Metamorphic Testing of Large Language Models for Natural Language Processing

Steven Cho
University of Auckland
Auckland, New Zealand
steven.cho@auckland.ac.nz

Stefano Ruberto
JRC European Commission
Ispra, Italy
stefano.ruberto@ec.europa.eu

Valerio Terragni
University of Auckland
Auckland, New Zealand
v.terragni@auckland.ac.nz

***Abstract*—Using Large Language Models (LLMs) to perform Natural Language Processing (NLP) tasks has been becoming increasingly pervasive in recent times. The versatile nature of LLMs makes them applicable to a wide range of such tasks. While the performance of recent LLMs is generally outstanding, several studies have shown that LLMs can often produce incorrect results. Automatically identifying these faulty behaviors is extremely useful for improving the effectiveness of LLMs.**

One obstacle to this is the limited availability of labeled datasets, necessitating an oracle to determine the correctness of LLM behaviors. Metamorphic Testing (MT) is a popular testing approach that alleviates this oracle problem. At the core of MT are Metamorphic Relations (MRs), defining the relationship between the outputs of related inputs. MT can expose faulty behaviors without the need for explicit oracles (e.g., labeled datasets).

This paper presents the most comprehensive study of MT for LLMs to date. We conducted a literature review and collected 191 MRs for NLP tasks. We implemented a representative subset (36 MRs) to conduct a series of experiments with three popular LLMs, running ∼560K metamorphic tests. The results shed light on the capabilities and opportunities of MT for LLMs, as well as its limitations.

***Index Terms*—large language models, metamorphic testing, machine learning testing, NLP, Software Engineering for AI**

I. Introduction

Large Language Models (LLMs) have rapidly gained traction in many applications due to their impressive natural-language understanding and generation capabilities [1]. However, concerns on the reliability and trustworthiness of LLMs persist—specifically, biases, hallucinations, and other faulty behaviors remain significant challenges [2], [3]. Exposing such issues is crucial as it is the first step for fixing them [4], [5].

Automated testing is crucial for evaluating the quality of LLM outputs [2]–[5]. It also plays a key role in the feedback loop that continually improves LLMs over time. When testing uncovers unwanted behaviors, targeted interventions—such as reinforcement learning [6], alignment adjustments [7], policy updates [8], focused fine-tuning [9], and other adaptive techniques—can be applied to substantially enhance a model's reliability [5]. Moreover, automated testing of LLMs is becoming increasingly important in industry contexts, where many organizations are transitioning away from generic, public (and supposedly well-tested) LLM services (e.g., OpenAI API) toward private, locally deployed, fine-tuned models [10], [11]. This shift is often driven by privacy, security, and compliance requirements, or simply by the need for better performance in company-specific environments [11]. By rigorously testing these locally hosted models, organizations can ensure they meet performance standards while avoiding exposure of sensitive data to third-party services.

One of the primary challenges in automatically testing LLMs is the **oracle problem** [12]; *the problem of distinguishing between correct and incorrect test executions*. In **Natural Language Processing (NLP)**, obtaining new text inputs for testing is straightforward thanks to abundant textual data. However, determining output correctness is far more complex. Typically, human-created labels serve as the "oracles", indicating correct or incorrect outputs. Because human-labeled datasets for NLP are limited and expensive to obtain, there is a pressing need for effective automated test oracles [3] that can evaluate correctness of textual outputs without relying on human-created labels.

Metamorphic Testing (MT) [13] is a widely used approach to alleviate the oracle problem. At the core of MT are **Metamorphic Relations (MRs)**, which define relationships between the outputs of related inputs. This approach is based on the intuition that, even if we cannot automatically determine the correctness of the output for an individual input, *we can use the relationships among the expected outputs of multiple related inputs as a test oracle* [14]. Metamorphic testing has been used extensively in NLP, across a wide range of tasks [15]–[19]. However, applying MT to LLMs is still an understudied problem [20] despite its several potential benefits.

Figure 1 shows an example of faulty behavior we detected in **GPT-4** using MT. The example involves *natural language inference* [21], a core NLP task that determines the relationship between a premise and a hypothesis. It classifies it as either `entailment` (hypothesis logically follows from the premise), `contradiction` (hypothesis is logically incompatible with the premise), or `neutral` (no clear logical relation). The MR in Figure 1 states that, *given two inputs where one is a paraphrase of the other, the inference results should be the same*. We create this input pair by applying a transformation on the first input to generate the second. In this case, the LLM responds `neutral` for the first input and `entailment` for the second, violating the metamorphic relation[1]. Notably,

[1]To note, the LLM's `neutral` response is correct, as the premise lacks evidence that the man is performing. He could be playing at home, for instance

This is the authors' version of the paper that was accepted for publication in the research track of the 41st IEEE International Conference on Software Maintenance and Evolution (ICSME 2025)

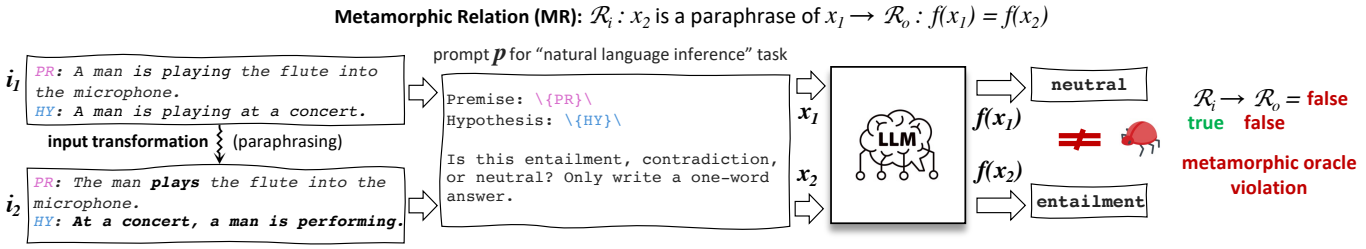

Fig. 1. Real faulty behaviour detected by LLMORPH in the OpenAI `gpt-4-1106` model.

we can identify the faulty behavior without needing to know the expected results for either input. This highlights the key advantage of MT: the same MRs can be applied to arbitrary inputs[2], enabling automated testing with a vast amount of data. This is crucial for testing LLMs, as faulty behavior often manifests only for specific inputs [5].

Hyun et al. [20] recently presented the only work on MT for LLMs on NLP tasks, showing its potential. They presented the framework METAL implementing 13 MRs and evaluated on six tasks on 3 LLMs. However, questions remain about the effectiveness of MT for LLMs, as well as their true positive rate and fault detection compared to traditional ML testing. Further research is needed to fully understand the effectiveness and limitations of MT for LLMs.

To address these gaps, this paper presents the most comprehensive study on MT for LLMs. We conducted the first systematic literature search on MRs for NLP, reviewing 1,024 papers and identifying 44 that explicitly defined MRs. This resulted in a catalog of 191 unique MRs across 24 tasks. We then developed **LLMORPH**, an automated framework for applying MT to LLMs, implementing a representative subset of 36 MRs. Finally, we conducted large-scale experiments using LLMORPH on three LLMs (GPT-4, LLAMA3, and HERMES 2) and four datasets, leading to 561,267 test executions. Our experiments yielded four key findings.

First, MT effectively exposes faulty behaviors, with an average failure rate of 18%. While a ground-truth oracle detects more faults, it requires costly manual labeling. MT, however, works on unlabeled data and identifies 11% of failures missed by traditional testing, highlighting its complementarity.

Second, a manual analysis of 937 metamorphic oracle violations showed an average true positive (FP) rate of around 60%, with most false positives arising from the intrinsic limitations of MT for NLP. The FP rate aligns with traditional MT for NLP, showing no LLM-specific issues, and remains unavoidable even with LLM-based input transformations.

Third, MR effectiveness depends on the relation and task, with minimal variation across LLMs. While the FP rate can be high, many MRs maintain an acceptable FP rate when applied to LLMs, making them practical for testing. In particular, some MRs are consistently more effective than others, with high failure rate and low false positive rate (e.g., MRs 9, 142, 154). Developers could prioritise these during testing.

[2]Inputs that satisfy the input relation, $\mathcal{R}_i$. See Section II.

Finally, our results show that several MRs are task-independent and thus useful for evaluating fine-tuned LLMs. Additionally, LLM flakiness is not a major concern.

In summary, this paper makes the following contributions:

- An exhaustive catalog of 191 MRs for NLP tasks[3].
- LLMORPH, a framework for performing MT on LLMs, implementing 36 MRs, of which can be easily extended.
- A series of experiments that shed light on the capabilities and limitations of MT for LLMs.
- Release of the source code for LLMORPH[4] and all experimental data[5] to foster future work in this area

II. METAMORPHIC TESTING FOR LLMS

This section provides the background of this work and formulates the problem of MT for LLMs. We follow the traditional definition of MRs by Chen et al. [13], [14]:

Definition 1: Let f be a target algorithm. A **Metamorphic Relation (MR)** is a property of f involving multiple inputs $\langle x_1, \ldots, x_n \rangle$ and their outputs $\langle f(x_1), \ldots, f(x_n) \rangle$, with $n \geq 2$. Formally[6], an MR is expressed as a logical implication $\mathcal{R}_i \Rightarrow \mathcal{R}_o$ from an **input relation** $\mathcal{R}_i$ to an **output relation** $\mathcal{R}_o$

$$\mathcal{R}_i\,(x_1, \cdots x_n) \Rightarrow \mathcal{R}_o(f(x_1), \cdots f(x_n))$$

Whenever a given input relation $\mathcal{R}_i$ holds between two or more inputs, a corresponding output relation $\mathcal{R}_o$ is expected to hold between the outputs. If $\mathcal{R}_i$ is true, then $\mathcal{R}_o$ should be true. Thus, an MR can be used as a metamorphic test oracle.

Definition 2: Given an MR ($\mathcal{R}_i \Rightarrow \mathcal{R}_o$), a **metamorphic test oracle** is an executable Boolean expression that reports a faulty behavior if, for a specific set of inputs, the input relation $\mathcal{R}_i$ is satisfied (true), but the output relation $\mathcal{R}_o$ is not (false).

For example, the MR of Figure 1 is formulated as: $\mathcal{R}_i :=$ *x_2 is a paraphrase of x_1* $\Rightarrow \mathcal{R}_o := f(x_1) = f(x_2)$. It reports a faulty behaviour in Figure 1 because the input relation is satisfied while the output relation is not.

Typically, metamorphic test cases are generated by first obtaining an initial test input x_1 (existing or generated), then

[3]https://mt4nlp.github.io/

[4]https://github.com/steven-b-cho/llmorph

[5]https://doi.org/10.5281/zenodo.16526643

[6]A more general definition of MRs allows inputs and outputs to also appear in the output and input relations, respectively [14]. However, we use the traditional definition (Definition 1) as it covers almost all collected MRs. Exceptions are MRs 115-119, 165-166.

