Postprint of article in IEEE/ACM 5th International Workshop on Metamorphic Testing (MET'20), *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*, ACM, New York, NY

Metamorphic Robustness Testing of Google Translate

Dickson T. S. Lee
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
u3556189@connect.hku.hk

Zhi Quan Zhou
School of Computing and IT
University of Wollongong
Wollongong, NSW 2522, Australia
zhiquan@uow.edu.au

T. H. Tse
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
thtse@cs.hku.hk

ABSTRACT

Current research on the testing of machine translation software mainly focuses on functional correctness for valid, well-formed inputs. By contrast, robustness testing, which involves the ability of the software to handle erroneous or unanticipated inputs, is often overlooked. In this paper, we propose to address this important shortcoming. Using the metamorphic robustness testing approach, we compare the translations of original inputs with those of follow-up inputs having different categories of minor typos. Our empirical results reveal a lack of robustness in Google Translate, thereby opening a new research direction for the quality assurance of neural machine translators.

CCS CONCEPTS

• Software and its engineering → Software creation and management → Software verification and validation → Software testing and debugging

KEYWORDS

Robustness testing, Oracle problem, Metamorphic testing, Metamorphic robustness testing, Machine translation, MT4MT

ACM Reference format:

Dickson T. S. Lee, Z. Q. Zhou, and T. H. Tse. 2020. Metamorphic Robustness Testing of Google Translate. IEEE/ACM 5th International Workshop on Metamorphic Testing (MET'20). In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*, ACM, New York, NY, USA, 8 pages.
https://doi.org/10.1145/3387940.3391484

1 Introduction

According to the *IEEE Standard Glossary of Software Engineering Terminology* [11], *robustness* is "the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions". While developers strive to ensure that the software performs the intended functionality correctly for valid inputs, there is often a lack of *negative testing* to ensure that the software can also reasonably handle invalid inputs.

Suppose we want to translate the sentence "Peter LOVES apples." into Chinese using Google Translate, which is available at https://translate.google.com. We want to stress the word "LOVE" and hence type it in caps. We end the sentence with a period as per usual practice. Suppose we have made a careless mistake by typing the first letter "L" in "LOVES" in lower case, giving "Peter lOVES apples.". For this sentence, Google Translate returns "彼得讨厌苹果。" (which means "Peter hates apples."), as shown in Fig. 1. This striking translation error reveals a lack of sturdiness of Google Translate in handling minor typos. It calls for the need to conduct robustness testing.

Figure 1: A striking failure of Google Translate, which translates "Peter lOVES apples." into a Chinese sentence meaning "Peter hates apples." (as at January 22, 2020)

Robustness testing involves the following steps:

1. Create a test case involving random, invalid, or unanticipated input.
2. Specify the expected test result.
3. Execute the software under test (SUT) with the test case.
4. Verify the test case execution outcome against the expected test result.

As pointed out by Pesu et al. [10], the automatic assessment of machine translation software is difficult because the expected test

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391484

result of the SUT may not be automatically available, while manual assessment by human experts is not only expensive but also subjective.

Metamorphic testing [2][3] is a popular testing technique that aims to tackle the problem of unavailable test results. It checks the SUT against *metamorphic relations* (*MRs*), which are expected relations among the inputs and outputs of *multiple* executions of the target program. Zhou and colleagues were the first to propose *metamorphic testing for machine translation* (*MT4MT*) [10][12] [14][15]. He et al. [8], and more recently, Sun et al. [13] followed up with further work.

In this research, we adopt the *metamorphic robustness testing* approach [18] to assess the robustness of machine translation. The theoretical contributions of this paper include the presentation of eight MRs for automated validation of the sturdiness of machine translation. While some of these MRs have been proposed in previous studies, the unique characteristic of the present research is the focus on robustness testing with respect to *erroneous* input.

The practical contributions of this work include a collection of findings of Google Translate failures. Some of them confirm previous results. Hence, we show the existence of long-term bugs in Google Translate, and highlight the challenge for the quality improvement of machine-learning-based applications.

The rest of this paper is organized as follows: Section 2 presents real-world examples of translation errors that motivate this research. Section 3 introduces the preliminaries of the research. Section 4 presents the eight MRs for testing the robustness of machine translation, and the findings that result. Section 5 discusses related work. Section 6 contains further discussions and concludes the paper.

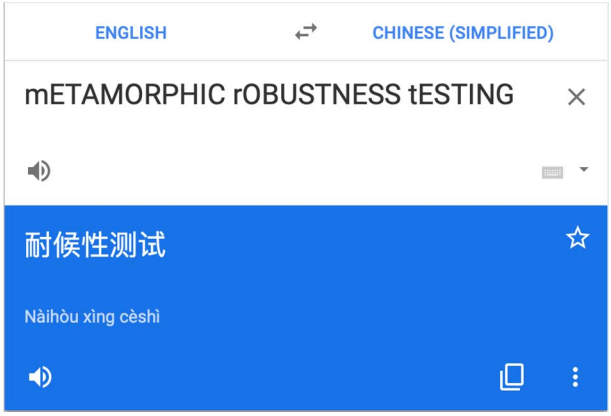


Figure 2: Google Translate result for "mETAMORPHIC rOBUSTNESS tESTING"

2 Motivating Examples

2.1 The Caps Lock

Suppose a researcher wants to translate "Metamorphic Robustness Testing" into Chinese. She accidentally presses the caps lock before typing. The phrase is therefore entered to Google Translate as "mETAMORPHIC rOBUSTNESS tESTING". The software returns "耐候性测试" (literally, "weatherability test") as shown in Fig. 2.

