Improving Machine Translation Systems via Isotopic Replacement

Zeyu Sun
Key Laboratory of High Confidence Software Technologies, MoE
School of Computer Science, Peking University
szy_@pku.edu.cn

Jie M. Zhang*
University College London
jie.zhang@ucl.ac.uk

Yingfei Xiong
Key Laboratory of High Confidence Software Technologies, MoE
School of Computer Science, Peking University
xiongyf@pku.edu.cn

Mark Harman
Meta platforms and University College London
mark.harman@ucl.ac.uk

Mike Papadakis
University of Luxembourg
michail.papadakis@uni.lu

Lu Zhang
Key Laboratory of High Confidence Software Technologies, MoE
School of Computer Science, Peking University
zhanglucs@pku.edu.cn

ABSTRACT

Machine translation plays an essential role in people's daily international communication. However, machine translation systems are far from perfect. To tackle this problem, researchers have proposed several approaches to testing machine translation. A promising trend among these approaches is to use word replacement, where only one word in the original sentence is replaced with another word to form a sentence pair. However, precise control of the impact of word replacement remains an outstanding issue in these approaches.

To address this issue, we propose CAT, a novel word-replacement-based approach, whose basic idea is to identify word replacement with controlled impact (referred to as isotopic replacement). To achieve this purpose, we use a neural-based language model to encode the sentence context, and design a neural-network-based algorithm to evaluate context-aware semantic similarity between two words. Furthermore, similar to TransRepair, a state-of-the-art word-replacement-based approach, CAT also provides automatic fixing of revealed bugs without model retraining.

Our evaluation on Google Translate and Transformer indicates that CAT achieves significant improvements over TransRepair. In particular, 1) CAT detects seven more types of bugs than TransRepair; 2) CAT detects 129% more translation bugs than TransRepair; 3) CAT repairs twice more bugs than TransRepair, many of which may bring serious consequences if left unfixed; and 4) CAT has better efficiency than TransRepair in input generation (0.01s v.s. 0.41s) and comparable efficiency with TransRepair in bug repair (1.92s v.s. 1.34s).

*Corresponding author.

ICSE '22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9221-1/22/05…$15.00
https://doi.org/10.1145/3510003.3510206

KEYWORDS

machine translation, testing and repair, machine learning testing, neural networks

ACM Reference Format:
Zeyu Sun, Jie M. Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. 2022. Improving Machine Translation Systems via Isotopic Replacement. In *44th International Conference on Software Engineering (ICSE '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3510003.3510206

1 INTRODUCTION

Machine translation systems, such as Google Translate, translate a text from a source language into a target language automatically. They play an essential role in overcoming communication barriers across the globe, providing translation services to millions of end users every day. In 2018, Facebook reported its deployment of machine translation supporting approximately 2,000 language pairs, serving approximately 4.5 billion translated post impressions every day, thereby allowing 600 million people to read translated posts in their mother tongue [14]. By January of 2021, Google Translate has been reported to support 109 languages, with more than 100 billion words translated per day [34].

Poor translations have been known to be a serious problem for a considerably long period of time. For example, mistranslations of Article 17 of the Treaty of Uccialli led to a war [9], while mistranslation of the Japanese government's response may have influenced American Government's decision to use the atom bomb during World War II [1]. These problems predate the advent of machine translation, but are arguably exacerbated by the apparent ease with which machines may now translate natural languages.

The profound consequences of mistranslation prior to automation are concerned with situations where there is a clear imperative for accurate translation and plenty of humans are involved in the translation process. How much more pernicious could such a situation become, as we rely more and more on automated machine translation; currently with translation processes that afford few checks and corrections?

As we move to a scenario in which machines routinely translate natural language sentences, the potential for misunderstandings may become magnified. This makes automated translation an important new application for testing and evaluation. Furthermore, given that machine translation is typically used in scenarios where there is no human in the loop, and thus no time to respond to failed test cases, it is also highly important that we have in place techniques for automated repair.

Machine translation systems currently have far more frequent mistranslations than human translations, causing problems in economics and safety, becoming a source of international tension, and also potentially violating human rights [32]. These issues highlight the importance of automatic improvement of machine translations.

Recently, quite a few approaches (e.g., SIT [15], TransRepair [32], and PathInv [12]) have been proposed for testing machine translation systems. In particular, approaches based on word replacement, where only one word in the original sentence is replaced with another word, have received intensive attention. Intuitively, since only one word in the original sentence is changed, the prediction of the corresponding change on the translation is thus easier than the situation where multiple words in the original sentence are changed. However, precise control of the impact of word replacement in word-replacement-based approaches is still difficult due to the fuzzy semantics of natural language words.

In this paper, we propose a novel word-replacement-based approach (named CAT[1]) to machine translation testing. Given a sentence in the source language, CAT performs isotopic replacement, which is a special type of word replacement, to generate new sentences. The goal of isotopic replacement is to control the impact of word replacement so that the translation of the sentence after word replacement should very likely be similar to the translation of the sentence before word replacement. Our insight is that the key to control the impact of word replacement is to control the semantic difference between the two words. In a typical natural language, the semantics of a word is comparatively stable, but the semantics may also vary to some extent in different sentences [8]. Therefore, it is necessary to consider the impact of contexts on the semantics of words when performing isotopic replacement. As a result, we design an algorithm to calculate context-aware semantic similarity between each pair of words. Our algorithm uses a neural architecture to encode the sentence context during replacement, thereby making the replacement context-aware. On top of context encoding, our algorithm further performs neural-based semantic evaluation to guarantee that the semantic difference between the words under replacement is small enough. Following TransRepair, our approach is also able to repair bugs revealed during testing.

We empirically evaluated CAT together with TransRepair on two state-of-the-art machine translation systems, Google Translate [10] and Transformer [33]. The translation is between the top two most widely used languages, English and Chinese, with 2,001 sentence pairs. Our experimental results show that CAT significantly outperforms TransRepair in both effectiveness and efficiency: CAT detects **seven more types** of bugs than TransRepair. It detects 129%/129% more bugs and repairs 199%/238% more bugs than TransRepair on Google Translate/Transformer, respectively. Over half of the detected mistranslations can be automatically fixed by CAT. Furthermore, CAT has better efficiency than TransRepair in input sentence generation (0.01s v.s. 0.41s) costs 80% and competitive efficiency with TransRepair in repairing (1.92s v.s. 1.34s).