TABLE I
TOP PUBLICATIONS VENUES OF THE REVIEWED PUBLICATIONS

Venue	# Papers
International Conference on Software Engineering (ICSE)	5
International Conference on Automated Software Engineering (ASE)	5
International Conf. on the Foundations of Software Engineering (FSE)	4
arXiv	4
International Workshop on Metamorphic Testing (MET)	3
ACM Transactions on Software Eng. and Methodology (TOSEM)	2
International Computer Software and Applications Conf. (COMPSAC)	2
Mathematics	2
Others	17
Total	**44**

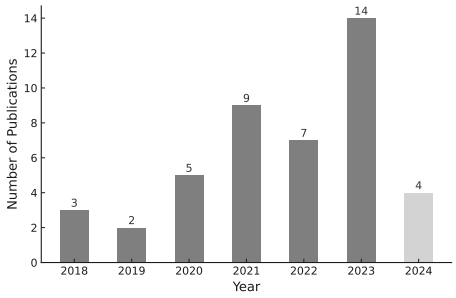

Fig. 2. Number of the 44 papers published each year from 2018 to 2024

applying a transformation [22] $\rightsquigarrow$ to x_1 to create new inputs that satisfy the input relation [23]. In such cases, the initial test input is called the **source test input** (or case), which is x_1 in Figure 1. The case(s) derived from it to satisfy the input relation are called the **follow-up test inputs** (or cases), which is x_2 in the example. We can thus formalise this as:

Definition 3: Given an MR ($\mathcal{R}_i \Rightarrow \mathcal{R}_o$) and a test input x_1, a **(metamorphic) input transformation** [24] $\rightsquigarrow$ is a transformation $x_1 \rightsquigarrow x_2$ that satisfies R_i (i.e., $x_1 \rightsquigarrow x_2 : \mathcal{R}_i(x_1, x_2) = \text{true}$)

Given a single MR, MT can consider an arbitrary number of source test inputs and use the *input transformation* to automatically create the corresponding follow-up test inputs. MT executes the function under test on each pair of inputs and reports an oracle violation if the output relation is false.

To apply **MT for LLMs**, we need a few considerations. First, we consider only the use of LLMs for NLP—that is, both the input x and the output $f(x)$ are natural language text. Second, as our target algorithm f is an LLM, it only provides next token prediction. To perform a task, LLMs require specific **prompts** (instructions). In the example in Figure 1, the prompt p is given for the natural language inference task, and is part of the input x to execute the task with an LLM. To obtain comparable results among outputs, the prompt must remain consistent between the source and follow-up inputs; it should not be altered by the input transformation. LLM prompt perturbation is a separate topic [25] that falls outside the scope of this work. Formally, an input x is a tuple $\langle i, p \rangle$, where i is the textual input and p is the prompt (instructions) on how to perform the NLP task using the input i. The input transformation $\rightsquigarrow (\mathcal{R}_i)$ only modifies (predicates on) the textual input i (see Figure 1).

III. LITERATURE SEARCH OF MRS

Our objective is to test LLMs using MRs for tasks involving natural language input and output. To that end, we systematically search for literature applying MT in natural language-based systems to create a comprehensive list of such relations.

Literature search: For our search, we follow the ACM/SIGSOFT systematic review standards[7] as closely as possible, though though some procedures are inapplicable as our goal was is search, not a review. As MT for NLP and LLMs is a relatively new area, we aimed for comprehensiveness and queried Google Scholar, which indexes both formal digital libraries (e.g., ACM, IEEE) [26] and preprint servers (e.g., arXiv). We used the following keyword search:

["metamorphic test" ∨ "metamorphic testing" ∨ "metamorphic relation(s)"] ∧ ["llm(s)" ∨ "language model" ∨ "nlp" ∨ "natural language processing"]

We include "`llm`" and "`language model`" as we wish to also find MRs that have already been used for testing LLMs and other language models. We searched papers indexed from 1 January 1998, the year when metamorphic testing was first introduced [13], to 30 June 2024. This resulted in 1,024 papers.

Literature selection: The literature search we performed was intentionally broad, as we intended to gather as many MRs as possible. Many of the papers chosen were thus out of scope. To filter these, we included only papers that use MT to test a natural language-based system explicitly mentioning the metamorphic relations used. We excluded papers not written in English, and those that only test for fairness or biases, which are outside our scope.

After this inspection, we performed backward and forward snowballing, examining references to identify any relevant papers we missed. The final result is **44 papers**.

Literature analysis: Table I and Figure 2 show the top publication venues and years of the 44 papers found, respectively. We did not find any papers that met our criteria published before 2018, despite MT being introduced 20 years earlier. This may indicate that the application of MT on NLP has only recently begun to be explored. Figure 2 shows an upward trend in the number of published papers over time, indicating growing interest and research activity in the field.

Table I shows that the most frequent publication venues are ICSE, ASE and FSE, the three top SE conferences. This is logical, given that MT is fundamentally a software testing concept. Four papers (9%) were published on ARXIV without a corresponding peer-reviewed conference or journal publication. However, given that the majority of the selected papers were published in peer-reviewed venues, we believe that the quality of the results is not compromised.

Catalog of metamorphic relations: Finally, we analysed the 44 papers and extracted **191 unique MRs** related to NLP. To

[7]https://www2.sigsoft.org/EmpiricalStandards/docs/standards?standard=SystematicReviews

TABLE II
191 METAMORPHIC RELATIONS FOR NLP TASKS.