We find that the result in Fig. 2 is completely different from Google Translate result for the original phrase "Metamorphic Robustness Testing". We see from this example that the software is not sufficiently robust. The translation result is problematic when the caps lock is on. However, this finding relies on human judgment with a good understanding of Chinese. Can the validation process be simplified?

2.2 The Extra Period

Now, suppose that the researcher not only has pressed the caps lock by mistake, but has also included a period at the end of the phrase because she is used to typing complete sentences. She has therefore entered "mETAMORPHIC rOBUSTNESS tESTING." to Google Translate. This time, the software returns "熔铸耐用性测试。" (literally, "Fusion casting durability test."), as shown in Fig. 3, which is completely different from the previous result in Fig. 2 for the same phrase without a period.

This second example also illustrates that Google Translate is not sufficiently robust. At least one of the results (either Fig. 2 or Fig. 3, or both) must be seriously wrong, but not obvious to people who rely fully on Google Translate to obtain the Chinese version.

Figure 3: Google Translate result for "mETAMORPHIC rOBUSTNESS tESTING."

3 Preliminaries

In this section, we briefly introduce the preliminary concepts in metamorphic testing and metamorphic robustness testing

3.1 Metamorphic Testing

A *test oracle* is a mechanism against which testers can decide whether the outcome of test case executions is correct. Sometimes, an oracle may be unavailable, or is available but is too expensive to be applied. We refer to such situations as the *oracle problem* [1].

Metamorphic testing (*MT*) [2][3] alleviates the problem by checking the software under test against *metamorphic relations*,

which are expected relations among the inputs and outputs of multiple executions, known as *source* and *follow-up executions*. In this way, the SUT can be verified against the metamorphic relations in the absence of an oracle. The input/output to the source/follow-up executions are called *source input*, *source output*, *follow-up input*, and *follow-up output*, respectively.

Consider the verification of an online search function using metamorphic testing [19]. We may define search criterion "*A*" as the source input, and search criterion "*A* and *B*" as the follow-up input. Let *count*( ) denote the number of search results. A metamorphic relation may be defined as "**MR$_{and}$.** *count*(*A*) ≥ *count*(*A* and *B*)". In other words, the number of search results satisfying criterion *A* should be greater than or equal to the number of results satisfying criterion *A* together with some other criterion *B*. A violation of **MR$_{and}$** reveals a failure of the search function.

3.2 Metamorphic Robustness Testing

The term *metamorphic robustness testing* has been introduced by Zhou et al. recently [18]. As the name implies, the new technique applies the concept of MT to robustness testing. It tests the sturdiness of the SUT by checking the outputs for erroneous or unanticipated inputs against the prescribed *metamorphic robustness relations*, which are MRs identified for the robustness of the target program. In order to verify whether the SUT complies with a metamorphic robustness relation, we need to execute the software at least twice with different inputs. The terms source and follow-up inputs/outputs are used in the same way.

4 Metamorphic Robustness Relations and Findings

In this section, we present eight MRs designed for metamorphic robustness testing of machine translation services, and the failures we have identified when applying them to Google Translate. Some of these MRs are instances of patterns proposed in previous studies. In contrast, however, this research has a focus on translation robustness for *erroneous* input.

4.1 Robustness Testing for Minor Typos in Upper/Lower Cases

Users may accidentally mistype an uppercase character in lowercase, or vice versa. This kind of minor typo should not affect the translation result, except when the change of case carries some specific meaning, such as when "March" is typed as "march", and "May" is entered as "may". We propose the following metamorphic robustness relation:

> **MR$_{case}$.** Minor typos in uppercase or lowercase should not drastically affect the translation.

Our experiment on Google Translate shows that **MR$_{case}$** is violated, revealing a failure in robustness testing. Consider the source input "PETER LOVES APPLES." in the first English line of Fig. 4. The source output is "彼得喜欢苹果。" (literally, "Peter loves apples."), as shown in the first output line. Next, refer to the follow-up input "PETER lOVES APPLES.", where the uppercase character "L" in "LOVES" is mistyped as "l" in lowercase, as shown in the second English line of Fig. 4.[1] The follow-up output is "爱上苹果。" (literally, "Fall in love with apples."), as shown in the second output line. Software testers with elementary knowledge of Chinese[2] can compare the two test results to find a contravention of **MR$_{case}$**, which uncovers a robustness failure.

