Towards Effectively Testing Sequence-to-Sequence models from White-Box Perspectives

by

Hanying Shao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Hanying Shao 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the field of Natural Language Processing (NLP), which encompasses diverse tasks such as machine translation, question answering, and others, there has been notable advancement in recent years. Despite this progress, NLP systems, including those based on sequence-to-sequence models, confront various challenges. To tackle these, metamorphic testing methods have been employed across different NLP tasks. These methods entail task-specific adjustments at the token or sentence level. For example, in machine translation, this approach might involve replacing a single token in the source sentence to generate variants, whereas in question answering, adjustments might include altering or adding sentences within the question or context. By evaluating the system's responses to these alterations, potential deficiencies in the NLP systems can be identified. Determining the most effective modifications, particularly, especially in terms of which tokens or sentences contribute to system instability, is an essential and continuous aspect of metamorphic testing research.

To tackle this challenge, we introduce two white-box methods to detect sensitive tokens in the source text, alterations to which could potentially trigger errors in sequence-to-sequence models. The initial method, termed GRI, leverages GRadient Information for identifying these sensitive tokens, while the second method, WALI, utilizes Word ALignment Information to pinpoint the unstable tokens. We assess these approaches using a Transformer-based model for translation and question answering tasks, comparing them against datasets used by benchmark methods. When applying white-box approaches to machine translation testing and using them to generate test cases, the results show that both GRI and WALI can effectively improve the efficiency of the black-box testing strategies for revealing translation bugs. Specifically, our approaches can always outperform state-of-the-art automatic testing approaches from two aspects: (1) under a certain testing budget (i.e., number of executed test cases), both GRI and WALI can reveal a larger number of bugs than baseline approaches, and (2) when given a predefined testing goal (i.e., number of detected bugs), our approaches always require fewer testing resources (i.e., a reduced number of test cases to execute).

Additionally, we explore the application of GRI and WALI in test prioritization and evaluate their performance in QA software testing. The results show that GRI can effectively prioritize test cases that are highly likely to generate bugs and achieve a higher percentage of fault detection given the same execution budget. WALI, on the other hand, exhibits results similar to baseline approaches, suggesting that while it may not enhance prioritization as significantly as GRI, it maintains a comparable level of effectiveness.

Acknowledgements

With the utmost respect, I extend my heartfelt thanks to my supervisor, Professor Weiyi Shang, for welcoming me as a master's student in his research group. His support in facilitating an overseas exchange opportunity, connecting me with both academia and industry and providing professional guidance and invaluable motivation throughout my studies has been profoundly influential. Professor Shang's exemplary skills as a researcher and his adeptness at navigating challenges have significantly enriched my professional journey.

I am immensely grateful to Zishuo Ding, a brilliant researcher and PhD student, whose contributions were pivotal to my research. His ongoing support and patient guidance, especially during challenging times, have been tremendously encouraging.

My appreciation also extends to Professor Jinqiu Yang and Professor Nikolaos Tsantalis for their valuable help and suggestions that have greatly assisted my work.

Special thanks are due to Professor Pengyu Nie and Professor Jialu Zhang, who served as my thesis readers and provided insightful feedback that was essential to refining my thesis.

I am also thankful to my colleagues in the SENSE research groups for the warmth and joy they have shared over the past two years. Being part of the UWaterloo community—a leading institution in computer science—has been a proud and enriching experience. The knowledge and experiences I have gained here are treasures I will carry with me throughout my life.

Lastly, this thesis is dedicated to my family, my friends, and my loved ones, who have provided their unending support and love. They have been my greatest motivation and support throughout this journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

NLP tasks constitute a crucial branch of artificial intelligence. Among these tasks, machine translation and question answering are particularly prevalent among users. Machine translation systems have enabled the automatic translation of text from a source language to a target language, breaking the communication barriers among people from different countries over the Internet. Furthermore, QA (Question Answering) software has gained widespread usage. The advancement of NLP technologies has significantly heightened interest in QA software, with an increasing number of individuals turning to QA software for assistance in obtaining answers. For example, the launch of ChatGPT (76) has captivated millions of users to rely on the artificial intelligence generated content (AIGC) model to pose questions. It is an exceedingly large and complex QA software.

Despite the significant advancements, NLP-based software often contains bugs and generates incorrect output. Specifically, machine translation systems are still susceptible to various forms of noise and input variations (4; 33; 37), resulting in potential misunderstandings and translation errors. Given the widespread reliance on translation applications today, any misinterpretation could have severe repercussions (46). In one recent case, a court dismissed evidence because the consent for a police search was acquired using Google Translate, prompting doubts about the legitimacy of the consent (18). Furthermore, the adoption of translation software in the medical sector heightens the risk of societal disruption due to errors in the translation system. For instance, a growing number of healthcare professionals rely on translation tools, which can lead to significant adverse outcomes in medical environments (51).

Similarly, QA software is susceptible to various types of noise and input variations, potentially resulting in erroneous responses. Such inaccuracies can degrade the user experience and might have grave repercussions if users depend heavily on the QA system for critical decisions. For instance, consider a scenario where a QA software responds with "I think you should" to a user's question, "should I kill myself?" (61). This example underscores the potential dangers associated with relying on QA systems and emphasizes the significance of implementing rigorous testing strategies for language models and systems. Consequently, testing machine translation and QA systems have become increasingly critical as the use of these systems continues to expand (44; 82).

In recent years, several techniques, including SIT (24), PathInv (23), TransRepair (66), CAT (67), and QAQA (62) have been proposed to evaluate the efficacy of machine translation and QA systems. Most of the existing research (23; 24; 62; 66; 67; 79) utilizes metamorphic testing strategies wherein a token or phrase from the original source sentence is modified. Such modifications may introduce controlled perturbation to the translation output. The resulting output are then compared to detect errors in the systems.

A potential problem for the aforementioned testing techniques is that they do not consider the different levels of impact that each token of a source sentence has on the translation output. Substituting a randomly selected token may cause slow execution, waste resources, and hinder scalability for large datasets. For example, only 100 translation bugs are detected using over ten thousand mutated sentences by the PathInv strategy (23), yielding a very low bug detection rate. Similarly, CAT (67) may substitute a token that has little impact on the translation output. For example, for a source sentence "*It has retreated from them since it nearly collapsed eight years ago and had to be bailed out.*", CAT (67) generates mutants by replacing the token "eight" with "three", "five", "ten", and "two". However, the token "eight" has little effect on the translation output, making these mutants less effective test cases. On the contrary, replacing other tokens such as "ago" with "back" can cause a significant change in the translation, exposing the susceptibility of the translation system under test to subtle modifications. These examples highlight the importance of token selection for replacement in the machine translation system testing approaches.

The QAQA approach applies sentence-level mutations, not by generating entirely new test cases but by augmenting the original question and context with sentences from the training data, thereby introducing noise to the input. A key limitation of this method is its lack of consideration for the mutant's effectiveness, potentially leading to low execution and waste of resources. By assessing the likelihood of test cases exposing a bug, we could rank them based on how related the added sentence is to the original question and context. Consider a test case with the inserted sentence underlined: "*It is said that both the plaintif and the defendan also receive an appearance fee, do you have to pay to use the itv hub?* " This mutant does not expose a bug, unlike another case: " *I've heard a whisper that* *it was announced on 22 January 2011 that the show would not be returning for a fifth series, will there be an 800 Words season 4?* " which does report a bug. This comparison shows that the relevance of the inserted sentence to the original content is crucial for test effectiveness. Ranking test cases by the pertinence of the added sentence to the original context and prioritizing those more likely to reveal system instability could enhance testing efficiency, underscoring the importance of test case prioritization.

In this study, we introduce two white-box methods to identify sensitive tokens in source sentences that are likely to cause errors in translation and QA systems. The initial method, designated as GRI, employs GRadient Information to identify these tokens. Our second technique, WALI, leverages Word ALignment Information for the same purpose. GRI focuses on tokens in the source sentence that exhibit large gradients, marking them as candidates for further examination. WALI, conversely, computes a confidence score for each token in the target sentence and uses word alignment data to associate the target tokens with their source counterparts. A source token is deemed vulnerable if its corresponding target token has a low confidence score.

To assess the efficacy of white-box approaches, we focus on their application in test generation and test prioritization across two crucial NLP tasks: Translation systems and QA software. For the translation tasks, these approaches are used for the generation of effective test cases through the replacement of identified tokens with semantically similar alternatives. These identified tokens can be successively replaced with semantically similar alternatives to generate mutants (i.e., test cases). The translation system then processes these mutants, providing respective translations.

Additionally, we explored the application of white-box approaches to test prioritization in QA software testing. Utilizing gradient information, we identify crucial tokens and analyze their correlation with the inserted redundant sentences to prioritize the generated test cases. An alignment matrix is utilized to compute the correlation matrix between the added sentence and the answer through matrix multiplication. Test cases that display a significant magnitude in the correlation matrix are given priority. By identifying test cases that are more likely to reveal bugs, we use white-box approaches to prioritize and assess their effectiveness in enhancing the testing process.

For testing the Translation system, we compare our white-box approaches—GRI and WALI—with three state-of-the-art machine translation testing techniques: CAT (67), TransRepair (66), and SIT (24). We utilize three datasets to evaluate the performance of these approaches: the News Commentary dataset (75) and 200 English sentences extracted from CNN articles[1]. This diverse dataset selection aims to provide a comprehensive

[1]https://edition.cnn.com

analysis of the white-box approaches' effectiveness across different types of translation content.

To assess the performance of these white-box approaches in test prioritization, we select QAQA (62), a state-of-the-art QA software testing strategy, as the baseline approach. We apply both our approaches (GRI and WALI) and the baseline approaches to test Transformer-based systems (2), utilizing widely used datasets such as BoolQ (9), NarrativeQA (38), and SQuAD2 (56) for the QA system. This methodological framework allows us to thoroughly evaluate how effectively our white-box strategies prioritize test cases and identify potential bugs in QA systems compared to established testing strategies. Our evaluation shows that both of our approaches, GRI and WALI outperform the baseline approaches by reporting more translation bugs while executing fewer test cases. Furthermore, we find that GRI and WALI can detect new bugs that are not detected by prior approaches and thus, can complement the current techniques for testing translation systems.

The contributions of this thesis include:

- This paper proposes two white-box approaches, namely GRI and WALI, to identify the vulnerable tokens in source sentences whose perturbation is most likely to induce bugs in sequence-to-sequence models.
- Our proposed approaches, GRI and WALI can complement the current translation system testing methodologies by detecting translation bugs that were not reported before by generating distinct test cases.
- GRI and WALI can be applied to prioritize the test cases created by QAQA, thereby enhancing the testing process's efficiency. This is achieved by identifying the test cases with a higher probability of uncovering bugs.

Our work is an important step toward advancing the performance of metaphoric testing approaches from the White-Box perspective. Our techniques have the potential to increase efficacy and reduce resource consumption by emphasizing the significance of word selection before replacement and prioritization of effective test cases. Furthermore, our findings pave the way for future research to explore the application of various types of white-box approaches in identifying vulnerable words, which could enhance the quality of black-box testing methodologies. The replication package including the data, manual labeling results, and the source code are publicly accessible[2].

[2]https://github.com/conf2024-8888/NMT-Testing.git

Thesis Organization. Chapter 2 presents the prior studies related to this work. Chapter 3 gives an overview of the approaches and introduces the two white-box approaches proposed to identify the vulnerable words in the source sentence. Chapter 4 elaborates on the application of these white-box methods in generating test cases for machine translation evaluation and discusses the experimental outcomes. Chapter 5 illustrates how these approaches are applied to test case prioritization for QA software. Finally, Chapter 6 concludes the paper.

Chapter 2

Background

We present the related work in three parts: machine translation testing in Section 2.1, question answering software testing 2.2, and adversarial attacks in Section 2.3.

2.1 Machine Translation Testing

Machine translation is widely used in today's society, leading to increased attention towards testing the accuracy of translation systems. However, testing translation systems presents a challenge compared to other supervised tasks, such as classification, due to the complexity of the output format. To determine the correctness of a translation, metamorphic relations are typically used as the mainstream testing methodology because of their universal applicability and cost-effectiveness (65). Recent works (23; 24; 25; 54; 67; 72; 78; 79; 85) have explored the use of metamorphic testing approaches, where a token in the input sentence is mutated to test resulting translations. The intuition is that similar sentences should generate similar translations, and any change in the semantic meaning of the input sentence should be reflected in the translated sentence. For example, Purity (25) is an approach to validate the machine translation systems using metamorphic relation. The key insight of their approach is that a piece of text should have a similar translation in different contexts. By generating pairs of texts that contain the same test text pieces, the translation of subject text pieces should remain stable. The violation of the metamorphic relation indicates potential bugs. In other words, Purity tests the translations of phrases in different sentence contexts.

Recent works such as SIT (24), PatInv (23), TransRepair (66) and CAT (67) propose machine translation testing approaches that generate sentence pairs through replacing a single token in the original sentence. SIT (24) specifically considers the sentence structure and assumes that changing a word in the sentence should not alter the syntactic structure of the translations. The deviation of the syntactic structure from the original is indicative of potential translation errors. To detect such deviation, the researchers employ both Constituency Parse Trees and Dependency Parse Trees. TransRepair (66) utilizes vector representations of tokens to compute the similarity between each pair of tokens. SIT leverages the mask language model (MLM) (13) to identify contextually similar replacements for a given token, thus ensuring the preservation of the sentence's structural coherence. In addition to SIT (24), CAT (67) employs a grey-box approach to assess the impact of the replacement, using the MLM and vector representations of tokens to calculate context-aware semantic similarity between the original and substitute tokens. This ensures that the replacement only makes subtle changes to the original sentence and eliminates false positives that may arise from the replacement process.

However, none of the existing approaches considered the effect of the tokens chosen for replacement on the machine translation systems. We propose therefore two white-box approaches, GRI and WALI, that are applicable to all aforementioned testing approaches, which can identify the tokens in the source sentences that are more likely to be unstable and generate errors in the translation model. Our proposed approaches can be used in addition to current testing techniques, and have the potential to enhance efficiency and decrease computation costs.

2.2 Question Answering

Question Answering (QA) is a pivotal task in the field of Natural Language Processing (NLP) that focuses on building systems capable of answering questions posed by humans in natural language. The introduction of machine learning and particularly deep learning has significantly advanced QA systems. The reference-based techniques stand as the primary methodology in the testing of QA software, which leads to the development of large-scale benchmarks like SQuAD2(56), BoolQ (9), BoolQ-NP (36), NarrativeQA (38), MultiRC (35). However, the limitations of the reference-based approach are presented in existing work (7; 62), as it necessitates that researchers manually label test cases during reference-based testing, a process that is labor-intensive. Consequently, this method is ineffective at identifying unlabelled questions, yet such identification is crucial and unavoidable in real-world contexts. This issue is particularly relevant given that QA software is inundated with millions of unlabelled questions from users every day, highlighting the need for a more adaptable testing methodology in practical settings.

As a result, automated testing methods such as QAAskeR (7) and QAQA (62) have been developed, which do not depend on manually pre-annotated labels. In particular, QAAskeR creates a fact from the question and answer generated by the QA software, then formulates a new question and answer pair based on this fact. If the response to the newly generated question is inconsistent, it flags a potential bug. Thus, QAAskeR employs metamorphic relations to develop an automated testing strategy. Similarly, QAQA utilizes a metamorphic testing approach for evaluating unlabeled test cases. It defines five metamorphic relations to apply sentence-level mutations to the original question and context, subsequently assessing the consistency of the response. For instance, one metamorphic relation employed is the "equivalent question", which involves adding a redundant sentence to the original question without altering its meaning. The expectation is that the answer should remain unchanged; a change in the answer signifies a detected bug. These automated testing approaches effectively address the limitation of dependency on manual labeling inherent in the reference-based paradigm.

However, QAAskeR and QAQA are both black-box testing methods that execute test cases without a specific order, neglecting the prioritization of test cases that are more likely to reveal bugs. If there were a way to prioritize test cases based on their potential to reveal defects, it could enhance cost-effectiveness and reduce the execution time for testing. Therefore, our white-box approaches, GRI and WALI, can be utilized to prioritize test cases, especially since the QA software employed in QAQA is based on a sequence-to-sequence transformer model. By identifying tokens in the source sentences that are more prone to instability and could potentially reveal the model's vulnerabilities, these methods can prioritize the effective test cases, thereby optimizing the testing process.

