Empirical Software Engineering (2025) 30:13
https://doi.org/10.1007/s10664-024-10549-2

Towards effectively testing machine translation systems from white-box perspectives

Hanying Shao[1] · Zishuo Ding [2] · Weiyi Shang[1] · Jinqiu Yang[3] · Nikolaos Tsantalis[3]

Accepted: 19 September 2024 / Published online: 30 October 2024

Abstract
Neural Machine Translation (NMT) has experienced significant growth over the last decade. Despite these advancements, machine translation systems still face various issues. In response, metamorphic testing approaches have been introduced for testing machine translation systems. Such approaches involve token replacement, where a single token in the original source sentence is substituted to create mutants. By comparing the translations of mutants with the original translation, potential bugs in the translation systems can be detected. However, the selection of tokens for replacement in the original sentence remains an intriguing problem, deserving further exploration in testing approaches. To address this problem, we design two white-box approaches to identify vulnerable tokens in the source sentence, whose perturbation is most likely to induce translation bugs for a translation system. The first approach, named GRI, utilizes the GRadient Information to identify the vulnerable tokens for replacement, and our second approach, named WALI, uses Word ALignment Information to locate the vulnerable tokens. We evaluate the proposed approaches on a Transformer-based translation system with the News Commentary dataset and 200 English sentences extracted from CNN articles. The results show that both GRI and WALI can effectively generate high-quality test cases for revealing translation bugs. Specifically, our approaches can always outperform state-of-the-art automatic machine translation 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).

Keywords Neural network · Neural machine translation · Software testing

1 Introduction

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

Communicated by: Andrea Stocco, Matteo Biagiola, Vincenzo Riccio, Foutse Khomh, Nicolás Cardozo and Dongwan Shin

This article belongs to the Topical Collection: *Special Issue on Innovations in Software System Testing with Deep Learning*

Extended author information available on the last page of the article different countries over the Internet. Despite the significant advancements, machine translation systems are still susceptible to various forms of noise and input variations (Belinkov and Bisk 2018; Karpukhin et al. 2019; Khayrallah and Koehn 2018), resulting in potential misunderstandings and translation errors. Consequently, testing machine translation systems have become increasingly critical as the use of these systems continues to expand (Lihua 2022; Zhang et al. 2022).

In recent years, several techniques, including SIT (He et al. 2020), PathInv (Gupta et al. 2020), TransRepair (Sun et al. 2020), and CAT (Sun et al. 2022), have been proposed to evaluate the efficacy of machine translation systems. Most of the existing research (He et al. 2020; Gupta et al. 2020; Sun et al. 2020, 2022; Xie et al. 2020) utilizes metamorphic testing strategies wherein a token or phrase from the original source sentence is substituted. Such substitutes may introduce controlled modifications to the translation output. The resulting translations are then compared to detect errors in the translation 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 (Gupta et al. 2020), yielding a very low bug detection rate. Similarly, CAT (Sun et al. 2022) 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 (Sun et al. 2022) 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.

Therefore, in this work, we propose two white-box approaches to identify vulnerable tokens in source sentences, whose perturbation is most likely to induce translation bugs for a translation system. The first approach, named GRI, utilizes the GRadient Information to identify the vulnerable tokens for replacement, and our second approach, named WALI, uses Word ALignment Information to locate the vulnerable tokens. GRI identifies the tokens in the source sentence with large gradients as candidates for replacement. WALI calculates the confidence score for each of the tokens in the target translation sentence and then adopts the word alignment information to map the target token to the source token. The source token whose aligned target token has a low confidence score is identified as a vulnerable token for replacement. The identified tokens can be iteratively replaced with semantically similar substitutions for mutant (i.e., test cases) generation. The translation system takes the newly generated 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.

We compare our approaches with three baseline approaches, CAT (Sun et al. 2022), TransRepair (Sun et al. 2020), and SIT (He et al. 2020), which are state-of-the-art machine translation testing techniques. We apply the approaches (our GRI and WALI, and baseline approaches) to test a Transformer-based translation system (Helsinki-NLP 2020b) with two commonly used datasets (Sun et al. 2022, 2020; He et al. 2020): the News Commentary dataset (WMT18 2018) and 200 English sentences from CNN[1] articles. Our evaluation shows

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

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 paper 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 translation bugs.
- Our proposed approaches, GRI and WALI can complement the current translation system testing methodologies by detecting translation bugs that were not reported before.
- A further analysis is conducted to investigate the factors that increase the likelihood of detecting translation bugs. This finding suggests that paying more attention to the selection of nouns and short sentences for translation system testing may expose more translation bugs.

Our work is an important step toward advancing the performance of machine translation 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. 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 machine translation testing methodologies. The replication package including the data, manual labeling results, and the source code are publicly accessible[2].

Paper organization Section 2 gives an overview of the approaches and introduces the two white-box approaches proposed to identify the vulnerable words in the source sentence. Section 3 describes the experimental settings used to evaluate the approaches. Section 4 demonstrates the results of our evaluation of the proposed approaches. Section 6 presents the prior studies related to this work. Section 7 discusses the threats to the validity of our results. Finally, Section 8 concludes the paper.

2 Approaches

In this section, we first formally define the task of testing neural machine translation systems. We then describe the details of our white-box approaches that leverage gradients and alignment information to efficiently generate new translation sentences by perturbing the original one.

2.1 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.

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

Formally, the optimization problem can be expressed as follows:

$$\text{maximize} \quad Sim\left(\mathbf{x}^{\text{new}}, \mathbf{x}^{\text{orig}}\right) \tag{1a}$$

$$\text{subject to} \quad Dist\left(\mathcal{F}\left(\mathbf{x}^{\text{new}}\right), \mathcal{F}\left(\mathbf{x}^{\text{orig}}\right)\right) > \xi \tag{1b}$$

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.

2.2 Approach overview

Figure 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., $Dist\left(\mathcal{F}(\mathbf{x}^{\text{new}}), \mathcal{F}\left(\mathbf{x}^{\text{orig}}\right)\right) > \xi$) 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.

2.3 Vulnerable token identification

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

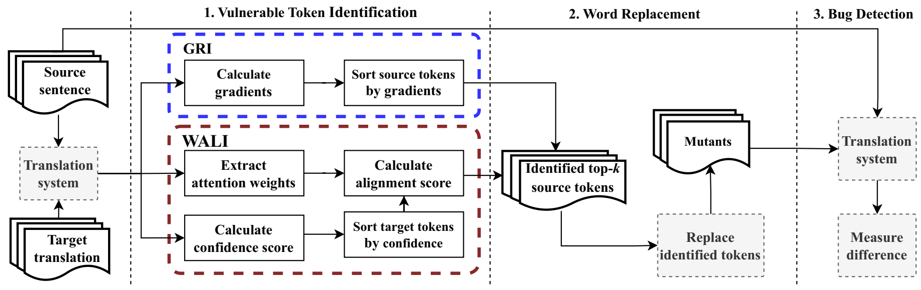

1. Vulnerable Token Identification
2. Word Replacement
3. Bug Detection
Source sentence
Translation system
Target translation
GRI
Calculate gradients
Sort source tokens by gradients
WALI
Extract attention weights
Calculate alignment score
Calculate confidence score
Sort target tokens by confidence
Identified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

Fig. 1 An overview of our approach

2.3.1 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) (Goodfellow et al. 2015; Fursov et al. 2020; Zhang et al. 2019; Grosse et al. 2016; Pei et al. 2019).

While gradient information is widely used for testing classification systems, there exists little work utilizing such information for testing neural machine translation systems. Machine translation 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 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 (Li et al. 2019), 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 translation once it is replaced.

Algorithm 1 GRI

```
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 $\mathbf{X}'$
```

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:

$$\frac{\partial L}{\partial x_i} = \sum_{j=1}^{|\mathcal{V}|} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} \tag{2}$$

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 $\mathbf{x} = (x_1, x_2, ..., x_N)$.

After obtaining the gradient information of each token in the source sequence, 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. For each token in the original sentences, the vulnerable tokens are identified using the GRI methodology. Subsequently, replacement words for each identified vulnerable token are selected based on the replacement strategies employed in baseline approaches. These replacement words are then integrated into the sentences to create the mutants. This systematic approach ensures a targeted modification of the source sentences, leveraging identified vulnerabilities to test and enhance the testing strategy.

2.3.2 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 (Wang and Zheng 2020; Nguyen et al. 2021; Zhang and van Genabith 2021; Song et al. 2020) 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 machine translation system 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, x_j will be finally identified as the vulnerable token for replacement. Figure 2 shows a visualization of word alignments between an English and Chinese sentence.

In WALI, we first calculate the generation probability score (also known as the confidence score) $p\left(y_i | y_{1:i-1}, \mathbf{x}\right)$ 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

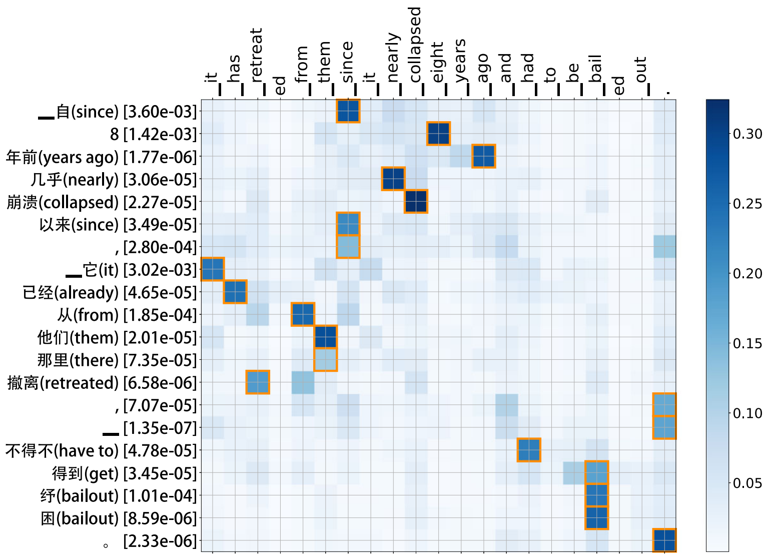


Fig. 2 An example of alignment using attention weights. The horizontal axis represents the source tokens and the left vertical axis represents the aligned target tokens(gold alignment)[generation probability after *softmax* function]. The circled squares highlight the alignments obtained using the alignment matrix $\boldsymbol{A}$ presented in Section 2.3.2

generated target sequence $y_{1:i-1}$. 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 Fig. 2, the third target token "年前 (years ago)" has the lowest score[3] and is identified as the vulnerable tokens of the target sentence.