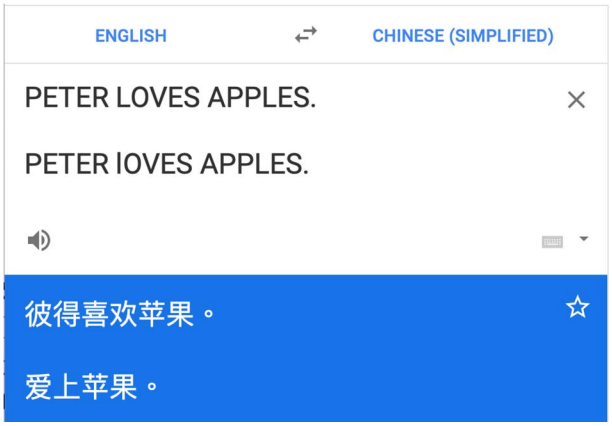


Figure 4: Google Translate results for "PETER LOVES APPLES." and "PETER lOVES APPLES."

4.2 Robustness Testing for Minor Mistakes in Caps Lock

Users may press the caps lock by mistake when typing. It is therefore possible that the first letter of a word is in lowercase while the remaining characters are in uppercase. For instance, a source input "Peter Loves Apples." could be accidentally mistyped into "pETER lOVES aPPLES.". The translator should be sturdy enough to handle this kind of mistake. We propose the following metamorphic robustness relation:

> **MR$_{caps}$.** Pressing the caps lock by mistake should not drastically affect the translation.

Our experiment on Google Translate shows that **MR$_{caps}$** is not satisfied, unveiling a failure in robustness testing. Consider the source input "Peter Loves Apples." in the first English line of Fig. 5. The source output from the translator is "彼得爱苹果。" (literally, "Peter loves apples."), as shown in the first output line. Then, refer to the follow-up input "pETER lOVES aPPLES.", where all uppercase characters of the input are changed to lowercase characters and vice versa because the caps lock is accidentally on. As such, "Peter" becomes "pETER", "Loves" becomes

[1] Note that this failure is different from that of Fig. 1.

[2] Basic knowledge of Chinese is necessary for confirming the violation of metamorphic relations. For example, a difference in the number of characters, such as 彼得喜欢苹果 and 彼得爱苹果, may not necessarily mean a violation, as both of the clauses mean "Peter loves apples". In contrast, two statements with the same number of characters, such as 彼得喜欢苹果 and 彼得讨厌苹果, may be a violation, as the latter clause means "Peter hates apples".

"lOVES", and "Apples" becomes "aPPLES", as shown in the second English line of Fig. 5. The follow-up output from the software is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the second output line. Testers with basic Chinese knowledge can compare the two test results to find a violation of $\mathbf{MR}_{\mathbf{caps}}$, which exhibits a robustness failure.

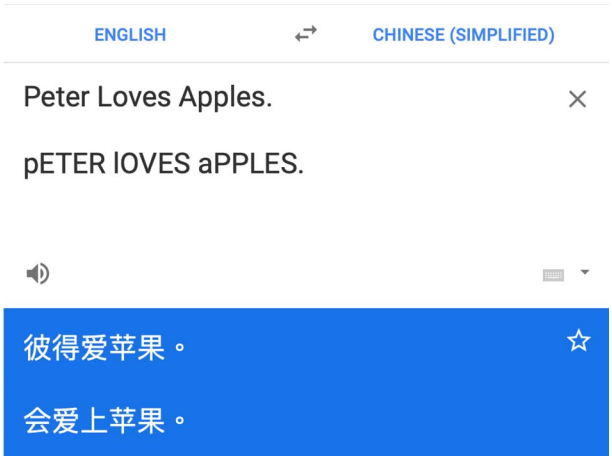


Figure 5: Google Translate results of a source input "Peter Loves Apples." and a follow-up input "pETER lOVES aPPLES."

4.3 Robustness Testing for Omission of Periods

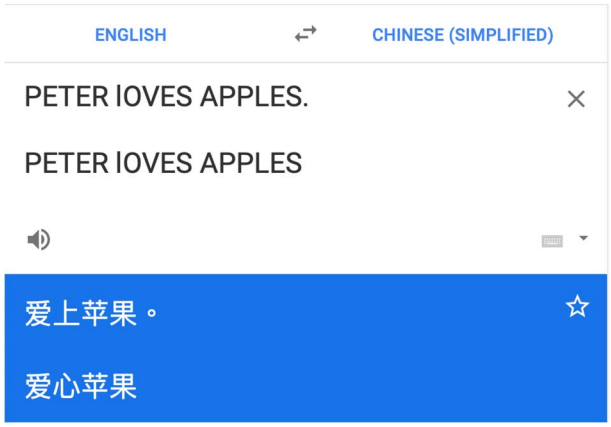
Users may accidentally forget to add the period at the end of a sentence. In other situations, users may even do it on purpose when doing copy-and-paste. Regardless of whether a period is included, the translation result should remain robust. For instance, "Mary loves apples" with or without a period at the end of the sentence should not affect the meaning of this sentence. We propose the following metamorphic robustness relation:

> $\mathbf{MR}_{\mathbf{period}}$**.** Omission of the period should not drastically affect the translation.