To summarise, this paper makes the following contributions: A novel approach (named CAT) to improving machine translation based on isotopic replacement, where the key technique is a novel algorithm for calculating context-aware semantic similarity to identify isotopic replacement. An extensive evaluation to demonstrate the effectiveness and efficiency of CAT, indicating that CAT significantly outperforms the state-of-the-art approaches.

The implementation code, the data we used, together with the full experimental results of this paper are available at https://github.com/zysszy/CAT.

2 RELATED WORK

We divide the related work of our paper into two parts: machine translation testing in Section 2.1 and machine translation repair in Section 2.2.

2.1 Machine Translation Testing

There are different properties to test in machine translation testing. Heigold et al. [17], Belinkov and Bisk [2], and Zhao et al. [39] focused on testing robustness, i.e., whether machine translators are influenced by minor errors, typos, or noises in the input sentences.

More approaches focus on testing the correctness of machine translators. The major differences lie in the heuristics adopted in these approaches.

Pesu et al. [27] proposed an approach based on cross-reference. In particular, this approach focuses on checking whether the direct translation (from a source language to a target language) and the indirect translation (from the source language to an intermediate language and then from the intermediate language to the target language) of the same sentence produce the same results. Cao et al. [3] proposed a similar approach, where they check whether the translation between different translation systems produce the similar translation results.

The mainstream approaches test correctness via the combination of input mutation and metamorphic relation [38]. They adopt the strategy of controlled input generation of new sentences or phrases in the source language. Given a sentence in the source language, such an approach generates a related sentence or phrase; then the original sentence and the generated sentence (phrase) are fed into the translator under test; and the approach compares the translation results of the two to determine whether a bug is found. In particular, Purity [16] breaks the original sentence into phrases and checks whether some phrases alone have different translations compared with their translations for the original sentence.

Except for Purity, other approaches based on controlled input generation adopt word replacement. That is to say, these approaches generate a new sentence via replacing one word in the original sentence with another related word. Gupta et al. [12] detect a translation bug by replacing a word with another word with completely different meaning, and expect that the word replacement should yield different translations. Sun and Zhou [31] replace a human

[1] Context-*A*ware replacement for improving machine *T*ranslation.

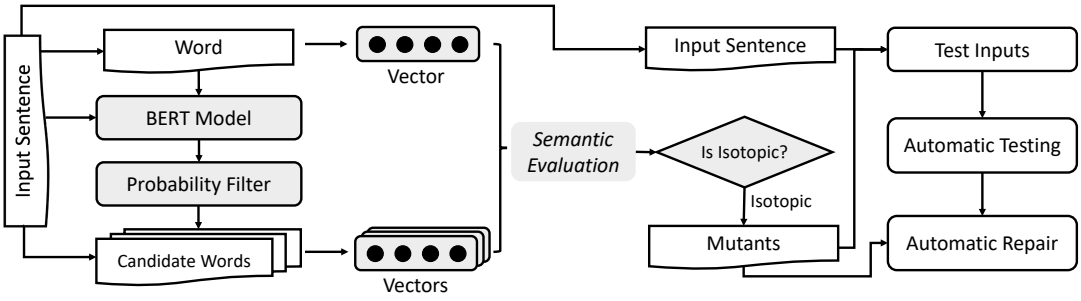

Figure 1: Overview of CAT.

name before "like" or "hate" with another human name. They expect that the translations of the sentences before and after word replacement should be similar. He et al. [15] replace one word in the original sentence with another word that would also fit into the structure of the original sentence, and expect that the two translations of the original sentence and the generated sentence should also have a similar structure. In particular, this approach (named SIT) adopts the masked language model (MLM) [5] to determine which words are likely to fit into the structure of the original sentence. Note that the word replacement in SIT is primarily syntactical without explicit consideration of semantics. Furthermore, since MLM considers the context of the original sentence when generating the replacement, many syntactically suitable replacements are excluded. For example, when facing *Make it a requirement that only half is to be used in polling stations (except in Wales).*, SIT generates a sentence by replacing "half" with "English". The approach (named TransRepair) to machine translation testing proposed in Sun et al. [32] replaces a word with another semantically similar word (e.g., girls→boys), and expect the generated sentence may lead to a similar translation with the original translation for the unchanged part in the sentence. In particular, TransRepair adopts vector representation of words [26, 29] to calculate the similarity of each pair of words. Since the vector representation of each word is calculated on the basis of all sentences containing the word in a corpus, the similarity values between words used in TransRepair do not consider the specific context of the original sentence.

Our approach (named CAT) to machine translation testing generally falls into the category of word-replacement-based approaches. Similar to SIT [15] and TransRepair [32], our approach also expects that the change of one word would not result in substantial difference in the translations. However, our approach is based on the concept of isotopic replacement, where one word is replaced with only words that differ subtly from the original word. Then we propose a specially designed algorithm to determine whether a word replacement is an isotopic replacement. In particular, our algorithm is based on calculating context-aware semantic similarity, for which to our knowledge no straightforward combination of MLM and vector representation of words are able to achieve our purpose. Another distinct merit of our approach is that isotopic replacement is applicable to all parts of speech, while existing word-replacement-based approaches are specific to only few parts of speech. For example, word replacement in SIT is specific to nouns and adjectives; and word replacement in TransRepair is specific to nouns, adjectives, and numbers.

There have been word replacement research for other tasks, such as attacking/testing/defending models on different classification tasks [11, 19, 20, 24, 28, 37]. The requirement of their word replacement is to keep the classification result unchanged. As a result, the replacement may lead to grammar errors, have completely different context, or affect the translation of the unchanged part in a sentence, thereby not being applicable for machine translation testing. For CAT, we use the isotopic replacement and the basic idea is to identify word replacement with controlled impact to the translation of the whole sentence.