ID	Task	Input Relation ($\mathcal{R}_\mathbf{i}$)	Output Relation ($\mathcal{R}_\mathbf{o}$)	Ref.
1	G_a	Replace characters with random	Equivalence	[20]
2	G_a	Delete characters	Equivalence	[20]
3	G_a	Leet format conversion	Equivalence	[20]
4	G_a	Add random characters	Equivalence	[20]
5	G_b	Add spaces	Equivalence	[20]
6	G_b	Swap characters	Equivalence	[20]
7	G_a	Shuffle characters	Equivalence	[20]
8	G_a	Synonym substitution	Equivalence	[20]
9	G_a	Word insertion	Equivalence	[20]
10	G_a	Antonyms substitution	Difference	[20]
11	G_c	Remove sentences	Equivalence	[20]
12	G_c	Replace sentences	Equivalence	[20]
13	G_d	Category-based substitution (names, pronouns)	Equivalence	[29]
14	G_d	Category-based substitution (country)	Equivalence	[29]
15	G_d	Category-based substitution (occupation)	Equivalence	[29]
16	G_d	Replace punctuation	Equivalence	[29]
17	G_d	Synonym substitution (noun)	Equivalence	[29]
18	G_d	Synonym substitution (verb)	Equivalence	[29]
19	G_d	Shuffle sentences	Equivalence	[29]
20	G_e	Append full stops to words or sentences	Equivalence	[32]
21	CM	Substitute characters with visually similar characters	Equivalence	[33]
22	CM	Split characters into visually similar characters	Equivalence	[33]
23	CM	Combine characters into visually similar characters	Equivalence	[33]
24	CM	Add random characters	Equivalence	[33]
25	CM	Replace characters with special symbols	Equivalence	[33]
26	CM	Swap characters	Equivalence	[33]
27	CM	Translate some words to other languages	Equivalence	[33]
28	CM	Homophone substitution	Equivalence	[33]
29	CM	Abbreviation substitution	Equivalence	[33]
30	CM	Split words into sub-words	Equivalence	[33]
31	CM	Add many irrelevant sentences	Equivalence	[33]
32	DS	Synonym substitution	Equivalence	[35]
33	DS	Category-based substitution (numbers)	Equivalence*	[35]
34	DS	Remove keywords	Difference	[35]
35	DS	Replace keywords with unrelated terms	Difference	[35]
36	DS	Change order of actions	Difference	[35]
37	DS	Back translate	Equivalence	[37]
38	DS	Word insertion	Equivalence	[37]
39	DSp	Synonymous sentence substitution in persona	Equivalence	[38]
40	DSp	Persona substitution	Equivalence	[38]
41	DSp	Swap characters in only persona	Equivalence	[38]
42	DSp	Swap characters in only input	Equivalence	[38]
43	DSp	Swap characters in both persona and input	Equivalence	[38]
44	CR	Synonym, antonym and mask substitution	Consistency	[40]
45	FN	Back translate	Equivalence	[41]
46	FN	Shuffle sentences	Equivalence	[42]
47	FN	Shuffle words within a sentence	Difference in confidence	[42]
48	FN	Add/remove negation	Difference	[42]
49	FN	No transformation	Equivalence	[42]
50	FN	Give input twice	Equivalence	[42]
51	FN	Paraphrase text	Equivalence	[42]
52	FN	Split a false x_1 into multiple inputs	At least one output is false	[42]
53	FN	Combine two inputs in which at least one is false	Combination is false	[42]
54	LSR	$f(x_1, x_2)$ and $f(x_2, x_3)$ are true	$f(x_1, x_3)$ is true	[43]
55	NER	Mask substitution (verbs, adjectives, noun phrases)	Equivalence	[44]
56	NER	Mask substitution (noun phrases)	Equivalence	[44]
57	NER	Declarative sentence into interrogative sentence	Is a declarative sentence	[44]
58	NER	Shuffle same-category entities in a sentence	Equivalence	[44]
59	NER	Transform company names to sub-companies	Recognise as sub-company	[45]
60	NER	Swap entities with the same type but different identities	Identify the new identities	[45]
61	NER	Add sentence to another sentence	Union of individual inputs	[46]
62	NER	Add sentence to a paragraph	Union of individual inputs	[46]
63	NER	Add paragraph to an article	Union of individual inputs	[46]
64	NER	Add list of random words to another list of random words	Union of individual inputs	[46]
65	NER	Remove list of words from sentence	x_1 minus removed output	[46]
66	NER	Remove sentence from paragraph	x_1 minus removed output	[46]
67	NER	Remove paragraph from article	x_1 minus removed output	[46]
68	NER	Remove some words from list of random words	x_1 minus removed output	[46]
69	NER	Shuffle paragraphs in an article	Equivalence	[46]
70	NER	Shuffle words in a list of random words	Equivalence	[46]
71	NLI	Synonym substitution (in both premise and hypothesis)	Equivalence	[47]
72	NLI	Synonym substitution (in premise)	Equivalence	[47]
73	NLI	Synonym substitution (in hypothesis)	Equivalence	[47]
74	NLI	Substitute gendered words with words of same gender	Equivalence	[47]
75	NLI	Substitute gendered words with words of different gender	Equivalence	[47]
76	NLI	Add negation (in hypothesis)	Opposite	[47]
77	NLI	Conjoin two premises that both imply hypothesis	Equivalence	[47]
78	NLI	Conjoin two hypotheses that are both implied by premises	Equivalence	[47]
79	NLI	x_1 implies x_2 and x_2 implies x_3	$f(x_1, x_3)$ is not contr.	[47]
80	NLI	x_1 implies x_2 and x_1 implies x_3	$f(x_2, x_3)$ is not contr.	[47]
81	PST	Change position of 'because' clause	Equivalence	[56]
82	PST	Change position of 'when' clause	Equivalence	[56]
83	PST	Add irrelevant sentence to beginning of input	Equivalence	[56]
84	PST	Add irrelevant sentence to end of input	Equivalence	[56]
85	PD	Paraphrase text	Equivalence	[57]
86	PDq	Swap first names and surnames	Equivalence	[58]
87	PDq	Reverse order of query	Equivalence	[58]
88	PDq	Swap pairs of numbers	Equivalence	[58]
89	PDq	Abbreviate words	Equivalence	[58]
90	PDq	Category-based substitution (nationality, gender, numbers)	Difference	[58]
91	PDq	Category-based substitution (gender, numbers)	Difference	[58]
92	PDq	Category-based substitution (numbers)	Difference	[58]
93	QAm	Numeric magnitude change	Equivalence	[27]
94	QAm	Numeric precision change	Equivalence	[27]
95	QAm	Synonym substitution (industry-specific)	Equivalence	[27]
96	QAm	Swapping order of options	Equivalence	[27]
97	QAm	Add negation (for multi-choice)	Difference	[27]
98	QAm	Add background for testing industrial sectors to question	Equivalence	[27]
99	QAm	Add security instruction to question	Equivalence	[27]
100	QAm	Add irrelevant option	Equivalence	[27]
101	QAc	Semantic invariance	Equivalence	[28]
102	QAc	Capitalisation (in question)	Equivalence	[19]
103	QAc	Rephrase comparative sentence (in question)	Equivalence	[19]
104	QAc	Replace comparative word with antonym (in question)	Difference	[19]
105	QAc	Replace subject with unrelated noun (in question)	Difference	[19]
106	QAc	Capitalisation (in context)	Equivalence	[19]
107	QAc	Reverse order of sentences (in context)	Equivalence	[19]
108	QAc	Add irrelevant sentences (in context)	Equivalence	[19]
109	QAc	Remove irrelevant sentences (in context)	Equivalence	[19]
110	QAc	Category-based substitution (numbers) (in context)	Equivalence*	[19]
111	QAc	Add irrelevant sentence (to question)	Equivalence	[30]
112	QAc	Add irrelevant sentence (to context)	Equivalence	[30]
113	QAc	Combine two questions and contexts	Answers both questions	[30]
114	QAc	Combine two questions and contexts	Answers first question	[30]
115	QA	Given a wh-ques. and answer, derive a wh-ques.	Consistency	[31]
116	QA	Given a wh-ques. and answer, derive a gen-ques.	Consistency	[31]
117	QA	Given a gen- or alt-question and answer, derive a wh-ques.	Consistency	[31]
118	QA	Given a wh-ques., answer and extra info., derive a wh-ques.	Consistency	[31]
119	QA	Given a wh-ques., answer and extra info., derive a gen-ques.	Consistency	[31]
120	QA	Back translate	Equivalence	[34]
121	QA	Change position of adverbial clause	Equivalence	[34]
122	QA	Word insertion	Equivalence	[34]
123	QA	Synonym substitution	Equivalence	[34]
124	QA	Alias substitution	Equivalence	[34]
125	QA	Abbreviation replacement	Equivalence	[34]
126	QA	Keyboard mistake	Equivalence	[34]
127	QA	Misspelled word	Equivalence	[34]
128	QA	OCR recognition error	Equivalence	[34]
129	QA	Repeated question mark	Equivalence	[34]
130	QAb	Antonym substitution (first adjective)	Difference	[36]
131	QAb	Change tense	Difference	[36]
132	QAb	Substitute between 'before' and 'after'	Difference	[36]
133	QAb	Add negation (for boolean)	Difference	[36]
134	QAb	Synonym substitution (adjective)	Equivalence	[36]
135	QAb	Change adverbial position	Equivalence	[36]
136	QAb	Change to active/passive voice	Equivalence	[36]
137	RE	Category-based substitution	Equivalence	[39]
138	RE	Replace head/tail with coarser-grained entity	Same or coarser relation	[39]
139	RE	Replace head/tail with co-related entity of different type	Equivalence	[39]
140	RE	Replace head/tail with coreferential entity	Equivalence	[39]
141	RE	Swap entities if they have a symmetrical relation	Equivalence	[39]
142	RE	Swap entities if they have an asymmetrical relation	Opposite relation	[39]
143	RE	$f(x_1, x_2)$ is asymmetrical and $f(x_1, x_3)$ is symmetrical	$f(x_3, x_2)$ equals $f(x_1, x_2)$	[39]
144	RE	$f(x_1, x_2)$ is asymmetrical and $f(x_2, x_3)$ is symmetrical	$f(x_1, x_3)$ equals $f(x_1, x_2)$	[39]
145	SA	x_1 is more pos./neg. than x_2, add random sentence to both	x_3 more pos./neg. than x_4	[43]
146	SA	Add positive/negative text	More positive/negative	[28]
147	SA	Abbreviation / contraction substitution	Equivalence	[18]
148	SA	Synonym substitution (nouns)	Equivalence	[18]
149	SA	Change singular/plural	Equivalence	[18]
150	SA	Change '.' to '!'	Stronger	[18]
151	SA	Add emphasising adverbs	Stronger	[18]
152	SA	Add negation	Difference	[18]
153	SA	Reverse case of all characters	Equivalence	[18]
154	SA	Make nouns and adjectives uppercase	Stronger	[18]
155	SA	Change tense	Equivalence	[18]
156	SA	Swap phrases around joining words	Equivalence	[18]
157	SA	Substitute between 'although' and 'but'	Equivalence	[18]
158	SA	Rephrase comparative sentence	Equivalence	[18]
159	SA	Swap comparative objects	Difference	[18]
160	SA	Shuffle sentences	Equivalence	[18]
161	SA	Group sentences of same sentiment together	Equivalence	[18]
162	SA	Group sentences by sentiment and confidence	Equivalence	[18]
163	SA	Sequentially add positive sentences	Gradually more positive	[18]
164	SA	Sequentially add negative sentences	Gradually more negative	[18]
165	SA	Remove all sentences with same label as document	Difference	[18]
166	SA	Remove all sentences with different label than document	Equivalence	[18]
167	SD	x_1 implies x_2 and x_1 implies x_3	x_2 implies x_3	[48]
168	TS	Deriving an abstract sentence/summary from the input	Difference	[28]
169	TR	Translate to a different language first	Equivalence	[49]
170	TR	Category-based substitution	Equivalence*	[50]
171	TR	Context-similar substitution	Equivalence	[51]
172	TR	Mask substitution (nouns, adjectives)	Structured format is similar	[52]
173	TR	Replace words with random	Difference	[53]
174	TR	Remove words	Difference	[53]
175	TR	Reverse case of some characters	Equivalence	[54]
176	TR	Reverse case of all characters	Equivalence	[55]
177	TR	Remove full stops	Equivalence	[55]
178	TR	Replace punctuation	Equivalence	[54]
179	TR	Add noise	Equivalence	[55]
180	TR	Category-based substitution (subject)	Equivalence*	[55]
181	TR	Synonym substitution (verb)	Equivalence	[55]
182	TR	Category-based substitution (object)	Equivalence*	[55]
183	TR	Random word substitution (word at end of sentence)	Equivalence*	[54]
184	TR	Translate to target language and back first	Equivalence	[54]
185	TR	Place a phrase in different contexts	Phrase is equivalent	[59]
186	TR	Insert adjunct sentence	Parse tree of x_1 found in x_2	[60]
187	TR	Place a phrase in different contexts then back translate	Phrase is equivalent	[15]
188	TR	Back translate	Proper nouns are the same	[61]
189	TR	Construct word closures	Word closure comparison	[62]
190	TR	Convert to tree then prune	Similarity of trees	[63]
191	TR	Back translate	Equivalence	[64]

[a] To improve table readability we grouped the tasks SA, TC, TD, QA, SM, IR into G_a, SA, TC, TD, QA into G_b, SM, IR into G_c, TS, SM, TC into G_d, and TR, NER into G_e

[b] Equivalence* means equivalence except for substitute

[c] Four papers: [31], [35], [42], [54]) are extensions of others: [16], [41], [55], [65]) of the same authors and share many MRs. Therefore, these shared MRs are attributed only to the latest paper.

[d] If we found the same MR in multiple papers applied on the same task we cite the most recent one.

TABLE III
DESCRIPTION AND BREAKDOWN OF THE NATURAL LANGUAGE PROCESSING (NLP) TASKS OF THE MRS IN TABLE II

ID	Task Name	Task Description	# MRs
CM	Content moderation	Detecting inappropriate or harmful content in a text	11
CR	Coreference resolution	Determining which words in a text refer to the same entity	1
DS	Dialogue system	Conversing with humans	7
DSp	Dialogue system (persona-based)	Conversing with humans while maintaining a consistent personality	5
FN	Fake news detection	Detecting whether a text has false or misleading information	9
IR	Information retrieval	Finding relevant information from a text	10
LSR	Lexical semantic relations	Analysing relationship between words	1
NER	Named entity recognition	Identifying and classifying words into categories	17
NLI	Natural language inference	Determining whether a hypothesis follows from a premise	10
PD	Plagiarism detection	Detecting instances of copied or unoriginal text	1
PDq	Plagiarism detection (query-based)	Detecting instances of copied or unoriginal text using a specific query	7
PST	Part-of-speech-tagging	Classifying words into grammatical categories	4
QA	Question answering	Answering questions in natural language, using no context	25
QAb	Question answering (boolean)	Answering true or false questions, using no context	7
QAc	Question answering (incl. context)	Answering questions in natural language, using context	14
QAm	Question answering (multi-choice)	Answering multi-choice questions, using no context	8
RE	Relation extraction	Identifying relationships between two entities in a text	8
SA	Sentiment analysis	Determining how positive or negative a text is	32
SD	Stance detection	Determining the agreement between two texts	1
SM	Summarisation	Condensing text while retaining key information	17
TC	Text classification	Determining whether a text falls into predefined categories	17
TD	Toxicity detection	Identifying whether a text is offensive or harmful	10
TR	Translation	Converting text from one language to another	24
TS	Text similarity	Determining the similarity between two texts	8

do so, the first author read each selected paper and identified all defined MRs, ensuring understanding of the MR, including any specific notation. They then reformulated the MR using our notation (for consistency) and wrote a brief description. This process ensured that all MRs in each paper were captured. Afterwards, to ensure consistency and correctness, all authors reviewed the collated list. We collectively discussed each MR, removed duplicates, and refined descriptions; and, when necessary, we revisited the papers together to ensure accurate interpretation and categorization until reaching a consensus.

The collected MRs are summarised in Table II, with the NLP task descriptions and breakdown in Table III. Table II presents a unique ID assigned to each MR, the input relation ($\mathcal{R}_i$), the output relation ($\mathcal{R}_o$), the NLP task for which the MR was presented, and the publication source. Note that, for improved table readability, the input relation ($\mathcal{R}_i$) implicitly describes the relation between the source and follow-up input(s), and the output relation ($\mathcal{R}_o$) among their outputs. A formal description of the 191 MRs can be found in our supplementary data.