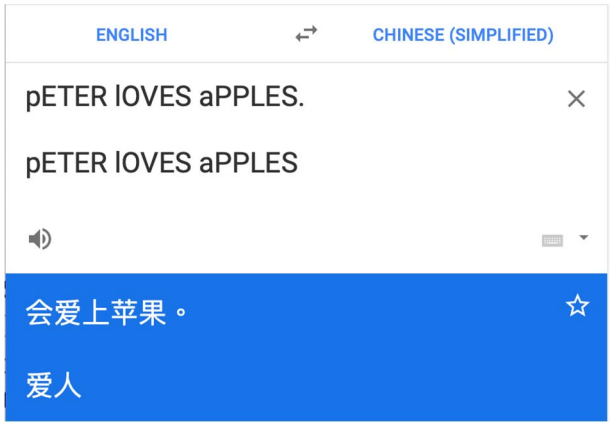
Remark. The impact of periods at the end of a sentence on Google Translate results was first studied by Wu, Sun, and Zhou as an instance of a "noise" *metamorphic relation pattern* [14], where an MR pattern is a general template from which concrete metamorphic relations can be identified. The present study follows up on this work to look at the period at the end of a sentence with respect to robustness testing.

Both the previous study [14] and our current study on Google Translate show that $\mathbf{MR}_{\mathbf{period}}$ has been violated, proving that this is a long-term error of Google Translate that has not been corrected for more than a year, thereby revealing the challenge for quality improvement of deep-learning-based machine translation software [9] even for correcting relatively simple faults.

Consider the source input "PETER lOVES APPLES." in the first English line of Fig. 6(a). The source output from the translator is "爱上苹果。" (literally, "Fall in love with apples."), as shown in the first output line. Note that the source input "PETER lOVES APPLES." is the same as the follow-up input in Fig. 4. Next, refer to the follow-up input "PETER lOVES APPLES", where the period at the end of the sentence is removed, as shown in the second English line of Fig. 6(a). The follow-up output from Google Translate is "爱心苹果" (literally, "apples of a loving heart"), as shown in the second output line. Testers with elementary knowledge of Chinese can compare the two test results to find a contravention of $\mathbf{MR}_{\mathbf{period}}$, revealing a robustness failure.



(a) Google Translate results of a source input "PETER lOVES APPLES." and a follow-up input "PETER lOVES APPLES"



(b) Google Translate results of a source input "pETER lOVES aPPLES." and a follow-up input "pETER lOVES APPLES"

Figure 6: Google Translate results related to omission of periods

Consider another source input "pETER lOVES aPPLES.", as shown in the first English line of Fig. 6(b). The source output from the translator is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the first output line. Note that the source input "pETER lOVES aPPLES." is reused from the follow-up input

in Fig. 5. Then, refer to the follow-up input "pETER lOVES aPPLES" without a period at the end of the sentence, as shown in the second English line of Fig. 6(b). The follow-up output from Google Translate is "爱人" (literally, "spouse"), as shown in the second output line. Testers with basic Chinese knowledge can compare the two test results to find a violation of $\mathbf{MR}_{\text{period}}$, which exposes a robustness failure.

4.4 Robustness Testing for Minor Typos in Punctuation

Punctuation errors are not uncommon. This kind of minor typo should not affect the translation result, except when the punctuation changes the meaning of the sentence drastically. For example, "a man eating chicken" is totally different from "a man-eating chicken". We propose the following metamorphic robustness relation:

> $\mathbf{MR}_{\text{punctuation}}$. A minor typo in punctuation should not drastically affect the translation.

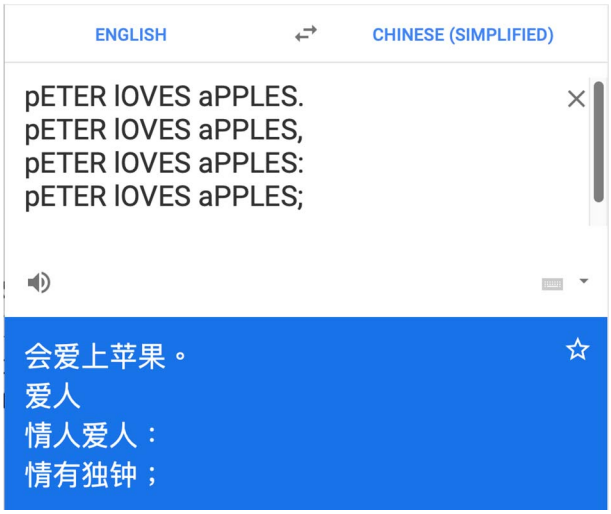


Figure 7: Google Translate results of a source input "pETER lOVES aPPLES." and three follow-up inputs "pETER lOVES aPPLES,", "pETER lOVES aPPLES:", and "pETER lOVES aPPLES;"