2.2 Machine Translation Repair

To repair the revealed bugs and improve machine translation, existing approaches are typically based on data augmentation. Heigold et al. [17], Sperber et al. [30], and Belinkov and Bisk [2] proposed to add the generated sentences (noises) used for robustness testing to the training data and retrain the model. Cheng et al. [4] and Ebrahimi et al. [7] used the gradient-based approach to generating sentences for model retraining. Different from these approaches, there are also some approaches improving robustness of machine translation by designing some additional neural components. Cheng et al. [4] added a component to distinguish the noises from the training set, while Belinkov and Bisk [2] used a character-level representation.

The above approaches need model retraining. To our knowledge, the only approach that is able to repair machine translation without model retraining is TransRepair by Sun et al. [32]. In particular, TransRepair adopts a post processing based strategy for repair. For a bug-revealing sentence, TransRepair generates several similar sentences via the same word replacement strategy in its testing phase; and then combines the translations of these similar sentences and the original sentence to repair the bug. Our CAT adopts a similar strategy with TransRepair for repair, but the generation of similar sentences for repair in CAT is also based on isotopic replacement. According to our evaluation results, the repair effectiveness of CAT significantly outperforms that of TransRepair.

3 APPROACH

3.1 Overview

Our approach (named CAT) follows the general process of word-replacement-based approaches: Given an input sentence s and a word w in s, our approach identifies a set of words (denoted as W_w), each of which can be used to replace w in s. For each word $w_f \in W_w$,

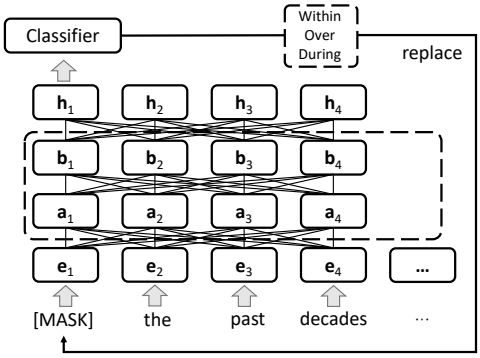

Figure 2: BERT-based context-aware word replacement.

our approach further ensures that the replacement of w with w_f in s is isotopic replacement, which only subtly impacts s. The key idea to realise isotopic replacement is to evaluate context-aware semantic similarity between w and each candidate replacement word $w_r \in C_w$ in the context of s.

Figure 1 depicts an overview of CAT. There are two stages in CAT for conducting isotopic replacement. In the first stage, CAT generates a set of word replacement candidates C_w for each word w in the input sentence s. In the second stage, CAT identifies the final set of words W_w from word candidates C_w to achieve isotopic replacement via evaluating context-aware semantic similarity between each word in $w_r \in C_w$ and the word w in sentence s.

This section mainly introduces these two parts (candidate word generation in Section 3.2 and semantic evaluation in Section 3.3). The remaining steps of translation testing and repair are similar to the corresponding steps in TransRepair. To make our presentation self-contained, we provide some necessary information in Section 3.4 for convenience.

3.2 Candidate Word Generation

In order to realise isotopic replacement, we first generate a set of candidate words C_w for each word w in the input original sentence s, where $W_w \subset C_w$, based on the context for further evaluation of context-aware semantic similarity.

Given an input sentence s, CAT generates such candidate words C_w by ensuring that each word in C_w is also suitable for the context. To realise our idea, we use the Bidirectional Encoder Representations from Transformers (BERT) [5] to encode the sentence context and check which words are suitable for the context.

Figure 2 shows the overview of context-aware word replacement in CAT. As a Transformer [33] based pre-trained model, BERT takes an input sentence and outputs a real-valued vector for each word within the sentence based on the context. When performing a word replacement, a word to be replaced in the input sentence is first masked by a special mark "[MASK]". For example, in Figure 2, the word "Within" is masked for the input sentence "Within the past decades...". The sentence with the masked word is then fed to the BERT model, which extracts a set of real-valued vectors. To conduct word replacement, BERT feeds the vector of the word "[MASK]" to a pre-trained linear classifier and gets a set of predicted words with different prediction probability.

Given an input sentence s with a word sequence $w_1, w_2, \cdots, w_N$, where N is the total number of words, we mask the words in the sentence in turn (one word being masked each time) and feed each masked sentence into BERT.

The outputs of BERT are a set of vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N$, which denotes the context-aware vector representation of the input words. Then, a pre-trained linear classifier takes the vector of the masked word $\boldsymbol{h}_{\text{mask}}$ as an input, and outputs a set of initial candidate words C_i. Each word in C_i has a predictive probability. The sum of the probabilities of all the candidate words is 1.0. We then use a probability filter to discard the words with low predictive probability (we set the threshold as 0.05 to remove the words that are most unlikely to be qualified). In addition, if the word is the same as the original word we masked, we discard this word. Finally, we treat the remaining words as word candidates $C_{w_{mask}}$ for the masked word w_{mask}, each of which can be used to fill the masked word in the original input sentence for replacement.

Note that, the above procedure for word generation is context-aware, because BERT predicts the masked word based on exactly the remaining parts of the sentence.

3.3 Semantic Evaluation

Isotopic replacement aims to identify a set of words W_w for each word w in input sentence s such that each word in W_w only subtly differs from w in the context of s. In candidate word generation, we generate a set of candidate words C_w for the word w in s. However, there are occasions that BERT predicts a masked word that has similar context with the original word, but with different meanings. Therefore, we next compute the context-aware semantic similarity and use it to discard the inappropriate words in C_w for identifying the final word set W_w.

To achieve our purpose, we propose a novel neural-based mechanism for the evaluation of context-aware semantic similarity. The key step to compute the context-aware semantic similarity in CAT is to measure the word vector similarity between the original word and its replaced word. In our approach, we again use BERT to get the output word vectors, which is the most widely used word2vec approach in natural language processing [5, 21, 40]. Unlike the usage of BERT in candidate word generation in the previous section, we do not feed BERT with a masked sentence, but directly take the whole tokenised sentence as its input. In this manner, BERT first represents a word as a vector trained from the training corpus. The vector indicates the semantics of this single word. BERT further considers the sentence context and computes a context-aware semantic representation. In this way, we get a series of word vectors for each word in each sentence, as the final word vectors.