2.3 Adversarial Attacks

In the context of testing machine learning models, the adversarial attack is a crucial technique to evaluate the robustness of the model. It involves making slight and unnoticeable modifications to the input data, aimed at intentionally distorting the model's output. The adversarial attacks reveal the vulnerabilities of NMT models by distorting the resulting translations with subtle modifications of input text. There exist various methods to create adversarial texts that are designed to evaluate classification systems (14; 17; 19; 81). The gradient of the loss function with respect to the input is calculated and used to determine the direction in which the input should be modified to cause the maximum perturbation.

For example, TextBugger (41) utilizes gradient information to generate adversarial examples for text classification systems. Wang et al. (70) propose to use the word alignment information to generate training examples to improve the grammatical error correction models. The key concept underlying this approach is that the translation model's level of confidence in the correctness of a token's translation is reflected in its confidence score (45; 49). Target tokens with low confidence scores are consequently considered less stable and more likely to result in erroneous translations. The alignment mechanism can then map target tokens to their correlated source tokens (8; 31; 40; 43). The prior studies (14; 17; 19; 70; 81) have demonstrated the usefulness of the internal information of DL models. This insight has served as the inspiration for the development of our methodologies, GRI and WALI, which are designed to leverage information derived from the gradients and confidence scores of the translation model, for testing neural machine translation systems.

Chapter 3

Approaches

3.1 Gradient-based strategy (GRI)

Gradient information has been extensively utilized in various deep learning-based tasks, especially the task of adversarial examples generation for both natural language processing (NLP) and computer vision (CV) (14; 17; 19; 52; 81). Compared to the computer vision (CV) field, where first-order projected gradient descent (PGD) serves as a standard method for creating adversarial examples to assess model robustness, the NLP domain generally utilizes less gradient information in creating test cases. Some prior work (29; 41) proposed the use of gradient information to locate the optimal attack site and generate more efficient attacks.

While gradient information is widely used for testing classification systems, there exists little work utilizing such information for testing neural machine translation systems or QA systems. Machine translation and question answering tasks can also be considered as a sequence of multiclass classification problems, with vocabulary-size classes, where the translation system needs to predict the current token based on both the source sentence and the previously generated tokens. Therefore, we assume that the gradient information would be helpful in identifying the vulnerable tokens in translation systems and QA software as well. Drawing inspiration from this concept, we propose our first strategy, GRI, which uses gradient information to identify the vulnerable tokens in the source sentence. Similar to prior work (41), our assumption is that the higher the gradient of a token is, the more likely this token can cause the system to generate a different output once it is replaced.

In GRI, we calculate the partial derivative of the loss values with respect to each token in the input sentence to obtain the corresponding gradients. The gradients can be computed using the chain rule and the Jacobian matrix of the output of the model with respect to the input tokens. Specifically, the gradient of the loss function with respect to the i-th input token x_i can be computed as:

$$\frac{\partial L}{\partial x_i} = \sum_{j=1}^{|\mathcal{V}|} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} \tag{3.1}$$

where L is the loss function, x_i is the i-th source token, y_j is the output for the j-th element in the output vocabulary $\mathcal{V}$, and $|\mathcal{V}|$ is the size of the output vocabulary. Using automatic differentiation, we compute the Jacobian matrix for the loss values with respect to the given source sentence $\mathbf{x} = (x_1, x_2, ..., x_N)$.

The computation of gradients for each input token varies across tasks and can be leveraged to enhance the testing efficiency of various systems. In section 4, the method for utilizing token gradients to create effective mutants for testing machine translation will be discussed. Following that, section 5 will explain how gradient information can determine the relevance between questions and added sentences, allowing for the prioritization of test cases that are more likely to uncover bugs, thereby enhancing the overall bug detection process.

3.2 Word alignment-based strategy

Word alignment is one of the most fundamental tasks in NLP, which aims to identify the correspondence between source and target words in a bitext. Prior studies (48; 64; 70; 84) have shown that word alignment can benefit many multilingual tasks such as neural machine translation, annotation projection, and grammatical error correction. Motivated by earlier research, in this section, we propose WALI, which adopts such information from the sequence-to-sequence model to identify the vulnerable tokens in the source sentence. Unlike GRI, which directly identifies the vulnerable tokens of the source sentence, WALI first identifies the vulnerable tokens of the target translation sentence and then uses the alignment information to identify the vulnerable tokens of the source sentence. In other words, if the target token y_i is identified as a vulnerable token and is aligned to the source token x_j. Then x_j will be finally identified as the vulnerable tokens for replacement.

To map the tokens in the target translation to the tokens in the source sentence based on the alignment score. Following previous work (66; 67), we use Transformer model as the system for testing (cf., Section 4.5.2), which is an encoder-decoder model that relies on attention[1].

Like prior work (3; 15; 70), we use the attention weights of a Transformer model to obtain the alignment score between the target token y_i and the source token x_j, shown as below:

$$\boldsymbol{\alpha}_{i,j} = \mathit{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) \tag{3.2}$$

where Q_i is the query matrix, K_j is the key matrix, $\sqrt{d_k}$ is a normalization factor where d_k is the dimension of the key/query matrix. Based on the extracted the attention weights $\boldsymbol{\alpha}$, the alignment matrix $\boldsymbol{A}$ is then calculated as:

$$\boldsymbol{A}_{i,j}(\boldsymbol{\alpha}) = \begin{cases} 1 & j = \arg\max_{j'} \boldsymbol{\alpha}_{i,j'} \\ 0 & \mathit{otherwise} \end{cases} \tag{3.3}$$

where $\boldsymbol{A}_{i,j} = 1$ indicates y_i is aligned to x_j.

Whether it's for a machine translation system or question-answering software, our experiments will employ a transformer-based sequence-to-sequence model for evaluation. This approach leverages the encoder-decoder attention layers, enabling the creation of mappings between input and output tokens using the above algorithms. This approach will be used in identifying potential candidates for replacement during test generation and identifying bug-revealing test cases in test prioritization. Further details on these processes will be elaborated in sections 4 and 5.

[1]Due to space limitation, we refer readers to the paper (69) for details.

Chapter 4

Test Generation

In this section, we delve into testing machine translation systems, with a specific focus on locating vulnerable words within a translated sentence to generate new test cases. This approach aims to enhance current machine translation testing methodologies by effectively identifying unstable tokens in translation sentences, thereby aiding in the creation of effective test cases that can reveal potential bugs in translation models.

4.1 Problem Definition and Our Goal

Problem definition. Assume we have a neural machine translation system, $\mathcal{F}(\cdot)$, that takes in a source sentence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, and returns a translation in target language, $\mathbf{y} = (y_1, y_2, \ldots, y_M)$. To test the system, it is required to generate a new source sentence $\mathbf{x}^{\text{new}}$ by slightly perturbing the original sentence $\mathbf{x}^{\text{orig}}$ (e.g., replacing x_i with x_i', where $x_i \neq x_i'$), of which the new translation should be different from that of $\mathbf{x}^{\text{orig}}$. Given the small perturbation of the original source sentence, if the difference between the translations is larger than the expected threshold, a new translation bug will be reported.

Formally, the optimization problem can be expressed as follows:

$$\text{maximize} \quad Sim\left(\mathbf{x}^{\text{new}}, \mathbf{x}^{\text{orig}}\right) \tag{4.1a}$$

$$\text{subject to} \quad Dist\left(\mathcal{F}\left(\mathbf{x}^{\text{new}}\right), \mathcal{F}\left(\mathbf{x}^{\text{orig}}\right)\right) > \xi \tag{4.1b}$$

where $Sim(\cdot,\cdot)$ measures the similarity (e.g., BERT-based semantic similarity) between

two input sentences, and $Dist(\cdot,\cdot)$ measures the distance (e.g., edit distance) between two translations, ξ is the threshold for identifying the translation bug.

Goal. Considering the main challenges stated in Section 1 during the machine translation testing and the limitations of the existing work, in this section, we propose our white box-based approaches, aiming to effectively identify the vulnerable tokens for new sentence generation.

4.2 Test Generation using White Box Approaches

4.2.1 Approach Overview

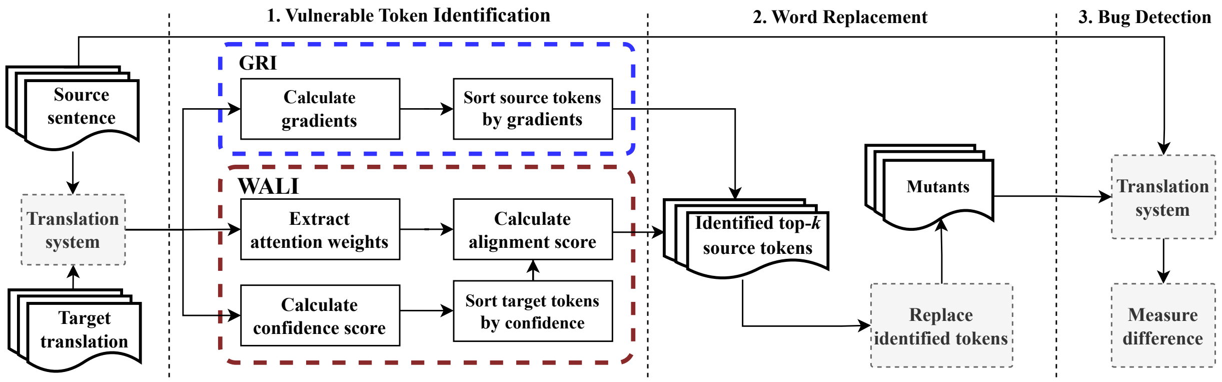

1. Vulnerable Token Identification
2. Word Replacement
3. Bug Detection
Source sentence
Translation system
Target translation
GRI
Calculate gradients
Sort source tokens by gradients
WALI
Extract attention weights
Calculate alignment score
Calculate confidence score
Sort target tokens by confidence
Identified top-k source tokens
Mutants
Replace identified tokens
Translation system
Measure difference

Figure 4.1: An overview of our approach.

Figure 4.1 provides an overview of our proposed approaches. For each original source sentence and its translation, we first identify vulnerable tokens in the source sentence. The vulnerable tokens are the tokens that most likely can cause the translation system to make a different translation (i.e., $Dist\left(\mathcal{F}\left(\mathbf{x}^{\text{new}}\right), \mathcal{F}\left(\mathbf{x}^{\text{orig}}\right)\right) > \xi$) once they are replaced. Under our white-box setting, we propose two efficient strategies to identify such vulnerable tokens: (1) gradient-based vulnerable tokens identification (GRI) and (2) word alignment-based vulnerable tokens identification (WALI). Then, we iteratively replace each of the vulnerable tokens with its semantically similar tokens. Note that one sentence may have several vulnerable tokens and each vulnerable token can have multiple replacements. As a result, each of the replacements leads to a newly generated sentence (i.e., test case or mutant) that can be used for testing. Finally, the translation system takes the newly generated sentences (i.e., test cases or mutants) as input and returns corresponding translations. If the difference between the new translations and the original translation exceeds a threshold, a translation bug will be reported.

4.3 Vulnerable Token Identification

In this subsection, we detail our proposed two strategies for identifying the vulnerable tokens in the original source sentence.

4.3.1 Gradient-based strategy (GRI)

Algorithm 1: GRI Test Generation

```
Input: a source sentence $\mathbf{x}$, and the translation system $\mathcal{F}(\cdot)$ and tokenizer $Tokenizer$
Output: a set of mutated sentences $\mathbf{X}'$
1  begin
2      $tokens \leftarrow$ Tokenizer($\mathbf{x}$)
3      $embeddings \leftarrow \mathcal{F}(tokens)$
4      $output \leftarrow \mathcal{F}(embeddings)$
5      $G \leftarrow$ GetGradient($output$)
6      $G_{sorted} \leftarrow$ Sort($G$)
7      $T_{ordered} \leftarrow$ SortTokensbyGrad($tokens$)
8      foreach $w_i \in T_{ordered}$ do
9          $C_w \leftarrow$ FindReplacementWord($\mathbf{x}$, $w_i$)
10         $\mathbf{X}' \leftarrow$ replace $w_i$ with $c_w \in C_w$
11     return $\mathbf{X}'$
```

As presented in section 3.1, the gradients can be used to identify volatile tokens, with the methods for computing these gradients outlined in the preceding section. This section explores how to employ these gradients to pinpoint and select tokens for substitution, thereby generating mutants for the evaluation of machine translation systems. This method is pivotal for detecting tokens that have a significant impact on the model, thus aiding in the enhancement of the machine translation systems testing approach.

After obtaining the gradient information of each token in the source sequence using the algorithms presented in 3.1, we sort the tokens in descending order based on the magnitude of their corresponding gradients. The top-k tokens are identified as the k most vulnerable tokens of the source sentence and will be replaced in the later stage. Algorithm 1 outlines the entire process of creating mutants for each source sentence using the GRI approach.

4.3.2 Word alignment-based strategy

As discussed in section 3.2, the attention weights in the transformer model provide the alignment between target and source tokens. Utilizing the alignment algorithms introduced in section 3.2, we focus on identifying vulnerable source tokens that correspond to target tokens with the lowest confidence scores. This methodology addresses the source tokens most likely to contribute to inaccuracies in the output.

Algorithm 2: WALI Test Generation

```
Input: a source sentence $\mathbf{x}$, and the translation system $\mathcal{F}(\cdot)$ and tokenizer $Tokenizer$
Output: a set of mutated sentences $\mathbf{X}'$
1  begin
2      $tokens \leftarrow$ Tokenizer($\mathbf{x}$)
3      $confidenceScore \leftarrow \mathcal{F}(tokens)$
4      $attnWeights \leftarrow \mathcal{F}(tokens)$
5      $C_{ordered} \leftarrow$ Sort($confidenceScore$)
6      $W_{ordered} \leftarrow$ Alignment($C_{ordered}$, $attnWeights$)
7      foreach $w_i \in W_{ordered}$ do
8          $C_w \leftarrow$ FindReplacementWord($\mathbf{x}$, $w_i$)
9          $\mathbf{X}' \leftarrow$ replace $w_i$ with $c_w \in C_w$
10     return $\mathbf{X}'$
```

To generate effective test sentences for translation systems, we first calculate the generation probability score (also known as the confidence score) $p(y_i | y_{1:i-1}, \mathbf{x})$ for each token y_i in the target translation. The generation probability captures the likelihood of the target token based on the source sentence $\mathbf{x}$ and the previously generated target sequence

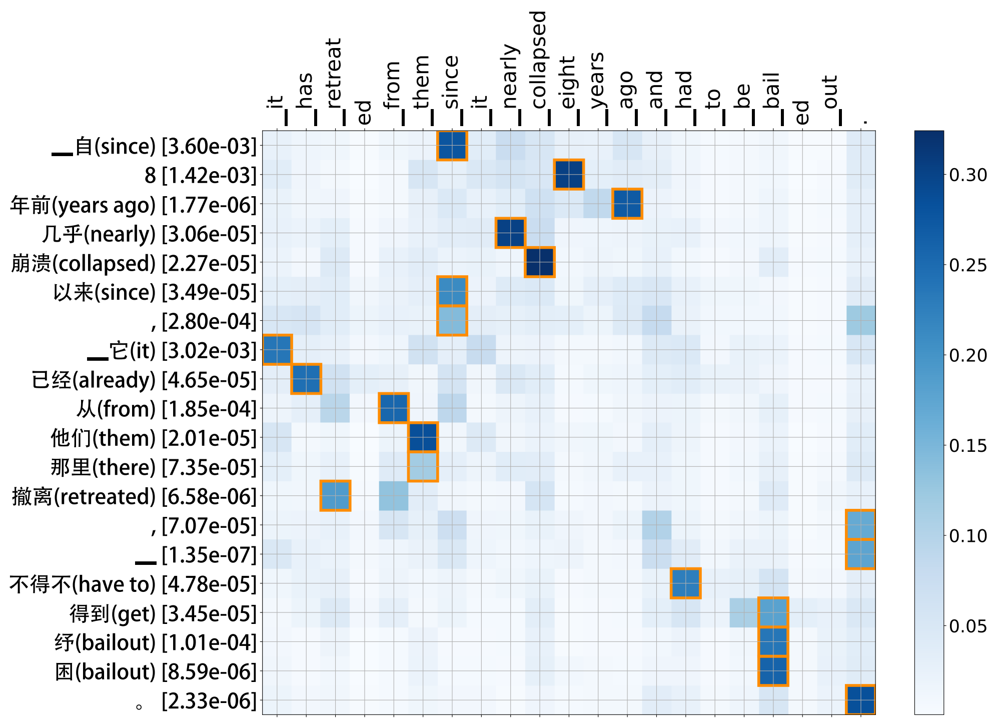


Figure 4.2: An example of alignment using attention weights. The horizontal axis represents the source tokens and the left vertical axis represents the aligned target tokens(gold alignment)[generation probability after *softmax* function]. The circled squares highlight the alignments obtained using the alignment matrix $\boldsymbol{A}$ presented in Section 3.2.