Our experiments on Google Translate show that $\mathbf{MR}_{\text{punctuation}}$ is not satisfied, indicating failures in robustness testing. Consider the source input "pETER lOVES aPPLES." in the first English line of Fig. 7. The source output from the translator is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the first output line. Next, refer to the three follow-up inputs that differ from the source input only in the punctuation at the end of the sentence: "pETER lOVES aPPLES,", "pETER lOVES aPPLES:", and "pETER lOVES aPPLES;", as shown in the second, third, and fourth input lines of Fig. 7, respectively. The follow-up outputs from Google Translate are "爱人" ("spouse" with missing comma), "情人爱人：" ("Lover spouse:"), and "情有独钟；" ("Show special favor to;"), as shown in the second to fourth output lines of the figure. Careful readers might criticize that the Chinese translation of the source input "pETER lOVES aPPLES." is already wrong so there is no need to do further checking. Strictly speaking, this observation may be true; but we are trying to verify the robustness of the translator against $\mathbf{MR}_{\text{punctuation}}$ instead of individual translation results. In this case, the translator is robust only if the difference in meaning between the source and follow-up translations is small. However, Google Translate gives drastically different translations. Testers with elementary knowledge of Chinese can compare the four test results to find violations of $\mathbf{MR}_{\text{punctuation}}$, which correspond to robustness failures.

4.5 Robustness Testing for Minor Noise

In a controlled experiment or a very careful use case scenario, users might pre-process all inputs to make sure they are clean without unwanted noise. However, it is impractical in many real-life situations. For example, computer files such as PDF or Microsoft Word files may contain special symbols or patterns that become input noise to the translator. Given significant input noise, the translator will naturally produce incorrect results. For minor noise, on the other hand, the translation software should function sturdily. We propose the following metamorphic robustness relation:

> $\mathbf{MR}_{\text{noise}}$. Minor noise after a sentence should not drastically affect the translation.

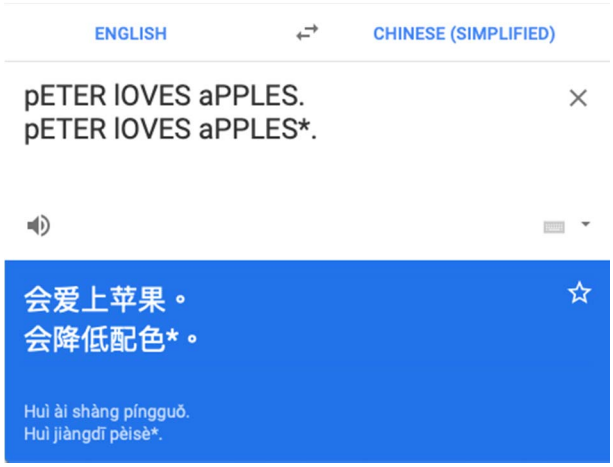


Figure 8: Google Translate results a source input "pETER lOVES aPPLES." and a follow-up input "pETER lOVES aPPLES*."

Remark. $\mathbf{MR}_{\text{noise}}$ is an instance of the "noise" metamorphic relation pattern proposed by Wu et al. [14].

Our experiment on Google Translate shows that $\mathbf{MR}_{\text{noise}}$ is violated, revealing a failure in robustness testing. Consider the source input "pETER lOVES aPPLES." in the first English line of Fig. 8. The source output from the translator is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the first output line. The noise we consider in this test is asterisk "*". Adding extra asterisks at the end of the sentence should not

drastically affect the translation result. Then, refer to the follow-up input "pETER lOVES aPPLES*.", where an asterisk is added before the period, as shown in the second English line of Fig. 8. The follow-up output from Google Translate is "会降低配色*。" (literally, "Will reduce color matching*."), as shown in the second output line. Testers with basic Chinese knowledge can compare the two test results to find a contravention of **MR$_{noise}$**, which uncovers a robustness failure.

4.6 Robustness Testing for Substitution of Subject

In addition to character-level metamorphic robustness relations proposed in Sections 4.1 to 4.5, we have also identified word-level metamorphic robustness relations. The latter involves word changes or replacements.

A typical sentence includes a subject, a verb, and an object, such as "Mary loves apples.". Replacing the subject by another one should not cause a remarkable change of the translation result, such as the entire sentence structure. In other words, modifying only the subject of an input sentence should only change the subject part of the output translation, and leave the remaining parts the same. We propose the following metamorphic robustness relation:

> **MR$_{subject}$.** Changing the subject of a sentence should not drastically affect the remaining parts in the translation.