Algorithm 2 WALI

```
    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 $\mathbf{X}'$
```

We then map the tokens in the target translation to the tokens in the source sentence based on the attention weights. Following previous work (Sun et al. 2022, 2020), we use Transformer as the translation system for testing (cf., Section 3.2), which is an encoder-decoder model that relies on attention[4].

Like prior work (Bahdanau et al. 2015; Ghader and Monz 2017; Wang and Zheng 2020), we use the attention weights of a Transformer model to obtain the alignment mapping between the target token y_i and the source token x_j, shown as below:

$$\boldsymbol{\alpha}_{i,j} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) \tag{3}$$

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. Query and key matrix are central components in the attention mechanism of Transformer models. The query matrix represents the set of queries in an attention mechanism where each query is associated with a token in the input sequence that is looking for correlated tokens in the sequence. The key matrix holds the keys associated with each element in the sequence. These keys are used to compute attention scores with the query matrix. Based on the extracted attention weights $\boldsymbol{\alpha}$, the alignment matrix $\boldsymbol{A}$ is then calculated as:

$$\boldsymbol{A}_{i,j}(\boldsymbol{\alpha}) = \begin{cases} 1 & j = \underset{j'}{\arg\max}\, \boldsymbol{\alpha}_{i,j'} \\ 0 & otherwise \end{cases} \tag{4}$$

where $\boldsymbol{A}_{i,j} = 1$ indicates y_i is aligned to x_j. To provide an example, as shown in Fig. 2, the third target token "年前 (years 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.

[3] Punctuations, like "。" and "_ _" are ignored.

[4] Due to space limitation, we refer readers to the paper (Vaswani et al. 2017) for details.

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.

Algorithm 2 outlines the procedure for generating mutants, leveraging the WALI algorithm. Initially, for each sentence, vulnerable tokens are identified and ranked based on WALI. For each vulnerable token, potential replacement words are sought using the replacement strategies defined in baseline approaches. After identifying valid replacement candidates, these are substituted into the original sentence to create test mutants. This systematic approach encapsulates the complete process of mutant generation.

2.4 Word replacement

In Section 2.3, 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 (1a), we need to maximally 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 (Sun et al. 2022, 2020; He et al. 2020). A brief description of the baseline approaches is provided below.

2.4.1 TransRepair

TransRepair employs a context-similar word replacement strategy, utilizing a pre-built context-similarity corpus. This corpus is constructed using two vector models: GloVe (Pennington et al. 2014) and SpaCy (Spa 2019). Words that exhibit a cosine similarity exceeding 0.9 across both vector models are included in the corpus. We adopted the identical word corpus provided by the original authors for consistency. For each word in the original sentence, a search is conducted within the corpus to find a suitable candidate for replacement. A structural filtering process, guided by the Stanford Parser (Manning et al. 2014), is then applied to ensure that the sentence structure remains intact. To maintain correct syntax, candidate words that have different part-of-speech (POS) tags compared to the original words are filtered out. This methodological approach ensures that replacements are contextually appropriate and grammatically coherent.

2.4.2 CAT

CAT utilizes the BERT (Devlin et al. 2019) model to identify candidates for word replacement. Initially, each word in the input sentence is replaced by the special marker "[MASK]", and this altered sentence is then processed by the BERT model. The model predicts substitute words for the masked position, which serve as potential replacements. Given that the BERT model is designed for context-aware masking tasks, it ensures that the predicted words are consistent with the original sentence's syntax and semantics. To confirm that these replacement words do not alter the sentence's semantic meaning, each candidate word is inserted back into the sentence and re-encoded using the BERT model. A cosine similarity is then computed between the context-aware vector representations of the original and the replacement words. This semantic evaluation process discards any replacements with a cosine similarity score below 0.85.

2.4.3 SIT

SIT employs the BERT model to identify replacement candidates as well. Each word in the original sentence is processed by the Masked Language Model (MLM), BERT, which predicts contextually appropriate words. To maintain the integrity of the sentence structure, only words that share the same part-of-speech (POS) tag as the original word are considered for replacement. This ensures that substitutions are syntactically coherent with the rest of the sentence.

The selected 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. For each original sentence, we substitute one token with a contextually appropriate synonym, allowing for the creation of up to five mutants per sentence.

2.5 Translation bug detection

For translation bug detection, we adopt the same test oracle presented in the prior studies (Sun et al. 2022, 2020; He et al. 2020). 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 $\mathcal{F}\left(\mathbf{x}^{\text{orig}}\right)$. 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 (Sun et al. 2022, 2020; He et al. 2020) 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 (Sun et al. 2020) and CAT (Sun et al. 2022) and the fifth metric is for SIT (He et al. 2020):

2.5.1 LCS-based metric

It measures the normalized length of the longest subsequence that is common between the original sentence and the mutated sentence (Hunt and Szymanski 1977). 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.

2.5.2 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 (Ristad and Yianilos 1998). It is widely used in applications that involve string comparison and spell-checking and can quantify the dissimilarity between two strings (Gu et al. 2018; Zhang et al. 2018).

2.5.3 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 (Robertson 2004). 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.

2.5.4 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 (Sun et al. 2020; Papineni et al. 2002) for additional information.

2.5.5 Relation distance between dependency parse trees metric

In order to assess the structural consistency of the translations, the relation distance metric measures the dissimilarity between two sets of constituency relations. This measurement is achieved by calculating the distance between two arrays of constituency grammars, defined as the sum of the absolute differences in the frequency of each phrasal type. Constituency relations represent a type of syntax representation in linguistics that organizes words into nested hierarchical structures called constituents or phrases. For instance, ”a flight” forms a Noun Phrase (NP). Sentences can be represented as parse trees with constituents like NP, VP, and PP. The distance is calculated by counting changes in the smallest constituents between original and new translations. The sentences are parsed using Stanford CoreNLP (de Marneffe et al. 2014), which provides a detailed analysis of the grammatical structure of sentences. We adhere to the Universal Dependencies annotation scheme, following the guidelines established in prior research (He et al. 2020).

For TransRepair (Sun et al. 2020) and CAT (Sun et al. 2022), 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 (Sun et al. 2020, 2022), the threshold is set to 0.963, 0.999, 0.906, 0.963 for LCS, TF-IDF, BLEU and ED respectively.

For SIT (He et al. 2020), 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 (He et al. 2020), the top-3 sentences exhibiting the greatest divergence are reported.

3 Experimental setup

3.1 Dataset

To ensure a fair comparison with baselines, following prior work (Sun et al. 2022, 2020), we utilize the News Commentary (NC) testing dataset (WMT18 2018) to evaluate CAT (Sun et al. 2022) and TransRepair (Sun et al. 2020) and 200 English sentences crawled from CNN (Cable News Network)[5] to evaluate SIT. The New Commentary dataset comprises

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

Table 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

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 the three datasets are illustrated in Table 1. Table 1 provides a detailed breakdown of the average number of words per sentence and the total number of distinct words in each dataset. The first column shows sentence length ranges in each dataset. The second and third columns present mean and median word counts per sentence. Specifically, the News Commentary dataset shows an average of 23.8 words per sentence, which is significantly higher compared to 19.2 and 19.5 words per sentence for the Politics and Business datasets, respectively. The final column lists total and unique word counts. This data suggests that sentences from the News Commentary dataset are relatively longer.

3.2 Translation system

The same as prior work (Sun et al. 2022; He et al. 2020; Gupta et al. 2020; He et al. 2021; Sun et al. 2020), we consider Transformer as the studied translation system[6]. It is one of the most widely used translation systems (Khan et al. 2022; Li et al. 2020) and has been widely studied in the research community (Sun et al. 2022; He et al. 2020; Gupta et al. 2020; He et al. 2021; Sun et al. 2020). Transformer-based pretrained language models have significantly advanced performance across NLP tasks. Many prominent Large Language Models are also based on the Transformer architecture, such as GPT and BERT. Following GPT and BERT, models like XLNet, RoBERTa, ELECTRA, ALBERT, T5, BART, and PEGASUS were developed (Kalyan et al. 2021). Consequently, the Transformer model represents a critical, state-of-the-art language model that merits study. Specifically, we utilize the pre-trained Transformer translation model, *opus-mt-en-zh* (Helsinki-NLP 2020b) as our tested translation system. The model is trained on the *opus-2020-07-14* dataset (Helsinki-NLP 2020a). We fine-tune the model with the News Commentary training set (WMT18 2018), which consists of 2, 513, 475 English-to-Chinese sentence pairs, with a learning rate of 2e−5 for 20 epochs.

3.3 Baseline approaches

We compare our approach with TransRepair (Sun et al. 2020), CAT (Sun et al. 2022), and SIT (He et al. 2020), 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.

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

3.4 Implementation settings

In this experiment, we identify the top 5 vulnerable tokens using GRI and WALI for replacement. For the purpose of a fair comparison, we adopt the same approach as in the previous study (Sun et al. 2022, 2020; He et al. 2020) for replacing the identified vulnerable tokens and generating a maximum of five new sentences for each original source sentence.[7] For each selected token, we first identify valid candidates using the replacement strategy presented in Section 2.4 and use them to create test cases. We then proceed to the next token until we have five test cases. For example, if one token has three valid substitutes, the strategy then looks for substitutes for the next token until five test sentences are generated. We conducted the experiment on Ubuntu 18.04 with an NVIDIA GTX 1080Ti GPU.

4 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 (Grosse et al. 2016; Li et al. 2019; Wang and Zheng 2020) 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 2.5). 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 (Sun et al. 2020, 2022; He et al. 2020), 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 Fig. 3. Figure 3 illustrates the cumulative count of bugs detected by different approaches, plotted against the number of test

[7] 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.