$y_{1:i-1}$. The higher the score, the greater the level of certainty exhibited by the model in making its prediction, and vice versa. Therefore, we assume that the lower the probability score of a target token is, the more likely its aligned source token can cause the system to generate a different translation once its aligned source token is replaced. For example, in Figure 4.2, the third target token "年前(years ago)" has the lowest score[1] and is identified as the vulnerable tokens of the target sentence.

We then map the target translation tokens to the source sentence tokens using the alignment method discussed in section 3.2. Figure 4.2 shows this alignment process through attention weights from the transformer model, illustrating the relationship between an English source sentence and its Chinese translation. This figure highlights the direct mapping between individual tokens in the English text and their counterparts in the Chinese translation. To provide an example, as shown in Figure 4.2, the third target token "年前(years

[1]Punctuations, like "。" and "__" are ignored.

ago)” is aligned to “ago” in the source sentence. Based on the lowest generation probability score and the alignment result, “ago” is thus identified as one of the vulnerable tokens of the source sentence.

Once the alignment between the target and source tokens is obtained, the source tokens are sorted in ascending order according to the confidence scores of their aligned target tokens. The top k tokens with the lowest confidence scores are identified as the k most vulnerable tokens in the source sentence and are considered candidates for replacement. Afterward, WALI approach is used to select and replace the vulnerable tokens in the input sentence. The entire process is described in Algorithm 2.

4.3.3 Word Replacement

In Section 4.3, we have introduced our two strategies: GRI and WALI for identifying the vulnerable tokens of a source sentence. In this part, we discuss how we replace the vulnerable tokens to generate new source sentences for testing.

As shown in Equation 4.1a, we need to maximumly preserve the meaning of the original source sentence during the word replacement process. To ensure a fair comparison with the existing baselines, we follow the replacement strategies proposed by baselines (24; 66; 67). Specifically, for comparison with TransRepair, we leverage a dictionary provided by the authors to identify contextually similar substitutions. This process is complemented by a validation of the part-of-speech (POS) tags (77) of the original and replacement words to ensure structural consistency. Meanwhile, for comparison with SIT, we utilize a neural language model (i.e., BERT) (13) to identify contextually analogous words for replacement and incorporate the POS tag (77) as a structural filter to verify sentence structure post-replacement. Likewise, for comparison with CAT, we employ BERT to generate a set of initial candidate words for each of the identified vulnerable tokens of the source sentence. Then, we adopt a neural network-based algorithm to evaluate the semantic similarity between the candidate and original words. The candidate words that have a low similarity are filtered out. The remaining candidates are used to replace the vulnerable tokens, and each replacement will result in a new source sentence $\mathbf{x}^{\text{new}}$ for testing the machine translation system. We adhere to the identical replacement strategies outlined in prior research to facilitate a fair evaluation, with and without the incorporation of GRI and WALI.

4.4 Translation Bug Detection

For translation bug detection, we adopt the same test oracle presented in the prior studies (24; 66; 67). Specifically, given a translation system $\mathcal{F}(\cdot)$ and an original source sentence $\mathbf{x}^{\text{orig}}$ and its newly generated one $\mathbf{x}^{\text{new}}$, the automatic bug detection oracle calculates the difference between the two translations, $\mathcal{F}(\mathbf{x}^{\text{new}})$ and $\mathcal{F}(\mathbf{x}^{\text{orig}})$. If the difference score is above a predefined threshold, a translation bug is reported. In our work, to ensure a fair comparison with baselines, we employ identical distance metrics and thresholds with the previous study (24; 66; 67) and avoid fine-tuning these settings only for our methods. We provide a brief definition of each metric below, where the first four metrics are used to evaluate TransRepair (66) and CAT (67) and the fifth metric is for SIT (24):

4.4.1 LCS-based metric

It measures the normalized length of the longest subsequence that is common between the original sentence and the mutated sentence (30). The LCS metric ranges from 0 to 1, where 1 indicates that the two sentences are identical and 0 indicates no overlap between the two sentences.

4.4.1.1 ED-based metric

The edit distance metric determines the minimum number of edit operations, such as insertion, deletion, substitution, and transposition, required to transform one string to another (59). It is widely used in applications that involve string comparison and spell-checking and can quantify the dissimilarity between two strings (20; 83).

4.4.1.2 TFIDF-based metric

The tf-idf (term frequency-inverse document frequency) metric is a statistical measure used to evaluate the similarity with the word frequency (60). In this study, we compute a weight w_{idf} for each word w using the same corpus used in CAT. The word vectors are then multiplied by their respective weights, and the cosine similarity between the resulting vectors of s_1 and s_2 is used to determine the similarity between the translations after the replacement process.

4.4.1.3 BLEU-based metric

BLEU score is a metric for evaluating the quality of machine-translated text against one or more human reference translations. It measures the overlap between the machine-generated translation and the reference translation(s) using n-gram analysis. It ranges from 0 to 1, where 1 indicates a perfect match between the candidate and reference translations. Those who seek further elaboration can refer to prior research studies (50; 66) for additional information.

4.4.1.4 Relation distance between dependency parse trees metric

In order to assess the structural consistency of the translations, it measures the dissimilarity between two sets of dependencies. This dissimilarity is determined by computing the total absolute difference in the counts of dependency relations parsed using Standford CoreNLP (10). We adhere to the Universal Dependencies annotation scheme, following the guidelines established in prior research (24).

For TransRepair (66) and CAT (67), the distance metrics, as outlined in 1)-4), quantify the similarity between the translation of the original input and the translation of the generated mutants. Consequently, a bug is flagged when the value of the distance metric falls below predefined thresholds. Following previous work (66; 67), the threshold is set to 0.963, 0.999, 0.906, 0.963 for LCS, TF-IDF, BLEU and ED respectively.

For SIT (24), the Dependency distance metric measures the distinction between the dependency lists of the translations of the original input and those of the generated mutants. The greater the metric, the larger the structural difference between the translations derived from the original. Following previous work (24), the top-3 sentences exhibiting the greatest divergence are reported.

4.5 Experimental Setup

4.5.1 Dataset

To ensure a fair comparison with baselines, following prior work (66; 67), we utilize the News Commentary (NC) testing dataset (75) to evaluate CAT (67) and TransRepair (66) and 200 English sentences crawled from CNN (Cable News Network)[2] to evaluate SIT. The

[2]https://edition.cnn.com

New Commentary dataset comprises 2,001 English-to-Chinese sentence pairs and the CNN dataset consists of articles from two categories: Politics and Business, where each contains 100 sentences. The statistics of all datasets are illustrated in Table 4.1.

Table 4.1: Statistics of the dataset for evaluation.

Corpus	**# of words per sentence**	**mean # of words per sentence**	**median # of words per sentence**	**# of words**	
				distinct	**total**
NC	$[2, 73]$	23.8	23	11,173	47,636
Politics	$[4, 32]$	19.2	19.5	1,918	933
Business	$[4, 33]$	19.5	19	1,949	944

4.5.2 Translation System

The same as prior work (23; 24; 25; 66; 67), we consider Transformer as the studied translation system[3]. It is one of the most widely used translation systems (34; 42) and has been widely studied in the research community (23; 24; 25; 66; 67). Specifically, we utilize the pre-trained Transformer translation model, *opus-mt-en-zh* (28) as our tested translation system. The model is trained on the *opus-2020-07-14* dataset (27). We fine-tune the model with the News Commentary training set (75), which consists of $2,513,475$ English-to-Chinese sentence pairs, with a learning rate of 2e$-$5 for 20 epochs.

4.5.3 Baseline approaches

We compare our approach with TransRepair (66), CAT (67), and SIT (24), which are by far the state-of-the-art approaches for machine translation system testing. All of these methods rely on the concept of metamorphic relations between the input sentence and its semantically similar variations. They operate under the assumption that input texts exhibiting semantic similarity should yield consistent translations.

4.5.4 Implementation Settings

In this experiment, we identify the top 10 vulnerable tokens using GRI and WALI for replacement. For the purpose of a fair comparison, we adopt the same approach as in the

[3]Note that we do not consider Bing or Google Translate for testing, as we cannot access the systems' architectures and parameters.

previous study (67) for replacing the identified vulnerable tokens and generating a maximum of five new sentences for each original source sentence. We conducted the experiment on Ubuntu 18.04 with an NVIDIA GTX 1080Ti GPU.

4.6 Evaluation

In this section, we evaluate our proposed approaches against the SOTA testing baseline (i.e., TransRepair, CAT and SIT). The effectiveness of our approach can be evaluated by considering two aspects: (1) the ability to identify a larger number of bugs using an equivalent number of test cases, and (2) the ability to detect an equal number of bugs with a reduced number of test cases, resulting in enhanced efficiency. Specifically, we aim to answer the following research questions (RQs):

RQ1: How effectively can GRI and WALI detect translation bugs compared to baselines?

Motivation. Extensive approaches have been proposed for testing machine translation systems, while few consider conducting the testing from a white-box perspective. Meanwhile, studies (19; 41; 70) have shown that using white-box methods can benefit many NLP tasks, such as the testing of the classification systems. Therefore, in this RQ, we would like to explore whether our two white-box-based approaches (i.e., GRI and WALI) can detect translation bugs with better performance than the baseline approaches (i.e., CAT, TransRepair and SIT). ***Approach.*** To answer this research question, we apply GRI and WALI as well as the baseline approaches on each source sentence in the NC, Politics, and Business dataset. With the generated mutants, we then examine whether the mutants can reveal translation bugs (cf. Section 4.4). We evaluate GRI and WALI using a combination of quantitative evaluation and human evaluation. For quantitative evaluation, we focus on 1) the number of translation bugs detected using the five distance metrics and 2) the number of test cases needed to detect a certain number of translation bugs. For human evaluation, following previous work (24; 66; 67), we randomly sample 100 test cases for CAT and TransRepair, and compute the Top-3 accuracy for SIT.

Result. **Both of our proposed approaches, GRI and WALI generally outperform the baseline approaches under identical experimental settings.** Our results comparing GRI and WALI with baseline approaches are presented in Figure 4.3. Figure 4.3 illustrates the cumulative count of bugs detected by different approaches, plotted against the number of test cases executed during the testing process[4]. It can be observed that our

[4]To ensure a fair comparison, we randomly shuffle all test cases before plotting the bug detection curve

approaches always outperform the baselines from two aspects: (1) under a certain testing budget (i.e., number of executed test cases, indicated by the vertical dotted line), both GRI and WALI can reveal a larger number of bugs, and (2) when given a predefined testing goal (i.e., number of detected bugs, indicated by the horizontal dotted line), our approaches always require less testing resource (i.e., a reduced number of test cases). For example, when running 6,000 test cases, GRI and WALI can detect 3662 and 3448 bugs, which is higher than the 3,018 detected by CAT (Figure 4.3(b)); meanwhile, GRI and WALI only need to execute 4,935 and 5,151 tests to detect 3,000 bugs, while CAT needs 5,961 tests. The results demonstrate that our approaches can detect translation bugs more effectively.

We also observe that our proposed approaches may not always improve the performance significantly. For example, in Figure 4.3(a), when running 4,000 test cases, TransRepair detects 1,826 bugs, relatively fewer than 1,918 and 1,875 reported by GRI and WALI. This may be due to the fact that TransRepair relies on a pre-established similarity dictionary[5] to provide suitable substitutions for specific tokens. Unlike the BERT model used in CAT and SIT (see Section 4.3.3), the dictionary only contains a limited number of tokens, which imposes constraints on the replacement of the tokens identified as vulnerable by GRI and WALI. In cases where a corresponding similarity pair does not exist in the dictionary, the replacement of these tokens becomes unfeasible. Upon reviewing the tokens that were substituted, it became evident that only 3,194 out of 18,387 tokens identified by GRI could be replaced within the corpus, leading to an insignificant improvement of our proposed approaches.

Meanwhile, we find that GRI and WALI always have a higher test success rate (i.e., the number of bugs detected divided by the number of total tests). The comparison results are shown in Table 4.2, with the best results highlighted in bold font. For example, GRI and WALI outperform SIT in terms of identifying more translation bugs across both the Politics and Business datasets. In addition, both GRI and WALI execute significantly fewer test cases, leading to an average increase in the success rate of 6.17% and 4.85%, respectively. This confirms that our approach can efficiently exploit the vulnerable spots of the translation systems by utilizing the gradients and word alignment information, thus leading to an increase in the success rate of the testing.

As observed in previous research on testing machine translation systems, it's acknowledged that automatic test oracles relying on distance metrics may yield results differing from those provided by human oracles. To address this, we carry out a manual inspection, adhering to the same evaluation process as the baseline approaches. The purpose of this

for each approach.

[5] We used the dictionary provided by the author of (66).

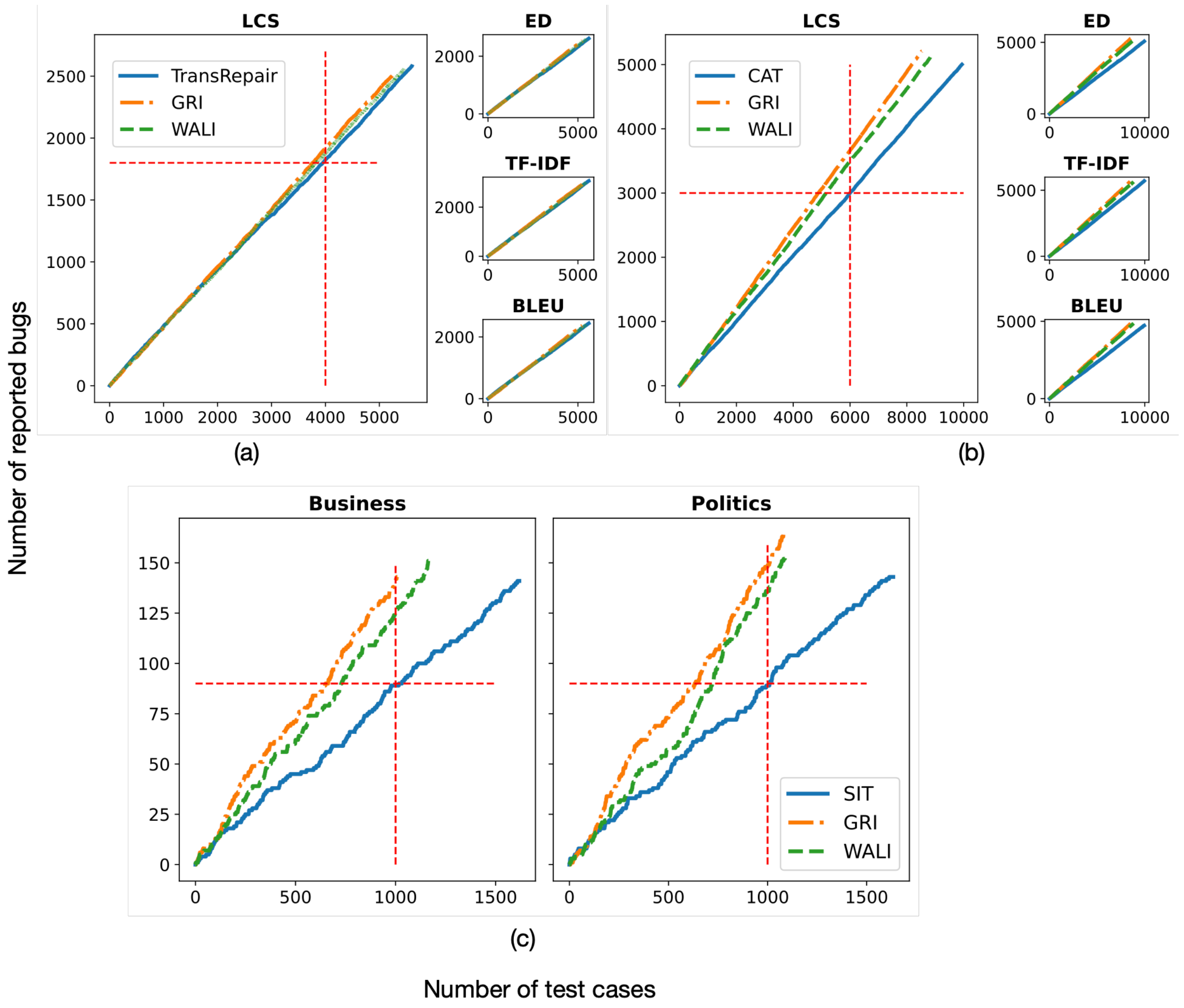


Figure 4.3: Cumulative counts of detected bugs for each of the approaches.

