of the perturbation quality between a set of original input texts, $original$, and a set of perturbed texts generated, $perturbed$. Lines 3 and 4 indicate the lists of constituent sentences for each input text. Line 5 presents the application of *AvgSTS* metric, which

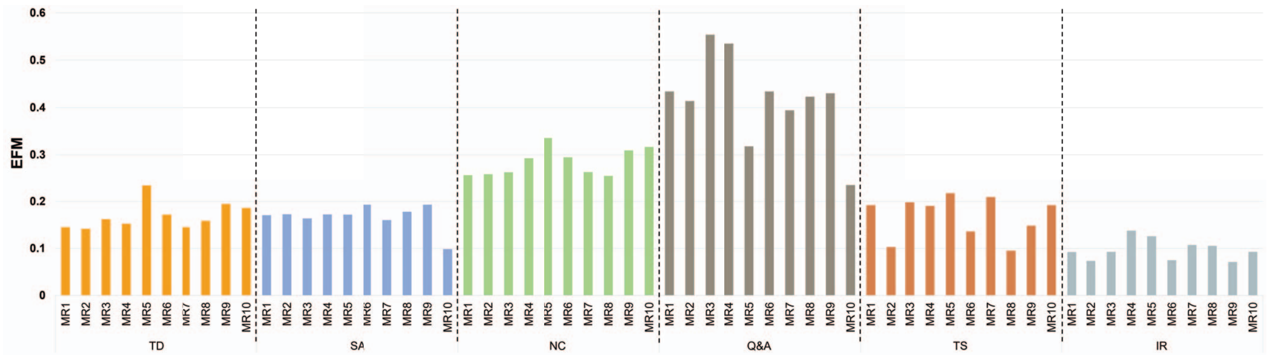


Fig. 5: The effectiveness of MR evaluation results for `Robustness` analysis

**Algorithm 1:** *PerturbQuality* calculation algorithm

```
   Input    : $original$, $perturbed$ are sets of $Text$
   Output   : PerturbQuality $\in \mathbb{R}$
1  PerturbMeasure $\leftarrow$ []
2  for org_text, pert_text in zip($original$,$perturbed$) do
3      org_list $\leftarrow$ org_text.split(“.”);
4      pert_list $\leftarrow$ pert_text.split(“.”);
5      ContextSim = $Avg\text{-}STS$(org_list, pert_list);
       // No perturbation applied
6      if ContextSim >= 0.98 then
7          PerturbMeasure.add(0);
8      else
9          PerturbMeasure.add(ContextSim $*$ $\frac{|\text{pert\_list}|}{|\text{org\_list}|}$);
10 return AVG(PerturbMeasure);
```

calculates the average contextual similarity of the two inputs. In Line 6, we empirically set the text identity threshold to detect cases where the generated perturbed text is identical to the original text. Line 9 calculates the similarity of the two sentences where ContextSim covers the character and word-level similarities. The ratio of the lengths of the perturbed and original texts handles the sentence-level similarity. Finally, the algorithm returns the average similarity as *PerturbQuality*.

**RQ2-1. The effectiveness of MRs on each Task.** Based on the proposed *EFM* metric, we analyzed the effectiveness of MRs by each task and QA. In Fig. 5, the results showed which MRs are more effective than others for evaluating specific tasks in LLMs. Higher values are better for this metric.

The experiment results revealed that the highest effectiveness was observed for the TD, NC, and TS tasks for MR type 5 that used the ConvertToI33tFormat perturbation. For the SA task, the perturbations of ShuffleCharacter and SwapCharacter, both of which were character-level perturbations, were most effective. In addition, the results demonstrated that word-level perturbations, such as ReplaceSynonym and AddRandomWord, were effective in the Q&A and IR tasks. On the other hand, the sentence-level perturbation, ReplaceRandomSentence, exhibited high effectiveness in the IR task.