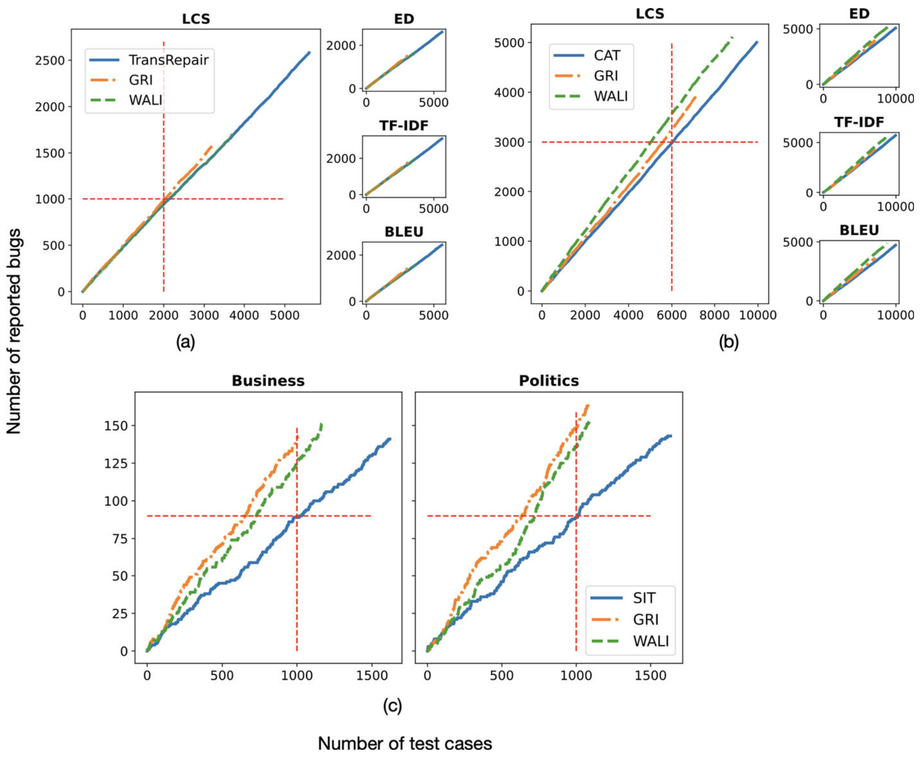


Fig. 3 Cumulative counts of detected bugs for each of the approaches

cases executed during the testing process[8]. It can be observed that our 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 3,252 and 3,568 bugs, which is higher than the 2,982 detected by CAT (Fig. 3(b)); meanwhile, GRI and WALI only need to execute 5,586 and 5,032 tests to detect 3,000 bugs, while CAT needs 6,036 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 Fig. 3(a), when running 2,000 test cases, TransRepair detects 940 bugs, relatively fewer than 979 and 940 reported by GRI and WALI. This may be due to the fact that TransRepair relies on a pre-established similarity dictionary[9] to provide suitable substitutions for specific tokens. Unlike the BERT model used in CAT and SIT (see Section 2.4), 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

[8] To ensure a fair comparison, we randomly shuffle all test cases before plotting the bug detection curve for each approach.

[9] We used the dictionary provided by the author of Sun et al. (2020).

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.

Table 2 details the number of reported bugs by the LCS metric at the 25%, 50%, and 75% percentiles of test cases executed for each approach, aligning with the Fig. 3. The first column lists approaches. Subsequent columns show bug counts at each percentile, with exact test numbers in parentheses. The percentiles are calculated based on the approach with the fewest generated test cases to ensure a standardized comparison. For example, for TransRepair, GRI, and WALI, the 75% percentile is computed based on 3,194 test cases, which is 2,395, ensuring that all approaches have a sufficient number of test cases. This table demonstrates that within a given testing budget, both GRI and WALI can detect a larger number of bugs, as indicated by the vertical dotted line in Fig. 3. Table 2 demonstrates explicitly the improved bug detection performance of white-box approaches within a fixed test budget.

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 3, with the best results highlighted in bold font. Each table presents the success rates of different approaches, with the number of reported test cases indicated in parentheses.

Table 2 Cumulative count of reported bugs by the LCS metric at 25%, 50%, 75% percentile of test cases executed

Percentile (Number of test cases)	**25% (798)**	**50% (1,597)**	**75% (2,395)**
NC			
TransRepair	388	755	1,112
GRI	394	783	1,194
WALI	379	745	1,114
Percentile (Number of test cases)	**25% (1,786)**	**50% (3,573)**	**75% (5,359)**
NC			
CAT	838	917	1,053
GRI	917	1,791	2,703
WALI	1,053	2,070	3,064
Percentile (Number of test cases)	**25% (270)**	**50% (540)**	**75% (810)**
Business			
SIT	20	40	37
GRI	44	83	73
WALI	67	118	115
Percentile (Number of test cases)	**25% (252)**	**50% (505)**	**75% (757)**
Politics			
SIT	15	35	40
GRI	35	80	61
WALI	40	112	100

Table 3 Comparison of the test success rate with TransRepair, CAT, and SIT

Metric	TransRepair	GRI	WALI
LCS	46.05% (2,581)	**49.27% (1,574)**	45.93% (1,704)
ED	46.65% (2,615)	**49.65% (1,586)**	46.47% (1,724)
TFIDF	54.75% (3,069)	**57.33% (1,831)**	54.45% (2,020)
BLEU	43.55% (2,441)	**46.49% (1,485)**	42.94% (1,593)
# of test cases	5,605	3,194	3,710

(a) Comparison with TransRepair

Metric	CAT	GRI	WALI
LCS	50.31% (5,002)	**54.95% (3,927)**	**57.92% (5,114)**
ED	51.00% (5,070)	**55.68% (3,979)**	**58.53% (5,168)**
TFIDF	57.39% (5,706)	**61.21% (4,374)**	**63.90% (5,642)**
BLEU	47.54% (4,726)	**51.25% (3,662)**	**55.15% (4,870)**
# of test cases	9,942	7,146	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 (a) specifically illustrates the success rates for TransRepair, GRI, and WALI across the four distance metrics detailed in Section 2.5. The final row of this table enumerates the total number of test cases generated by each approach. The remaining two tables (Table (b) and (c)) follow the same layout. 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. Specifically, False Positives (FP)—correct outputs mistakenly reported as errors by the automatic oracle—can arise due to the limitations inherent in using distance metrics as the basis for error detection. To address this, we carry out a manual inspection, adhering to the same evaluation process as the baseline approaches. The purpose of this 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.

To ensure statistical validity, we employed a uniform sampling strategy at a 95% confidence level to derive sample sizes that accurately represent the generated test cases for each method (Ding et al. 2023; Chen et al. 2023). We uniformly sampled from all test cases labeled as Positive and Negative by the automatic test oracle, specifically the distance metrics. The calculated representative sample sizes, shown in Table 4, were calculated to achieve

Table 4 The Precision, Recall, F1-Score, False Positive, and False Negative of different metrics for the samples from all approaches and datasets

Dataset		**News Commentary**			**News Commentary**		
Approaches		**TransRepair**	**GRI**	**WALI**	**CAT**	**GRI**	**WALI**
Sample Size		360	343	349	370	365	369
Precision	**LCS**	0.7005	0.8103	0.7528	0.7435	0.7591	0.7609
	ED	0.6906	0.7966	0.7458	0.7397	0.7551	0.7511
	TF-IDF	0.5943	0.6826	0.6186	0.6682	0.6651	0.7035
	BLEU	0.7798	0.8402	0.8170	0.7647	0.7914	0.8020
Recall	**LCS**	0.9763	0.9724	0.9781	0.9797	0.9666	0.9750
	ED	0.9842	0.9724	0.9854	0.9797	0.9866	0.9812
	TF-IDF	0.9921	0.9793	0.9708	0.9797	0.9666	0.9937
	BLEU	0.9763	0.9793	0.9781	0.9661	0.9866	0.9875
F1-score	**LCS**	0.8157	0.8840	0.8507	0.8454	0.8504	0.8547
	ED	0.8116	0.8757	0.8490	0.8430	0.8554	0.8509
	TF-IDF	0.7433	0.8045	0.7556	0.7945	0.7880	0.8238
	BLEU	0.8671	0.9044	0.8903	0.8537	0.8783	0.8851
FP	**LCS**	53	33	44	50	46	49
	ED	56	36	46	51	48	52
	TF-IDF	86	66	82	72	73	67
	BLEU	35	27	30	44	39	39
FN	**LCS**	3	4	3	3	5	4
	ED	2	4	2	3	2	3
	TF-IDF	1	3	4	3	5	1
	BLEU	3	3	3	5	2	2
Dataset		**Business**			**Politics**		
Approach		**SIT**	**GRI**	**WALI**	**SIT**	**GRI**	**WALI**
Sample Size		312	284	285	311	279	289
Precision		0.7037	0.7692	0.7250	0.7500	0.7391	0.7446
Recall		0.8261	0.9375	0.8285	0.7058	0.9189	0.8750
F1-Score		0.7600	0.7450	0.7733	0.7272	0.8192	0.8045
FP		8	9	11	4	12	12
FN		4	2	6	5	3	5

the necessary accuracy and to identify significant variations or correlations within the study population.

For reliable ground truth, the primary authors manually annotated the sampled test cases, creating human-generated labels that served as the benchmark for comparing automated metric scores. Our evaluation framework included multiple performance indicators such as precision, recall, F1-score, false positive rate, and false negative rate. These metrics were computed for each method and distance metrics are detailed in Table 4. The third row in Table 4 indicates the number of samples extracted from each approach. The sample sizes were calculated to be representative at a 95% confidence level. For TransRepair and CAT, Table 4 presents the precision, recall, F1-score, false positive (FP), and false negative (FN) rates for each distance metric: LCS, ED, TF-IDF, and BLEU. Conversely, for SIT, which employs the Distance between Dependency Parse Tree for the Business and Politics datasets, the second half of the table displays the precision, recall, F1-score, FP, and FN of the distance metric specifically for the Business and Politics datasets.