Table 4.2: Comparison of the test success rate with TransRepair, CAT, and SIT.

Metric	**TransRepair**	**GRI**	**WALI**
LCS	46.05% (2,581)	**47.80% (2,504)**	46.90% (2,561)
ED	46.65% (2,615)	**48.37% (2,534)**	47.50% (2,594)
TFIDF	54.75% (3,069)	**55.80% (2,923)**	55.10% (3,009)
BLEU	43.55% (2,441)	**45.10% (2,363)**	43.95% (2,400)
# of test cases	5,605	5,239	5,461

(a) Comparison with TransRepair

Metric	**CAT**	**GRI**	**WALI**
LCS	50.31% (5,002)	**61.07% (5,205)**	57.92% (5,114)
ED	51.00% (5,070)	**61.82% (5,269)**	58.53% (5,168)
TFIDF	57.39% (5,706)	**67.02% (5,712)**	63.90% (5,642)
BLEU	47.54% (4,726)	**56.96% (4,855)**	55.15% (4,870)
# of test cases	9,942	8,523	8,830

(b) Comparison with CAT

Dataset	**Metrics**	**SIT**	**GRI**	**WALI**
Politics	**Dependency**	8.74% (143)	**15.17% (164)**	14.11% (154)
	# of test cases	1,635	1,081	1,091
Business	**Dependency**	8.71% (141)	**14.16% (143)**	12.96% (151)
	# of test cases	1,619	1,010	1,165

(c) Comparison with SIT

Table 4.3: An average of manual evaluation results.

Approach	**TransRepair**	**CAT**	**GRI**	**WALI**	**SIT**	**GRI**	**WALI**
Precision	67.6	72.4	73.9	77.8	70.8	71.3	70.6
Recall	94.2	81.8	90.9	96.2	–	–	–
F1 Score	78.7	73.2	80.9	85.9	–	–	–

manual examination is to confirm that introducing our approach does not compromise the validity of the testing methods. Therefore, the bugs reported by our approaches are reliable and comparable to those reported by the baseline approaches.

Following previous work (66; 67), we uniformly sample 100 generated test cases for CAT and TransRepair and manually inspect the results. Table 4.3 demonstrates that the similarity metric achieves a precision of 0.73 and 0.77, recall of 0.9 and 0.96, and an F1 score of 0.8 and 0.85 for GRI and WALI, respectively. Notably, prior studies, CAT and TransRepair, reported similar precision, recall, and F1 scores, with values of 0.7, 0.95, and 0.8 for TransRepair, and 0.72, 0.9, and 0.8 for CAT in their experiment. The precision, recall, and F1 scores for GRI, WALI, and CAT in our experiment closely match the results of their original experiment evaluations. These findings suggest that the bug reports generated by the baseline approaches align closely with the outcomes of their original experiments especially when adopting the identical evaluation process. In addition, it is noteworthy that GRI and WALI exhibit relatively higher recall, indicating that most erroneous translations are correctly detected.

SIT employs Top-3 accuracy as its manual evaluation metric, and we also leverage identical metrics. In our experiment, using the transformer model, SIT obtained an accuracy of 0.7 on average for the Politics and Business dataset, whereas in the original experiment, they achieved a Top-3 accuracy of 0.73 and 0.78 for Google and Bing translators. The variance in accuracy is marginal and can be attributed to the difference in the translation systems used. In the case of GRI and WALI, the accuracy remains at 0.71 and 0.7, which closely aligns with that of SIT in our experiment and the accuracy reported in the original experiment. Consequently, our methods do not impede the performance of the testing approaches.

Overall, our manual inspection demonstrates that the validity of the bug reported by GRI and WALI is similar to those reported by the baseline approaches. Therefore, when using the same distance metrics and experimental settings, our results are reliable and can be compared to those of the baseline approaches. The distance metrics do not compromise the superiority of our approaches over the baseline methods.

Our proposed approaches, i.e., GRI and WALI, generally outperform the state-of-the-art baseline approaches in quantitative evaluations while preserving the accuracy of human evaluations. The results illustrate the promising future research opportunity of using more white-box approaches for testing machine translation systems efficiently and effectively.

RQ2: Can GRI and WALI complement the existing baseline approaches in terms of the detected translation bugs?

Motivation. In addition to the efficiency improvement offered by our approaches, it is also worth exploring whether our detected translation bugs are different from those of the existing approaches when employing the same maximum number of generated mutants, as utilized in previous studies.[6] Such analysis allows us to determine whether GRI and WALI can detect bugs that are not previously detected by the baseline approaches, thus complementing the existing baseline approaches towards a more comprehensive translation system testing. As the focus of this work is on effectively identifying the vulnerable tokens for a source sentence, therefore, in this RQ, we will investigate the overlap between the replaced tokens in the bugs detected by our proposed approaches and those by the baseline approaches. ***Approach.*** To address this research question, we examine the intersection of the tokens identified for replacement by different approaches among the detected translation bugs. The degree of overlap among these approaches signifies the distinctiveness of the bugs detected by different approaches[7].

Result. **Our approaches GRI and WALI can identify different tokens for replacement from that of baseline approaches and thus can detect unique bugs that were not detected previously.** Figure 4.4[8]shows the overlap of the replaced tokens in the detected translation bugs by different approaches. As shown in Figure 4.4, the majority of the tokens replaced by our approaches are not replaced by CAT. For example, under the LCS metric, CAT only shares an overlap of 39% (1,959/5,002) and 21% (1,353/5,002) with GRI and WALI, respectively. As a result, each of the distinct tokens leads to at least one bug-revealing test case. Similar results are observed with SIT on both Politics and Business datasets. With the Politics dataset, SIT shares an overlap of 25% (37/143) with GRI and WALI. Additionally, we notice that there is a limited overlap between GRI

[6]Note that the baseline approaches utilized a predefined maximum number of generated mutants to prevent an excessive number of test cases. This condition is maintained in our experiments to ensure a fair comparison.

[7]Note that each different replaced token can at least result in one different translation bug.

[8]Due to the space limitation, we only show the results using LCS metric for comparison with TransRepair and CAT for Figures 4.4 4.5 5.5. More results are shared in Appendix 6.

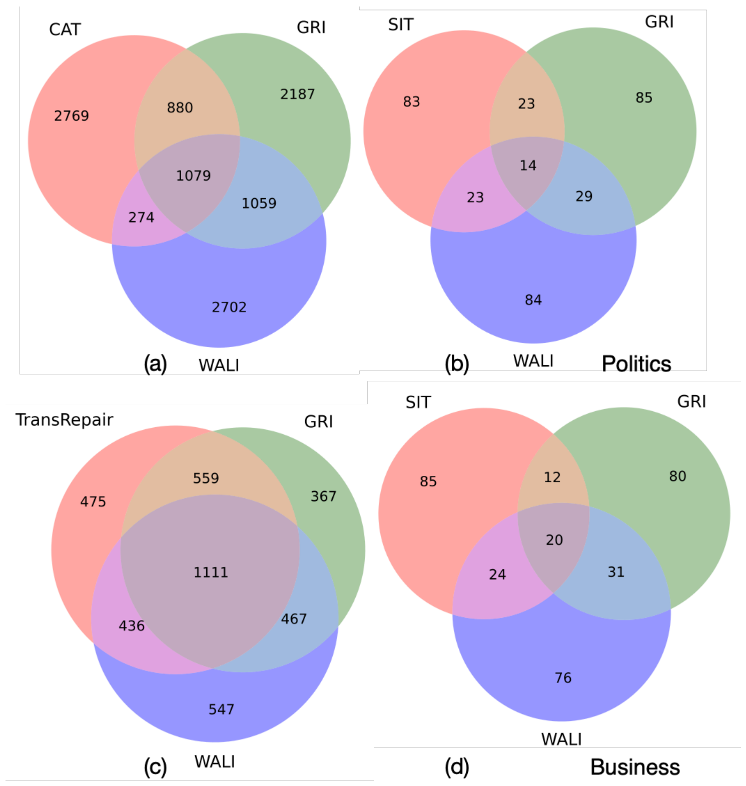


Figure 4.4: Overlap of the replaced tokens in translation bugs detected by GRI, WALI, and baseline approaches.

and WALI, meaning they can reveal bugs that the other approach didn't identify. This indicates that combining both approaches can complement the existing testing strategy by identifying a greater variety of susceptible tokens that were previously not replaced by the baseline approaches, which can further be used for generating new test cases with a word replacement strategy. Note that for TransRepair, we observe a more significant overlap. As it relies on a restricted dictionary to locate substitutes for specific tokens (cf. Section 4.6-RQ1), a considerable number of tokens identified by GRI and WALI do not have corresponding substitutes in the corpus. Consequently, the overlap among these three approaches is quite extensive.

To better demonstrate our complementary to the existing baseline approach, Table 4.4 provides examples of metamorphic test cases generated by SIT, CAT, GRI, and WALI where only GRI and WALI have successfully detected bugs. For instance, the first example pertains to a sentence in which CAT generates four mutants by replacing the word "eight"

Table 4.4: Examples of test cases generated by CAT, SIT, GRI and WALI where only GRI and WALI have detected bugs.

Original	**Baseline**	**GRI**	**WALI**
It has retreated from them since it nearly collapsed eight years ago and had to be bailed out. 自8年前几乎崩溃以来，它已经从他们那里撤离，不得不得到纾困。	It has retreated from them since it nearly collapsed [two] years ago and had to be bailed out. 自[两]年前几乎崩溃以来，它已经从他们那里撤离，不得不得到纾困。(It has retreated from them since it nearly collapsed [two] years ago and had to be bailed out.) - **CAT**	It has [moved] from them since it nearly collapsed eight years ago and had to be bailed out. 自8年前几乎崩溃以来，[政府就已经摆脱了这种困境]，不得不得到纾困。([The government has already got rid of the predicament] since it nearly collapsed eight years ago and had to be bailed out.)	It has retreated from them since it nearly collapsed eight years [back] and had to be bailed out. 自从八年前它几乎[倒塌后]，它已经从他们那里撤离，不得不得到纾困。(It has retreated from them since it nearly [fell apart] eight years ago and had to be bailed out.)
Meanwhile, whites still fill most of the seats at the most selective institutions, which spend the most on their students. 同时,白人仍然占据了最有选择的院校的大多数席位,这些院校在学生身上花费最多。	Meanwhile, whites still fill most of the seats at the most [important] institutions, which spend the most on their students. 同时,白人仍然占据着大多数[重要]院校的席位,这些院校花费在学生身上的[钱]最多。(Meanwhile, whites still fill most of the seats at the most [important] institutions, which spend the most [money] on their students.) - **SIT**	Meanwhile, whites still fill most of the seats at the most selective institutions, which [reflect] the most on their students. 同时,白人仍然占据了[大多数]院校的席位,这最能[反映白人学生的成绩]。(Meanwhile, whites still fill most of the seats at the [most of] institutions, which reflect the most [the whites' academic grades].)	Meanwhile, whites still fill most of the seats at the most selective institutions, which spend the most [as] their students. 同时,白人仍然占据了大多数[选择最多]的院校的席位,这些院校[作为学生]的花费最多。(Meanwhile, whites still fill most of the seats at the institutions [having most choices], [these institutions] spend the most [as students].)

with "three", "five", "ten", and "two". However, such replacements have little effect on the final translations, rendering CAT incapable of detecting translation bugs for this sentence. On the contrary, GRI identifies the word "retreated" as a replacement target, which is substituted with "moved", a word that has a similar semantic meaning. As the result shows, this perturbation has a relatively large impact on the translation, specifically affecting the segment "It has moved from them," which, due to the substitution of "moved" is translated to a Chinese phrase with a totally different meaning (i.e., "The government has already got rid of the predicament"). Similarly, WALI identifies "ago" as a vulnerable token and replaces it with "back", which does not alter the sentence's meaning. Interestingly, this substitution affects the translation of the unchanged word "collapsed", which was initially translated as "crumbled" and subsequently translated as "fell apart" due to the replacement of "ago" with "back".

Our proposed approaches GRI and WALI can detect unique translation errors that are not detected by the baseline approaches, and thus, can complement current translation system testing methodologies.

RQ3: What are the contributing factors that could raise the probability of generating bug-revealing test cases for translation systems?

Motivation. Our previous results indicate that the impact of token substitution on the system's performance varies depending on the specific token being replaced. To gain a deeper understanding of the factors that contribute to the generation of bug-revealing test cases for translation systems, we conduct further analysis of the results. The such analysis aims to identify the key factors that increase the likelihood of generating effective test cases for translation systems, with the goal of providing valuable insights for future research in this field. ***Approach.*** To answer this research question, we focus on the source sentences that can detect translation bugs and conduct the analysis from two aspects: 1) the distribution of parts-of-speech (POS) tags (77) in these source sentences and 2) the length of the source sentences for the three studied approaches.

Result. **To a greater extent, the substitution of nouns tends to expose more bugs for translation systems.** Figure 4.5 shows the distribution of the POS tags in the bug-revealing source sentences. We can see that translation bugs can be detected by replacing the tokens of various parts-of-speech (POS) tags in the source sentences. For example, GRI can detect translation bugs aroused by over 20 types of POS tags. Meanwhile, for all three approaches, it is clear that nouns are the predominant POS tag that contributes to bug detection. This observation can be attributed to the significance of lexical items in determining the meaning of the sentence. Nouns often play a crucial role in shaping the meaning of the sentence and are frequently associated with polysemous words, which further complicates their interpretation (80). Therefore, perturbing nouns can increase the likelihood of detecting translation bugs.

Furthermore, sentences with more tokens are relatively difficult to expose translation bugs for translation systems. As illustrated in Figure 5.5, under all distance metrics except TF-IDF, with the increase of the length of source sentences, the percentage of reported bugs decreases. We further calculate the Spearman correlation (47) between the proportion of reported bugs and the length of source sentences, and we find that overall, they have a negative correlation between each other (e.g., for BLEU-based metric, the correlation coefficients are -0.79, -1.0, and -1.0 for CAT, GRI, and WALI respectively). This is attributed to the fact that, as the length of the source sentence increases, substituting a single token has less impact on the overall semantic meaning of the sentence. When a sentence is short (e.g., length $\leqslant 5$), a single substitution can cause a

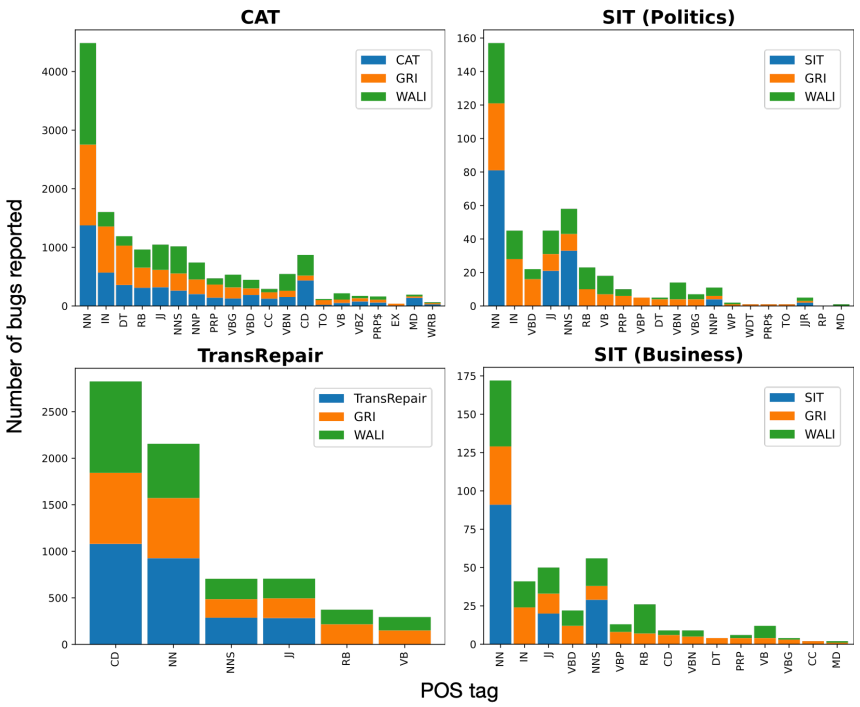


Figure 4.5: Number of reported bugs (y-axis) per mutant type (x-axis). This figure shows the distribution of bugs in terms of POS tags.

significant deviation in the sentence's meaning, as the sentence is short, it may not provide enough context for the translation system to comprehend. This finding suggests that putting more attention on the selection of short sentences for translation system testing, may expose more translation bugs (11; 22).

The replacement of nouns in a source sentence has a higher chance of exposing bugs in translation systems. Additionally, there exists a negative correlation between the length of source sentences and the percentage of reported bugs.