**RQ2-2. Optimized MR on each Task.** On top of the results on RQ2-1, we investigated to determine the most optimized MR for each task. We utilized the Shapley-Value (SV) [53] to identify the most dominant MRs for each task, considering the combination of perturbations. By selecting the top-5 MRs based on the results on RQ2-1, we applied all the combinations of the selected perturbations on the input texts. The *SV* value for $MR_i$ is computed by adding up the marginal contributions of $MR_i$ in different sets and dividing the sum by the total number of permutations. A marginal contribution measures the difference in *EFM* between a combination that includes $MR_i$ and one that doesn't. For instance, if the *EFM* for $\{MR_0, MR_1\}$ is 0.3 and the one for $\{MR_0, MR_1, MR_2\}$ is 0.45, the marginal contribution of $MR_2$ is 0.15.

TABLE IV: The top-3 SV evaluation results for each task

| | TD | SA | NC | Q&A | TS | IR |
|---|---|---|---|---|---|---|
| Top-1 | ConvertTo I33tFormat | Shuffle Character | AddSpace | Replace Synonym | Replace Synonym | AddRandom Word |
| Top-2 | AddRandom Character | AddRandom Character | ConvertTo I33tFormat | AddRandom Word | AddRandom Word | Introduce Typo |
| Top-3 | Delete Character | Delete Character | Delete Character | Introduce Typo | ConvertTo I33tFormat | Replace Antonym |

The top-3 SV results of MRs for each task are presented in Table IV. Our investigation revealed that character and word-level perturbations are more effective than sentence-level ones. In the results, MRs based on character-level perturbations are the most optimized for Classification tasks, while word-level ones are the most effective for Generative tasks. The results suggest that character-level and word-level perturbations can be adapted based on the number of tokens in the input texts.

In the MR effectiveness analysis on `Robustness` results, we found that character-level and word-level perturbations achieved higher effectiveness for all tasks.

**RQ3. Self and Cross-examination of LLMs.** Moreover, we examined the MR effectiveness results in terms of self and cross-examination of LLMs. We aimed to validate the feasibility of self and cross-examination of LLMs by determining which MR generation methods were most effective in evaluating the qualities of each target LLM.

Table V presented the average *EFM* results on each MR generation method to the corresponding LLMs. When targeting GooglePaLM, all MR generation methods except for GooglePaLM achieved similar *EFM* results. When targeting ChatGPT, we observed that MRs generated by CreatedFunc-

TABLE V: Average *EFM* results for MR generation methods on target LLMs

| | CreatedFunctions | GooglePaLM | ChatGPT | Llama2 |
|---|---|---|---|---|
| GooglePaLM | **0.22** | 0.09 | **0.22** | 0.20 |
| ChatGPT | **0.24** | 0.13 | **0.28** | 0.23 |
| Llama2 | 0.26 | **0.35** | **0.30** | 0.12 |

tion and ChatGPT methods presented higher effectiveness. Lastly, for Llama2, GooglePaLM and ChatGPT presented higher *EFM* values compared to other methods.

During the self and cross-examination of LLMs for `Robustness`, ChatGPT conventionally generated highly effective MRs when targeting all LLMs.

## C. Threats to Validity

**Internal Validity.** In general, LLMs can understand the context of conversations with users. In an experiment, we repeatedly provided LLMs with the original and perturbed texts modified from the original texts. This can be a threat that the LLMs would consistently return the same output from the first given input to other similar inputs. To mitigate the impact of contextual contamination in LLM outputs, we adopted a strategy of opening a new session for each request by using APIs and downloaded models instead of continuously providing similar inputs as in web-based usage. Moreover, we varied the order of the original and perturbed inputs, avoiding always providing the original inputs at the first request.

**External Validity.** Our framework includes several prompts, such as generating perturbations on original input texts by LLMs or executing various tasks. We referred to recent studies on prompt engineering to develop precise prompts for each functionality. For instance, we used the prompt "Please summarize the given text in 5 sentences" to request the text summarization task. We used the prompt "Please randomly swap characters a maximum of 3 times in each sentence in the given text" to request the SwapCharacter perturbation to LLMs. The prompts used in the experiment are entirely open in our repository. Further, we plan to include prompt perturbations for LLM quality evaluation in future studies.

**Conclusion Validity.** We executed the proposed framework on three different LLMs provided by prominent vendors. First, GPT-3.5-turbo is the base model of ChatGPT, consisting of 175 billion parameters. OpenAI offers APIs for every GPT model for a fee. Next, PaLMAPI consists of 750 billion parameters provided by Google. It is open for free use by developers through its API service. Finally, Meta provides a downloadable LLM called Llama 2, which comes in different types based on the parameter sizes. Our experiment used a 7 billion-sized model due to limited resources, instead of a 70 billion model requiring at least 30GB of GPU memory.