Table 4 shows the number of false positives and false negatives in the samples. It is evident that false positives significantly outnumber false negatives. To illustrate the false positives, consider the following example. For the test sentence "But the finding was not distributed outside Central Command, The Times reported in September." the token "the" was replaced by "it" to generate a test case using GRI. The translation changed from "但《纽约时报》9月报道说,这一发现并未在中央指挥部以外分发。(But the finding was not distributed outside Central Command, The Times reported in September.)" to "但据《纽约时报》9月报道,该发现并未在中央指挥部外分发。(But the finding was not distributed outside Central Command, The Times reported in September.)" Although the translation's meaning remained unchanged, the distance metrics flagged this as a potential bug because "这一" and "该" in Chinese are synonyms meaning "this," and there are subtle token changes in the translation such as "但 (but)" to "但据 (but according to)" and "报道说 (reported)" to "报道 (reported)," which do not alter the translation's meaning. This exemplifies a false positive in our experiment. The complexity of languages results in expression variations, posing a significant challenge for automatic oracles to distinguish between actual semantic changes and false positives in translation testing. False positives continue to pose a major challenge in all NLP tasks involving automatic testing. Consequently, manual evaluation is essential to verify the validity of the results.

As observed in Table 4, the precision, recall, and F1-score of GRI and WALI remain consistent compared to the baseline approaches across all distance metrics. For instance, the BLEU metric for GRI and WALI achieved precision scores of 0.79 and 0.80, recall scores of 0.98 and 0.98, and F1-score of 0.87 and 0.88, respectively with the CAT-based replacement strategy. Previous studies, such as CAT and TransRepair, reported similar metrics, with TransRepair showing precision, recall, and F1-scores of 0.7, 0.95, and 0.8, and CAT showing scores of 0.72, 0.9, and 0.8. The precision, recall, and F1-score for GRI, WALI, TransRepair, and CAT in our experiment closely align with the results from their original evaluations. These findings indicate that the bug reports generated by the baseline approaches correspond closely with the outcomes of their original experiments, especially when using identical evaluation processes. Additionally, 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 a precision of 0.70 for the 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 0.82 and 0.76 for the Business dataset, which closely aligns with that of SIT in our experiment and the accuracy reported in the original experiment. Consequently, Table 4 demonstrates that 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.[10] 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. Moreover, leveraging these two types of white-box information can more effectively identify vulnerable tokens, enhancing testing efficiency. If different approaches identify different sets of vulnerable tokens (and thus generate different test cases), practitioners with sufficient testing resources might consider combining these approaches to detect more translation bugs. 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 bug-inducing test inputs generated 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 overlap is defined as the test cases generated by substituting the same token. The degree of overlap among these approaches signifies the distinctiveness of the bugs detected by different approaches[11].

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[12] shows the overlap of the replaced tokens in the detected translation bugs by different approaches. As shown in Fig. 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 25% (594 + 694 = 1,288/5,002) and 21% (659 + 694 = 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 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

[10] 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.

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

[12] Due to the space limitation, we only show the results using LCS metric for comparison with TransRepair and CAT for Figs. 4, 5 and 6. More results are shared in Appendix A.

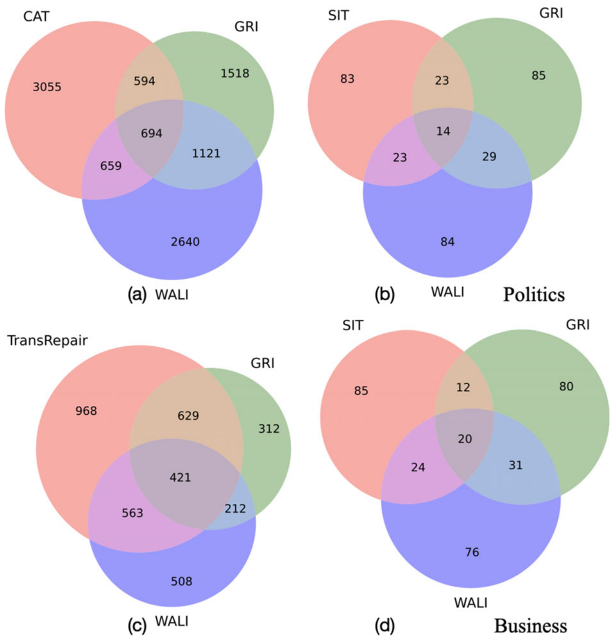


Fig. 4 Overlap of the replaced tokens in translation bugs detected by GRI, WALI, and baseline approaches

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-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 5 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” 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 “崩溃(crumbled)” and subsequently translated as “倒塌(fell apart)” due to the replacement of “ago” with “back”. “崩溃(crumbled)” generally refers to a breakdown or collapse, often used metaphorically to describe emotional or system failure. “倒塌(fell apart)” specifically means a physical collapse or falling of a structure.

Table 5 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].)

This demonstrates a unique translation bug caused by the token substitution selected by WALI.

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. 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 part-of-speech (POS) tags (Xia 2000) 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 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 part-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,

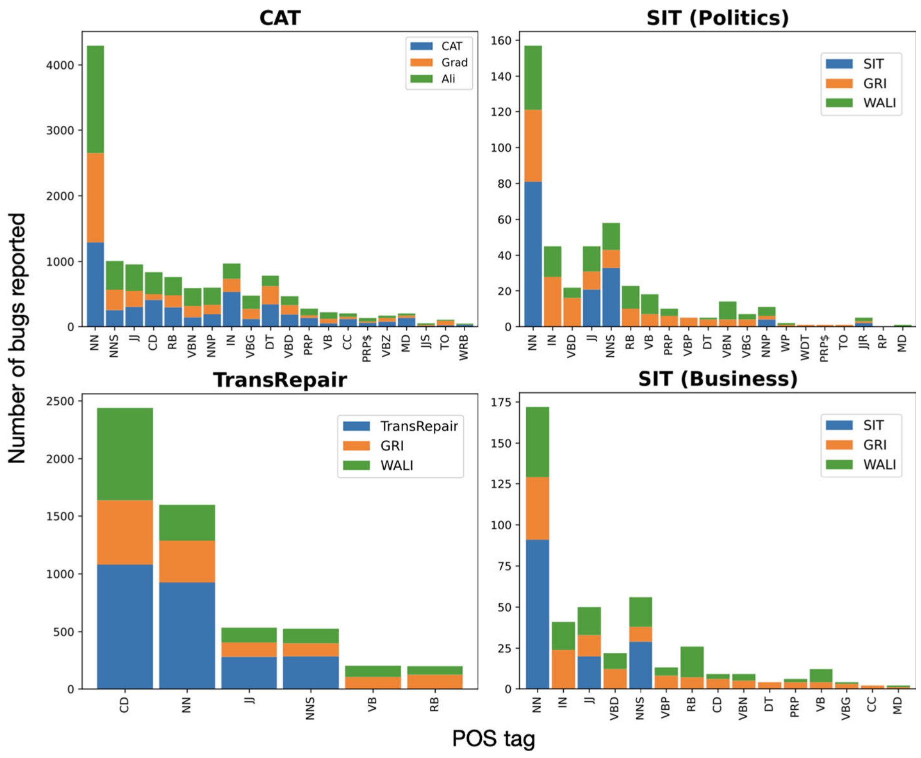


Fig. 5 Number of reported bugs (y-axis) per mutant type (x-axis). This figure shows the distribution of bugs in terms of POS tags

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 (Zhang et al. 2017). Therefore, perturbing nouns can increase the likelihood of detecting translation bugs.

Furthermore, sentences that contain a larger number of tokens tend to be more challenging for exposing translation bugs in translation systems. As illustrated in Fig. 6, 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 (Mood et al. 1973) 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 $\leq$ 5), a single substitution can cause a 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 (Guo et al. 2021; Deng et al. 2022).

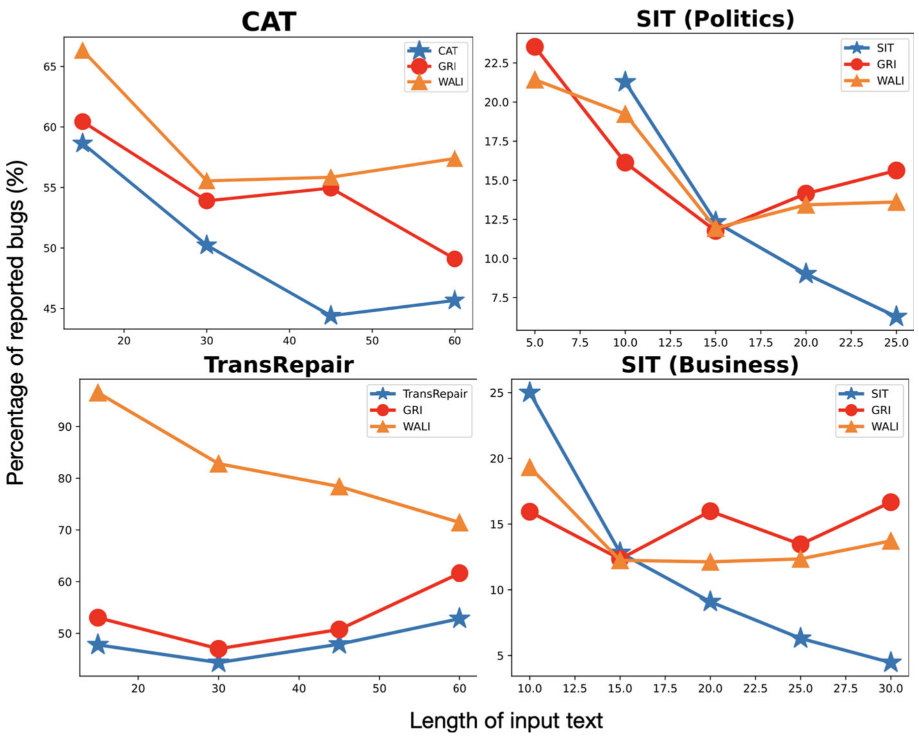


Fig. 6 Percentage of reported bugs (y-axis) vs. length of the input sentence (x-axis). This figure shows the distribution of bugs in terms of sentence length

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.

5 Discussion

In the previous section, we conducted several experiments and showed that both of our proposed approaches, GRI and WALI can effectively detect bugs in translation systems. In this section, we would like to discuss the possibility of combining GRI and WALI into one approach, and our initial attempt of trying to include large language models (LLMs) for generating tests and as target translation systems.

The possibility of combining GRI and WALI We included both GRI and WALI in our study because they utilize different types of white-box information. GRI leverages gradient information, while WALI employs word alignment information to identify vulnerable tokens. By presenting both approaches, we aim to provide a comprehensive evaluation, enabling researchers and practitioners to select the most suitable method for their specific requirements. However, as indicated by the intersection of tokens highlighted by both approaches (see Fig. 4), these intersecting tokens warrant further study.