MR discussion: We see that the most common output relation is `Equivalence`. This is perhaps the simplest relation an input relation could imply, leading to its prevalence. In the context of NLP, `Equivalence` can be defined in two ways: syntactic and semantic. *Syntactic equivalence* occurs when the two values have the same structure or form, while *semantic equivalence* refers to when they convey the same meaning, regardless of structure. The choice between syntactic and semantic equivalence depends on the task. For instance, NLI with its triary output would use syntactic equivalence (see Figure 1), while free-form QA would use semantic equivalence (often implemented with BERT-like word embeddings [19], [30]). Similarly, the second most common output relation is `Difference`, and follows the same principle.

Many MRs share commonalities, but are distinct enough to be considered separate. One example is the varying levels of granularity that different papers use in defining MRs. For instance, MR-8 [20] involves creating semantically similar sentences by substituting words with their synonyms. MR-17 [27] does this also, but only substitutes nouns. MR-72 [47] adapts this specifically for NLI, substituting words only in the `premise` of the input. These variations may have different fault detection capabilities, and so are kept separate.

Task discussion: We identified 24 distinct NLP tasks, described in Table III. It is important to mention that while some MRs are specific to certain tasks (e.g., MR-188 is specific to Translation (TR)), others have been applied to multiple tasks. For example, we found that MR-1 to MR-20 were applied across various tasks, with MR-1 to MR-4 being applied to the highest number (SA, TC, TD, QA, SM, IR) (see *footnote*[a] of Table II for the description of grouped notation G_x). Moreover, Table II associates tasks to MRs based on their use in the papers we reviewed, not necessarily the tasks to which they could be applied in other contexts. Indeed, MR-1 to MR-4 are presumably general enough to be applied to any NLP task.

To note, the number of MRs for a task (Table III) is not an indication of how well-studied that task is with MT. For instance, sentiment analysis (SA)—with the highest number of MRs—derives 20 out of its 32 relations from a single paper (Table II), the rest from only three others. In contrast, Translation (TR) sources its 24 MRs from 14 different studies.

IV. EXPERIMENTS

We conducted a series of experiments to address three research questions:

RQ1 Failure Rate. *To what extent does MT exhibit failures in LLMs?*

RQ2 Comparison. *How does MT compare with traditional testing that uses labeled data?*

RQ3 Manual Validation. *What is the true positive rate of the reported failures?*

RQ1 evaluates MT's effectiveness at detecting oracle violations by running 561,267 test groups (source and follow-up pairs) from 36 MRs on three popular LLMs. **RQ2** compares the results with the source input's ground truth provided by the datasets. Although MT is typically not applied when oracles (e.g., labeled data) are present, we used labeled data to better understand MT's fault detection capabilities. **RQ3** assesses MT's reliability through manual analysis of detected failures, identifying true positives and limitations.

Chosen tasks and relations: We selected four tasks from Table III—question answering with context (QAc), natural language inference (NLI), sentiment analysis (SA), and relation extraction (RE)—as they provide a good range to evaluate LLMs' language understanding.

In Table II, we categorized the MRs based on the tasks they were proposed for. However, this does not mean these MRs cannot be applied to other tasks; indeed, several of the relations were presented for multiple tasks or explicitly stated to be generally applicable. Consequently, we chose **36 metamorphic relations to implement**, spread across the four selected tasks, including MRs not necessarily originally proposed for that task. We selected these based on: (i) inclusion in METAL [20] (MRs 1–12), (ii) common use across tasks, (iii) MRs proposed for one task but suited to many, (iv) task-specific MRs, and (v) ease of understanding to reduce misimplementation. The first five columns of Table V show the implemented MRs and the tasks to which we applied them.

LLMORPH implementation: To implement the input transformations $\rightsquigarrow$ of the 36 MRs, we use two methods: function-based, and LLM-based. For simple procedures, we use traditional function-based methods to generate follow-up inputs (for MRs 1-7, 9, 19, 49, 84, 102, 120, 126, and 128). For instance, randomly adding a sentence to the end of the input (MR-84) was done through concatenation, obtaining the sentences from an online source[8]. As another instance, introducing keyboard mistakes (MR-126) was done through the NLPAug library[9].

For more complex cases, we utilise an LLM (HERMES 2). This is done via few-shot prompting. For example, we used the prompt `Paraphrase the following text: "{TEXT}" Only output the changed text, nothing else.` for the MR in Fig. 1. LLMs, unlike traditional techniques, consistently outputs text that is grammatically correct and thus viable test cases [66]. Without using LLMs, one would need check for grammatical consistency and text sensicality [67].

While we did not formally validate HERMES 2 for input transformation, preliminary tests showed reasonable alignment with the MRs. GPT-4 may perform better, but HERMES 2 was more practical due to GPT-4 rate limits.

We crafted the LLM prompts for both the tasks and input transformations through an iterative process, following prompt engineering principles [68] and refining them until we felt them sufficiently effective. For the transformation prompts, as we aimed to compare MRs across multiple tasks, we designed the few-shot examples generically to be applicable across different tasks, keeping the prompts and examples the same for all tests. We used zero-shot for the implementation of tasks as we believe this would give more insight into the generalisability of the task performance, as well as minimising bias from any examples we were to choose. While we did not rigorously evaluate the prompts, preliminary checks show that they perform reasonably well for the tasks at hand.

[8]https://randomwordgenerator.com/sentence.php

[9]https://github.com/makcedward/nlpaug

TABLE IV
DATASETS USED IN OUR EXPERIMENTS

NLP Task	Dataset Name	Avg. words	Size	Ref.
QA	SQUAD2	137.0	142,000	[70]
NLI	SNLI	20.4	570,000	[71]
SA	SST2	19.2	70,000	[72]
RE	RE-DOCRED	182.4	4,050	[73]

Although there is the risk of nondeterminism in the transformations performed by HERMES 2 (unfortunately not supporting a random seed), we did not deem this a critical issue. This is because: (i) the stochastic effect is diluted due to applying the same input transformation to a myriad of different inputs; (ii) in preliminary runs, we did not observe significant differences when repeatedly transforming the same input; and (iii) repeating the experiments would require excessive time and resources.

Some tasks have multiple components to their inputs—for example, NLI has a `premise` and a `hypothesis` (see Figure 1). We observe that, for these tasks, oftentimes previous research would apply the same transformation to each component and regard them as separate MRs (for example, MR-71 to 73). This process can be applied to many of the 36 MRs we selected. Thus, for our chosen multiple-component tasks (QA and NLI), we test MRs on every combination when possible. For instance, in QA, we may apply the input transformation on the first component, then the second, then both, resulting in three distinct metamorphic test cases. This effectively increases the variations of those MRs for these tasks as well as the number of metamorphic test groups generated.

To measure semantic equivalence for computing the output relation, we use BERT [69] to assess the similarity between two texts, a common practice in NLP-related MT studies [19], [30]. Specifically, we use PARAPHRASE-MINILM-L6-V2, a popular model on HuggingFace. If the similarity score is above a threshold, we consider the texts to be semantically equivalent. To decrease potential false positives, we selected different thresholds for MRs with equivalence and difference relations. To obtain these thresholds, we initially adopted a threshold of 0.6 (inspired by METAL [20], though they employed a different output comparison method) and applied it to preliminary test data consisting of metamorphic output pairs. Through manual analysis, we selected the thresholds where 75% of the true positives were retained. This resulted in a similarity threshold of 0.8, and a dissimilarity threshold of 0.4.

Datasets: For each task, we selected an existing labeled dataset that is popularly used in the reviewed papers. Table IV provides a summary of the four datasets. Due to the high computational cost of running experiments with LLMs and the large size of the datasets, we sampled them as described in Section IV-E. The sampled instances serve as our source input texts for metamorphic testing. It is important to note that MT does not require ground truth (e.g., labeled data); however, we chose these datasets with labeled data to compare the effectiveness of traditional testing with MT (RQ2).

SQuAD2 [70] is a popular question answering dataset, with five of the six MT for free-form question answering (QA) papers using it [19], [30], [31], [34], [65]. SNLI [71] is a widely used NLI dataset from Stanford University and is used in the primary source of MRs for NLI [47]. SST2 [72] is a popular sentiment analysis dataset from Stanford University. RE-DocRED [73] is a newer version of DocRED, a popular RE dataset. For each input, RE-DocRED provides several possible relations; we randomly selected one for each instance.

LLMs under test: We performed our experiments on three LLMs. These are chosen as they are state-of-the-art models, and are readily available at our Institution.

- `gpt-4-1106`[10] (**GPT-4**) from OpenAI.
- `llama-3.1-70b-instruct`[11] (**LLAMA3**) from META, comparable to GPT-4 in many areas.
- `nous-hermes-2-mixtral-8x7b-dpo`[12] (**HERMES 2**), an open-source model. A fine-tuned version of MIXTRAL8X7B.

Experimental setup: Using our four chosen tasks and 36 relations, LLMORPH takes input data from a dataset and transforms each input into one or more follow-up inputs, forming so-called **metamorphic test groups**—groups composed of a source input and its corresponding follow-up(s). It then checks against the output relation to determine possible metamorphic oracle violations. Throughout, it assesses whether all source/follow-up inputs/outputs adhere to any additional requirements of the MR. If they do not, that metamorphic group is discarded. We run 1,000 instances randomly sampled from each dataset for each task. This is performed for each MR and corresponding task, resulting in 108 task-MR pairs for each LLM. Through this, we get 178,180 unique metamorphic groups. We run these same groups for each LLM to obtain comparable results. In total, discounting discarded groups, we run 561,267 metamorphic groups, spanning four tasks, three LLMs, and 36 MRs.

A. RQ1: Failure Rate

Table V shows the results for RQ1 and RQ2 by MRs, aggregating the outcomes for different tasks and LLMs.

Column "*# groups*" gives the number of metamorphic test groups executed for each MR (range: 2,165–24,217; average: 15,265; median: 20,120). Each metamorphic group $t = \langle x_1, ...x_n \rangle$ involves the inputs/outputs of two or more invocations of the LLM under test (source and follow-up inputs, see Sec. II). The number of groups varies across MRs as some were applied to only one task, and others to more (see Column "*NLP Tasks*"). Additionally, some instances from the datasets did not satisfy the input relation and were thus discarded.