Remark. **MR$_{subject}$** is an instance of **MR$_{replace}$** proposed by Sun and Zhou [12], who presented the concept of *replacement* as a pattern for the construction of metamorphic relations. More specifically, they wrote that "Our metamorphic relation is named **MR$_{replace}$**. In some situations, if we change the value of a relatively independent component in a system's input, then only a small part (or no part) in the system's output should be changed." They observed that "some changes to the input should not have an impact on the overall structure of the output." They further explained that this concept can be applied to construct concrete metamorphic relations for different application domains such as *autonomous driving* and *machine translation*. Moreover, they showed that their replacement MR pattern in MT4MT can be used to construct MRs by modifying the *subject*, *verb*, or *object* of an English sentence. Using this approach, they found various failures of Google Translate and Microsoft Translator.

The present study is complementary to Sun and Zhou's work [12]. Both their work and our current, more recent experiments on Google Translate show that the replacement MR has been violated, proving that this is a long-term error of the software, which has not been corrected for more than two years, thereby revealing the challenge in quality improvement of deep-learning-based machine translation software. On the other hand, the present study covers the robustness of machine translation, which is not considered in the previous work.

Consider the source input "MARY lOVES APPLES." in the first English line of Fig. 9. The source output from the translator is "玛丽爱苹果。" (literally, "Mary loves apples."), as shown in the first output line. Next, refer to the follow-up input "PETER lOVES APPLES.", where the subject "MARY" of the source input is replaced by "PETER" in the follow-up input, as shown in the second English line of Fig. 9. The follow-up output from Google Translate is "爱上苹果。" (literally, "Fall in love with apples."), as shown in the second output line. Testers with elementary knowledge of Chinese can compare the two test results to find a violation of **MR$_{subject}$**, which exhibits a robustness failure.

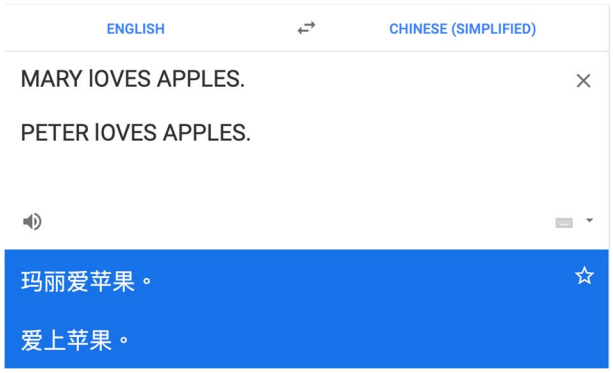


Figure 9: Google Translate results of a source input "MARY lOVES APPLES." and a follow-up input "PETER lOVES APPLES."

4.7 Robustness Testing for Substitution of Verb with Similar Semantic Meaning

Following the same concept of word substitution as explained in Section 4.6, we can also change the verb of the source input with a semantically similar, syntactically equivalent word. Consider, for instance, the sentences "Peter loves apples." and "Peter likes apples.". Since the verbs "loves" and "likes" are so similar, the translation results should be similar, too, both in terms of meanings and sentence structures. We propose the following metamorphic robustness relation:

> **MR$_{verb}$.** Replacing the verb with a similar verb should not drastically affect the remaining parts in the translation.

Remark. **MR$_{verb}$** is an instance of both the "replacement" MR pattern proposed by Sun and Zhou [12] and the "structure-invariant" MR developed by He et al. [8]. He and team followed up on our work of metamorphic testing for machine translation [12] by developing *structure-invariant* MRs based on the principle that "the translation results of similar source sentences should typically exhibit a similar sentence structure." For example, their approach revealed that the Chinese translations given by Google Translate for the sentences "I live on campus with *cute* people." and "I live on campus with *tall* people." had very different sentence structures, hence revealing a translation failure.

Our experiment on Google Translate shows that **MR$_{verb}$** is violated, unveiling a failure in robustness testing. Consider the source input "pETER lOVES aPPLES." in the first English line of Fig. 10. The source output from the translator is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the first output line.

Then, refer to the follow-up input "pETER lIKES aPPLES.", where the verb "lOVES" in the source input is substituted by a semantically similar word "lIKES" in the follow-up input, as shown in the second English line of Fig. 10. The follow-up output is "像喜欢的人。" (literally, "Similar to the person I like."), as shown in the second output line. Software testers with basic Chinese knowledge can compare the two test results to find a contravention of $\mathbf{MR}_{\mathbf{verb}}$, which exposes a robustness failure.

Figure 10: Google Translate results of a source input "pETER lOVES aPPLES." and a follow-up input "pETER lIKES aPPLES."

4.8 Robustness Testing for Substitution of Object

We have covered substitution of the subject and verb in Sections 4.6 and 4.7. The same observation can be applied to substitution of the object of a sentence. Changing or replacing the object should not drastically affect the structure of the translation result. For instance, the translation result of source input "Peter loves apples." should have a similar structure as the follow-up input "Peter loves oranges.". The translation results should differ only in the meaning of the object, while the rest of the translation remains the same. We propose the following metamorphic robustness relation:

> **$\mathbf{MR}_{\mathbf{object}}$.** Changing the object of a sentence should not drastically affect the remaining parts in the translation.