Therefore, to explore the effectiveness of combining the two approaches, we analyzed the success rate of test cases generated by replacing the tokens identified by both methods.

Specifically, we examined how many test cases in the intersection of the GRI and WALI approaches were flagged as potential bugs.

Using a simple combination strategy, the results indicate that integrating GRI and WALI achieves a slightly higher success rate compared to each method used alone.

Table 6 exhibits the number of reported test cases for each metric across different approaches. The first row lists the distance metrics for TransRepair and CAT, while the following rows present the success rate in the intersection of GRI and WALI using the TransRepair and CAT replacement strategies, with the number of reported bugs in parentheses. The second half of the table reports the success rate reported by the Distance between the Dependency Parse Tree for SIT in the intersection of GRI and WALI for the Business and Politics datasets. For example, for the TransRepair-based approach, there are 3,216 total test cases at the intersection of the GRI and WALI approaches. Using the LCS metric, this approach reported 1,578 suspicious bugs, resulting in a 49.06% success rate. Each row shows the success rate of the intersection of GRI and WALI on top of each baseline approach. Comparing Table 6 to the results in Table 3, we observe that the success rate is similar to that of GRI and WALI separately and sometimes slightly better. This discussion utilizes a simple combination strategy to explore the possibility of employing both white-box approaches. The results show that the combination can slightly improve the performance (i.e., success rate) of the proposed approaches. Thus, it could be a promising direction to explore other strategies to combine these two types of information. In RQ 2, we further explored the intersection and the complementarity of these approaches, showing that each white-box method can identify unique bugs separately. Therefore, we believe that both approaches are valuable and should be considered for future research.

LLM-based Experiments In this part, we first explore the application of white-box approaches to Large Language Models (LLMs), which currently play a dominant role in NLP tasks. Given the significant influence and extensive use of LLMs in various domains, it is crucial to assess whether our white-box methodologies are compatible with these advanced models. We aim to apply our white-box strategies to one of the prominent LLMs to determine their applicability and effectiveness in providing insights and improvements for these models.

We selected LLaMA-3, Large Language Model Meta AI (Touvron et al. 2023), as our target model for this discussion. This choice was made because LLaMA-3 is an autoregressive language model based on a transformer architecture, recognized as one of the state-of-the-art large language models and is open-sourced for research. Released by Meta in 2023, it is available in two configurations: one with 8 billion (8B) and another with 70 billion (70B) parameters. This model has been trained on trillions of tokens sourced from publicly available datasets. Research indicates that LLaMA-13B, one of its variants, outperforms GPT-3 (175B) in most benchmarks. Due to hardware and cost limitations, we opted to work with the LLaMA-3 8B version, specifically fine-tuned on the deepctrl-sft-data (deepctrl 2024), known

Table 6 Success rate of test cases in the intersection of GRI and WALI for different approaches

	LCS	ED	TFIDF	BLEU	# of test cases
TransRepair	49.06% (1578)	49.59% (1595)	55.94% (1799)	46.21% (1486)	3216
CAT	59.22% (607)	59.71% (612)	65.27% (669)	55.71% (571)	1025
SIT	**Business**		**Politics**		
	# of reported bugs	# of test cases	# of reported bugs	# of test cases	
	14.32% (60)	419	10.87% (45)	414	

as Llama3-Chinese (Zhichen Zhang 2024). This choice enables us to assess the applicability of our white-box approaches within feasible resource constraints.

Given the resource demands associated with experimenting on LLaMA-3, we chose the Politics and Business datasets as the test sentences for our evaluation. Importantly, as LLaMA-3 is not a Sequence-to-Sequence model and lacks a cross-attention layer, the WALI approach is not applicable to this model. Therefore, for this discussion, we focus exclusively on evaluating the Gradient-based Identification (GRI) approach on the LLaMA-3 Chinese model, detailed as Llama-Chinese (Zhichen Zhang 2024), using the Politics and Business datasets as outlined in Section 3.1.

Upon reviewing the results and test cases, we noted that, although LLaMA-3 functions as a chat model and was set with the context ”Translate from English to Chinese,” some test cases did not translate successfully. The occurrences of unsuccessful translations in the chat model are random and arbitrary. In addition due to the inherent complexities of LLMs, the translations given by LLMs are often unstable. For example, given the same input sentence “Imagine if you’re a very large retailer that has a large number of locations, said Martin Fleming, chief economist at IBM,” Llama-Chinese produces different translations on two runs: first outputting “想象一下，如果您是一家拥有大量分店的大型零售商, IBM首席经济学家马丁弗林姆将会说。(Let’s imagine, if you are a very large retailer that has a large number of locations, the chief economist at IBM, Martin Fleming will say.),” and the second time, “如果你是一家非常大的零售商，并拥有许多分店，那么Martin Fleming, IBM首席经济学家所说的想象会变得非常有意义。(If you’re a very large retailer and has a large number of locations, then Martin Fleming, what the chief economist at IBM said becomes meaningful.)” Therefore, it is challenging to induce controlled modifications to the translations of LLMs, which is the fundamental idea behind the metamorphic testing strategy.

This discussion explores the applicability of the white-box approaches on LLMs, except for the hardware and cost limitations, it is possible to apply the white-box testing approaches on the LLMs. However, applying the approaches to the LLMs as a target model is challenging. However, using LLMs in the NLP task testing is a promising direction that is worth further study.

In addition to utilizing LLMs as target translation systems, we further investigate their potential as a baseline replacement approach. Unlike replacement strategies that use BERT (Devlin et al. 2019) to find substitutes and generate test cases, LLMs possess the capability to directly produce test cases. For this experiment, we utilized Llama3-8b (Touvron et al. 2023) as the baseline replacement strategy. The model was provided with the following prompt to generate test sentences: “when I give you a sentence, you can only replace ONE random word with its synonym and try not to change the semantic of the phrase, generate FIVE sentences as required.” This experiment was conducted using the Politics and Business datasets. For GRI and WALI approaches, we employed the prompt: “when I give you a sentence, you can only replace {*selected tokens*} with its synonym and try not to change the semantic of the phrase, output ONLY the generated sentence.” In this case, we explicitly specified the token to be replaced, rather than selecting a random token from the sentence. We utilized the same model, *opus-mt-en-zh* (Helsinki-NLP 2020b), as the target translation system across all experimental conditions.

The experimental results are presented in Table 7. Table 7 details the number of test cases generated by Llama3 using random replacement and the specified replacements for GRI and WALI, as listed in the last column. We employed the same distance metric as the test oracle to detect translation bugs, following the predefined threshold mentioned in Section 2.4. The number of reported bugs detected by different metrics is listed in Table 7. Analysis of Table 7 reveals that Llama3 successfully generated approximately 500 test cases for all approaches

Table 7 Number of bugs reported by Llama-based approach VS. GRI and WALI

		LCS	ED	TF-IDF	BLEU	# test cases
Business	**Llama**	451	453	461	449	500
	GRI	318	321	335	303	463
	WALI	353	358	364	341	479
Politics	**Llama**	443	444	446	438	499
	GRI	306	311	322	298	453
	WALI	315	319	327	311	464

as specified (five test cases for each of the 100 sentences from the dataset). Notably, the random replacement method implemented by Llama3 yielded a higher number of reported bugs compared to both the GRI and WALI approaches.

By inspecting the test cases generated by Llama3 and the results, we have identified several notable insights. First, Llama3 demonstrates the ability to generate semantically equivalent test cases by replacing one word in the sentence with a context-appropriate synonym. This capability surpasses that of baseline approaches such as TransRepair, CAT, and SIT, which rely on the BERT-masked language model. Llama3's superior performance can be attributed to its larger corpus and enhanced comprehension of semantic and syntactic sentence structures. Furthermore, Llama3 achieves this substitution under a black-box setting, eliminating the need for sentence encoding and word embedding extraction. Consequently, without the complex computation typically associated with encoding and embedding processes, the replacement procedure is executed more rapidly.

On the other hand, the test cases generated by Llama3 exhibit occasional instability. As previously noted, the output of Large Language Models (LLMs) can be inconsistent, particularly when the prompt imposes fewer constraints. When instructed to generate five test cases by replacing one random word in a sentence, Llama3 sometimes alters more than three tokens and even the sentence structure. For instance, given the sentence from the Business dataset, "Anxiety over flying on Boeing's 737 Max planes reached a fever pitch after the crash in Ethiopia.", Llama3 produced "Panic set in as anxiety over flying on Boeing's 737 Max planes reached a peak after the crash in Ethiopia." with the provided prompt. This example demonstrates the replacement of multiple words and the uncontrolled modification to the sentence rather than the specified single-word substitution. These inconsistencies can be mitigated by explicitly specifying the word to be replaced in the prompt. The higher frequency of such deviations in the baseline Llama3 approach accounts for the increased number of flagged bugs observed in our results.

In summary, these experiments motivate us to explore alternative approaches for utilizing LLMs for generating test cases and detecting translation bugs. For example, instead of directly asking LLMs to generate test cases, it can be used to identify the vulnerable tokens in the original source sentence for replacement and then apply the word replacement strategy to generate test cases. The potential of this methodology to enhance both the efficiency and accuracy of translation testing presents a compelling direction for our future research.

Evaluation of GRI and WALI with top-10 tokens. In our evaluation of the proposed approaches, we selected the top 5 tokens for both GRI and WALI for token substitution and test case generation. We then conducted an identical experiment using the top 10 identified tokens. The number of generated test cases and reported bugs are presented in Table 8.

A comparative analysis of Tables 8 and 3 reveals a positive correlation between the number of selected tokens and the quantity of generated test cases. Notably, the test cases generated by GRI for the TransRepair-based approach increased from 3,194 to 5,239. Concurrently, an

Table 8 Test cases generation and test success rate for TransRepair, CAT, and SIT with 10 selected tokens

Metric	**GRI**	**WALI**
LCS	2,504 (47.80%)	2,561 (46.90%)
ED	2,534 (48.37%)	2,594 (47.50%)
TFIDF	2,923 (55.80%)	3,009 (55.10%)
BLEU	2,363 (45.10%)	2,400 (43.95%)
# of test cases	5,239	5,461

(a) TransRepair

Metric	**GRI**	**WALI**
LCS	4069 (55.05%)	4530 (61.28%)
ED	4123 (55.78%)	4530 (61.28%)
TFIDF	4530 (61.28%)	4530 (61.28%)
BLEU	4530 (61.28%)	4530 (61.28%)
# of test cases	7,392	9,615

(b) CAT

Dataset	**Metrics**	**GRI**	**WALI**
Business	**Dependency**	146 (6.98%)	152 (7.59%)
	# of test cases	2089	2001
Politics	**Dependency**	155 (7.37%)	152 (7.49%)
	# of test cases	2103	2028

(c) SIT

increase in reported bugs was observed. However, this was accompanied by a decrease in the success rate.