TABLE V
RESULTS BY METAMORPHIC RELATIONS (RQ1 AND RQ2)

	NLP Tasks				RQ1 – failure rate		RQ2 – labelled data			
MR	**QA**	**NLI**	**SA**	**RE**	**# groups**	λ	①	②	③	④
1	•	•	•	•	23,288	0.23	0.53	0.23	0.15	0.08
2	•	•	•	•	23,288	0.19	0.56	0.24	0.12	0.07
3	•	•	•	•	23,256	0.11	0.63	0.27	0.06	0.05
4	•	•	•	•	23,256	0.17	0.58	0.25	0.11	0.06
5	•	•	•	•	23,256	0.12	0.62	0.26	0.07	0.05
6	•	•	•	•	23,288	0.18	0.57	0.25	0.12	0.07
7	•	•	•	•	23,292	0.15	0.60	0.26	0.09	0.06
8	•	•	•	•	24,175	0.19	0.56	0.25	0.11	0.08
9	•	•	•	•	11,216	0.17	0.55	0.29	0.09	0.07
10	•	•	•	•	18,131	0.27	0.41	0.23	0.22	0.14
19	•			•	5,178	0.14	0.46	0.40	0.03	0.11
25	•	•	•	•	23,284	0.21	0.50	0.22	0.19	0.09
34	•	•	•	•	23,284	0.39	0.34	0.14	0.35	0.17
49	•	•	•	•	11,206	0.10	0.59	0.31	0.04	0.05
51	•	•	•	•	23,262	0.17	0.59	0.24	0.10	0.07
57				•	2,165	0.36	0.08	0.56	0.05	0.31
77		•			3,013	0.02	0.70	0.26	0.02	0.02
78		•			3,013	0.01	0.71	0.27	0.02	0.01
79		•			3,013	0.00	0.72	0.27	0.01	0.00
80		•			3,013	0.00	0.72	0.27	0.00	0.00
84	•	•		•	14,217	0.11	0.60	0.29	0.05	0.06
102	•	•	•	•	23,256	0.10	0.63	0.27	0.05	0.05
120	•	•	•	•	23,262	0.16	0.59	0.25	0.09	0.06
126	•	•	•	•	24,211	0.21	0.53	0.26	0.13	0.07
127	•	•	•	•	24,217	0.20	0.54	0.26	0.13	0.07
128	•	•	•	•	24,211	0.15	0.58	0.27	0.09	0.06
136	•	•		•	20,243	0.15	0.58	0.27	0.09	0.06
137	•	•	•	•	11,216	0.37	0.43	0.20	0.21	0.16
141				•	2,165	0.06	0.05	0.47	0.07	0.42
142				•	2,165	0.80	0.01	0.05	0.13	0.82
149	•		•	•	14,223	0.10	0.62	0.28	0.05	0.05
150			•		3,013	0.13	0.72	0.13	0.12	0.03
151			•		3,013	0.14	0.70	0.13	0.14	0.02
152	•	•	•	•	19,997	0.27	0.44	0.20	0.20	0.17
154			•		3,013	0.21	0.64	0.11	0.20	0.05
155	•	•		•	20,249	0.10	0.62	0.28	0.05	0.05
AVG					15,265	0.18	0.54	0.26	0.11	0.10
MED					20,120	0.15	0.58	0.26	0.09	0.06
TOT					549,548	0.18				

Column "λ" gives the **failure rate** $\lambda = \frac{\#\text{violations}}{\#\text{test groups}}$, the ratio of test groups that lead to metamorphic oracle violations (i.e., the input relation $\mathcal{R}_i$ is true, but the output relation $\mathcal{R}_o$ is false). The failure rates range from 0.00 (0%) to 0.80 (80%), with an average of 0.18 (18%) and median of 0.15 (15%).

Tables VI and VII summarize results by tasks and LLMs, respectively. QA tasks have the fewest failures ($\lambda = 0.12$), while RE tasks have the most ($\lambda = 0.32$). Among LLMs, GPT-4 has the lowest failure rate ($\lambda = 0.14$).

> **Answering RQ1:** The failure rate λ of the 36 MRs averages to 18%. There is significant variability in the ratio across MRs (λ from 0% to 80%), even within the same task.

B. RQ2: Comparison with Traditional Testing

To compare MT with traditional testing in the context of LLMs, we computed the confusion matrix of two oracles: the **metamorphic oracle**, and the **ground truth of the source output** as provided by the original labeled dataset. Table VIII describes the notation for each of the four possible outcomes.

Following Section II, consider a metamorphic test group $t = \langle x_1, \ldots, x_n \rangle$ with an associated MR $\mathcal{R}_i(x_1, \cdots, x_n) \Rightarrow$

[10]https://platform.openai.com/docs/models
[11]https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct
[12]https://huggingface.co/NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO

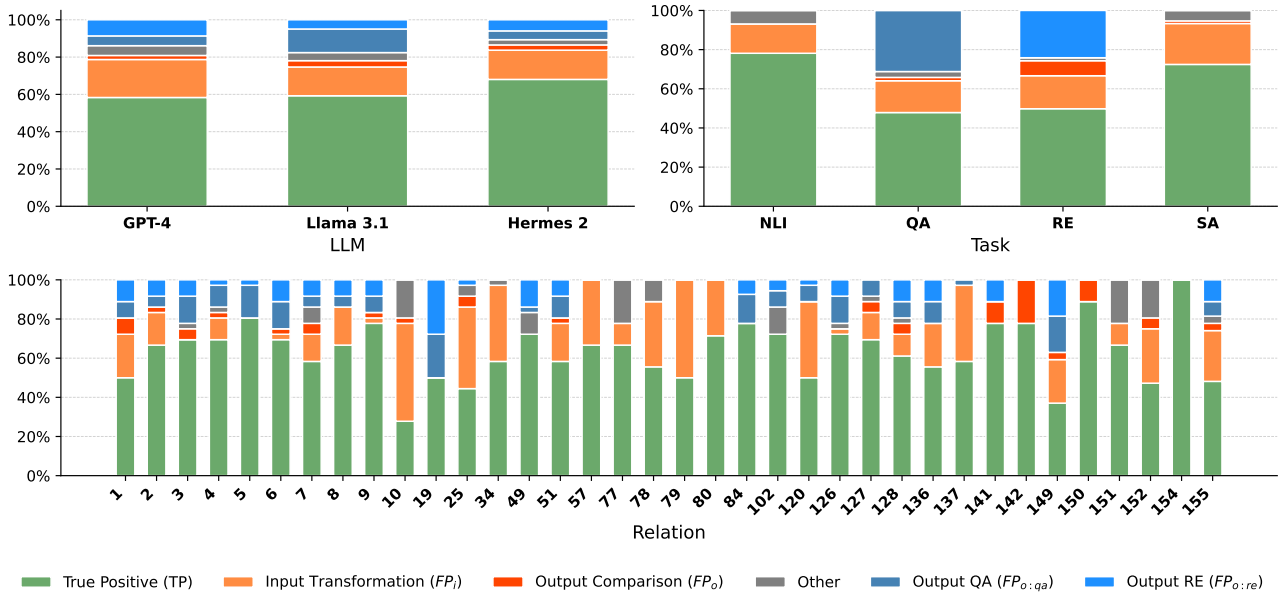

Fig. 3. Stacked bar chart of the manual labels by LLM, Task, and MR (RQ3)

TABLE VI
RESULTS BY NLP TASKS (RQ1 AND RQ2)

	RQ1 – failure rate		RQ2 – labelled data			
Task	**# tests**	λ	①	②	③	④
NLI	200,987	0.17	0.59	0.23	0.13	0.06
QA	205,179	0.12	0.67	0.21	0.09	0.03
RE	67,958	0.32	0.09	0.57	0.04	0.30
SA	75,424	0.25	0.59	0.13	0.22	0.05
AVG	137,387	0.22	0.48	0.29	0.12	0.11
MED	138,205.5	0.21	0.59	0.22	0.11	0.05

TABLE VII
RESULTS BY LARGE LANGUAGE MODELS (RQ1 AND RQ2)

	RQ1 – failure rate		RQ2 – labelled data			
LLM	**# tests**	λ	①	②	③	④
GPT-4	169,787	0.14	0.64	0.21	0.10	0.05
LLAMA3	191,082	0.18	0.53	0.28	0.11	0.08
HERMES 2	188,679	0.21	0.51	0.26	0.13	0.10
AVG	183,183	0.18	0.56	0.25	0.11	0.08
MED	188,679	0.18	0.53	0.26	0.11	0.08

$\mathcal{R}_o(f(x_1), \cdots, f(x_n))$. The metamorphic test oracle fails (an oracle violation indicating incorrect behavior) if this Boolean implication is false (③ or ④). Otherwise, it passes (① or ②).

For the same group t, let $\hat{f}(x_1)$ be the ground truth (from a labeled dataset) of the expected output for the source input x_1. The ground truth oracle fails if $\hat{f}(x_1) \neq f(x_1)$ (② or ④), and passes otherwise (① or ③). When tasks require semantic equivalence for the comparison, we follow the method described in Section IV; otherwise, we compare the strings syntactically (e.g., `neutral` versus `entailment` in Figure 1).

It is important to clarify that these oracle results are not directly comparable. On one hand, we only have the ground truth for the expected output of the source input, not for the metamorphic relation or any follow-up outputs. On the other hand, a failure in the metamorphic oracle does not necessarily indicate that the source output is incorrect; it may also point to issues with the follow-up output(s), or both. Nevertheless, we can still draw meaningful conclusions by analysing the results.

TABLE VIII
CONFUSION MATRIX OF THE MR AND GROUND-TRUTH ORACLES (RQ2)

MR(t)	$\hat{\mathbf{f}}(\mathbf{x_1}) \neq \mathbf{f}(\mathbf{x_1})$ True	False
True	①	②
False	③	④

① $\sim$ 53% Both oracles are passing
② $\sim$ 27% MT oracle passes, source output is incorrect
③ $\sim$ 11% MT oracle fails, source output is correct
④ $\sim$ 10% Both oracles are failing

We partition the 561,267 test groups into four disjoint sets (①, ②, ③, ④) depending on whether the MR was satisfied or violated, and whether the source output was correct or not according to the ground truth. Columns ①, ②, ③, and ④ in Tables V, VI, and VII present the confusion matrix results grouped by MRs, tasks, and LLMs, respectively. Interestingly, the average ratios across the three are quite similar (with some exceptions). Thus, we can make some common observations:

① ($\sim$53% of all groups) Both oracles pass. This is the most common case, which makes sense as the tested LLMs generally have high capabilities, leading to mostly correct executions.

② ($\sim$27% of all groups) The metamorphic oracle fails to detect that the source output is incorrect. An incorrect source output often leads to similarly incorrect follow-up outputs, likely making metamorphic fault detection more challenging.

③ ($\sim$11% of all groups) The metamorphic oracle detects a faulty behavior that the ground truth oracle does not. As the ground truth oracle on the source output passes, there is likely a problem in the follow-up output. This is the most interesting case, showing the complementary nature of the two oracles.

④ ($\sim$10% of all groups) Both oracles fail, indicating overlapping fault detection capabilities.