Then, we measure the context-aware semantic similarity between the word w in s (denoted by $\boldsymbol{h}_a$) and the candidate word $w_r \in C_w$ in s_r, which is generated via replacing w in s with w_r, (denoted by $\boldsymbol{h}_b$) by using the cosine similarity CosSim of their word vectors:

$$\text{CosSim}(\boldsymbol{h}_a, \boldsymbol{h}_b) = \frac{\boldsymbol{h}_a \boldsymbol{h}_b}{|\boldsymbol{h}_a||\boldsymbol{h}_b|}. \quad (1)$$

The overall process of our semantic evaluation is detailed by Algorithm 1. For an input sentence s, a word w in s, and the set of the candidate words C_w generated via candidate word generation, we aim at getting a final word set W_w. We first define an empty set

W_w for collecting the words after evaluation (Line 1). For the word w in s, we further extract its word vector $\boldsymbol{h}_a$ via BERT (Lines 2). Next, for each candidate word $w_r \in C_w$, we use it to replace w in s and generate a mutant sentence s_r (Lines 3 and 4). For the word w_r in s_r, we extract its word vector $\boldsymbol{h}_b$ (Line 5). Then, to capture the context-aware semantic similarity of the above two words, we compute the introduced cosine similarity $CosSim$ of their word vectors $\boldsymbol{h}_a$ and $\boldsymbol{h}_b$ (Line 6). If the similarity is above the predefined threshold $SThreshold$ (we set the threshold to be 0.85), it passes the evaluation (Line 7). Thus, we add the word w_r to the set W_w as a final generated one (Line 8). After we have evaluated all words automatically, we output the final word set W_w for word w in s (Line 11). This process ensures that the replacement of w with $w_f \in W_w$ in s is isotopic.

Finally, we in turn use the word $w_f \in W_w$ to replace the word w in s and repeat this process for each word in s to generate a final set of sentences M_f. We refer to each of these finally generated sentences as a mutant [18, 25]. This generated set of mutants is further used for automatic testing (each mutant is paired with the original sentence as one test input pair) and repair.

Algorithm 1: Process of semantic evaluation

Data: s: the input sentence; w the word in the input sentence s; C_w the set of the candidate words for the word w (the output of the candidate word generation)
Result: W_w: the final set of words, each of which can be used to replace w in s

```
1  W_w = {}
2  h_a = BERT(s, w)
3  for each candidate word w_r ∈ C_w do
4      s_r = Replace(s, w, w_r)
5      h_b = BERT(s_r, w_r)
6      Similarity = CosSim(h_a, h_b)
7      if Similarity >= SThreshold then
8          W_w = W_w ∪ w_r
9      end
10 end
11 return W_w
```

3.4 Testing and Repair

The process of test oracle generation and black/grey-box repair in CAT is the same as TransRepair. We briefly introduce the details in this section.

3.4.1 Test Oracle Generation. For each test input (s, m), where $m \in M_f$, let $r(s)$ and $r(m)$ be the translation results of input sentences s and its mutant m. Mutant m is generated by replacing a word w_1 in s with another word w_2. CAT generates the test oracle by computing the similarity between $r(s)$ and $r(m)$ excluding the translation changes caused by w_1 and w_2. To remove the context changes, we computes the similarity of the subsequences of $r(s)$ and $r(m)$ and further selects the largest similarity to approximate the similarity of $r(s)$ and $r(m)$. When the similarity is below the predefined threshold, CAT reports an inconsistency bug.

3.4.2 Black/Grey-box Repair. Black-box and grey-box repair transform the original translation based on the best translation among the translations for mutants. There are two ways of choosing the best translation, one using predictive probability (grey-box), the other using cross-reference (black-box).

In detail, CAT first repairs the translation of the original sentences in a bug-revealing test case and then it seeks to find a translation for the mutant, which passes the consistency test.

For the original sentence, CAT generates a set of mutants and gets their translations. It further ranks these translations based on the predictive probability of the machine translation or cross-reference in decreasing order. CAT maps back the translation with the highest ranking to the translation of original sentence via word alignment [23]. If the mapping back fails, it selects the translation with a lower ranking.

For the mutant, CAT uses the same repair process as the original sentence but marks the output as a candidate solution. We then checks whether the candidate solution passes the testing with the repaired translation of the original sentence as test input. If not, CAT proceeds by checking other candidate solutions.

4 EXPERIMENTAL SETUP

In this section, we introduce the procedure we follow to evaluate CAT.

4.1 Research Questions

To evaluate CAT we begin by investigating the extent to which isotopic replacement brings value in testing, i.e., helps the construction of test cases. Thus, we ask:

RQ1: How effective is CAT in generating test inputs?
To answer this question, we consider the quantity, distribution, validity, and diversity of the test inputs generated with isotopic replacement. To ensure validity, we conduct a manual check on whether the replaced word in the mutant leads to grammatical errors, whether the semantic meaning of the mutant sentence is reasonable, and whether the mutant ought to have consistent translations with the original sentence.

By showing that CAT helps designing test inputs, we then turn our attention to the actual value of interest, bug detection and bug repair. We first focus on bug detection and ask:

RQ2: How effective is CAT in bug detection?
Here, we focus on the bug-revealing ability of CAT. To this end, we test Google Translate (GT) and Transformer with the test inputs generated in RQ1 to investigate: 1) the number of inconsistency bugs reported by automated test oracles; 2) the diversity of the detected inconsistency bugs; 3) the precision, recall, and F-1 measure of bug detection based on manual inspection.

Having presented the bug-revealing ability of CAT, we turn our attention to its repairing ability. Hence, we ask:

RQ3: How effective is CAT in bug repair?
To answer this question, for each reported bug on Google Translate (GT) and Transformer, we use CAT to repair the bug automatically (with both black-box and grey-box repair). We thus follow the same procedure as in RQ2 and investigate: 1) the repair effectiveness measured by automated test oracles; 2) the diversity of repaired bugs; and 3) the validity of reported fixes based on manual inspection.