4.7 Threats to Validity

External Validity. In our study, we have utilized three datasets (i.e., News Commentary (75) and 200 English sentences extracted from CNN articles) and a translation system (i.e., Transformer (28)). The dataset and translation system are for English-to-Chinese

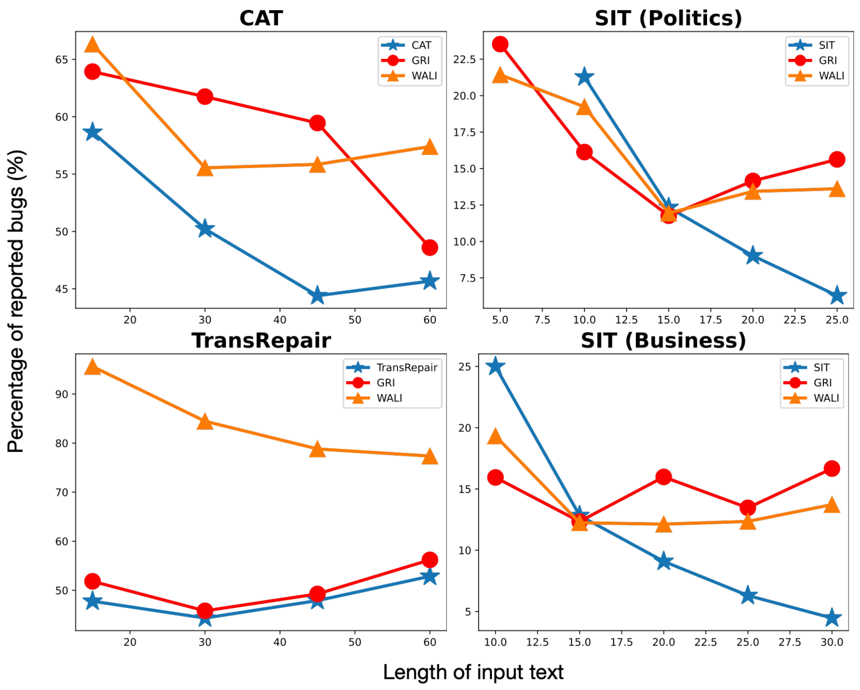


Figure 4.6: Percentage of reported bugs (y-axis) vs. length of the input sentence (x-axis). This figure shows the distribution of bugs in terms of sentence length.

translation, following the experimental setup of CAT, TransRepair and SIT. Therefore, it is necessary for future research to examine the effectiveness of our proposed approaches on other translation models and datasets covering various languages.

Internal Validity. In WALI, we use the encoder-decoder attention weights of the Transformer model to align the target and source tokens, which although effective, may not always be precise. For future research, we can employ a more sophisticated alignment approach to enhance the approach. In addition, the automatic test oracle used for bug detection relies on distance metrics to evaluate the quality of translations. However, the metrics used for evaluating translations may not accurately reflect human perceptions of translation quality, and human evaluation may be subject to individual biases. We followed the same human evaluation criteria as used in the baseline approaches and randomized the test inputs to reduce bias. To address this issue, future research could incorporate more manual evaluations to assess the effectiveness of GRI and WALI with a larger number of evaluators to minimize bias.

Chapter 5

Test Prioritization

In this chapter, we explore how to test QA software, applying two white-box methods to rearrange the test sentences we've generated and to give priority to those test cases that are closely related to specific tokens. The rationale is that the stronger the correlation between an inserted sentence and vulnerable tokens, the higher the likelihood that the sentence insertion will reveal a bug. This approach helps to streamline the testing process by focusing on the most impactful test cases.

5.1 Test Prioritization White Box Approaches

5.1.1 Overview

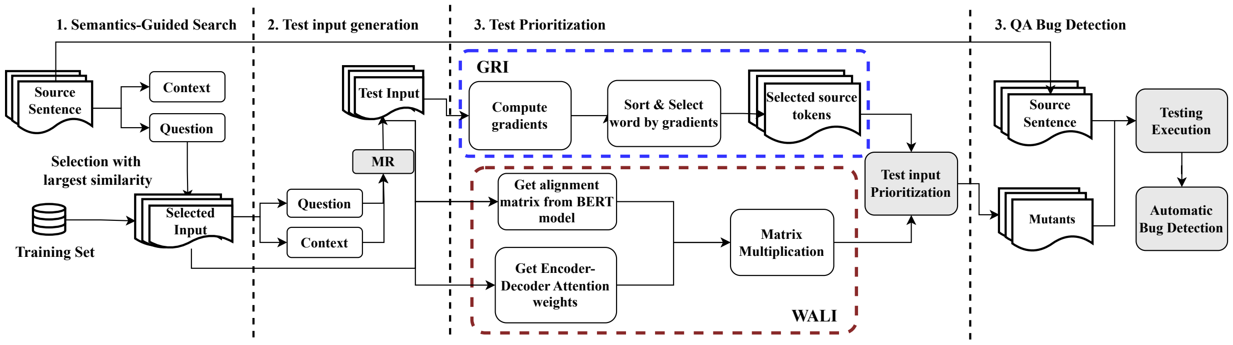


Figure 5.1: An overview of our approach.

Figure 5.1 outlines our proposed approaches for testing QA software. Initially, for each source sentence, we identify related sentences from the training dataset using methods from baseline research (62). Next, we create a new input by adding a redundant sentence to either the question or the context, following a consistent strategy. Within our white-box framework, we employ two specific strategies to order the test cases: (1) identifying vulnerable tokens based on gradient information (GRI) and (2) identifying them through word alignment (WALI). Subsequently, we organize the new test inputs by ranking the test cases according to the relevance of the selected sentences and the targeted tokens, specifically ordering them by the maximum gradient or attention values associated with the selected tokens within the sentences. This reordering of test cases ensures they are executed in a prioritized manner. The QA software then processes these reordered sentences (test cases or mutants), generating corresponding responses. If the variance between a new response and the original response surpasses a predetermined threshold, it indicates a potential bug.

5.1.2 GRI

Using the algorithm presented in Section 3.1, we are able to compute the gradients of tokens in the question. This section explores how to employ these gradients to reorder the test cases that are more likely to induce a bug in QA software.

By applying the semantics-guided search strategy outlined in the baseline work, we determine gradients for tokens both in the original question and in the selected sentence. We then prioritize test cases based on the highest gradient value among tokens within the inserted sentence, effectively gauging the inserted sentence's relevance to the original question. The underlying hypothesis is that a sentence more relevant to the original question could potentially confuse the QA software more, even if the semantic essence of the question remains unchanged.

We pinpoint the top-k tokens by their gradient magnitudes, cataloging these values. The test cases are then sequenced according to the highest gradient value found in the inserted sentences, with a value assigned as 0 if the inserted sentence lacks any of the selected tokens. Algorithm 3 outlines the entire process of prioritizing test cases using the GRI approach.

Algorithm 3: GRI Prioritization

Input: a source sentence $\mathbf{x}$, and the QA system $\mathcal{F}(\cdot)$ and tokenizer *Tokenizer*

Output: a set of reordered sentences $\mathbf{X}'$

```
1  begin
2      $tokens \leftarrow$ Tokenizer($\mathbf{x}$)
3      $embeddings \leftarrow \mathcal{F}(tokens)$
4      $output \leftarrow \mathcal{F}(embeddings)$
5      $G \leftarrow$ GetGradient($output$)
6      $G_{sorted} \leftarrow$ Sort($G$)
7      $T_{ordered} \leftarrow$ SortTokensbyGrad($tokens$)
8      $S_{selected} \leftarrow$ Semantics-GuidedSearch($Training\ Set$)
9      $g_{max} \leftarrow 0$
10     foreach $w_i, g_i \in T_{ordered}$ do
11         if $w_i \in S_{selected}$ then
12             $g_{max} \leftarrow \max\{g_{max}, g_i\}$
13     $\mathbf{X}'_{ordered} \leftarrow$ SortbyMaxGradient($\mathbf{X}'$)
14     return $\mathbf{X}'_{ordered}$
```

5.1.3 Word alignment-based strategy

As explained in section 3.2, attention weights in the transformer model illustrate the connections between target and source tokens. While translation tasks typically exhibit a straightforward one-to-one token mapping, only a few exceptions feature a one-to-many relationship. In contrast, question-answering scenarios present a more intricate mapping due to the disproportionate lengths of source inputs and outputs. Consequently, the emphasis shifts towards analyzing the alignment weights that link target and source tokens.

To enhance the identification of bug-revealing test sentences, the hypothesis is that a stronger correlation between the inserted sentence and the answer increases the likelihood of influencing the answer. Thus, the goal is to assess the correlation between the inserted sentence and the answer. While the transformer model provides attention weights showing input-output token mappings, it does not directly offer the correlation with inserted sentences. The proposed method involves using these attention weights, as detailed in section 3.2, to derive the attention matrix for the input sentences (question and context) against the output answer. An alignment matrix is then calculated for the original input and the inserted sentence using a pre-trained BERT model (12). The BERT model is also a transformer-based model pre-trained on a large corpus of multilingual data that can provide the alignment between the input and output sentences. By matrix multiplication of these two matrices, a new matrix emerges, illustrating the relationship between the selected tokens and the answer, aiding in prioritizing the test cases effectively.

To mathematically formalize the algorithm described for prioritizing bug-revealing test sentences, consider the following equations:

Let $A \in \mathbb{R}^{m\times n}$ represent the attention matrix from the transformer model, where m is the number of input tokens and n is the number of output tokens (answer). An element a_{ij} in A indicates the attention weight from the i-th input token to the j-th output token.

$$A = [a_{ij}] \quad \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n$$

Let $B \in \mathbb{R}^{m\times o}$ denote the alignment matrix obtained using the BERT model (12), where o is the number of tokens in the inserted sentence. An element b_{ik} in B represents the alignment score between the i-th input token and the k-th token in the inserted sentence.

$$B = [b_{ik}] \quad \text{for } i = 1, \ldots, m \text{ and } k = 1, \ldots, o$$

The correlation matrix $C \in \mathbb{R}^{o\times n}$ is computed by multiplying matrices A and B, where an element c_{kj} in C corresponds to the correlation between the k-th token in the inserted sentence and the j-th token in the output answer.

$$C = B^\top \times A$$

$$C = [c_{kj}] \quad \text{for } k = 1, \ldots, o \text{ and } j = 1, \ldots, n$$

This product $C = B^\top \times A$ provides a detailed view of how tokens in the inserted sentence is related to tokens in the answer. Figure 5.2 illustrates the matrix multiplication between A and $B^\top$.

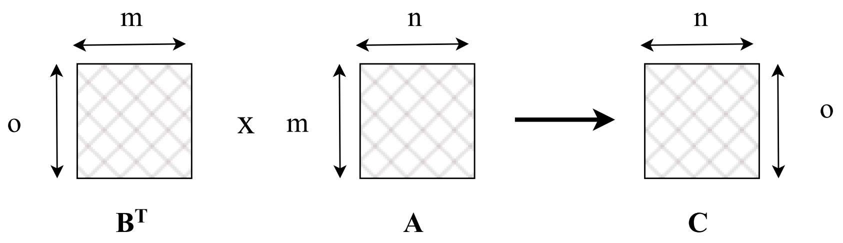


Figure 5.2: Matrix multiplication where A represent the attention matrix, m is the number of input tokens and n is the number of output tokens, B denote the alignment matrix obtained using the BERT model (12), o is the number of tokens in the inserted sentence.

The magnitude of the correlation matrix is utilized to sequence the test sentences, prioritizing those that have a higher likelihood of revealing bugs during testing. Test sentences are ordered in descending order according to the magnitude of the correlation matrix, ensuring that those with the greatest potential impact are tested first.

5.1.4 Test Input Generation

For the test input generation, we follow the methodology from a previous study. The process, known as QAQA, involves searching the training data for a sentence that closely resembles the original question in terms of semantic content. To identify the core semantics of the question, it is first condensed using the compression model SLAHAN (32). Subsequently, both the truncated question and the training data sentences are transformed into embeddings using Sentence-BERT (SBERT) (57). By computing the cosine similarity between these embeddings, the sentence from the training set with the highest similarity score to the truncated question is chosen for mutation. This selected sentence is then used to generate a new test input.

To generate mutants, QAQA leverages five metamorphic relations: Equivalent Question (EQ), Equivalent Context (EC), Test Integration (TI), Equivalent Question Context (EQC), and Equivalent Test Integration (ETI). These relations guide how the selected sentence is inserted into the question and/or context to create a new test input. We employ this same methodology to produce mutants in our testing strategy. For a more detailed explanation of these metamorphic relations and their application in the mutant generation, refer to the study detailed in (62).

5.1.5 Bug Detection

The bug detection approach aligns with the baseline method to ensure comparability, focusing on the consistency of metamorphic relations. Answers provided by the QA software are compared; if the discrepancy between the answer to the generated input and the original exceeds a predefined threshold, it signals a potential bug. Considering the linguistic variability where identical semantics can be presented differently (e.g., "UK" vs. "United Kingdom"), the approach also includes measuring the embeddings' similarity to ensure semantic consistency between answers. Phrase-Bert (PBERT) (71), an advanced model for phrase-level representation, is employed to convert the answers into embedding vectors, followed by computing cosine similarity to detect semantic variations. When the similarity score falls below the established threshold (0.76, consistent with the baseline methodology), a bug is reported. More details can be found in prior research on QA software testing (62).

5.1.6 Test Prioritization Evaluation Metric

To assess the effectiveness of test prioritization, it is essential to have an evaluation metric. A common metric is a graph depicting the percentage of detected faults versus the fraction of the test suite used. This curve illustrates the cumulative percentage of faults detected as testing progresses. The shaded area under this curve signifies the weighted average of the percentage of faults detected throughout the lifespan of the test suite. This area is referred to as the prioritized test suite's average percentage faults detected measure.

In our experiments, we will evaluate the area under the curve (AUC) of the percentage of detected faults during test execution. We will compare this with the ideal scenario, where all bugs are scheduled in priority, to assess the performance of our white-box approaches. (Figure 5.3 provide an example of the curves) This comparison will allow us to quantify how effectively our approaches can identify and prioritize critical test cases that are likely to reveal faults, thereby optimizing the testing process and ensuring more efficient resource utilization.

5.2 Experimental Setup

5.2.1 Dataset

The study employs two well-regarded QA datasets previously utilized in existing research. The baseline QAQA used these datasets as well for evaluation therefore, we adopted the same datasets to maintain consistency in comparison. The training datasets are used to build the UnifiedQA model, while the test sets are used as test cases for the proposed approaches. The details of datasets are listed below.

BoolQ. BoolQ (9) is a boolean QA dataset, where the answers to the questions are either "yes" or "no". This dataset comprises questions sourced from Google search engine queries, with corresponding contexts extracted from Wikipedia articles.

SQuAD2. SQuAD2 (56) is an extractive QA dataset in which the answers is a segment of text. The questions and answers are collected from Wikipedia articles. SQuAD2 includes unanswerable questions, meaning that some questions are designed with “<No Answer>” indicating that the text does not contain information to answer the query.

NarrativeQA. NarrativeQA (38) is an abstractive QA dataset, in which the answer is not a span of the context. The dataset is collected from books and movie scripts to test the comprehension of the model in the context.

5.2.2 QA Software

We used the pre-trained T5-large-based UnifiedQA (2) model as the testing subject, which aligns with the baseline approach, QAQA. It is one of the state-of-the-art models that is proven to perform well across diverse datasets. In addition, it was widely studied in prior researches (7; 62) as well as an experimental subject, employing this model facilitates direct comparisons with prior results.

5.2.3 Implementation Settings

In this experiment, we improve the testing efficiency using GRI and WALI for prioritization. For the purpose of a fair comparison, we adopt the same approach as in the previous study (62) for generating new test sentences using metamorphic relations for each original source sentence. We conducted the experiment on Ubuntu 18.04 with an NVIDIA GTX 1080Ti GPU.

5.3 Evaluation

In this section, we evaluate our proposed approaches against the SOTA testing baseline (i.e., QAQA). The effectiveness of our approach can be evaluated by the ability to identify the bug-revealing test sentences and positioning these potential bug-triggering sentences at the beginning of the testing sequence, thus improving the fault detection rates over the testing phase. Specifically, we aim to answer the following research questions (RQs):

RQ1: How effective are the white-box approaches in prioritizing the bug-revealing test cases?