TABLE IX
CLASSIFICATION OF MANUAL VALIDATION RESULTS (RQ3)

Label	Description - True Positive (TP) and False Positive (FP)
TP	The metamorphic relation correctly identified a faulty behavior according to human judgment.
FP_i	The MR failed because the input transformation altered the source input either too much or too little, depending on whether equivalence or difference was intended.
FP_o	The output relation failed to correctly compare outputs (e.g., BERT's semantic similarity misidentifying equivalence or difference).
$FP_{o:qa}$	In QA tasks, outputs indicate the answer is unknown, but the output comparison fails to recognize equivalence.
$FP_{o:re}$	In RE tasks, outputs specify different but correct textual relations, and the output comparison fails to recognize correctness.
FP_{MR}	The input and output relations are correct, but the MR is nonsensical for the given inputs.
FP_{other}	Uncategorized cases (e.g., empty LLM output).

The exceptions to this trend are MR-57, 141, and 142—all applied to RE—and the overall RE task itself. They show fewer MR satisfactions (①) and more violations (③). This likely stems from RE allowing multiple correct answers (e.g., "David Bowie" and "Space Oddity" could relate as both "singer" and "author"). As a result, our semantic similarity method may misclassify valid variations as failures.

> **Answering RQ2:** Traditional label-based testing detects more faults overall, but MT complements it by identifying faults missed by labeled data, at zero human cost. Thus, MT is valuable when labeled data is limited or costly.

C. RQ3 - Manual Validation

To validate the results, we manually assessed a sample of the metamorphic violations to determine whether they were true positive faulty behaviors. To reduce bias toward high-violation MRs, we equally sampled from all combinations of LLM, task, and MR. We randomly selected three for each combination, totaling 967 violations distributed among three authors.

Each evaluator examined the inputs, outputs, and MR descriptions. To facilitate this, we built a web interface highlighting the textual difference between source and follow-up inputs/outputs. The evaluators chose from the seven labels described in Table IX. Note that FP_o, $FP_{o:qa}$, and $FP_{o:re}$ are disjoint sets. We chose to label $FP_{o:qa}$ and $FP_{o:re}$ separately from other output relation issues due to their prominence.

Figure 3 shows the stacked bar charts of the results grouped by MR, task, and LLM, respectively. As only one violation (of MR-155) was deemed nonsensical, we incorporated FP_{MR} into FP_{other}. On average, the TP rate is 62% across all 967 metamorphic oracle violations manually analysed. While the bar chart of LLMs shows a similar proportion across different LLMs (left top corner), the chart divided by task (right top corner) provides more insight into which tasks MT is more prone to have false positives. As expected, QA and RE pose the greatest challenges because their free-form outputs require semantic comparisons. The NLI task had no issues with the output relation as it uses syntactic equivalence among three possible outputs (see Figure 1). Similarly, sentiment analysis had fewer issues due to the ease of comparison of its numerical sentiment score. False positives derived from input transformation issues is the most common category.

TABLE X
REPORTED FALSE POSITIVE RATES OF PREVIOUS WORK ON MT FOR NLP

Task	Ref.	Lower	Upper	Task	Ref.	Lower	Upper
TR	[63]	0.16	0.38	QAb	[36]	0.07	0.07
TR	[62]	0.30	0.43	QA	[65]	0.00	0.19
TR	[60]	0.13	0.37	QA	[31]	0.00	0.38
TR	[59]	0.00	0.22	QAc	[30]	0.02	0.03
TR	[53]	0.23	0.57	NER	[44]	0.02	0.20
TR	[52]	0.22	0.30	CR	[40]	0.00	0.00
TR	[51]	0.09	0.23	Various	[28]	0.02	0.08
TR	[15]	0.22	0.30				

> **Answering RQ3:** True positives are the most common (62%), but some MRs have significant false positives.

V. DISCUSSION AND ANALYSIS

False positive discussion: We observe a high false positive rate in our results. This is somewhat expected, as it is an issue inherent to MT for NLP. In traditional MT for software systems, MRs are usually defined exactly, such that the description and the implementation are one and the same [14]. Under such conditions, if a false positive occurs, the MR itself must have been incorrectly specified and thus not a necessary property of the system under test [13]. For example, consider the MR $x^2 = (-x)^2$ for the square function. Given a number x, we can apply the input transformation precisely to obtain $-x$. The output relation of numerical equivalence ($=$) admits no FPs.

Conversely, in NLP, the MR serves more as a guideline; there may be many possible implementations of the same MR. Consider MR-35. The input relation *"Replace keywords with unrelated terms"* specifies a vague implementation of detecting keywords and identifying unrelated terms, a task complicated by the inherently ambiguous nature of language. The semantic nature of the output relation further increases FPs.

Table X summarizes the false positive rates reported by previous work on MT for NLP, showing both the lower and upper bounds from their experiments. Our results are comparable to these. This suggests that MT for LLMs does not produce substantially more false positives than traditional MT for NLP, which is an important finding. Notably, prior MT research on LLMs [20] did not evaluate FPs.

We find that the most common cause FPs is input transformation errors. These issues typically fall into one of two categories: (i) the transformation changes too little even though the MR expects a different outcome, or (ii) it changes too much while the MR still expects an equivalent result (e.g. MR-127 misspell "What is the **capital** of Chile?" $\rightsquigarrow$ "What is the **apitpul** of Chile?"). Regarding output relation FPs, the BERT-based semantic comparison commonly used in NLP cannot as easily be used in LLM systems. In the QA task, 31% of the false positives was due to the cosine similarity not being able

to recognise the equivalence between, for instance, "unknown" and a similar response. In RE, 24% of FPs occur because standard NLP assumes a single correct label, while LLMs can produce multiple valid answers (e.g., *"son"* or *"successor"* for *"What is the relation between Charles I and Charles II?"*). These are intrinsic issues stemming from natural language's ambiguity and complexity.

While our results indicate that further research is needed to enhance the effectiveness of input transformations and output comparisons, it is unrealistic to expect these to perform flawlessly across all tasks and contexts. A more attainable goal is to devise methods for assessing the confidence or quality of such transformations and comparisons, thereby filtering out cases likely to be false positives. For instance, prioritising failures by metrics such as MR type and the magnitude of similarity score would help reduce the cost of inspecting failing test cases in practical use. In addition, if the oracle violations are used in an automated pipeline (e.g., for self-supervised fine-tuning or adversarial training), some FPs are less critical.

Task dependency discussion: Our experiments confirm that some MRs exhibit varying TP rates depending on the task. For example, synonym substitution in QA (*"What's the tallest building?"* $\rightsquigarrow$ *"What's the highest building?"*, which remains equivalent) behaves differently in SA (*"The movie was great"* $\rightsquigarrow$ *"The movie was superb"*, which may increase the sentiment score). However, certain MRs (e.g., MR-9, MR-102) prove effective across multiple tasks, consistently maintaining a high TP rate. Notably, we applied some MRs to tasks they were not originally designed for and found them to be effective. This is crucial, as it suggests that some MRs can be leveraged universally, demonstrating the potential of MT for evaluating locally deployed, fine-tuned LLMs.

Flakiness analysis: LLMs, compared to many NLP techniques, are non-deterministic in nature[13]. Therefore, it is important to understand how flaky (inconsistent) the results may be. We selected all 99,099 metamorphic groups that initially failed and re-ran them 9 additional times to measure flakiness across 10 total runs. We used exactly the same groups for each run, thus guaranteeing no variability in generating the follow-up test cases and ensuring consistency across all experiments.

Table XI shows the results (**all** row). Most metamorphic groups (62%) consistently failed in the majority of runs (6-10 times), with 28% failing in all 10 runs. These results indicate that the observed issues reliably trigger MR violations and are not artifacts of randomness behaviors inherent to LLMs models. Moreover, although we re-ran only failing metamorphic groups rather than all 561,267 groups (due to computational constraints), our results suggest that even a few runs may still reliably detect MR violations.

Table XI also breaks down the failure rates between failure types. The average 10/10 rates between true (TP 21%) and false (FP_{all} 23%) positives are similar, unexpectedly. This is likely coincidental, however, as the 10/10 rates for input-based false positives (46%) differ greatly from that of output-based (0%-11%). The former is LLM-independent, leading to a higher chance of consistent violation; the latter largely depends on the LLM's nondeterministic output, often resulting in true satisfaction once the output has changed.

[13]Even when the temperature parameter is set to zero, some models or tokenizers may still randomly choose between tokens with equal scores.

TABLE XI
FLAKINESS OF METAMORPHIC GROUPS WITH AT LEAST ONE FAILURE

	# failures per 10 re-runs									
Failure Type	**1/10**	**2/10**	**3/10**	**4/10**	**5/10**	**6/10**	**7/10**	**8/10**	**9/10**	**10/10**
all	0.08	0.07	0.08	0.08	0.08	0.08	0.08	0.08	0.10	0.28
TP	0.08	0.08	0.14	0.06	0.11	0.08	0.06	0.09	0.10	0.21
FP_{all}	0.08	0.10	0.11	0.10	0.11	0.06	0.06	0.07	0.09	0.23
FP_{i}	0.03	0.04	0.03	0.03	0.04	0.04	0.10	0.05	0.16	0.46
FP_{o}	0.11	0.05	0.16	0.21	0.11	0.00	0.05	0.11	0.11	0.11
$FP_{o:qa}$	0.22	0.18	0.22	0.13	0.13	0.09	0.02	0.02	0.00	0.00
$FP_{o:re}$	0.00	0.19	0.06	0.19	0.25	0.06	0.06	0.16	0.03	0.00
FP_{other}	0.00	0.05	0.16	0.05	0.16	0.11	0.00	0.11	0.05	0.32

Envisioned use of LLMORPH: LLMORPH can be combined with traditional testing by using MT to automatically identify inputs where the LLM fails to satisfy an MR. Traditional testing can then be applied to these inputs to define human-specified oracles, enabling more targeted validation while reducing the high cost of human labelling.

LLMORPH could be integrated into existing continuous integration/continuous deployment pipelines for LLM-based services. In particular, in the regression testing scenario, LLMORPH could automatically detect if changes in the training, prompts, or model architecture impacted the performance of the system. This could be fully automated by checking if the failure rate significantly increases after modifications.

VI. THREATS TO VALIDITY

Data leakage: A possible threat is data leakage, as the data we used was likely trained on by the LLMs. Indeed, during our manual inspection, we found that the LLM was sometimes responding too well on very perturbed follow-up inputs. However, this is not a severe issue; it simply means that the LLM performs better than it should. We are still able to detect issues using MT.

Implementation issues: Another potential threat concerns our implementation of prompts, input transformations, and output comparison. To mitigate this, we followed established practices from related work and applied prompt engineering techniques to refine our prompts where needed. All experimental data are publicly available[5], and we welcome external validation.

Input transformation bias: In our experiments, we relied on HERMES 2 to perform complex input transformations. However, HERMES 2 is also one of the LLM under test, so the input transformation and output generation was done by exactly the same MR. This might have introduced biases. Performing more experiments possibly using a dedicated LLM for the input transformation is an important future work, as well as evaluating LLMORPH on more recent and advanced LLMs (e.g., OpenAI's O3, which has advanced reasoning).