Up to this point, in our analysis, we only investigated test and repair effectiveness. However, this analysis tells nothing about the time required to conduct the testing and repair tasks. Therefore, we ask:

RQ4: What is the efficiency of CAT?

To answer this question, we repeat the experiments of CAT in RQ1-3 and record the time cost. Then, we investigate: 1) the time cost of isotopic replacement; 2) the time cost of bug detection for each sentence; 3) the time cost of repairing a bug.

Since CAT follows TransRepair for improving machine translation, for ***each*** research question, we set TransRepair [32] as a baseline, and report the improvement of CAT over TransRepair under identical configuration. The comparison with other testing approaches can be referred to the extended analysis in Section 6.1.

4.2 Machine Translators

The same as TransRepair, we consider the following two state-of-the-art machine translators: 1) Transformer [33], the most widely studied machine translation system by the research community; 2) Google Translate [10], the most widely used end-to-end machine translation system developed by Google.

Transformer: Transformer is an attention-based sequence-to-sequence model designed for natural language processing tasks. It have been shown to be effective in various areas including machine translation [33], question answering [5], and others [6]. In this paper, we used identical parameters Transformer model used in [32][2] to achieve a fair comparison.

Google Translate: Google Translate is a neural machine translation system developed by Google. We select Google Translate due to it being a mainstream translation system, which has over 500 million total users with more than 100 billion words translated daily [34].

4.3 Implementation Settings

The implementation of CAT is based on the standard BERT that is publicly available [36]. The BERT model contains 24-layers, 1024 hidden size, 16 heads. The model was trained on 16 TPU chips for one million steps with a batch size of 256. For TransRepair, we implement it with the same setting according to the paper [32]. To perform a fair comparison, we follow the setting of the previous approach [32] and generate at most 5 valid mutants for each input sentence during testing and at most 16 valid mutants for each sentence in a each buggy test case during repair. We perform an additional experiment in Section 6.2 to study the influence of this mutant number upper bound.

We conduct experiments on Ubuntu 16.04 with 256GB RAM and four Intel E5-2620 v4 CPUs. The neural networks (BERT and Transformer) are all trained and inference on 8 Nvidia Titan RTXs.

4.4 Dataset

For ease of comparison, we use the same dataset as TransRepair [32], the News Commentary dataset [35]. This dataset has been adopted as a standard benchmark for translator evaluation [13]. It contains 2,001 parallel sentences that are different from the training set and validation set used for Transformer training.

[2]We use the same hyper-parameters, random seeds, deep learning library and trained the model for the same epochs.

5 RESULTS

In this section, we introduce the results of our experiment to answer the research questions.

5.1 Effectiveness of Test Generation (RQ1)

To answer this question, for each test sentence in our dataset (2,001 all together), we generate mutants with CAT and TransRepair. Each mutant is paired with the original sentence to form a test input. We then record the quantity (i.e., total number of test inputs), distribution (i.e., the number of generated test inputs per sentence), validity (i.e., whether the mutant and the original sentence in a test input should yield consistent translations for bug detection), and diversity (i.e., the types of generated mutants).

Quantity: For the 2,001 input sentences, 11,045 candidate words are generated by isotopic replacement, and 1,103 are further discarded by candidate filtering (using automatic semantic evaluation). In total, CAT generates 9,942 mutants that could be paired with the original sentences as test inputs for translation testing. For TransRepair, it generates 21,960 mutant candidates by word replacement with 17,268 discarded by candidate filtering (using Standford parser). It finally yields 4,692 test inputs.

These results show that CAT generates significantly more test inputs than TransRepair. In particular, CAT's isotopic replacement increases candidates' possibility of passing semantic evaluation. For CAT, 90% of the mutant candidates pass the filtering; for TransRepair, only 21% mutant candidates pass the filtering due to its ignorance of the context information of the original input sentences.

In the following, we dig deep into the distribution, diversity, and validity of the generated mutants.

Distribution: We use violin graphs to demonstrate the entire distribution of test inputs generated for each sentence. Figure 3 shows the results. In this figure, we observe that the number of test inputs generated by CAT reaches 5 (the upper bound, more details in Section 4.3) for most of the input sentences. However, for TransRepair, many sentences have fewer than 5 test inputs.

In particular, the isotopic replacement enables CAT to generate test inputs for almost all the sentences (with only three sentences, i.e., 0.15%, having no generated test inputs). However, for TransRepair, it fails to generate test inputs for as many as 43.7% of the sentences. Consequently, TransRepair is not able to detect any translation bugs for those sentences.

Diversity: We further compare CAT and TransRepair from the aspect of mutant diversity, i.e., the type of generated mutants in terms of part-of-speech of the replaced word. The part-of-speech types include Noun, Adj. (Adjective), Adv. (Adverb), Num. (Numeral), Verb, Deter. (Determiner), Conj. (Conjunction), Pron. (Pronoun), Prep. (Preposition), and Others.

Figure 4 shows the number of mutants belonging to each type. Overall, we observe that most mutants TransRepair generated are for nouns, adjectives, and numerals as mentioned in Section 1.

By contrast, CAT generates mutants for seven more types of part-of-speech. This demonstrated diversity is important because

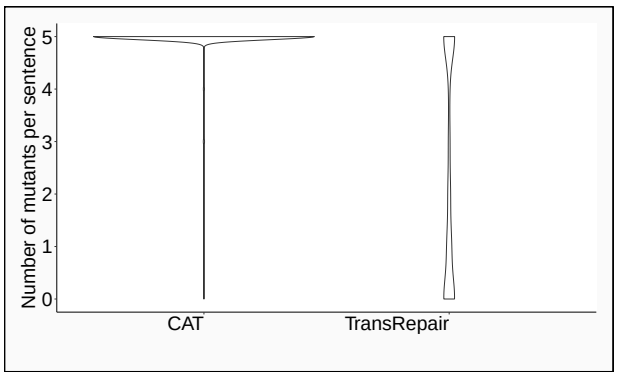

Figure 3: Distribution of the generated mutants for each sentence. CAT generates 5 test inputs (the upper bound) for almost all the input sentences, while TransRepair fails to generate any test inputs for 43.7% of the sentences.

different types of test inputs reveal different types of inconsistency bugs. For example, TransRepair will miss the detection of inconsistency bugs aroused by verbs, which plays a key role in the understanding of the sentence translation.