We would like to point out again that this is a study complementary to the work by Sun and Zhou [12], who used this kind of "replacement" MR (involving a change of the subject, verb, or object) to detect failures of Google Translate and Microsoft Translator. Most of the previous studies (except the "noise" MR pattern proposed by Wu et al. [14]) have focused on normal and valid input, that is, well-structured sentences, whereas our present study focuses on erroneous inputs.

Our experiment on Google Translate shows that $\mathbf{MR}_{\mathbf{object}}$ is not satisfied, indicating a failure in robustness testing. Consider the source input "pETER lOVES aPPLES." in the first English line of Fig. 11. The source output from the translator is "会爱上苹果。" (literally, "Will fall in love with apples."), as shown in the first output line. Next, refer to the follow-up input "pETER lOVES oRANGES.", where the object "aPPLES" in the source input is changed to "oRANGES" in the follow-up input, as shown in the second English line of Fig. 11. The follow-up output from Google Translate is "可能会导致错误。" (literally, "May cause errors."), as shown in the second output line. Testers with elementary knowledge of Chinese can compare the two test results to find a violation of $\mathbf{MR}_{\mathbf{object}}$, which corresponds to a robustness failure.

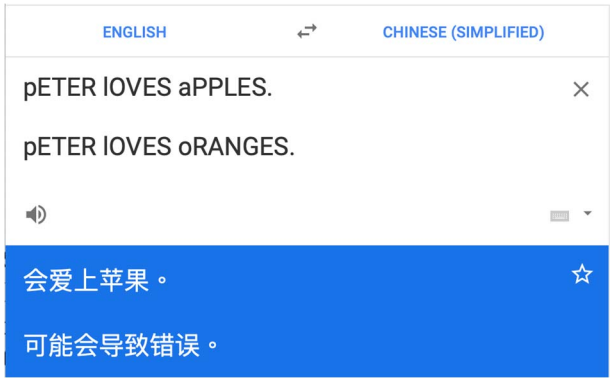


Figure 11: Google Translate results of a source input "pETER lOVES aPPLES." and a follow-up input "pETER lOVES oRANGES."

5 Related Work

Fuzzing [6], also known as fuzz testing, is a main approach to robustness testing. It involves the use of random or semi-random data as input to the SUT. The SUT execution is monitored for erroneous behavior especially for crashes and hangs which, if detected, could reveal vulnerabilities of the SUT. Fuzzing has been used to detect some specific types of software faults and vulnerabilities in a variety of real-life systems. Due to the lack of a test oracle, however, fuzzing alone can hardly detect *logic errors*, which are bugs that do not crash or hang the SUT but instead produce incorrect output values, such as the errors presented in this paper.

Zhou et al. [17][19] were the first to combine metamorphic testing and fuzzing (testing search engines with random ASCII strings) to address the oracle problem. Even if the search engine under test did not crash or hang, a failure could still be detected if the MR was violated. More recently, metamorphic testing and fuzzing have been combined to test security-critical applications [4], and to test graphics drivers for the Android ecosystem at Google [5][7].

Another limitation of fuzzing is that it cannot fuzz the *environment*. For example, it is not difficult to apply a fuzzer to generate random query terms to test a search engine, but it is infeasible to change the real-world Internet where the search engine's crawler runs. To address this problem, as well as the oracle problem, Zhou et al. [18] introduced the term *metamorphic robustness testing*, and applied the approach to test the citation database systems Scopus and Web of Science.

More recently, Zhang et al. [16] conducted a meticulous survey on the testing of machine learning, covering 138 papers from 2007 to 2019.

6 Discussions and Conclusions

Our research group initiated the application of metamorphic testing to alleviate the test oracle problem in machine translation software [10]. The present paper is part of a series of studies that we have conducted since then on metamorphic testing for machine translation [10][12][14][15], which we refer to as MT4MT for short.

The first contribution of this article is the presentation of eight metamorphic robustness relations that can be used to test the sturdiness of machine translation services. Some of these relations may not be completely new, but a unique characteristic of the present investigation is that we focus on erroneous input, in contrast to previous studies where most of the MRs were designed with anticipated, normal input in mind. Designing effective MRs for erroneous inputs is the most important contribution of this research.

The second contribution of this work is the detection of several categories of striking translation errors in Google Translate, the most popular machine translation service in the world (outside of the Mainland of China). While Google Translate bugs have also been reported previously, those papers mainly focused on "positive testing", which attempts to ensure that the translator returns reasonable results for normal, valid, and well-constructed input sentences. By contrast, the translation errors reported in the present study focus on "negative testing", which attempts to ensure that the translator gives reasonable translations when the input sentence contains unanticipated errors. To the best of our knowledge, this is the first study on machine translation testing with a focus on robustness against erroneous inputs.

A potential limitation of this work is that we have presented the test results as case studies in a qualitative way, rather than using a systematic quantitative approach to generate failure-related statistics such as failure rates, failure patterns, and distributions and characteristics of the failure-causing inputs. These will be studied in future research. The test method presented in this paper can be fully automated, as it does not require an advanced oracle to decide on the correctness of the translations. It only needs a simple mechanism to detect an MR violation. For instance, we may adopt a standard similarity metric for text comparison in order to identify dissimilar translations [10].