VII. Related Work

To the best of our knowledge, this is the first comprehensive study of MT for LLMs through NLP. We now discuss the most related work in MT surveys, LLMs for MT, and MT for LLMs.

MT surveys: Previous surveys on MT have not specifically targeted MRs in the NLP domain. METWIKI[14] [74] is a repository for retrieving MRs, containing MRs extracted from the MT survey by Segura et al. [75]. They do not discuss MRs for NLP, however. Their survey was conducted in 2016, while our literature search indicates that the first papers on MT for NLP date to 2018. Indeed, Figure 5 of Segura et al.'s survey, the MT applications over the years [75], does not include NLP (or a related application). Another survey on MT by Xie et al. [76] (conducted in 2011) provides a broader overview of MT techniques for machine learning but does not mention MT for NLP, which again aligns with our findings. Moreover, both surveys discuss the properties of MT rather than specific MRs.

LLMs for MT: There is growing interest in leveraging LLMs for various aspects of automated software testing, such as generating test cases and oracles [77]–[80]. In the context of MT, recent research has explored using LLMs for MT of software systems [81]–[83], including generating MRs [84]–[88]. Similarly, LLMorph employs LLMs for implementing MT, specifically for transformations. However, our focus is on testing LLMs, whereas these work leverages LLM to target MT for code-based software systems.

MT for LLMs: Adversarial attacks on prompts aim to perturbing *prompts* to evaluate the qualities of LLMs [25], [89]–[91]. Some of these adversarial attacks partially resemble MRs and MT [90]. However, these approaches focus on altering the prompt to make the LLM respond differently, including jailbreak techniques that assign specific roles to prompts to induce privacy and security leaks [25]. In contrast, we focus on MRs applied to inputs rather than prompts, which is an orthogonal problem. Indeed, the MRs discussed in this paper are generic to any NLP technique, whereas adversarial attacks on prompts are specific to particular LLMs, with prompt-based jailbreak methods working for some LLMs but not others [25].

Regarding MT for LLMs in the NLP domain, METAL [20] is a recent framework for MT of LLMs. While the experimental settings of METAL are similar to LLMorph, including investigating using LLMs to implement input transformations, the two works differ substantially: (i) METAL experimented with 13 MRs, whereas we chose 36, which includes all relevant MRs in METAL[15]. (ii) The number of metamorphic test cases generated in our experiments is orders of magnitude higher. (iii) METAL did not manually investigate the false positive rate of metamorphic oracle violations. We found that this is crucial to truly understand the capabilities and effectiveness of MT for LLMs. (iv) We compared the MT results with the ground truth of the source inputs to provide more insights into the fault detection capabilities of MT.

Since we performed the literature search on MT for LLMs via NLP (up to 30 June 2024), a few papers have been published in this field. There have been studies on detecting fact-conflicting hallucinations in LLM using MT [92]–[95], mainly in the areas of question answering [92], [93]. In addition, there have been papers on LLM-based dialogue systems [96] and evaluating LLM in-context learning via sentiment analysis and question answering [97]. However, these studies investigate specific applications of MT for LLMs under specific tasks, domain, or contexts. Conversely, our work aim at examining MT for LLM more generally; indeed, we still exceed all of these works in the breadth of our study, considering both the number of MRs and the range of NLP tasks evaluated.

VIII. Conclusion and Future Work

This paper presented the most comprehensive study on applying Metamorphic Testing (MT) to Large Language Models (LLMs). We systematically reviewed the existing literature and compiled a list of 191 metamorphic relations for NLP. We then introduced a framework called LLMorph that implements 36 of these MRs, executing 561,297 metamorphic test groups on three LLMs across four NLP tasks. Using this collection of MRs, we explored the interactions between MRs, tasks, and LLM response semantics.

As this framework uses the general NLP testing method of Metamorphic Testing, it can also be used on any NLP system, not just LLMs. While LLMorph could still rely on the power of LLMs for implementing input transformations, we can replace the model under test with any NLP tool. In particular, our catalog of 191 MRs represents the largest knowledge base of MRs for NLP to date, which we believe is a useful contribution for the entire NLP community beyond LLMs.

While MT for LLMs via NLP is promising, it remains a relatively new research direction, and several challenges remain. Based on our findings, we highlight several opportunities for future research.

LLMorph currently implements 36 out of the 191 collected MRs. While implementing all MRs is a substantial task, we hope that releasing the source code[4] will encourage the research community to contribute to LLMorph.

We investigated the same MRs applied across several tasks and found that many exhibit task independence. With LLMs increasingly being fine-tuned for tasks, it is important to find more of such task-independent MRs to more effectively test these systems automatically.

We found that false positive metamorphic violations is still a major challenge in MT for LLMs. Given the inherent ambiguity of language, completely eliminating FPs is difficult. Future research should focus on detecting and filtering FPs to improve the effectiveness of metamorphic testing for LLMs.

[14]Unreachable website http://metwiki.net/

[15]METAL includes additional MRs targeting more general LLM qualities (e.g., efficiency, fairness), which are outside the scope of this work.

REFERENCES

[1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[2] W. Wang, J. Shi, Z. Tu, Y. Yuan, J.-t. Huang, W. Jiao, and M. R. Lyu, "The earth is flat? unveiling factual errors in large language models," *arXiv preprint arXiv:2401.00761*, 2024.

[3] 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.

[4] Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, and Y. Goldberg, "Measuring and improving consistency in pretrained language models," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1012–1031, 2021.

[5] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.

[6] H. Sun and M. van der Schaar, "Inverse reinforcement learning from demonstrations for llm alignment," in *ICML 2024*, 2024.

[7] Y. Wang, W. Zhong, L. Li, F. Mi, X. Zeng, W. Huang, L. Shang, X. Jiang, and Q. Liu, "Aligning large language models with human: A survey," *arXiv preprint arXiv:2307.12966*, 2023.

[8] C. Tang, Z. Liu, C. Ma, Z. Wu, Y. Li, W. Liu, D. Zhu, Q. Li, X. Li, T. Liu *et al.*, "Policygpt: Automated analysis of privacy policies with large language models," *arXiv preprint arXiv:2309.10238*, 2023.

[9] K. VM, H. Warrier, Y. Gupta *et al.*, "Fine tuning llm for enterprise: Practical guidelines and recommendations," *arXiv:2404.10779*, 2024.

[10] C. Irugalbandara, A. Mahendra, R. Daynauth, T. K. Arachchige, J. Dantanarayana, K. Flautner, L. Tang, Y. Kang, and J. Mars, "Scaling down to scale up: A cost-benefit analysis of replacing openai's llm with open source slms in production," in *ISPASS*. IEEE, 2024, pp. 280–291.

[11] V. Hanke, T. Blanchard, F. Boenisch, I. E. Olatunji, M. Backes, and A. Dziedzic, "Open llms are necessary for current private adaptations and outperform their closed alternatives," *arXiv:2411.05818*, 2024.

[12] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[13] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology, Tech. Rep., 1998.

[14] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[15] W. Gao, J. He, and V.-T. Pham, "Metamorphic testing of machine translation models using back translation," in *DeepTest*, 2023, pp. 1–8.

[16] J. Bozic and F. Wotawa, "Testing chatbots using metamorphic relations," in *Testing Software and Systems*, C. Gaston, N. Kosmatov, and P. Le Gall, Eds., 2019, pp. 41–55.

[17] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of NLP models with CheckList," in *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4902–4912.

[18] M. Jiang, T. Y. Chen, and S. Wang, "On the effectiveness of testing sentiment analysis systems with metamorphic testing," *Inf. Softw. Technol.*, vol. 150, no. C, 2022.

[19] K. Tu, M. Jiang, and Z. Ding, "A metamorphic testing approach for assessing question answering systems," *Mathematics*, vol. 9, no. 7, 2021.

[20] S. Hyun, M. Guo, and M. A. Babar, "Metal: Metamorphic testing framework for analyzing large-language model qualities," in *ICST*, 2024, pp. 117–128.

[21] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing*, 2015, pp. 632–642.

[22] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, 2018.

[23] C. Xu, V. Terragni, H. Zhu, J. Wu, and S.-C. Cheung, "MR-Scout: Automated Synthesis of Metamorphic Relations from Existing Test Cases," *TOSEM*, vol. 33, no. 6, 2024.

[24] J. Ayerdi, V. Terragni, G. Jahangirova, A. Arrieta, and P. Tonella, "GenMorph: Automatically Generating Metamorphic Relations via Genetic Programming," *IEEE TSE*, vol. 50, no. 7, pp. 1888–1900, 2024.

[25] H. Li, D. Guo, W. Fan, M. Xu, J. Huang, F. Meng, and Y. Song, "Multi-step jailbreaking privacy attacks on chatgpt," *arXiv:2304.05197*, 2023.

[26] Z. Zhang, B. G. Patra, A. Yaseen, J. Zhu, R. Sabharwal, K. Roberts, T. Cao, and H. Wu, "Scholarly recommendation systems: a literature survey," *Knowledge and Information Systems*, vol. 65, no. 11, pp. 4433–4478, 2023.

[27] Z. Li, W. Qiu, P. Ma, Y. Li, Y. Li, S. He, B. Jiang, S. Wang, and W. Gu, "An empirical study on large language models in accuracy and robustness under chinese industrial scenarios," *arXiv:2402.01723*, 2024.

[28] P. Ji, Y. Feng, W. Huang, J. Liu, and Z. Zhao, "Intergenerational test generation for natural language processing applications," *arXiv:2302.10499*, 2023.

[29] L. Jin, Z. Ding, and H. Zhou, "Evaluation of chinese natural language processing system based on metamorphic testing," *Mathematics*, vol. 10, no. 8, 2022.

[30] Q. Shen, J. Chen, J. M. Zhang, H. Wang, S. Liu, and M. Tian, "Natural test generation for precise testing of question answering software," in *ASE '22*, 2023.

[31] X. Xie, S. Jin, and S. Chen, "qaasker+: a novel testing method for question answering software via asking recursive questions," *Automated Software Engg.*, vol. 30, no. 1, mar 2023.

[32] C. Wu, L. Sun, and Z. Q. Zhou, "The impact of a dot: Case studies of a noise metamorphic relation pattern," in *2019 IEEE/ACM 4th Int. Workshop on Metamorphic Testing (MET)*, 2019, pp. 17–23.

[33] 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 *ICSE*, 2023, p. 2387–2399.

[34] Z. Liu, Y. Feng, Y. Yin, J. Sun, Z. Chen, and B. Xu, "Qatest: A uniform fuzzing framework for question answering systems," in *ASE*, 2023.

[35] J. Božić, "Ontology-based metamorphic testing for chatbots," *Software Quality Journal*, vol. 30, no. 1, p. 227–251, mar 2022.

[36] S. Chen, S. Jin, and X. Xie, "Validation on machine reading comprehension software without annotated labels: a property-based method," in *ESEC/FSE 2021*, 2021, p. 590–602.