Motivation. Extensive approaches have been proposed for testing question answering systems, while few consider conducting the testing from a white-box perspective. Meanwhile, studies (19; 41; 70) have shown that using white-box methods can benefit many NLP tasks. Therefore, in this RQ, we would like to explore whether our two white-box-based approaches (i.e., GRI and WALI) can improve the AUC (Area Under Curve) of the cumulative count of bug reported during the testing which indicates the rate of fault detection of the prioritization techniques. ***Approach.*** To answer this research question, we apply GRI and WALI as well as the baseline approaches on each source sentence in the BoolQ, NarrativeQ, and SQuAD2 datasets. With the generated mutants, we then examine whether the mutants can reveal a bug (cf. Section 4.4). For quantitative evaluation, we focus on the number of bugs detected using the similarity metrics over the number of test cases executed and. To ensure the validity of the results, following previous work (62), we randomly sample 100 test cases for manual evaluation.

Result. **Our proposed approaches, GRI can outperform the baseline approaches under identical experimental settings.** The results comparing GRI and WALI with baseline methods are shown in Figure 5.3. This figure demonstrates the cumulative bugs detected by different strategies against the number of test cases executed in various sequences across three datasets. It's evident that GRI surpasses the baseline in two key

Table 5.1: AUC for different appraoches

	BoolQ	**NarrativeQA**	**SQuAD2**
QAQA	51.37	51.04	50.19
GRI	**58.24**	**55.54**	**52.68**
Ideal	89.72	84.59	80.86

areas: (1) it uncovers more bugs within a specific testing budget, and (2) it achieves set testing goals with fewer resources post-prioritization. For instance, GRI detects 315 bugs in BoolQ with 1,000 test cases, surpassing the 221 bugs identified by QAQA, as illustrated in Figure 5.3(a). Moreover, to find 200 bugs, GRI requires 522 tests, whereas QAQA needs 890 tests. This figure illustrates the effectiveness of GRI in identifying test cases with a higher likelihood of revealing bugs, thereby enhancing the testing process's efficiency through prioritization.

Additionally, GRI consistently shows a higher percentage of AUC when set against the QAQA baseline. The AUC (Area Under the Curve) in the Figure 5.3 illustrates the rate at which bugs are discovered as testing progresses. A higher AUC indicates that a larger number of bugs are identified early in the testing cycle, suggesting that the testing process is effective in quickly uncovering faults in the software. Table 5.1 displays these comparisons, with the best results in bold. Specifically, GRI outperforms the baseline approach in identifying more bugs within the same testing budget across BoolQ, NarrativeQA, and SQuAD2 datasets, showing an increase in AUC percentage of 6.87%, 4.50%, and 2.49%, respectively. This emphasizes GRI's effectiveness in exploiting QA systems' vulnerabilities by using gradient information, thus enhancing testing success rates.

From Figure 5.4, it is evident that the performance of WALI is not particularly impressive. The curve and the Area Under the Curve (AUC) of WALI are quite similar to those of the original QAQA curve. For the SQuAD2 dataset, the percentage of AUC is only about 5% higher than the original baseline approach. However, since this improvement is not consistent across all three datasets and is relatively modest, it is not sufficient to conclude that WALI significantly enhances detection efficiency. Thus, the prioritization based on the WALI approach does not offer a significant improvement over the established QAQA method. However, our proposed approach offers the potential to explore the relationship between the input and output using the alignment information of the sequence-to-sequence models.

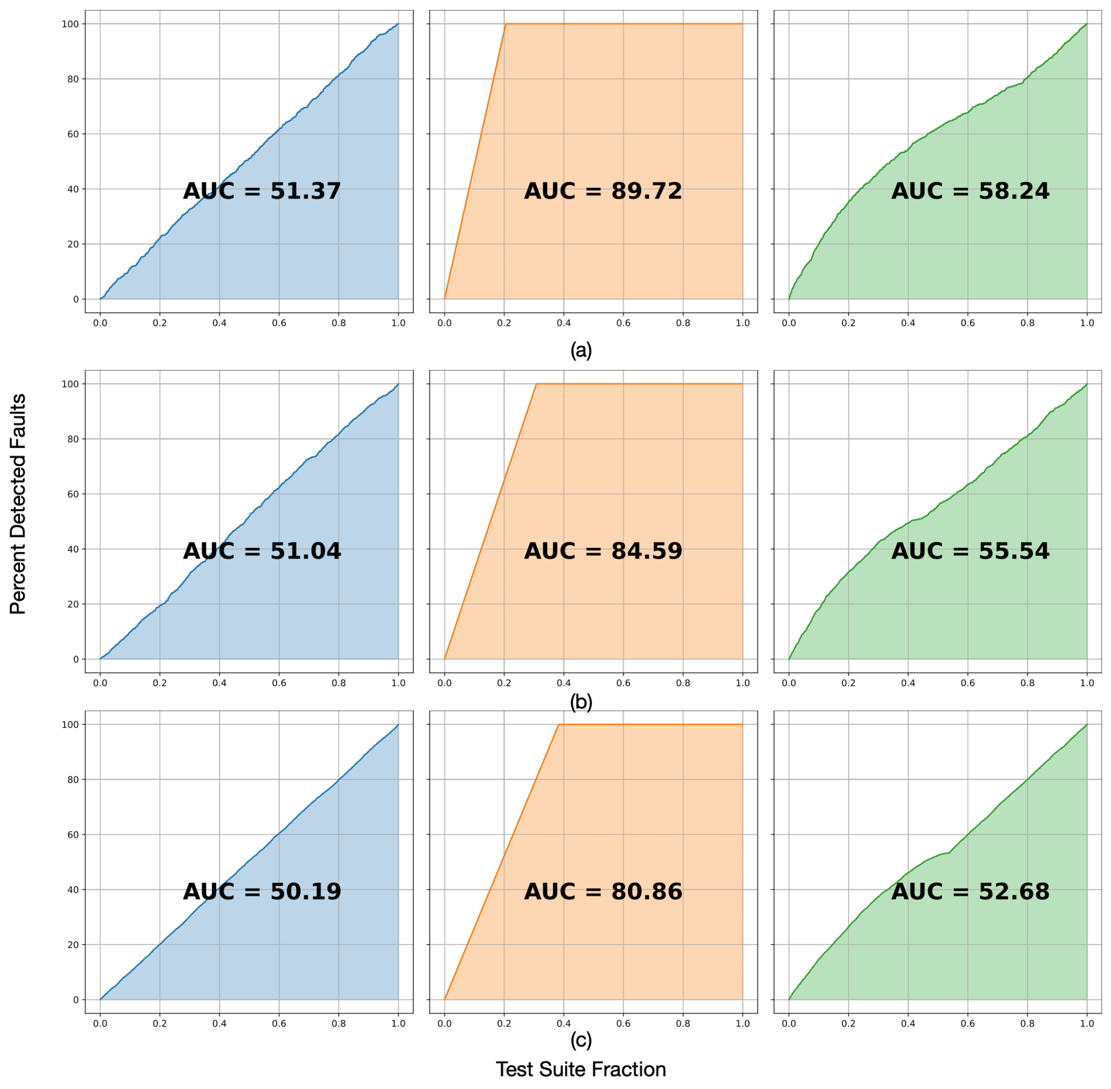


Figure 5.3: Percentage of detected bugs(y-axis) vs. fraction of test suite executed(x-axis) using the initial test order, the ideal scenario, and GRI-based prioritization for (a) BoolQ, (b) NarrativeQA and (c) SQuAD2 dataset respectively.

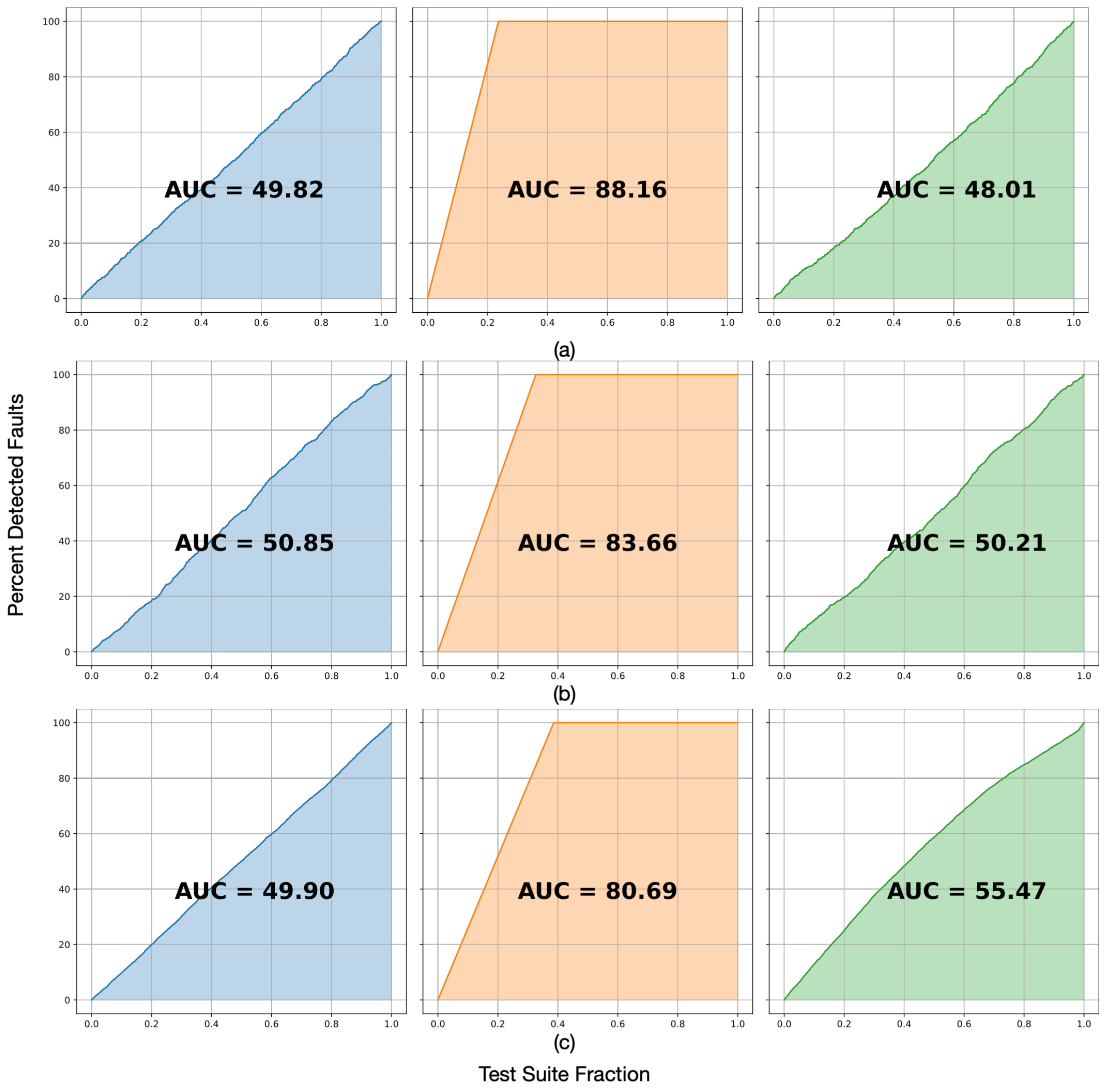


Figure 5.4: Percentage of detected bugs(y-axis) vs. fraction of test suite executed(x-axis) using the initial test order, the ideal scenario, and WALI-based prioritization for (a) BoolQ, (b) NarrativeQA and (c) SQuAD2 dataset respectively.

Table 5.2: Manual evaluation results.

Dataset	NarrativeQA	SQuAD2
Precision	79.49%	86.67%
Recall	96.87%	83.87%
F1 Score	87.32%	85.24%

As the test oracle depends on the similarity score between the original and new answers, ensuring the validity of the results necessitates a manual evaluation. Given that the original test generation methodology of QAQA remains unchanged, the precision of the test oracle should remain consistent. To validate the results, we employed the same evaluation approach, sampling 100 test cases from the NarrativeQA and SQuAD2 datasets. These test cases and their results were manually labeled to assess accuracy. The precision, recall, and F1-score of this evaluation are presented in Table 5.2.

From Table 5.2, the precision, recall, and F1 score align closely with those reported in the earlier study (62), demonstrating the test oracle's reliability. Specifically, in the NarrativeQA dataset, there were 8 false positives and 1 false negative out of 100 random samples. For the SQuAD2 dataset, the evaluation revealed 4 false positives and 5 false negatives among the 100 samples. Additionally, the QAQA report noted 5 false positives and 27 false negatives from 300 samples of generated question and answer pairs. By comparing these outcomes with the false positives and false negatives documented in the paper (62), it is clear that our testing strategy maintains its accuracy. This consistent performance results from our adherence to the original test generation process and the use of the same threshold values, ensuring robust and reliable testing across different datasets. Thus, the validity of the results is maintained.

Our proposed approach, which prioritizes tests using GRI, outperforms the state-of-the-art baseline approaches, QAQA, in quantitative evaluations, while maintaining the accuracy validated through human evaluations. Conversely, WALI yields results comparable to the baseline. The results highlight the potential for future research to explore the use of white-box approaches in prioritizing test cases for QA testing, aimed at enhancing testing efficiency.

RQ2: What are the characteristics of the test cases that are not identified by GRI and WALI?

Motivation. While GRI and WALI have effectively identified several test cases more likely to uncover bugs, there are still instances where test cases evade detection. This research question (RQ) aims to investigate the characteristics of the test cases that GRI and WALI failed to identify and to explore the factors that may increase the likelihood of generating bug-revealing test cases for QA systems. ***Approach.*** To tackle this research question, our investigation centers on test cases that, despite exhibiting low gradient or attention mapping values, nevertheless reported a bug. We intend to conduct an in-depth analysis of the length of the source input to determine its influence on the effectiveness of bug detection. This examination will help ascertain any potential correlations between the length of input and the probability of a test case revealing a bug, thereby aiming to refine and enhance the overall testing strategy for QA systems.

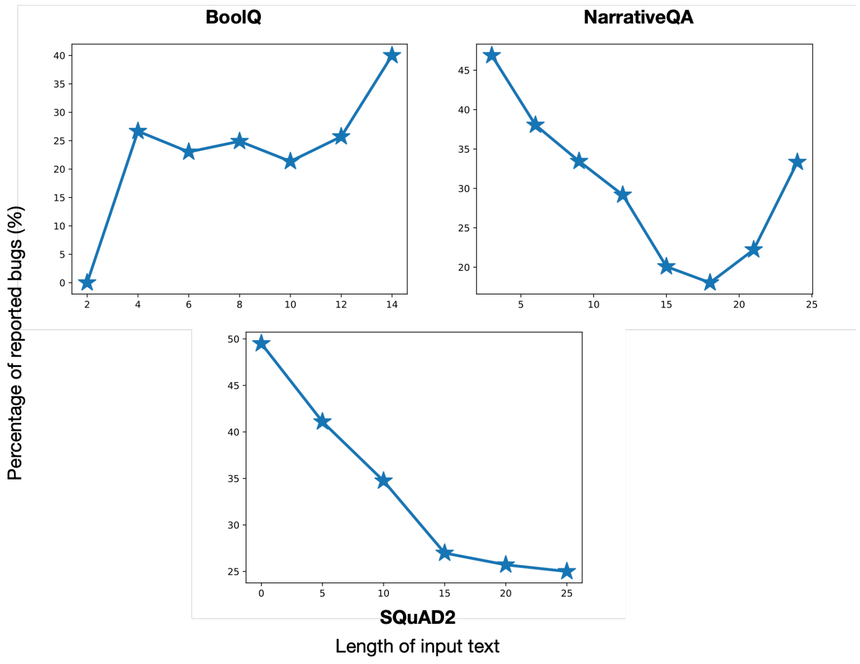
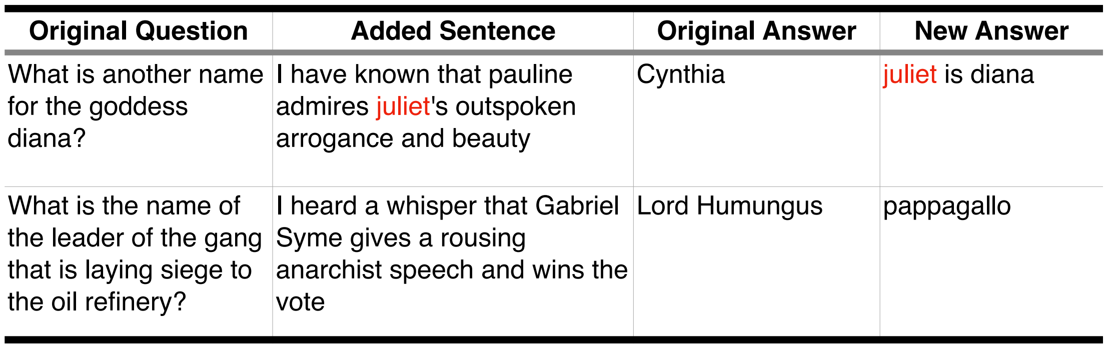


Figure 5.5: Percentage of reported bugs (y-axis) vs. length of the input sentence (x-axis). This figure shows the distribution of bugs in terms of sentence length.