We evaluated all QAs and tasks for 5 iterations during our experiment to ensure consistent results for all LLMs since GPT-APIs are not offered for free use. We requested about 33 million tokens and received similar tokens from LLMs. However, because the charging policy in GPT-APIs is both on input and output tokens, we decided to limit the repetitions instead of decreasing the scalability of the experiment. To evaluate `Non-determinism`, we executed all original inputs every time we tested different MR generation methods. Therefore, we utilized 20 execution results (i.e., 5 repetitions * 4 MR generation methods) for each original input.

## V. Conclusion

This study presented the METAL framework, a systematic approach for evaluating the quality of LLMs through MT techniques. The framework offers a comprehensive set of MR templates covering important QAs and tasks in LLMs, followed by automated MR generation modules. Additionally, we proposed novel metrics to measure the effectiveness of MRs by integrating the *Attack Success Rate* with several semantic and structural similarity metrics for text data. The framework can be utilized to assess various quality attributes of LLMs, identify areas for improvement, and enhance the overall quality of outputs generated by LLMs.

The experimental results conducted on three LLMs demonstrated the effectiveness of the proposed framework. GooglePaLM generally achieved the high quality outputs across different tasks. The newly proposed metrics guided the prioritization of MRs for each task, providing insights on which MRs are most effective in evaluating specific tasks of LLMs. The feasibility of self/cross-examination of LLMs is empirically validated in this study, with Chat-GPT generating highly effective MRs for the target LLMs.

We expect our framework to provide multiple benefits to different users. For ML engineers, our approach facilitates the testing process by using unlabelled inputs for LLMs. This allows LLMs to be executed in infinite scenarios and conditions, revealing several edge failure cases that were previously unreported. For industry engineers, the METAL framework can serve as an open assessment platform for fine-tuned LLMs in various domains. Small businesses, in particular, can benefit from our framework as it can address their difficulties in assessing the qualities of the fine-tuned LLMs. Additionally, as we have made the first attempt to apply MT techniques in evaluating LLMs, we believe our framework has the potential for further improvements. We plan to extend its coverage to more QAs and tasks. We also aim to involve prompt perturbations in MRs and propose optimization techniques to generate more effective MRs for testing LLMs.

## Appendix

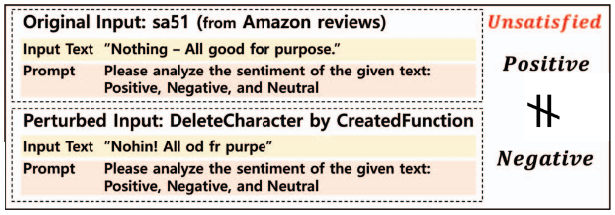

Original Input: sa51 (from Amazon reviews)
Input Text "Nothing – All good for purpose."
Prompt Please analyze the sentiment of the given text: Positive, Negative, and Neutral
Perturbed Input: DeleteCharacter by CreatedFunction
Input Text "Nohin! All od fr purpe"
Prompt Please analyze the sentiment of the given text: Positive, Negative, and Neutral
Unsatisfied
Positive
Negative

Fig. 6: Example execution of MRs for sentiment analysis task on `Robustness` evaluation

## Acknowledgment

We thank Navaneeth Sivakumar for his assistance with developing the framework, conducting experiments with data sampling, and improving the usability of the open repository.

## References

[1] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann et al., “Palm: Scaling language modeling with pathways,” arXiv preprint arXiv:2204.02311, 2022.

[2] OpenAI. (2023) Chatgpt 3.5 turbo api. [Online]. Available: https://openai.com/blog/openai-api

[3] Meta. (2023) Llama 2 api. [Online]. Available: https://ai.meta.com/llama/

[4] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou, “Large language models can be easily distracted by irrelevant context,” in International Conference on Machine Learning. PMLR, 2023, pp. 31 210–31 227.