For the CAT and SIT approaches, the success rate using top-10 tokens was found to be comparable to or lower than that achieved with top-5 tokens. This experimental outcome indicates that while an increase in the number of selected tokens leads to a higher volume of test cases, it does not necessarily correlate with improved performance of the approaches. Consequently, we select top-5 tokens for the experiment.

6 Related work

We present the related work in two parts: machine translation testing in Section 6.1 and adversarial attacks in Section 6.2.

6.1 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 (Sun et al. 2021). Recent works (Sun et al. 2022; He et al. 2021, 2020; Gupta et al. 2020; Pesu et al. 2018; Wang et al. 2019; Zhou and Sun 2018; Xie et al. 2020, 2022) 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.

Recent works such as SIT (He et al. 2020), PatInv (Gupta et al. 2020), TransRepair (Sun et al. 2020) and CAT (Sun et al. 2022) propose machine translation testing approaches that generate sentence pairs through modifying a single token in the original sentence.

TransRepair (Sun et al. 2020) utilizes vector representations of tokens to compute the similarity between each pair of tokens. SIT leverages the mask language model (MLM) (Devlin et al. 2019) to identify contextually similar replacements for a given token, thus ensuring the preservation of the sentence's structural coherence. In addition to SIT (He et al. 2020), CAT (Sun et al. 2022) 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. In contrast, PathInv generates pairs of sentences that are syntactically similar but semantically different to evaluate the translation system. The idea behind this approach is that sentences with distinct semantic meanings should not result in the same translation, which is beyond the scope of our study. Therefore, we selected TransRepair, CAT, and SIT as our baseline approaches.

In addition, a recently proposed method, DCS (Liu et al. 2023), similarly to SIT, employs constituency parsing to partition original sentences. The central concept of DCS is to achieve more precise similarity metrics between the translations of original and mutated sentences through the compositionality algorithm (Hintikka 1984).

In contrast to SIT, which computes the relation distance between constituency grammars, DCS calculates similarity scores between phrases. This approach mitigates the issue of collapse in long sentences, where the encoding of extended sentences tends to converge (Yan et al. 2021). By leveraging constituency parsing, DCS enhances detection precision. It represents a black-box testing approach that utilizes constituency parsing to detect translation bugs in the sub-structures of sentences.

However, it is important to note that DCS remains a black-box approach, like all previously mentioned methodologies. As such, it does not account for token-level differences in the substitution process. The primary focus of DCS is to address the collapse issue inherent in sentence embeddings. This is achieved through a strategic approach of sentence segmentation, effectively truncating sentences into constituent parts. This segmentation strategy allows for more granular analysis, potentially mitigating the loss of information that often occurs when encoding long sequences.

It is an intriguing idea to enhance the precision of distance metrics using DCS. However, this approach introduces the risk of generating more false positives or false negatives due to the segmentation of the translation. In many languages, sentence segmentation can be problematic as it might completely alter the semantics of the sentence. Nonetheless, experiments employing this approach could be of interest for future research.

STP (Zhang et al. 2024), introduced in 2024, represents one of the most recent strategies for testing machine translation. Unlike traditional methods that focus on word replacement, STP advocates for the removal of contextual information from the original sentence. Based on the principles of linguistic rhetorical structure theory (MANN and Thompson 1988), this approach posits that removing such contextual elements should not affect the translation outcomes of the core content, or "trunk" of the sentence. Compared to prior replacement-based strategies, STP aims to remove redundant parts of the sentence to assess the consistency of translation.

STP is a black-box approach that incrementally removes contextual words from the sentence, generating sentence pairs. However, this method may lead to inefficiencies, as it could produce a large number of test cases, potentially resulting in a waste of execution resources. To enhance performance, GRI and WALI can be employed to identify words that are more vulnerable to attack (i.e., those that should be removed).

In summary, none of the existing approaches considered the effect of the tokens chosen for replacement or deletion 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.

6.2 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. There exist various methods to create adversarial texts that are designed to evaluate classification systems (Goodfellow et al. 2015; Fursov et al. 2020; Zhang et al. 2019; Grosse et al. 2016). For example, TextBugger (Li et al. 2019) utilizes gradient information to generate adversarial examples for text classification systems. Wang et al. (Wang and Zheng 2020) propose to use the word alignment information to generate training examples to improve the grammatical error correction models. The prior studies (Goodfellow et al. 2015; Fursov et al. 2020; Zhang et al. 2019; Grosse et al. 2016; Wang and Zheng 2020) 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.

7 Threats to validity

External validity In our study, we have utilized three datasets (i.e., News Commentary (WMT18 2018) and 100 English sentences extracted from each of the CNN Business and Politics articles) and a translation system (i.e., Transformer (Helsinki-NLP 2020b)). The dataset and translation system are for English-to-Chinese 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. GRI and WALI can be extended to other models. Specifically, GRI can be utilized with other systems since gradient information is accessible for any model that can be fine-tuned or trained. Conversely, WALI relies on the cross-attention layer for word alignment information, which may limit its application to models without this layer. In Section 5, we discuss the potential of applying these white-box approaches on LLMs, highlighting both promising insights and challenges. These experiments encourage us to investigate alternative methods for detecting translation bugs.

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.

8 Conclusion

In this paper, we present two white-box approaches, GRI and WALI, which can 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. 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. Our research illuminates the advances in using white-box approaches to improving neural machine translation testing techniques. Furthermore, our approaches demonstrate the potential of using white-box-based information in the quality assurance of AI software.

A Figures

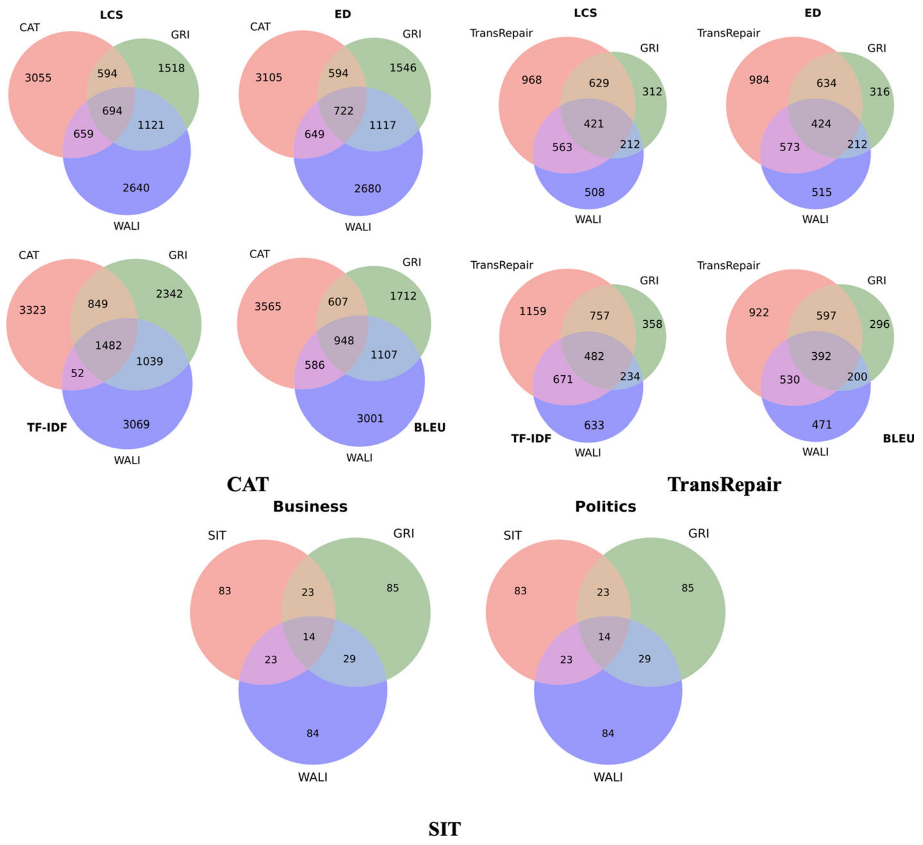


Fig. 7 Overlap of the replaced tokens in translation bugs detected by GRI, WALI, and baseline approaches

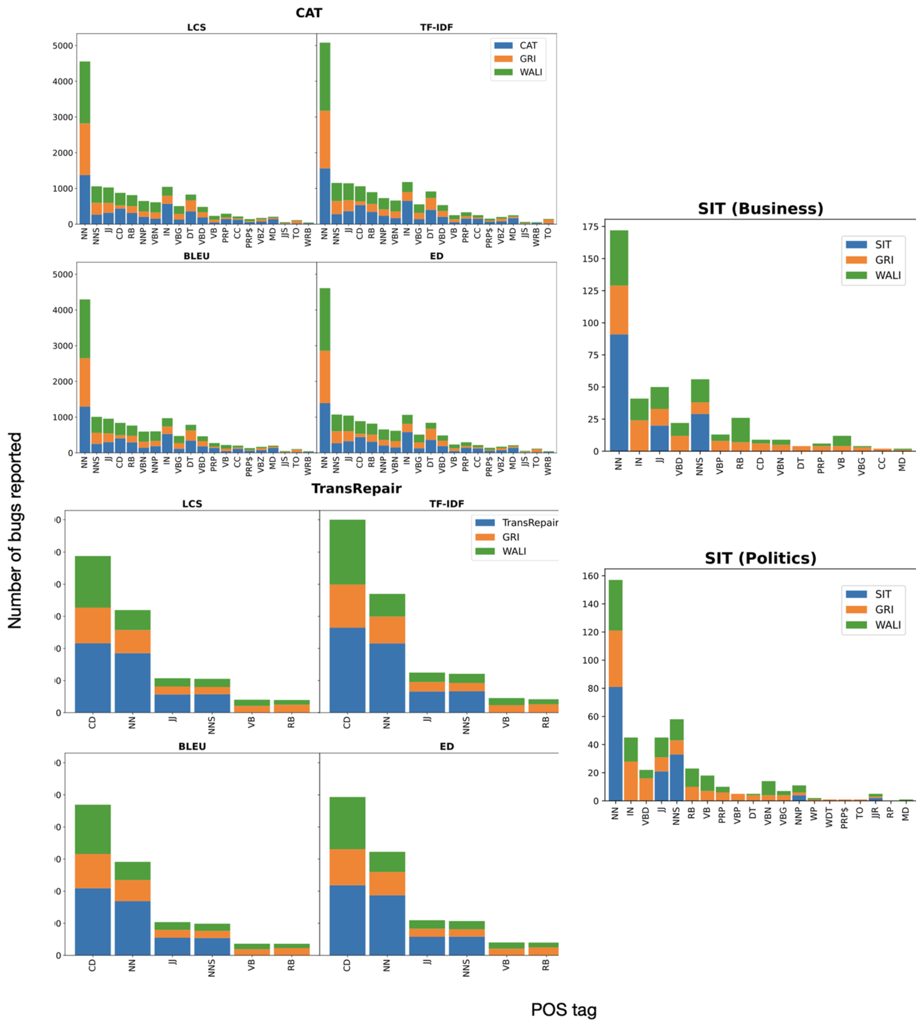


Fig. 8 Number of reported bugs (y-axis) per mutant type (x-axis). This figure shows the distribution of bugs in terms of POS tags