First, we examine the correlation between the length of the question and the likelihood of reporting a bug after a mutation is inserted. Figure 5.5 illustrates a negative relationship between the percentage of bugs reported and the length of the question in the input across all datasets except BoolQ. This trend arises because shorter questions are more significantly affected by the addition of redundant sentences to the input question and context, logically leading to a negative relationship between the likelihood of bug reporting and question length. For the dataset NarrativeQA, the Spearman correlation coefficient (47) ρ is -0.69 with a p-value of 0.057, which indicates a strong negative correlation between the variables. This finding implies that shorter questions in NarrativeQA are particularly susceptible to generating detectable bugs when altered with test case mutations.

The exception observed with the BoolQ dataset can be attributed to the binary nature of its answers and the consistent structure of boolean questions. Typically, boolean questions demand a straightforward "yes" or "no" response, which may not vary significantly with changes in question length. This consistency in the question format and answer type reduces the variability in how questions are processed and understood, potentially diminishing the impact of additional or redundant information introduced through mutations. Consequently, the correlation between question length and bug occurrence that is evident in other datasets may not manifest as strongly in BoolQ.



Original Question	Added Sentence	Original Answer	New Answer
What is another name for the goddess diana?	I have known that pauline admires juliet's outspoken arrogance and beauty	Cynthia	juliet is diana
What is the name of the leader of the gang that is laying siege to the oil refinery?	I heard a whisper that Gabriel Syme gives a rousing anarchist speech and wins the vote	Lord Humungus	pappagallo

Figure 5.6: Example of test cases that reported a bug but not identified by GRI and WALI.

In Table 5.6, two test cases are presented where GRI and WALI did not effectively prioritize as bug-revealing yet they reported bugs. Observing these examples, it is evident that the added sentences do not initially correlate with the question. However, when incorporated into the question using the QAQA methodology—prefaced with phrases like “I have known” or “I have heard”—they are likely to create an implication that influences the answer.

For the first example, GRI identified “*Diana*” and “*goodness*” as significant tokens within the question. Upon analyzing the keywords in the added sentence, it failed to find any direct correlation with these identified tokens. WALI calculated a score of 0.45 for the magnitude of the correlation matrix of the added sentence and the answer, suggesting a moderate connection. Despite the initial absence of a direct link, the way the sentence was integrated and the phrase, “*juliet's outspoken arrogance and beauty*” implies a semantic connection. This integration subtly misled the model, causing it to provide an incorrect

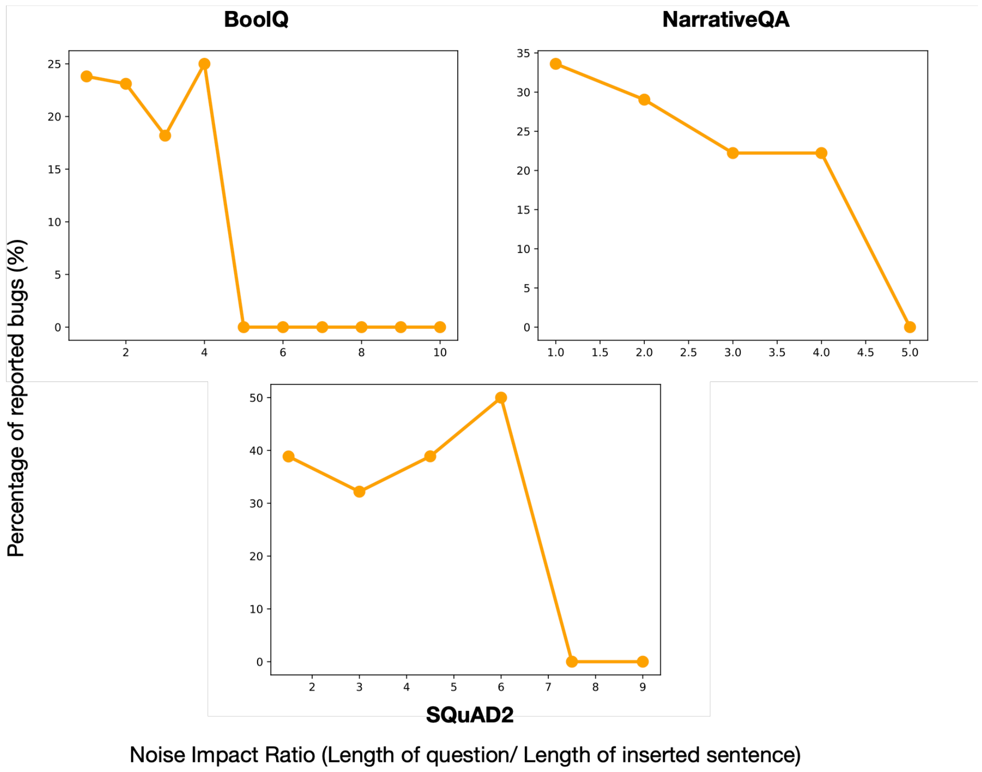


Figure 5.7: Percentage of reported bugs (y-axis) vs. length of question/ length of inserted sentence (x-axis). This figure shows the distribution of bugs in terms of ratio between texts.

response. This outcome underscores the complexity of detecting and interpreting the semantic implications of added sentences in QA systems, highlighting areas for improvement in the tools used to prioritize potential bug-revealing test cases.

In the second example, there is a potential implication established even though there is no direct correlation or similarity between the added sentence and the question. However, while the answer changes, it is not influenced by the added sentence but rather presents a completely different response. This scenario is another typical example of test cases that GRI and WALI fail to identify. The addition of a new sentence to the input text introduces noise and can impact the model's output. To explore this effect further, we analyze whether the ratio of the length of the question to the length of the added sentence correlates with the number of bugs reported. We hypothesize that the longer the added sentence relative to the shorter original question, the greater the impact of noise on the model.

From Figure 5.7, we observe a negative correlation between the ratio, which we will refer to as the “Noise Impact Ratio” (Length of question/ Length of inserted sentence), and the number of reported bugs. This metric helps us quantify how the proportion of added content to original content influences the likelihood of generating bugs, underscoring the sensitivity of the QA model to variations in input length. This observation emphasizes the importance of the length of the inserted sentence in relation to the original input text. A lengthy inserted sentence can introduce significant noise, causing the model to produce a different answer even if there is no semantic correlation between the texts. In such cases, the model struggles to accurately comprehend the question and identify the crucial segments, leading to incorrect responses.

There is a negative correlation between the length of source sentences and the percentage of reported bugs. Additionally, a negative correlation is observed between the ratio of the length of the question to the length of the inserted sentence and the number of reported bugs. Although there is no direct correlation or similarity between the question and the added sentence, the aggregation of the sentence into the question can create implications that might mislead the model into generating an incorrect answer.

5.4 Threats to Validity

External Validity. In our study, we have utilized three datasets (i.e., BoolQ (9), NarrativeQA (38) and SQuAD2 (56)) and a QA software (i.e., UnifiedQA (2)). We followed strictly the experimental setup of QAQA in the research to ensure a fair comparison. Therefore, it is necessary for future research to examine the effectiveness of our proposed approaches on other QA models and datasets.

Internal Validity. In WALI, we use the encoder-decoder attention weights of the UnifiedQA model to align the target and source tokens, which although effective, may not always be precise. For future research, we can employ a more sophisticated alignment approach to enhance the approach. In addition, the automatic test oracle used for bug detection relies on cosine similarity of embedding vectors to evaluate the quality of answers. However, the metrics used for evaluating output answers may not accurately reflect human perceptions of quality, and human evaluation may be subject to individual biases. We followed the same human evaluation criteria as used in the baseline approaches and randomized the test inputs to reduce bias. To address this issue, future research could incorporate more manual evaluations to assess the effectiveness of GRI and WALI with a larger number of evaluators to minimize bias.

Chapter 6

Conclusion

In this thesis, we present two white-box approaches, GRI and WALI, which can be applied to the existing testing technique on the sequence-to-sequence model. They are proven to improve current machine translation testing techniques. By comparing the bug detection outcomes of our approaches with those of the baseline approach, we demonstrate that our approaches can enhance the efficacy of the existing testing methods on translation systems. Specifically, our approaches are able to detect a greater number of bugs with fewer test cases and can identify previously unnoticed bugs, complementing the existing testing approaches.

Next, we explore the possibility of prioritizing test cases in the QA software testing strategy. We slightly modify our approaches and apply them to prioritize the sentences generated by QAQA, aiming to enhance the efficiency of test execution. The results indicate that GRI can prioritize bug-revealing test cases more effectively, while WALI's performance is comparable to the baseline approach. Specifically, these white-box approaches are capable of identifying test cases that are more likely to reveal bugs, prioritizing them for early execution. This prioritization improves execution efficiency and helps in saving computational costs.

Our research illuminates the advances in using white-box approaches to improving testing techniques on sequence-to-sequence models. Furthermore, our approaches demonstrate the potential of using white-box-based information in the quality assurance of AI software.

References

[1] Spacy, 2019.

[2] allenai, 2020.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[4] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[5] Google AI Blog. Recent advances in google translate: Understanding languages and enhancing accessibility. https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html, 2020. [Online; accessed March 20, 2023].

[6] Angana Borah, Manash Pratim Barman, and Amit Awekar. Are word embedding methods stable and should we care about it? In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, HT '21, page 45–55, New York, NY, USA, 2021. Association for Computing Machinery.

[7] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. Testing your question answering software via asking recursively. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*, pages 104–116. IEEE, 2021.

[8] Yun Chen, Yang Liu, Guanhua Chen, Xin Jiang, and Qun Liu. Accurate word alignment induction from neural machine translation. In Bonnie Webber, Trevor Cohn,

Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 566–576. Association for Computational Linguistics, 2020.

[9] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics, 2019.

[10] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[11] Chuyun Deng, Mingxuan Liu, Yue Qin, Jia Zhang, Hai-Xin Duan, and Donghong Sun. ValCAT: Variable-length contextualized adversarial transformations using encoder-decoder language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1735–1746, Seattle, United States, July 2022. Association for Computational Linguistics.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[14] Ivan Fursov, Alexey Zaytsev, Nikita Kluchnikov, Andrey Kravchenko, and Eugeny Burnaev. Differentiable language model adversarial attacks on categorical sequence classifiers. *CoRR*, abs/2006.11078, 2020.

[15] Hamidreza Ghader and Christof Monz. What does attention in neural machine translation pay attention to? In Greg Kondrak and Taro Watanabe, editors, *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 30–39. Asian Federation of Natural Language Processing, 2017.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[18] Pierre Grosdidier. Lost in translation: Google translate might not always be the best way to obtain search consent. *Texas Bar Journal*, 82(2):94, 2019.

[19] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *CoRR*, abs/1606.04435, 2016.

[20] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. Search engine guided neural machine translation. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5133–5140. AAAI Press, 2018.

[21] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5747–5757. Association for Computational Linguistics, 2021.

[22] Junliang Guo, Zhirui Zhang, Linlin Zhang, Linli Xu, Boxing Chen, Enhong Chen, and Weihua Luo. Towards variable-length textual adversarial attacks. *CoRR*, abs/2104.08139, 2021.

[23] Shashij Gupta, Pinjia He, Clara Meister, and Zhendong Su. Machine translation testing via pathological invariance. In Prem Devanbu, Myra B. Cohen, and Thomas

Zimmermann, editors, *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 863–875. ACM, 2020.

[24] Pinjia He, Clara Meister, and Zhendong Su. Structure-invariant testing for machine translation. In Gregg Rothermel and Doo-Hwan Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 961–973. ACM, 2020.

[25] Pinjia He, Clara Meister, and Zhendong Su. Testing machine translation via referential transparency. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pages 410–422. IEEE, 2021.

[26] Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael R. Lyu, and Shuming Shi. Towards understanding neural machine translation with word importance, 2019.

[27] Helsinki-NLP. opus-2020-07-14, 2020.

[28] Helsinki-NLP. opus-mt-en-zh, 2020.

[29] Bairu Hou, Jinghan Jia, Yihua Zhang, Guanhua Zhang, Yang Zhang, Sijia Liu, and Shiyu Chang. Textgrad: Advancing robustness evaluation in NLP by gradient-driven optimization. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[30] James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest subsequences. *Commun. ACM*, 20(5):350–353, 1977.

[31] Ayyoob Imani, Lütfi Kerem Senel, Masoud Jalili Sabet, François Yvon, and Hinrich Schütze. Graph neural networks for multiparallel word alignment. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1384–1396. Association for Computational Linguistics, 2022.

[32] Hidetaka Kamigaito and Manabu Okumura. Syntactically look-ahead attention network for sentence compression. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8050–8057. AAAI Press, 2020.

[33] Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. Training on synthetic noise improves robustness to natural noise in machine translation. In Wei Xu, Alan Ritter, Tim Baldwin, and Afshin Rahimi, editors, *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 42–47. Association for Computational Linguistics, 2019.

[34] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s), sep 2022.

[35] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 252–262. Association for Computational Linguistics, 2018.

[36] Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. More bang for your buck: Natural perturbation for robust question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 163–170. Association for Computational Linguistics, 2020.

[37] Huda Khayrallah and Philipp Koehn. On the impact of various types of noise on neural machine translation. In Alexandra Birch, Andrew M. Finch, Minh-Thang Luong, Graham Neubig, and Yusuke Oda, editors, *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 74–83. Association for Computational Linguistics, 2018.

[38] Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328, 2018.

[39] Frédéric Latrémolière, Sadananda Narayanappa, and Petr Vojtechovský. Estimating the jacobian matrix of an unknown multivariate function from sample values by means of a neural network. *CoRR*, abs/2204.00523, 2022.

[40] Bryan Li. Word alignment in the era of deep learning: A tutorial. *CoRR*, abs/2212.00138, 2022.

[41] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[42] Jinyu Li, Yu Wu, Yashesh Gaur, Chengyi Wang, Rui Zhao, and Shujie Liu. On the comparison of popular end-to-end models for large scale speech recognition. In Helen Meng, Bo Xu, and Thomas Fang Zheng, editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 1–5. ISCA, 2020.

[43] Xintong Li, Guanlin Li, Lemao Liu, Max Meng, and Shuming Shi. On the word alignment from neural machine translation. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1293–1303. Association for Computational Linguistics, 2019.

[44] Zhao Lihua. The relationship between machine translation and human translation under the influence of artificial intelligence machine translation. *Mobile Information Systems*, 2022:1–8, 07 2022.

[45] Yu Lu, Jiali Zeng, Jiajun Zhang, Shuangzhi Wu, and Mu Li. Learning confidence for transformer-based neural machine translation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2353–2364. Association for Computational Linguistics, 2022.

[46] Minako O'Hagan Lucas Nunes Vieira and Carol O'Sullivan. Understanding the societal impacts of machine translation: a critical review of the literature on medical and legal use cases. *Information, Communication & Society*, 24(11):1515–1532, 2021.

[47] Chao Meng, Xuesong Jiang, Jian Wang, and Xiumei Wei. The complex network model for industrial data based on spearman correlation coefficient. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2019, Atlanta, GA, USA, July 14-17, 2019*, pages 28–33. IEEE, 2019.

[48] Thien Nguyen, Lam Nguyen, Phuoc Tran, and Huu Nguyen. Improving transformer-based neural machine translation with prior alignments. *Complex.*, 2021:5515407:1–5515407:10, 2021.

[49] Jan Niehues and Ngoc-Quan Pham. Modeling confidence in sequence-to-sequence models. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 575–583. Association for Computational Linguistics, 2019.

[50] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[51] Sumant Patil and Patrick Davies. Use of google translate in medical communication: evaluation of accuracy. *Bmj*, 349, 2014.

[52] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: automated whitebox testing of deep learning systems. *Commun. ACM*, 62(11):137–145, 2019.

[53] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[54] Daniel Pesu, Zhi Quan Zhou, Jingfeng Zhen, and Dave Towey. A monte carlo method for metamorphic testing of machine translation services. In Xiaoyuan Xie, Laura L. Pullum, and Pak-Lok Poon, editors, *3rd IEEE/ACM International Workshop on Metamorphic Testing, MET 2018, Gothenburg, Sweden, May 27, 2018*, pages 38–45. ACM, 2018.