[5] B. Liu, B. Xiao, X. Jiang, S. Cen, X. He, W. Dou et al., “Adversarial attacks on large language model-based system and mitigating strategies: A case study on chatgpt,” Security and Communication Networks, vol. 2023, 2023.

[6] X. Liu, H. Cheng, P. He, W. Chen, Y. Wang, H. Poon, and J. Gao, “Adversarial training for large neural language models,” arXiv preprint arXiv:2004.08994, 2020.

[7] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” in Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 05, 2020, pp. 8018–8025.

[8] B. Wang, C. Xu, X. Liu, Y. Cheng, and B. Li, “Semattack: natural textual attacks via different semantic spaces,” arXiv preprint arXiv:2205.01287, 2022.

[9] F. Perez and I. Ribeiro, “Ignore previous prompt: Attack techniques for language models,” arXiv preprint arXiv:2211.09527, 2022.

[10] J. Wang, X. Hu, W. Hou, H. Chen, R. Zheng, Y. Wang, L. Yang, H. Huang, W. Ye, X. Geng et al., “On the robustness of chatgpt: An adversarial and out-of-distribution perspective,” arXiv preprint arXiv:2302.12095, 2023.

[11] C.-H. Chiang and H.-y. Lee, “Can large language models be an alternative to human evaluations?” arXiv preprint arXiv:2305.01937, 2023.

[12] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao et al., “A survey of safety and trustworthiness of large language models through the lens of verification and validation,” arXiv preprint arXiv:2305.11391, 2023.

[13] J. Wang, Z. Liu, K. H. Park, M. Chen, and C. Xiao, “Adversarial demonstration attacks on large language models,” arXiv preprint arXiv:2305.14950, 2023.

[14] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, “Metamorphic testing: A review of challenges and opportunities,” ACM Computing Surveys (CSUR), vol. 51, no. 1, pp. 1–27, 2018.

[15] F. Tambon, G. Laberge, L. An, A. Nikanjam, P. S. N. Mindom, Y. Pequignot, F. Khomh, G. Antoniol, E. Merlo, and F. Laviolette, “How to certify machine learning based safety-critical systems? a systematic literature review,” Automated Software Engineering, vol. 29, no. 2, p. 38, 2022.

[16] B. Wang, C. Xu, S. Wang, Z. Gan, Y. Cheng, J. Gao, A. H. Awadallah, and B. Li, “Adversarial glue: A multi-task benchmark for robustness evaluation of language models,” arXiv preprint arXiv:2111.02840, 2021.

[17] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” arXiv preprint arXiv:1707.07328, 2017.

[18] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, “Adversarial nli: A new benchmark for natural language understanding,” arXiv preprint arXiv:1910.14599, 2019.

[19] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” arXiv preprint arXiv:1508.05326, 2015.

[20] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” arXiv preprint arXiv:1806.03822, 2018.

[21] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, “Hellaswag: Can a machine really finish your sentence?” arXiv preprint arXiv:1905.07830, 2019.

[22] S. Guo, C. Xie, J. Li, L. Lyu, and T. Zhang, “Threats to pre-trained language models: Survey and taxonomy,” arXiv preprint arXiv:2202.06862, 2022.

[23] H. Li, D. Guo, W. Fan, M. Xu, and Y. Song, “Multi-step jailbreaking privacy attacks on chatgpt,” arXiv preprint arXiv:2304.05197, 2023.

[24] X. Shen, Z. Chen, M. Backes, and Y. Zhang, “In chatgpt we trust? measuring and characterizing the reliability of chatgpt,” arXiv preprint arXiv:2304.08979, 2023.

[25] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, “Universal and transferable adversarial attacks on aligned language models,” arXiv preprint arXiv:2307.15043, 2023.

[26] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N. Z. Gong, Y. Zhang et al., “Promptbench: Towards evaluating the robustness of large language models on adversarial prompts,” arXiv preprint arXiv:2306.04528, 2023.

[27] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar et al., “Holistic evaluation of language models,” arXiv preprint arXiv:2211.09110, 2022.

[28] S. Qiu, Q. Liu, S. Zhou, and W. Huang, “Adversarial attack and defense technologies in natural language processing: A survey,” Neurocomputing, vol. 492, pp. 278–307, 2022.