[37] Z. Liu, Y. Feng, and Z. Chen, "Dialtest: automated testing for recurrent-neural-network-driven dialogue systems," in *ISSTA*, 2021, p. 115–126.

[38] Y. Chen, L. Li, X. Tao, and D. Zhou, "Persona-centric metamorphic relation guided robustness evaluation for multi-turn dialogue modelling," *arXiv:2401.12483*, 2024.

[39] Y. Sun, Z. Ding, H. Huang, S. Zou, and M. Jiang, "Metamorphic testing of relation extraction models," *Algorithms*, vol. 16, no. 2, 2023.

[40] J. Cao, Y. Lu, M. Wen, and S.-C. Cheung, "Testing coreference resolution systems without labeled test sets," in *FSE*, 2023, p. 107–119.

[41] Y. Ma, D. Towey, T. Yueh Chen, and Z. Quan Zhou, "Metamorphic testing of fake news detection software," in *COMPSAC*, 2021, pp. 1508–1513.

[42] L. Miao, D. Towey, Y. Ma, T. Y. Chen, and Z. Quan Zhou, "Exploring metamorphic testing for fake-news detection software: A case study," in *COMPSAC*, 2023, pp. 912–923.

[43] E. Manino, J. Rozanova, D. Carvalho, A. Freitas, and L. Cordeiro, "Systematicity, compositionality and transitivity of deep NLP models: a metamorphic testing perspective," in *ACL*, 2022, pp. 2355–2366.

[44] B. Yu, Y. Hu, Q. Mang, W. Hu, and P. He, "Automated testing and improvement of named entity recognition systems," in *FSE*, 2023, p. 883–894.

[45] Y. Xu, Z. Q. Zhou, X. Zhang, J. Wang, and M. Jiang, "Metamorphic testing of named entity recognition systems: A case study," *IET Software*, vol. 16, no. 4, p. 386–404, 2022.

[46] M. Srinivasan, M. P. Shahri, I. Kahanda, and U. Kanewala, "Quality assurance of bioinformatics software: A case study of testing a biomedical text processing tool using metamorphic testing," in *2018 IEEE/ACM 3rd Int. Workshop on Metamorphic Testing (MET)*, 2018, pp. 26–33.

[47] M. Jiang, H. Bao, K. Tu, X.-Y. Zhang, and Z. Ding, "Evaluating natural language inference models: A metamorphic testing approach," in *ISSRE*, 2021, pp. 220–230.

[48] A. Arno, F. Iwama, and M. Takeuchi, "Automated metamorphic testing using transitive relations for specializing stance detection models," in *ICSE-SEIP*, 2023, pp. 467–470.

[49] D. Pesu, Z. Q. Zhou, J. Zhen, and D. Towey, "A monte carlo method for metamorphic testing of machine translation services," in *2018 IEEE/ACM 3rd Int. Workshop on Metamorphic Testing (MET)*, 2018, pp. 38–45.

[50] L. Sun and Z. Q. Zhou, "Metamorphic testing for machine translations: Mt4mt," in *ASWEC*, 2018, pp. 96–100.

[51] Z. Sun, J. M. Zhang, M. Harman, M. Papadakis, and L. Zhang, "Automatic testing and improvement of machine translation," in *ICSE*, 2020, p. 974–985.

[52] P. He, C. Meister, and Z. Su, "Structure-invariant testing for machine translation," in *ICSE*, 2020, p. 961–973.

[53] S. Gupta, P. He, C. Meister, and Z. Su, "Machine translation testing via pathological invariance," in *ESEC/FSE*, 2020, p. 863–875.

[54] Z. Li, L. Xiao, R. Lin, and Z. Xiao, "Metamorphic robustness testing for deepl translation," *Journal of Physics: Conf. Series*, vol. 2456, no. 1, p. 012018, 2023.

[55] D. T. S. Lee, Z. Q. Zhou, and T. H. Tse, "Metamorphic robustness testing of google translate," in *ICSEW*, 2020, p. 388–395.

[56] S. Jin, S. Chen, and X. Xie, "Property-based test for part-of-speech tagging tool," in *ASE*, 2021, pp. 1306–1311.

[57] P. Y. P. Chan, J. Keung, and Z. Yang, "Uncertainty-based metamorphic testing for validating plagiarism detection systems," in *Technology in Education. Innovative Practices for the New Normal*, 2024, pp. 299–314.

[58] ——, "Validating plagiarism detection systems with metamorphic testing," in *ISET*, 2023, pp. 132–137.

[59] P. He, C. Meister, and Z. Su, "Testing machine translation via referential transparency," in *ICSE*, 2021, p. 410–422.

[60] P. Ji, Y. Feng, J. Liu, Z. Zhao, and B. Xu, "Automated testing for machine translation via constituency invariance," in *ASE*, 2021, pp. 468–479.

[61] B. Yu, Q. Mang, Q. Guo, and P. He, "Retromorphic testing: A new approach to the test oracle problem," *arXiv:2310.06433*, 2023.

[62] X. Xie, S. Jin, S. Chen, and S.-C. Cheung, "Word closure-based metamorphic testing for machine translation," *arXiv:2312.12056*, 2023.

[63] Q. Zhang, J. Zhai, C. Fang, J. Liu, W. Sun, H. Hu, and Q. Wang, "Machine translation testing via syntactic tree pruning," *ACM Trans. Softw. Eng. Methodol.*, 2024.

[64] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2020.

[65] S. Chen, S. Jin, and X. Xie, "Testing your question answering software via asking recursively," in *ASE*, 2021, pp. 104–116.

[66] Z. Wang, W. Wang, Q. Chen, Q. Wang, and A. Nguyen, "Generating valid and natural adversarial examples with large language models," in *CSCWD*, 2024, pp. 1716–1721.

[67] J.-t. Huang, J. Zhang, W. Wang, P. He, Y. Su, and M. R. Lyu, "Aeon: a method for automatic evaluation of nlp test cases," in *ISSTA*, 2022, p. 202–214.

[68] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering in large language models: a comprehensive review," *arXiv:2310.14735*, 2023.

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019, pp. 4171–4186.

[70] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 784–789.

[71] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.

[72] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMENLP*, 2013, pp. 1631–1642.

[73] Q. Tan, L. Xu, L. Bing, H. T. Ng, and S. M. Aljunied, "Revisiting docred – addressing the false negative problem in relation extraction," in *EMNLP*, 2022.

[74] X. Xie, J. Li, C. Wang, and T. Y. Chen, "Looking for an mr? try metwiki today," in *2016 IEEE/ACM 1st Int. Workshop on Metamorphic Testing (MET)*, 2016, pp. 1–4.

[75] S. Segura, S. Poulding, and G. Fraser, "Metamorphic testing: Testing the untestable," in *ICSE*, 2016, pp. 1–12.

[76] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[77] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "Adaptive test generation using a large language model," *arXiv e-prints*, 2023.

[78] Z. Yuan, Y. Lou, M. Liu, S. Ding, K. Wang, Y. Chen, and X. Peng, "No more manual tests? evaluating and improving chatgpt for unit test generation," *arXiv:2305.04207*, 2023.

[79] S. Ruberto, J. Perera, G. Jahangirova, and V. Terragni, "From implemented to expected behaviors: Leveraging regression oracles for non-regression fault detection using llms," in *ICSTW*, 2025, pp. 37–40.

[80] R. Ravi, D. Bradshaw, S. Ruberto, G. Jahangirova, and V. Terragni, "LLMLOOP: Improving LLM-Generated Code and Tests through Automated Iterative Feedback Loops," in *ICSME*. IEEE, 2025.

[81] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Transactions on Software Engineering*, 2024.

[82] M. L. Siddiq, J. C. Da Silva Santos, R. H. Tanvir, N. Ulfat, F. Al Rifat, and V. Carvalho Lopes, "Using large language models to generate junit tests: An empirical study," in *Proc. of the 28th Int. Conf. on Evaluation and Assessment in Software Engineering*, 2024, pp. 313–322.

[83] N. Alshahwan, J. Chheda, A. Finogenova, B. Gokkaya, M. Harman, I. Harper, A. Marginean, S. Sengupta, and E. Wang, "Automated unit test improvement using large language models at meta," in *FSE*, 2024, pp. 185–196.

[84] S. Y. Shin, F. Pastore, D. Bianculli, and A. Baicoianu, "Towards generating executable metamorphic relations using large language models," *arXiv:2401.17019*, 2024.

[85] C. Tsigkanos, P. Rani, S. Müller, and T. Kehrer, "Large language models: The next frontier for variable discovery within metamorphic testing?" in *SANER*. IEEE, 2023, pp. 678–682.

[86] Q.-H. Luu, H. Liu, and T. Y. Chen, "Can chatgpt advance software testing intelligence? an experience report on metamorphic testing," *arXiv:2310.19204*, 2023.

[87] D. Srinivas, R. Das, S. Tizpaz-Niari, A. Trivedi, and M. L. Pacheco, "On the potential and limitations of few-shot in-context learning to generate metamorphic specifications for tax preparation software," *arXiv:2311.11979*, 2023.

[88] C. Xu, S. Chen, J. Wu, S.-C. Cheung, V. Terragni, H. Zhu, and J. Cao, "Mr-adopt: Automatic deduction of input transformation function for metamorphic testing," in *ASE*, 2024, pp. 557–569.

[89] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv:2211.09527*, 2022.

[90] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Z. Gong, and X. Xie, "Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts," 2024.

[91] X. Shen, Z. Chen, M. Backes, and Y. Zhang, "In chatgpt we trust? measuring and characterizing the reliability of chatgpt," *arXiv:2304.08979*, 2023.

[92] W. Wu, Y. Cao, N. Yi, R. Ou, and Z. Zheng, "Detecting and reducing the factual hallucinations of large language models with metamorphic testing," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025.

[93] B. Yang, M. A. Al Mamun, J. M. Zhang, and G. Uddin, "Hallucination detection in large language models with metamorphic relations," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025.

[94] N. Li, Y. Li, Y. Liu, L. Shi, K. Wang, and H. Wang, "Drowzee: Metamorphic testing for fact-conflicting hallucination detection in large language models," *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA2, Oct. 2024.

[95] N. Li, Y. Song, K. Wang, Y. Li, L. Shi, Y. Liu, and H. Wang, "Detecting llm fact-conflicting hallucinations enhanced by temporal-logic-based reasoning," *arxiv:2502.13416*, 2025.

[96] G. Guo, A. Aleti, N. Neelofar, C. Tantithamthavorn, Y. Qi, and T. Y. Chen, "Mortar: Multi-turn metamorphic testing for llm-based dialogue systems," *arxiv:2412.15557*, 2025.

[97] T. Racharak, C. Ragkhitwetsagul, C. Sontesadisai, and T. Sunetnanta, "Test it before you trust it: Applying software testing for trustworthy in-context learning," in *Natural Language Processing and Information Systems*, R. Ichise, Ed., Cham, 2026, pp. 243–258.