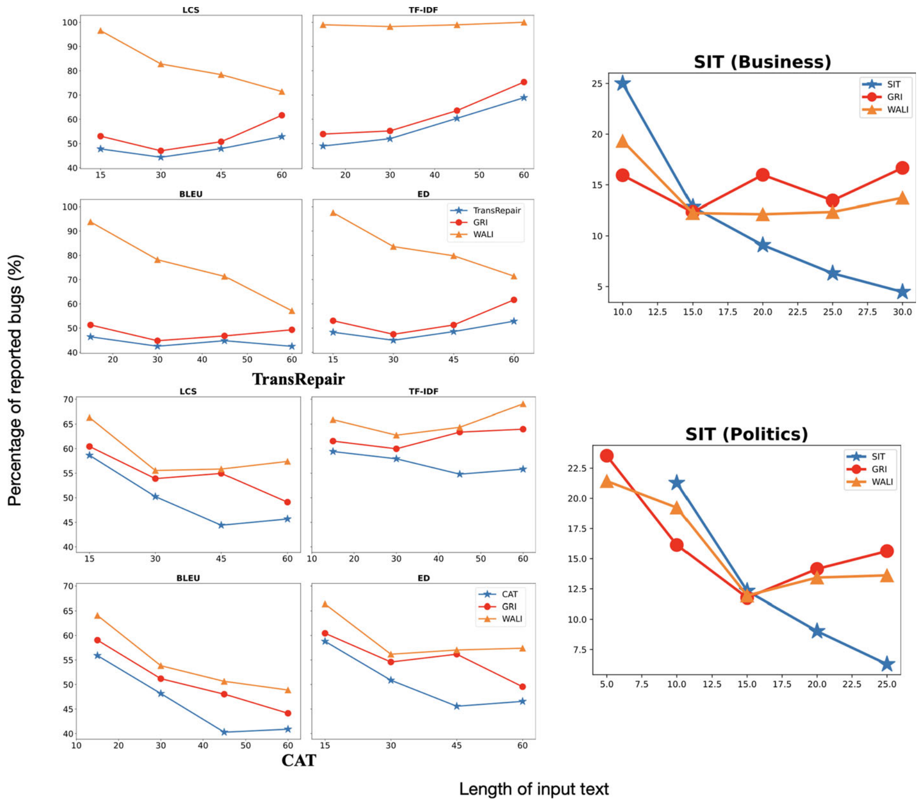


Fig. 9 Percentage of reported bugs (y-axis) vs. length of the input sentence (x-axis). This figure shows the distribution of bugs in terms of sentence length

Data Availability Statements The replication package including the data, manual labeling results, and the source code are publicly accessible at https://github.com/conf2024-8888/NMT-Testing.git

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

(2019) Spacy. https://spacy.io/

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, arXiv:1409.0473

Belinkov Y, Bisk Y (2018) 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, https://openreview.net/forum?id=BJ8vJebC-

Chen J, Ding Z, Tang Y, Sayagh M, Li H, Adams B, Shang W (2023) Iopv: On inconsistent option performance variations. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, New York, NY, USA, ESEC/FSE 2023, pp 845–857, https://doi.org/10.1145/3611643.3616319

deepctrl (2024) deepctrl-sft-data. https://www.modelscope.cn/datasets/deepctrl/deepctrl-sft-data/summary

Deng C, Liu M, Qin Y, Zhang J, Duan HX, Sun D (2022) 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, Association for Computational Linguistics, Seattle, United States, pp 1735–1746, https://doi.org/10.18653/v1/2022.naacl-main.125, https://aclanthology.org/2022.naacl-main.125

Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T (eds) 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), Association for Computational Linguistics, pp 4171–4186, https://doi.org/10.18653/v1/n19-1423

Ding Z, Tang Y, Li Y, Li H, Shang W (2023) On the temporal relations between logging and code. In: 2023 IEEE/ACM 45th International conference on software engineering (ICSE), pp 843–854, https://doi.org/10.1109/ICSE48619.2023.00079

Fursov I, Zaytsev A, Kluchnikov N, Kravchenko A, Burnaev E (2020) Differentiable language model adversarial attacks on categorical sequence classifiers. arXiv:2006.11078

Ghader H, Monz C (2017) What does attention in neural machine translation pay attention to? In: Kondrak G, Watanabe T (eds) Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers, Asian Federation of Natural Language Processing, pp 30–39, https://aclanthology.org/I17-1004/

Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, arXiv:1412.6572

Grosse K, Papernot N, Manoharan P, Backes M, McDaniel PD (2016) Adversarial perturbations against deep neural networks for malware classification. arXiv:1606.04435

Gu J, Wang Y, Cho K, Li VOK (2018) Search engine guided neural machine translation. In: McIlraith SA, Weinberger KQ (eds) 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, AAAI Press, pp 5133–5140, https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17282

Guo J, Zhang Z, Zhang L, Xu L, Chen B, Chen E, Luo W (2021) Towards variable-length textual adversarial attacks. arXiv:2104.08139

Gupta S, He P, Meister C, Su Z (2020) Machine translation testing via pathological invariance. In: Devanbu P, Cohen MB, Zimmermann T (eds) 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, ACM, pp 863–875, https://doi.org/10.1145/3368089.3409756

He P, Meister C, Su Z (2020) Structure-invariant testing for machine translation. In: Rothermel G, Bae D (eds) ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020, ACM, pp 961–973, https://doi.org/10.1145/3377811.3380339

He P, Meister C, Su Z (2021) Testing machine translation via referential transparency. In: 43rd IEEE/ACM International conference on software engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021, IEEE, pp 410–422, https://doi.org/10.1109/ICSE43902.2021.00047

Helsinki-NLP (2020a) opus-2020-07-14. https://github.com/Helsinki-NLP/Tatoeba-Challenge/blob/master/models/eng-zho/README.md

Helsinki-NLP (2020b) opus-mt-en-zh. https://huggingface.co/Helsinki-NLP/opus-mt-en-zh

Hintikka J (1984) A hundred years later: The rise and fall of frege's influence in language theory. Synthese 59:27–49, https://api.semanticscholar.org/CorpusID:46975316

Hunt JW, Szymanski TG (1977) A fast algorithm for computing longest subsequences. Commun ACM 20(5):350–353

Kalyan KS, Rajasekharan A, Sangeetha S (2021) Ammus : A survey of transformer-based pretrained models in natural language processing. arXiv:2108.05542

Karpukhin V, Levy O, Eisenstein J, Ghazvininejad M (2019) Training on synthetic noise improves robustness to natural noise in machine translation. In: Xu W, Ritter A, Baldwin T, Rahimi A (eds) Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019, Association for Computational Linguistics, pp 42–47, https://doi.org/10.18653/v1/D19-5506

Khan S, Naseer M, Hayat M, Zamir SW, Khan FS, Shah M (2022) Transformers in vision: A survey. ACM Comput Surv 54(10s), https://doi.org/10.1145/3505244

Khayrallah H, Koehn P (2018) On the impact of various types of noise on neural machine translation. In: Birch A, Finch AM, Luong M, Neubig G, Oda Y (eds) Proceedings of the 2nd Workshop on Neural Machine

Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018, Association for Computational Linguistics, pp 74–83, https://doi.org/10.18653/v1/w18-2709

Li J, Ji S, Du T, Li B, Wang T (2019) 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, https://www.ndss-symposium.org/ndss-paper/textbugger-generating-adversarial-text-against-real-world-applications/

Li J, Wu Y, Gaur Y, Wang C, Zhao R, Liu S (2020) On the comparison of popular end-to-end models for large scale speech recognition. In: Meng H, Xu B, Zheng TF (eds) Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020, ISCA, pp 1–5, https://doi.org/10.21437/Interspeech.2020-2846

Lihua Z (2022) The relationship between machine translation and human translation under the influence of artificial intelligence machine translation. Mobile Inf Syst 2022:1–8. https://doi.org/10.1155/2022/9121636

Liu S, Dou S, Chen J, Zhang Z, Lu Y (2023) Differential testing of machine translators based on compositional semantics. IEEE Trans Softw Eng 49(12):5046–5059. https://doi.org/10.1109/TSE.2023.3323969

Mann W, Thompson S (1988) Rethorical structure theory: Toward a functional theory of text organization. Text 8:243–281. https://doi.org/10.1515/text.1.1988.8.3.243

Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: Bontcheva K, Zhu J (eds) Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Baltimore, Maryland, pp 55–60, https://doi.org/10.3115/v1/P14-5010, https://aclanthology.org/P14-5010

de Marneffe MC, Dozat T, Silveira N, Haverinen K, Ginter F, Nivre J, Manning CD (2014) Universal Stanford dependencies: A cross-linguistic typology. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), European Language Resources Association (ELRA), Reykjavik, Iceland, pp 4585–4592, http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf

Mood A, Graybill F, Boes D (1973) Introduction to the Theory of Statistics. International Student edition, McGraw-Hill, https://books.google.ca/books?id=Viu2AAAAIAAJ

Nguyen T, Nguyen L, Tran P, Nguyen H (2021) Improving transformer-based neural machine translation with prior alignments. Complex 2021:5515407:1–5515407:10, https://doi.org/10.1155/2021/5515407

Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: A method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, association for computational linguistics, USA, ACL '02, pp 311–318. https://doi.org/10.3115/1073083.1073135

Pei K, Cao Y, Yang J, Jana S (2019) Deepxplore: automated whitebox testing of deep learning systems. Commun ACM 62(11):137–145. https://doi.org/10.1145/3361566

Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Moschitti A, Pang B, Daelemans W (eds) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, ACL, pp 1532–1543, https://doi.org/10.3115/V1/D14-1162

Pesu D, Zhou ZQ, Zhen J, Towey D (2018) A monte carlo method for metamorphic testing of machine translation services. In: Xie X, Pullum LL, Poon P (eds) 3rd IEEE/ACM International Workshop on Metamorphic Testing, MET 2018, Gothenburg, Sweden, May 27, 2018, ACM, pp 38–45, https://doi.org/10.1145/3193977.3193980

Ristad E, Yianilos P (1998) Learning string-edit distance. IEEE Trans Pattern Anal Machine Intell 20(5):522–532. https://doi.org/10.1109/34.682181

Robertson S (2004) Understanding inverse document frequency: on theoretical arguments for IDF. J Documentation 60(5):503–520. https://doi.org/10.1108/00220410410560582

Song K, Zhou X, Yu H, Huang Z, Zhang Y, Luo W, Duan X, Zhang M (2020) Towards better word alignment in transformer. IEEE ACM Trans Audio Speech Lang Process 28:1801–1812. https://doi.org/10.1109/TASLP.2020.2998278

Sun C, Fu A, Poon P, Xie X, Liu H, Chen TY (2021) Metric^{+}+ : A metamorphic relation identification technique based on input plus output domains. IEEE Trans Software Eng 47(9):1764–1785. https://doi.org/10.1109/TSE.2019.2934848

Sun Z, Zhang JM, Harman M, Papadakis M, Zhang L (2020) Automatic testing and improvement of machine translation. In: Rothermel G, Bae D (eds) ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020, ACM, pp 974–985, https://doi.org/10.1145/3377811.3380420

Sun Z, Zhang JM, Xiong Y, Harman M, Papadakis M, Zhang L (2022) 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, ACM, pp 1181–1192, https://doi.org/10.1145/3510003.3510206

Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G (2023) Llama: Open and efficient foundation language models. CoRR abs/2302.13971, https://doi.org/10.48550/ARXIV.2302.13971, arXiv:2302.13971

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R (eds) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp 5998–6008, https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Wang L, Zheng X (2020) Improving grammatical error correction models with purpose-built adversarial examples. In: Webber B, Cohn T, He Y, Liu Y (eds) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, Association for Computational Linguistics, pp 2858–2869, https://doi.org/10.18653/v1/2020.emnlp-main.228

Wang W, Zheng W, Liu D, Zhang C, Zeng Q, Deng Y, Yang W, He P, Xie T (2019) 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, IEEE, pp 1–4, https://doi.org/10.1109/DSN-Industry.2019.00007

WMT18 (2018) Wmt.2018.news-commentary. http://data.statmt.org/wmt18/translation-task/

Xia F (2000) The part-of-speech tagging guidelines for the penn chinese treebank (3.0)

Xie X, Zhang Z, Chen TY, Liu Y, Poon P, Xu B (2020) METTLE: A metamorphic testing approach to assessing and validating unsupervised machine learning systems. IEEE Trans Reliab 69(4):1293–1322. https://doi.org/10.1109/TR.2020.2972266

Xie X, Yin P, Chen S (2022) 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, ACM, pp 17:1–17:13, https://doi.org/10.1145/3551349.3556919

Yan Y, Li R, Wang S, Zhang F, Wu W, Xu W (2021) Consert: A contrastive framework for self-supervised sentence representation transfer. arXiv:2105.11741

Zhang C, Laroche M, Richard MO (2017) The differential roles of verbs, nouns, and adjectives in english and chinese messages among bilingual consumers. J Business Res 72:127–135

Zhang H, Zhou H, Miao N, Li L (2019) Generating fluent adversarial examples for natural languages. In: Korhonen A, Traum DR, Màrquez L (eds) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, Association for Computational Linguistics, pp 5564–5569, https://doi.org/10.18653/v1/p19-1559

Zhang J, van Genabith J (2021) 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), Association for Computational Linguistics, Online, pp 283–292, https://doi.org/10.18653/v1/2021.acl-long.24, https://aclanthology.org/2021.acl-long.24

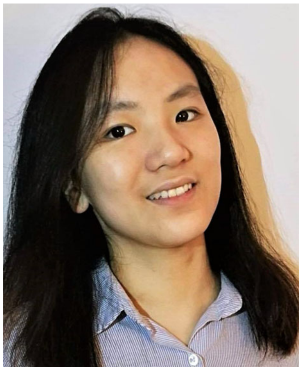
Zhang J, Utiyama M, Sumita E, Neubig G, Nakamura S (2018) Guiding neural machine translation with retrieved translation pieces. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, pp 1325–1335, https://doi.org/10.18653/v1/N18-1120, https://aclanthology.org/N18-1120

Zhang JM, Harman M, Ma L, Liu Y (2022) Machine learning testing: Survey, landscapes and horizons. IEEE Trans Software Eng 48(2):1–36. https://doi.org/10.1109/TSE.2019.2962027

Zhang Q, Zhai J, Fang C, Liu J, Sun W, Hu H, Wang Q (2024) Machine translation testing via syntactic tree pruning. CoRR abs/2401.00751, https://doi.org/10.48550/ARXIV.2401.00751, arXiv:2401.00751

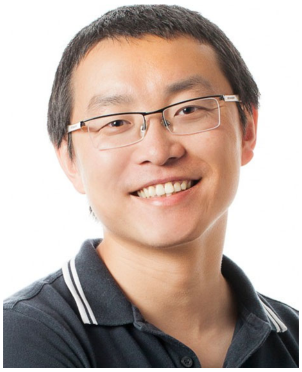
Zhichen Zhang LC Xin LU (2024) Llama3-chinese. https://github.com/seanzhang-zhichen/llama3-chinese

Zhou ZQ, Sun L (2018) Metamorphic testing for machine translations: MT4MT. In: 25th Australasian Software Engineering Conference, ASWEC 2018, Adelaide, Australia, November 26-30, 2018, IEEE Computer Society, pp 96–100, https://doi.org/10.1109/ASWEC.2018.00021



Hanying Shao successfully obtained her Master's degree in 2024 from the University of Waterloo under the supervision of Prof. Weiyi Shang. She completed her B.Sc. degree in Statistics and Computer Science at McGill University. Her research interests focus on leveraging intelligent approaches, such as machine learning, deep learning, and large language models, to enhance various software engineering tasks.

Zishuo Ding is an Assistant Professor at the Hong Kong University of Science and Technology (Guangzhou). Prior to this, he received his Ph.D. from the University of Waterloo in 2024. His research primarily focuses on leveraging intelligent approaches (e.g., ML/DL/LLMs) to enhance various software engineering tasks. His work has been published in flagship conferences and journals including ICSE, FSE, ASE, TOSEM, and EMSE, and recognized with the SIGSOFT Distinguished Paper Award at ICSE 2020. More information at https://personal.hkust-gz.edu.cn/ding/.



Weiyi Shang is a an Associate Professor in the Department of Electrical and Computer Engineering at the University of Waterloo. His research interests include AIOps, big bata software engineering, software log analytics and software performance engineering. He serves as a Steering committee member of the SPEC Research Group. He is ranked top worldwide SE research stars in a recent bibliometrics assessment of software engineering scholars. He is a recipient of various premium awards, including the SIGSOFT Distinguished paper award at ICSE 2013 and ICSE 2020, best paper award at WCRE 2011 and the Distinguished reviewer award for the Empirical Software Engineering journal. His research has been adopted by industrial collaborators (e.g., BlackBerry and Ericsson) to improve the quality and performance of their software systems that are used by millions of users worldwide.

Jinqiu Yang is an Assistant Professor in the Department of Computer Science and Software Engineering at Concordia University, Montreal, Canada. Her research interests include automated program repair, software testing, quality assurance of machine learning software, and mining software repositories. Her work has been published in flagship conferences and journals such as ICSE, FSE, EMSE. She serves regularly as a program committee member of international conferences in Software Engineering, such as ASE, ICSE, ICSME and SANER. She is a regular reviewer for Software Engineering journals such as EMSE, TSE, TOSEM and JSS. Dr. Yang obtained her BEng from Nanjing University, and MSc and PhD from University of Waterloo. More information at: https://jinqiuyang.github.io/

Nikolaos Tsantalis is an Associate professor in the department of Computer Science and Software Engineering at Concordia University, Montreal, Canada, and was a Concordia University Research Chair in Web Software Technologies between 2015-2020. His research interests include software maintenance, software evolution, empirical software engineering, refactoring recommendation systems, refactoring mining, and software quality assurance. He has developed tools, such as the Design Pattern Detection tool, JDeodorant and RefactoringMiner, which are used by many practitioners, researchers, and educators.
He has received three Most Influential Paper awards at SANER 2018, SANER 2019 and CASCON 2023, and two ACM SIGSOFT Distinguished Paper awards at FSE 2016 and ICSE 2017.

Authors and Affiliations

Hanying Shao[1] · Zishuo Ding [2] · Weiyi Shang[1] · Jinqiu Yang[3] · Nikolaos Tsantalis[3]

✉ Zishuo Ding
zishuoding@hkust-gz.edu.cn

Hanying Shao
h9shao@uwaterloo.ca

Weiyi Shang
wshang@uwaterloo.ca

Jinqiu Yang
jinqiu.yang@concordia.ca

Nikolaos Tsantalis
nikolaos.tsantalis@concordia.ca

1 Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

2 Data Science and Anaytics Thrust, Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong, China

3 Department of Computer Science and Software Engineering, Concordia University, Montreal, QC, Canada