[55] Keval Pipalia, Rahul Bhadja, and Madhu Shukla. Comparative analysis of different transformer based architectures used in sentiment analysis. In *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, pages 411–415, 2020.

[56] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,*

ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers, pages 784–789. Association for Computational Linguistics, 2018.

[57] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[58] Lucas Rettenmeier. Word embeddings: Stability and semantic change, 2020.

[59] E.S. Ristad and P.N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[60] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Documentation*, 60(5):503–520, 2004.

[61] Patrick Schramowski, Cigdem Turan, Nico Andersen, Constantin A. Rothkopf, and Kristian Kersting. Language models have a moral dimension. *ArXiv*, abs/2103.11790, 2021.

[62] Qingchao Shen, Junjie Chen, Jie M. Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. Natural test generation for precise testing of question answering software. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*, pages 71:1–71:12. ACM, 2022.

[63] Harald Søndergaard and Peter Sestoft. Referential transparency, definiteness and unfoldability. *Acta Informatica*, 27(6):505–517, 1990.

[64] Kai Song, Xiaoqing Zhou, Heng Yu, Zhongqiang Huang, Yue Zhang, Weihua Luo, Xiangyu Duan, and Min Zhang. Towards better word alignment in transformer. *IEEE ACM Trans. Audio Speech Lang. Process.*, 28:1801–1812, 2020.

[65] Chang-Ai Sun, An Fu, Pak-Lok Poon, Xiaoyuan Xie, Huai Liu, and Tsong Yueh Chen. Metric$ˆ{+}$+: A metamorphic relation identification technique based on input plus output domains. *IEEE Trans. Software Eng.*, 47(9):1764–1785, 2021.

[66] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. Automatic testing and improvement of machine translation. In Gregg Rothermel and Doo-Hwan Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 974–985. ACM, 2020.

[67] Zeyu Sun, Jie M. Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. Improving machine translation systems via isotopic replacement. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pages 1181–1192. ACM, 2022.

[68] Barak Turovsky. Ten years of google translate, April 2016.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[70] Lihao Wang and Xiaoqing Zheng. Improving grammatical error correction models with purpose-built adversarial examples. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2858–2869. Association for Computational Linguistics, 2020.

[71] Shufan Wang, Laure Thompson, and Mohit Iyyer. Phrase-bert: Improved phrase embeddings from BERT with an application to corpus exploration. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10837–10851. Association for Computational Linguistics, 2021.

[72] Wenyu Wang, Wujie Zheng, Dian Liu, Changrong Zhang, Qinsong Zeng, Yuetang Deng, Wei Yang, Pinjia He, and Tao Xie. Detecting failures of neural machine translation in the absence of reference translations. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN (Industry Track) 2019, Portland, OR, USA, June 24-27, 2019*, pages 1–4. IEEE, 2019.

[73] Laura Wendlandt, Jonathan K. Kummerfeld, and Rada Mihalcea. Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018.

[74] Jason Wise. How many people use google translate in 2022, 2022.

[75] WMT18. Wmt.2018.news-commentary, 2018.

[76] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE CAA J. Autom. Sinica*, 10(5):1122–1136, 2023.

[77] Fei Xia. The part-of-speech tagging guidelines for the penn chinese treebank (3.0). 11 2000.

[78] Xiaoyuan Xie, Pengbo Yin, and Songqiang Chen. Boosting the revealing of detected violations in deep learning testing: A diversity-guided method. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*, pages 17:1–17:13. ACM, 2022.

[79] Xiaoyuan Xie, Zhiyi Zhang, Tsong Yueh Chen, Yang Liu, Pak-Lok Poon, and Baowen Xu. METTLE: A metamorphic testing approach to assessing and validating unsupervised machine learning systems. *IEEE Trans. Reliab.*, 69(4):1293–1322, 2020.

[80] Chun Zhang, Michel Laroche, and Marie-Odile Richard. The differential roles of verbs, nouns, and adjectives in english and chinese messages among bilingual consumers. *Journal of Business Research*, 72:127–135, 2017.

[81] Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. Generating fluent adversarial examples for natural languages. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5564–5569. Association for Computational Linguistics, 2019.

[82] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Trans. Software Eng.*, 48(2):1–36, 2022.

[83] Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[84] Jingyi Zhang and Josef van Genabith. A bidirectional transformer based alignment model for unsupervised word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers), pages 283–292, Online, August 2021. Association for Computational Linguistics.

[85] Zhi Quan Zhou and Liqun Sun. Metamorphic testing for machine translations: MT4MT. In *25th Australasian Software Engineering Conference, ASWEC 2018, Adelaide, Australia, November 26-30, 2018*, pages 96–100. IEEE Computer Society, 2018.

[86] Zhi Quan Zhou and Liqun Sun. Metamorphic testing for machine translations: MT4MT. In *25th Australasian Software Engineering Conference, ASWEC 2018, Adelaide, Australia, November 26-30, 2018*, pages 96–100. IEEE Computer Society, 2018.

[87] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

APPENDICES

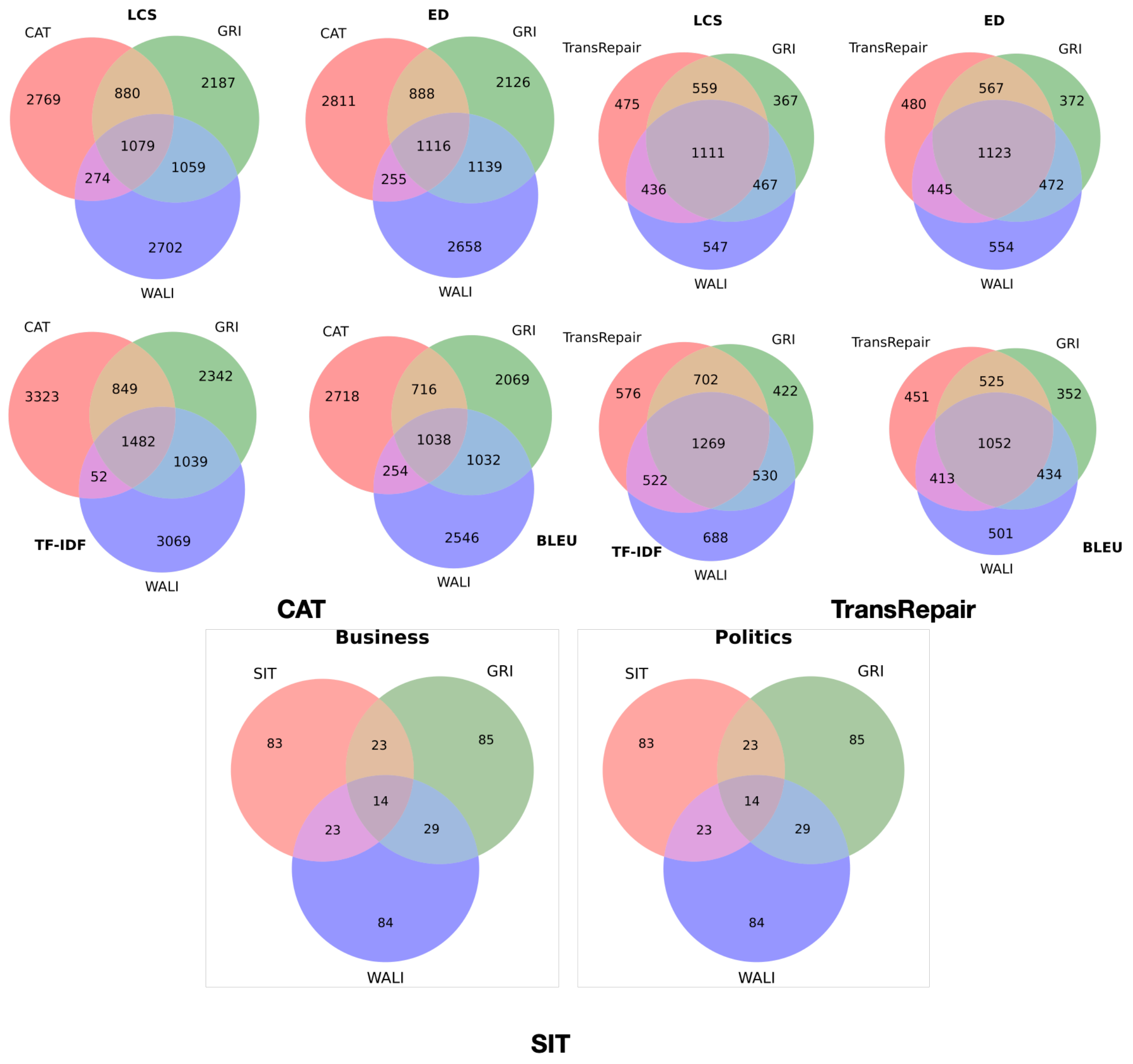


Figure 1: Overlap of the replaced tokens in translation bugs detected by GRI, WALI, and baseline approaches.

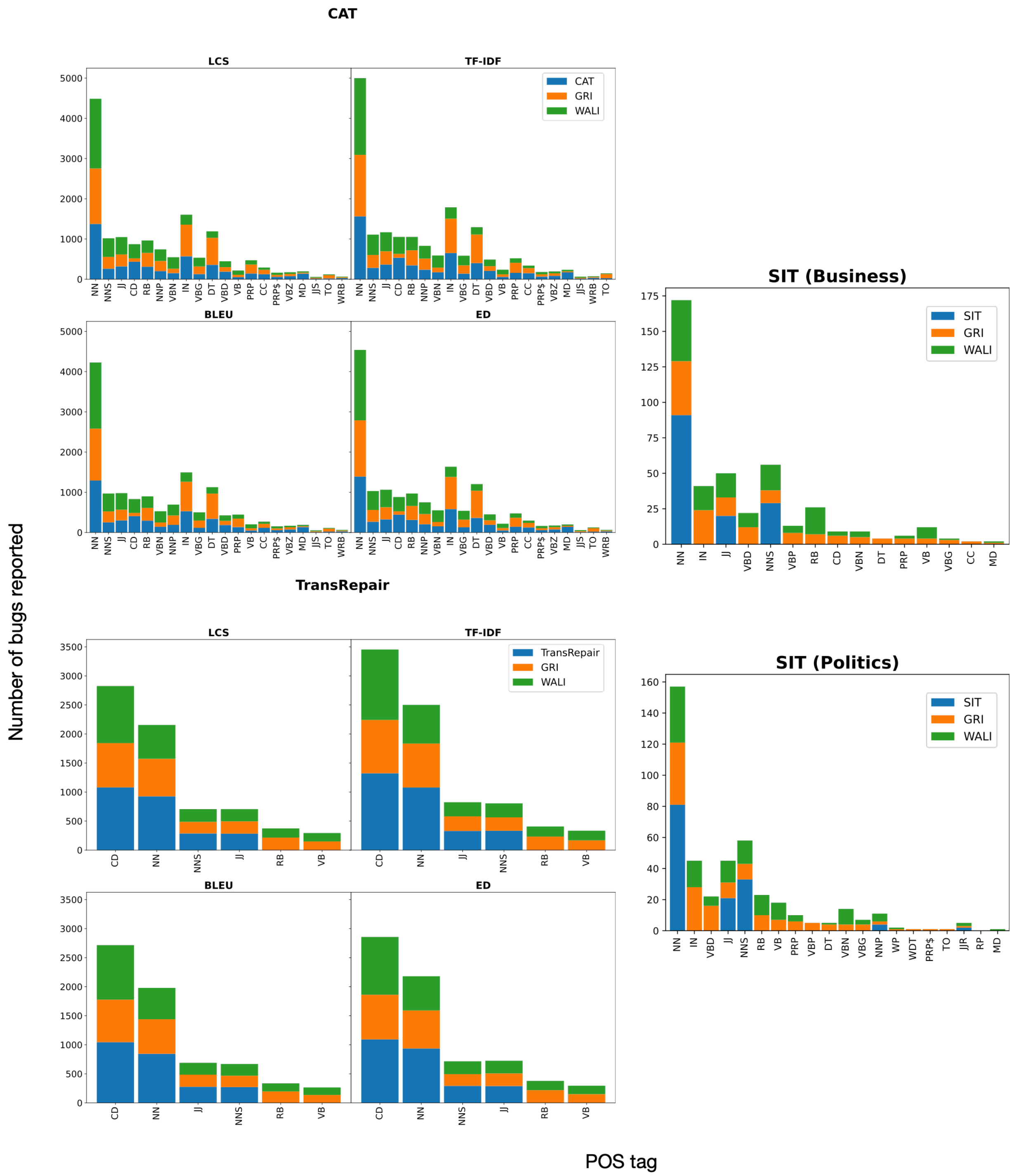


Figure 2: Number of reported bugs (y-axis) per mutant type (x-axis). This figure shows the distribution of bugs in terms of POS tags.

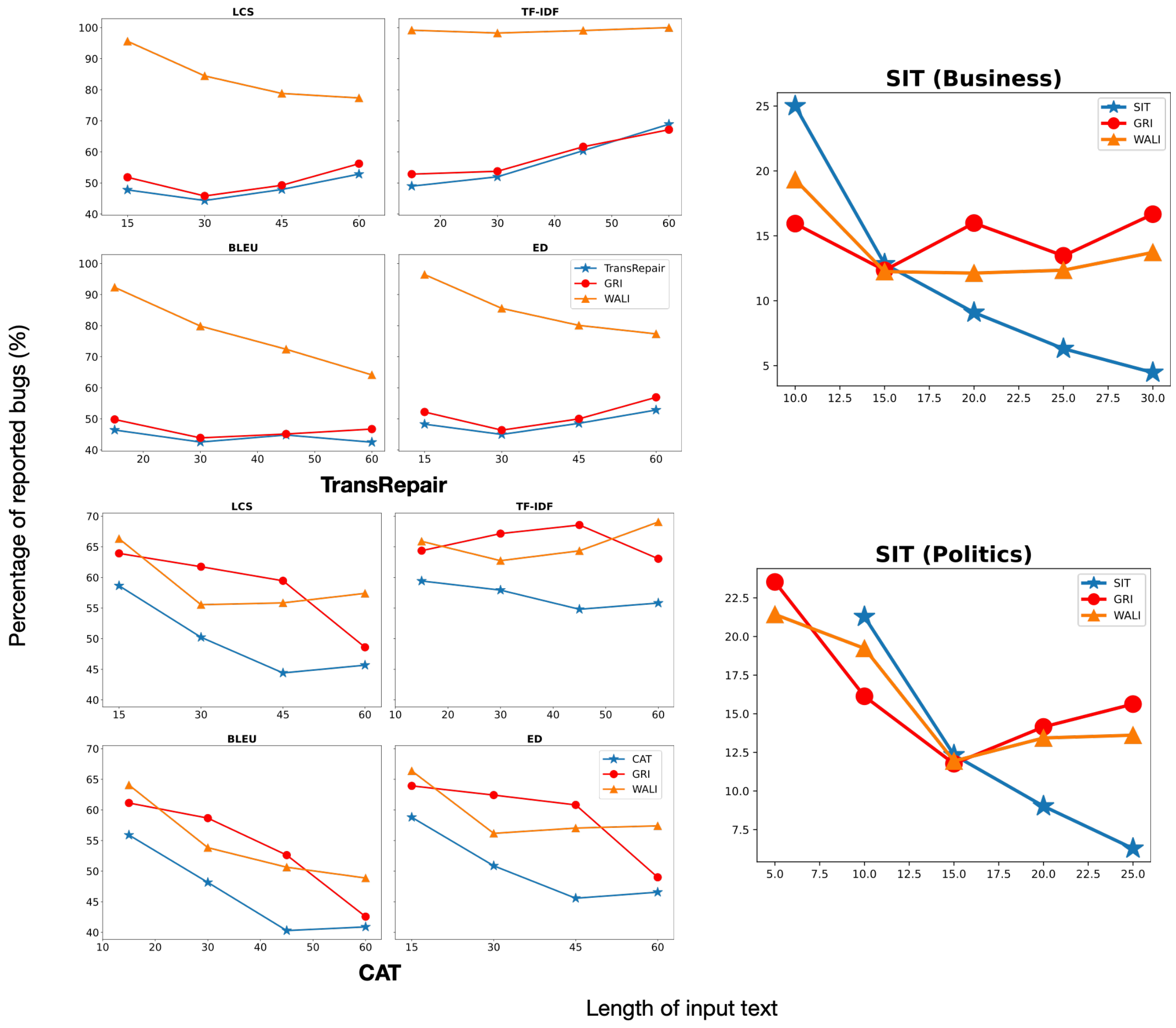


Figure 3: Percentage of reported bugs (y-axis) vs. length of the input sentence (x-axis). This figure shows the distribution of bugs in terms of sentence length.