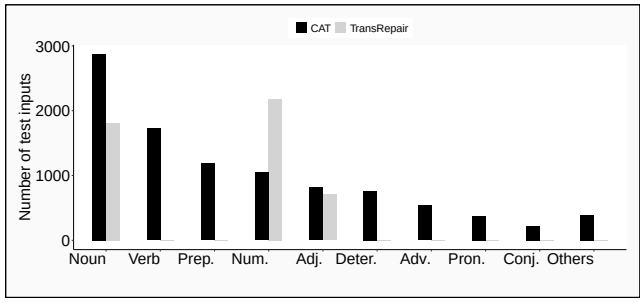

Figure 4: Number of generated test inputs (y-axis) per mutant type (x-axis). This figure shows that CAT generates more diverse mutants than TransRepair.

Validity: The validity measures the proportion of the generated test input sentences that are qualified for translation testing. The first two authors manually check the validity separately. If a generated mutant is 1) grammatically correct; 2) semantically reasonable; and 3) ought to yield semantic-identical translation with the original sentence (excluding the replaced word), we deem the test input to be valid. We randomly sample 200 test cases for each approach.

Our manual inspection[3] indicates that 96.5% of the test inputs generated by CAT are valid. Invalid mutant examples are 1) *Greater (protection⟶priority) should be given to whistleblowers, Sir Eric says;* 2) *The (Supplement⟶Suppression) of Public Sports Facilities;*. For TransRepair, 93.0% of its test inputs are valid. These results show that CAT not only generates more test inputs, but also generates more qualified test inputs for inconsistency bug detection.

[3]The Cohen's Kappa is 0.96 on average, which indicates that our manual inspection results are highly consistent.

Remember that we use semantic evaluation to filter word candidates to improve mutant validity. Thus, we are interested to investigate the effectiveness of semantic evaluation. It turns out that without semantic evaluation, 10.0% of the test inputs are invalid[4]. With semantic evaluation, this proportion is reduced to 3.5%. Thus, semantic evaluation removes (10% - 3.5%) / 10% = 65% invalid test inputs.

We further investigate the validity of test inputs on different types of mutants. Table 1 shows the results. In each cell, the first/second number is for the valid/total number of inputs belonging to the corresponding type. The ratio in brackets is the proportion of valid inputs. Interestingly, we observe that the "Adj." mutant type has the lowest validity for both CAT and TransRepair. This observation shows possibilities for further validity improvement for test input generation in machine translation testing.

Overall, for RQ1, we have the following conclusion:

> **Answer to RQ1**: CAT outperforms TransRepair in the quantity, distribution, diversity, and validity of the generated test inputs. In particular, CAT generates 9,942 test inputs, covering 99.9% of the input sentences and 10 types of mutants, with a validity score of 96.5% based on manual inspection. For TransRepair, these numbers are 4,692, 66.3%, 3, and 93.0%, respectively.

5.2 Effectiveness of Bug Detection (RQ2)

To answer RQ2, for each test input generated by CAT or TransRepair, we feed it into Google Translate and Transformer to get its translations, then apply the similarity metrics to automatically decide the test input reveals a bug.

Table 2 shows the number of bugs reported by CAT and TransRepair with each similarity metric. We observe that CAT significantly outperforms TransRepair in the number of reported bugs. On average, CAT detects 129% more bugs than TransRepair on both Google Translate and Transformer.

We further investigate the diversity of the reported bugs. The results are shown in Figure 5, where we select the reported bugs on the LCS similarity metric as an example. We observe that CAT reports diverse bugs aroused by 10 types of part-of-speeches. TransRepair can only detect the bugs for nouns, adjectives, and numerals but could hardly detect bugs aroused by the words in other part-of-speeches.

The overall results are as expected because as shown by RQ1, CAT generates a larger quantity of, more diverse, and more widely distributed mutants than TransRepair, which contributes to its overall bug detection ability.

Using the similarity metric as an approximation of test oracles, as mentioned by Sun et al. [32], may have difference with human oracles (i.e., human judgement about whether the translations are consistent). To investigate such a threat, Sun et al. [32] sampled sentences and conducted manual inspection to get the performance of oracle approximation. In this paper, following their work, we

[4]The Cohen's Kappa is 0.86.

Table 1: Validity of test inputs on different types of mutants based on part-of-speech (RQ1).

Method	Noun	Adj.	Adv.	Num.	Verb	Deter.	Conj.	Pron.	Prep.	Others
TransRepair	83/92 (90%)	16/20 (80%)	0/0 (0.0%)	86/87 (99%)	0/0 (0.0%)	0/0 (0.0%)	0/0 (0.0%)	0/0 (0.0%)	1/1 (100.0%)	0/0 (0.0%)
CAT	65/69 (94%)	11/14 (79%)	11/11 (100%)	18/18 (100%)	29/29 (100%)	18/18 (100%)	6/6 (100%)	5/5 (100%)	24/24 (100%)	6/6 (100%)

Table 2: Number of reported bugs (RQ2).

	Metric	TransRepair	CAT
GT	LCS	2,198	5,109
	ED	2,210	5,128
	TFIDF	2,430	5,381
	BLEU	2,126	4,852
Transformer	LCS	1,957	4,545
	ED	1,963	4,573
	TFIDF	2,146	4,798
	BLEU	1,897	4,325

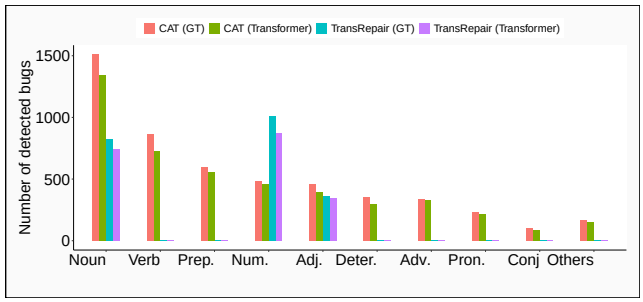

Figure 5: Number of reported bugs (y-axis) per mutant type (x-axis). This figure shows that CAT reports more diverse bugs than TransRepair.

uniformly sampled[5] 100 test inputs for each approach. For each sampled test input and its translation results, the first two authors manually check whether the test input and their translations reveal inconsistency bugs[6], then compare the human judgement with the judgement of similarity metrics. The purpose is to check whether using the similarity metric to report bugs is a threat to the superiority of CAT over TransRepair shown by Table 2.