[29] W. Wang, J.-t. Huang, W. Wu, J. Zhang, Y. Huang, S. Li, P. He, and M. R. Lyu, “Mttm: metamorphic testing for textual content moderation software,” in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023, pp. 2387–2399.

[30] H. B. Braiek and F. Khomh, “On testing machine learning programs,” Journal of Systems and Software, vol. 164, p. 110542, 2020.

[31] P.-O. Côté, A. Nikanjam, R. Bouchoucha, I. Basta, M. Abidi, and F. Khomh, “Quality issues in machine learning software systems,” arXiv preprint arXiv:2306.15007, 2023.

[32] S. H. A. Harbi, L. N. Tidjon, and F. Khomh, “Responsible design patterns for machine learning pipelines,” arXiv preprint arXiv:2306.01788, 2023.

[33] A. Nikanjam, M. M. Morovati, F. Khomh, and H. Ben Braiek, “Faults in deep reinforcement learning programs: a taxonomy and a detection approach,” Automated software engineering, vol. 29, no. 1, p. 8, 2022.

[34] L. N. Tidjon and F. Khomh, “The different faces of ai ethics across the world: A principle-to-practice gap analysis,” IEEE Transactions on Artificial Intelligence, 2022.

[35] A. E. Jim Ormond. (2023) World’s largest association of computing professionals issues principles for generative ai technologies. [Online]. Available: https://www.acm.org/binaries/content/assets/public-policy/principles-generative-ai.pdf

[36] F. Hutter, L. Kotthoff, and J. Vanschoren, Automated machine learning: methods, systems, challenges. Springer Nature, 2019.

[37] G. A. Lewis, I. Ozkaya, and X. Xu, “Software architecture challenges for ml systems,” in 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2021, pp. 634–638.

[38] A. Aleti, “Software testing of generative ai systems: Challenges and opportunities,” arXiv preprint arXiv:2309.03554, 2023.

[39] L. N. Tidjon and F. Khomh, “Threat assessment in machine learning based systems,” arXiv preprint arXiv:2207.00091, 2022.

[40] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, “Software engineering for ai-based systems: a survey,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 31, no. 2, pp. 1–59, 2022.

[41] B. Meyer, Object-oriented software construction. Prentice hall Englewood Cliffs, 1997, vol. 2.

[42] (2023) Mitigating bias in artificial intelligence. [Online]. Available: https://haas.berkeley.edu/equity/industry/playbooks/mitigating-bias-in-ai/

[43] J.-M. John-Mathews, D. Cardon, and C. Balagué, “From reality to world. a critical perspective on ai fairness,” Journal of Business Ethics, vol. 178, no. 4, pp. 945–959, 2022.

[44] (2023) Fairness in machine learning. [Online]. Available: https://github.com/fairlearn/fairlearn/what-we-mean-by-fairness

[45] M. Lee, P. Liang, and Q. Yang, “Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities,” in Proceedings of the 2022 CHI conference on human factors in computing systems, 2022, pp. 1–19.

[46] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, “Llm is like a box of chocolates: the non-determinism of chatgpt in code generation,” arXiv preprint arXiv:2308.02828, 2023.

[47] A. F. Cooper, J. Frankle, and C. De Sa, "Non-determinism and the lawlessness of machine learning code," in Proceedings of the 2022 Symposium on Computer Science and Law, 2022, pp. 1–8.
[48] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 11, no. 3, pp. 1–41, 2020.
[49] M. H. Asyrofi, Z. Yang, I. N. B. Yusuf, H. J. Kang, F. Thung, and D. Lo, "Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems," IEEE Transactions on Software Engineering, vol. 48, no. 12, pp. 5087–5101, 2021.
[50] S. Segura, A. Durán, J. Troya, and A. R. Cortés, "A template-based approach to describing metamorphic relations," in 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET). IEEE, 2017, pp. 3–9.
[51] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar et al., "Universal sentence encoder," arXiv preprint arXiv:1803.11175, 2018.
[52] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," SIAM Journal on discrete mathematics, vol. 17, no. 1, pp. 134–160, 2003.
[53] A. E. Roth, The Shapley value: essays in honor of Lloyd S. Shapley. Cambridge University Press, 1988.