The detection of the translation errors as reported in this paper is meaningful also in the sense that they have confirmed several categories of previously reported bugs in Google Translate, although the previous studies were conducted in the context of positive testing. This means that some of these bugs have existed in Google Translate for more than a couple of years, demonstrating the challenges and practical difficulties in the quality assurance and quality improvement of deep-learning software. These findings raise a further alert on the increasing (risky) use of machine learning technology for mission-critical tasks.

As future work, we also plan to study the robustness of other popular machine translators as well as the robustness of the translation of other languages.

ACKNOWLEDGMENT

This work was supported in part by a linkage grant of the Australian Research Council (Project ID: LP160101691) and a Western River entrepreneurship grant. All correspondence should be addressed to Dr. Z. Q. Zhou at the address shown on the first page of this paper.

REFERENCES

[1] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. 2015. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering* 41, 5, 507–525.

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

[3] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou. 2018. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys* 51, 1, 4:1–4:27.

[4] T. Y. Chen, F.-C. Kuo, W. Ma, W. Susilo, D. Towey, J. Voas, and Z. Q. Zhou. 2016. Metamorphic testing for cybersecurity. *Computer* 49, 6, 48–55.

[5] A. F. Donaldson and A. Lascu. 2016. Metamorphic testing for (graphics) compilers. In *Proceedings of the 1st International Workshop on Metamorphic Testing* (*MET'16*) (*in conjunction with the 38th International Conference on Software Engineering* (*ICSE'16*)). ACM Press, New York, NY, 44–47.

[6] P. Godefroid. 2020. Fuzzing: Hack, art, and science. *Communications of the ACM* 63, 2, 70–76.

[7] Google/GraphicsFuzz. 2018. A testing framework for automatically finding and simplifying bugs in graphics shader compilers. https://github.com/google/graphicsfuzz/

[8] P. He, C. Meister, and Z. Su. 2019. Structure-invariant testing for machine translation. *CoRR* abs/1907.08710.

[9] D. Monroe. 2017. Deep learning takes on translation. *Communications of the ACM* 60, 6, 12–14.

[10] D. Pesu, Z. Q. Zhou, J. Zhen, and D. Towey. 2018. A Monte Carlo method for metamorphic testing of machine translation services. In *Proceedings of the 3rd International Workshop on Metamorphic Testing* (*MET'18*) (*in conjunction with the 40th International Conference on Software Engineering* (*ICSE'18*)). ACM Press, New York, NY, 38–45. DOI: https://doi.org/10.1145/3193977.3193980

[11] Standards Coordinating Committee of the IEEE Computer Society. 1990. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990, IEEE Computer Society Press, Los Alamitos, CA.

[12] L. Sun and Z. Q. Zhou. 2018. Metamorphic testing for machine translations: MT4MT. In *Proceedings of the 25th Australasian Software Engineering Conference* (*ASWEC'18*). IEEE Press, Piscataway, NJ, 96–100.

[13] Z. Sun, J. M. Zhang, M. Harman, M. Papadakis, and L. Zhang. 2020. Automatic testing and improvement of machine translation. In *Proceedings of the 42nd International Conference on Software Engineering* (*ICSE'20*). ACM, New York, NY.

[14] C. Wu, L. Sun, and Z. Q. Zhou. 2019. The impact of a dot: Case studies of a noise metamorphic relation pattern. In *Proceedings of the IEEE/ACM 4th International Workshop on Metamorphic Testing* (*MET'19*) (*in conjunction with the 41st International Conference on Software Engineering* (*ICSE'19*)). IEEE Press, Piscataway, NJ, 17–23.

[15] B. Yan, B. Yecies, and Z. Q. Zhou. 2019. Metamorphic relations for data validation: A case study of translated text messages. In *Proceedings of the IEEE/ACM 4th International Workshop on Metamorphic Testing* (*MET'19*) (*in* conjunction *with the 41st International Conference on Software Engineering* (*ICSE'19*)). IEEE Press, Piscataway, NJ, 70–75.

[16] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*. DOI: https://doi.org/10.1109/TSE.2019.2962027

[17] Z. Q. Zhou, T. H. Tse, F.-C. Kuo, and T. Y. Chen. 2007. Automated functional testing of Web search engines in the absence of an oracle. Technical Report TR-2007-06, Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong.

[18] Z. Q. Zhou, T. H. Tse, and M. Witheridge. 2019. Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors. *IEEE Transactions on Software Engineering*. DOI: https://doi.org/10.1109/TSE.2019.2915065

[19] Z. Q. Zhou, S. Zhang, M. Hagenbuchner, T. H. Tse, F.-C. Kuo, and T. Y. Chen. 2012. Automated functional testing of online search services. *Software Testing, Verification and Reliability* 22, 4, 221–243.