We present the precision, recall, and F1-score of the manual inspection results. When calculating these metrics, false positive (FP) means that the similarity metric judges the translations as inconsistent (buggy) but manual inspection judges them as consistent (non-buggy); false negative (FN) means that the similarity metric judges the translations as consistent (non-buggy) but manual inspection says they are inconsistent (buggy).

The results show that for CAT, similarity metric has a precision of 0.72, a recall of 0.90, and an F1 score of 0.80 on average. For TransRepair, similarity has a precision of 0.70, a recall of 0.95, and an F1 score of 0.80 on average. These results indicate that the validity of the bugs reported by CAT is similar to that by TransRepair. TransRepair has slightly better recall than CAT. This is because many mutants TransRepair generated are via numerals replacement (as shown by Figure 4), which influence the translation consistency less than other mutant types.

Overall, our manual inspection demonstrates that the validity of the bugs reported by CAT is similar to those reported by TransRepair. Thus, using the similarity metric to report bugs is not a threat to the superiority of CAT over TransRepair shown by Table 2.

> **Answer to RQ2**: CAT detects seven more types of bugs than TransRepair in diversity, as well as 129% more bugs than TransRepair in quantity.

5.3 Effectiveness of Bug Repair (RQ3)

For ease of comparison, we let CAT and TransRepair fix the same set of detected bugs, i.e., the bugs detected by CAT. For each bug, we let CAT and TransRepair generate at most 16 mutants (the mutant number upper bound for repair introduced in Section 4.3) on both the original sentence and the mutant in the test case to conduct automatic translation repair. For Transformer, we use cross-reference to conduct black-box repair, and predictive probability to conduct grey-box repair. The predictive probability of Google Translate is inaccessible, we thus only conduct black-box repair.

To answer RQ3, we first present the number of repaired bugs accessed by similarity metrics. Then, we present the diversity of repair bugs aroused by the words in different part-of-speeches. Finally, we present the translation improvement accessed by manual inspection.

Repair effectiveness accessed by similarity metrics. The results are shown in Table 3. Each cell presents the number of repaired bugs based on the similarity metrics, as well as the proportion of repaired bugs (against the total number of bugs detected by CAT). The upper rows are for Google Translate (GT), the bottom rows are for Transformer.

We observe that CAT repairs much more bugs than TransRepair. For example, for Google Translate with the LCS similarity metric, CAT repairs 53% of the reported bugs, while TransRepair only repairs 17%. On average, CAT repairs 199% / 238% more bugs than TransRepair on Google Translate/Transformer with black-box repair (using cross-reference), and 190% more bugs than TransRepair on Transformer with grey-box repair (using probability).

Diversity of repaired bugs. We further investigate the effectiveness of CAT in different types of repaired bugs detected by the word replacement in different part-of-speeches. Since different types have different numbers of reported bugs, we focus on the proportion of repaired bugs against the number of reported bugs for each type.

[5]We sample the test case that reveals a bug and the test case that fail to reveal a bug with equal probability.

[6]The Cohen's Kappa is 0.97 on average.

Table 3: Number and proportion of repaired bugs (RQ3).

Approach	Metric	Probability	Cross-reference
TransRepair (GT)	LCS	-	890 (17%)
	ED	-	890 (17%)
	TFIDF	-	980 (18%)
	BLEU	-	886 (18%)
CAT (GT)	LCS	-	2,729 (53%)
	ED	-	2,730 (53%)
	TFIDF	-	2,741 (51%)
	BLEU	-	2,719 (56%)
TransRepair (Transformer)	LCS	738 (16%)	684 (15%)
	ED	744 (17%)	689 (15%)
	TFIDF	810 (17%)	748 (16%)
	BLEU	739 (17%)	708 (16%)
CAT (Transformer)	LCS	2,193 (48%)	2,399 (53%)
	ED	2,196 (48%)	2,399 (52%)
	TFIDF	2,231 (47%)	2,415 (50%)
	BLEU	2,176 (50%)	2,344 (54%)

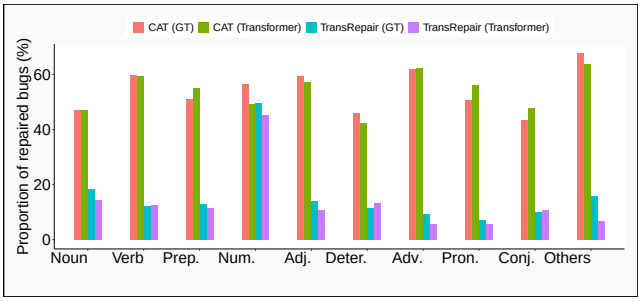

Figure 6: Proportion of repaired bugs against the number of reported bugs (y-axis) per bug type (x-axis).

The results are shown in Figure 6[7]. We observe that CAT achieves a larger proportion of repaired bugs than TransRepair among all bug types. Furthermore, we find that CAT repairs 42%-64% / 43%-68% of bugs on Transformer / Google Translate among all bug types. TransRepair repairs 45% / 50% of bugs among numerals but only 5%-14% / 7%-18% of bugs among other bug types on Transformer / Google Translate. TransRepair generates mutants for nouns, adjectives, and numerals, but it can still repair bugs in other types. This is because the semantics of the unchanged part yielded by different types of replacement could be the same.

Validity of repaired translations. The first two authors manually check the bug fixes on the LCS similarity metric with cross-reference on Transformer. The purpose is to check whether a bug fix based on the similarity metric indeed improves the translation consistency.

Although CAT aims to repair translation inconsistency, it has a "bonus" benefit of improving translation acceptability, which captures the property that a translation meets human assessment of a reasonable (aka acceptable) translation. Thus, the manual inspection considers two aspects: 1) the consistency of translations before and after repair; 2) the acceptability of the translations for the original sentence as well as the mutants before and after repair. For each dimension, we set up three labels "Improved", "Unchanged", and "Decreased". We randomly sampled 100 reported fixes. For acceptability, it checks the translation improvement of both the original sentence and the mutant, thus all together 200 inspections are conducted.

Table 4: Manual inspection results on reported fixes (RQ3)

	Aspect	Improved	Unchanged	Decreased
TransRepair	**Consistency**	87	12	1
	Acceptability	32	149	19
CAT	**Consistency**	93	7	0
	Acceptability	32	153	15

Table 4 shows the results[8]. From this table, for consistency improvement, CAT successfully improves the consistency for 93 (93%) fixes, the remaining 7 (7%) fixes have unchanged consistency. Whereas for TransRepair, it improves the consistency for 87 (87%) fixes, 12 (12%) fixes have an unchanged consistency, 1 (1%) case has a dropped consistency. For the improvement in translation acceptability, CAT improves the translation acceptability for 32 (16.0%) sentences, with 15 (7.5%) sentences' acceptability being dropped (due to possible trade-off between quality and consistency as well as the incorrect alignment from the word alignment tool during repair). TransRepair improves acceptability for the same number of sentences (32), but leads to decreased acceptability for 19 (9.5%) cases.

Table 5 shows some examples of translations that could be repaired by **only** CAT (TransRepair could repair none of them). In this table, the first column is the input sentence. The second column is the Chinese translation of the input as well as the detected translation bug (explained in blue text). The last column shows the repaired translations.

> **Answer to RQ3**: CAT repairs twice more bugs than TransRepair. For diversity, CAT repairs approximately half of the bugs under each bug type, whereas TransRepair achieves competitive performance on only numerals (53% for CAT v.s. 47% for TransRepair), failing to repair 89% of the bugs under other types.

5.4 Efficiency (RQ4)

To answer RQ4, we record the time cost of CAT and TransRepair during the process of mutant generation, bug detection, and bug repair. The results are shown in Table 6. Overall, both CAT and TransRepair have good efficiency under our configuration (see more details in Section 4.3) in their automatic testing and repair tasks. This high efficiency is important to guarantee that end users do not need to wait for a long time to get the final repaired translation while using machine translators on line.

Specifically, for mutant generation, the mean time cost is 0.01s per mutant for CAT and 0.41s for TransRepair. The reason is that

[7]We present the proportion of bugs repaired by cross-reference on the LCS similarity metric as an example

[8]The Cohen's Kappa score of the translation acceptability and translation consistency are 0.96 and 0.97, respectively.

Table 5: Examples of buggy translations that CAT can repair whereas TransRepair cannot.

Original input sentence	Original translation and the bug (explained in blue text)	Repaired translation
Around 140,000 people have a heart attack in England every year, and a quarter of these go on to have another attack or a stroke.	每年在英国约有140万人心脏病发作，其中四分之一继续发作或中风。 [Bug: "140,000" is incorrectly translated to "1.4 million"]	每年在英国约有140,000人心脏病发作，其中四分之一继续发作或 中风。 ["140,000" is correctly translated to "140,000".]
Your patience never fails to disappoint me.	你的耐心永远不会让我失望 [Bug: "never fails to" is incorrectly translated to its opposite meaning "永远不会", which means "never".]	你的耐心总是让我失望 ["never fails" is correctly translated to "总是".]
There he managed presidential protocol and government staff, the Kremlin website says (in Russian).	克里姆林宫网站说，他在那里管理着总统协议和政府工作人员。 [Bug: "in Russian" is not translated.]	那里，他管理了总统协议和政府工作人员，克里姆林宫网站说（俄文）。 ["in Russian" is correctly translated to "俄文".]

TransRepair uses Stanford Parser to parse each sentence for applying word replacement as well as filtering mutant candidates. CAT directly uses the inference of the BERT model. It avoids the parsing process, thereby greatly improving the efficiency.

Table 6: Efficiency of CAT and TransRepair (RQ4).

Approach	TransRepair	CAT
Mutant generation	0.41s	0.01s
Bug detection	0.99s	0.27s
Bug Repair	1.34s	1.92s

The lower cost of CAT in mutant generation also contributes to its lower cost in bug detection: CAT needs 0.27s on average for examining each sentence to report bugs, while TransRepair needs 0.99s.

For each reported bug, CAT spends 1.92s to conduct the automatic repair, TransRepair spends 1.34s. The reason is that TransRepair is not able to generate many mutants when fixing a bug, yet each mutant requires a fetch of translation from Google Translate/Transformer for the repair process. Consequently, its time spent on each bug is lower than CAT. However, this lower time comes at a significant cost: most bugs could not be repaired successfully (as we have observed in RQ2) due to the insufficiency of mutants.

The overall results lead to the following conclusion:

> **Answer to RQ4**: CAT is significantly more efficient than TransRepair in mutant generation (0.01s v.s. 0.41s) and bug detection (0.27s v.s. 0.99s). CAT has comparable efficiency (slightly worse) with TransRepair in bug repair (1.92s v.s. 1.34s), which is attributed to the larger number of mutants that it employs.

6 EXTENDED ANALYSIS

In this section, we provide an analysis beyond the scope of our research questions that can improve the understanding and better explain the effectiveness of CAT.

6.1 Comparison with Other Translation Testing Approaches

As introduced in Section 2, apart from TransRepair, there are also some other translation testing approaches. This section explores their effectiveness and discusses their differences with CAT.

We compare CAT (with LCS metric) with three most recently published testing methods: SIT [15], PatInv [12], and Purity [16]. For SIT, PatInv, and Purity we used their released code and parameterise them according to their best performing parameters, as shown by their papers. Table 7 presents the number of bugs detected by these four approaches on Google Translate (GT) and Transformer. As can be seen by these results, CAT and SIT detect many more bugs than TransRepair, PatInv, and Purity.

Table 7: Number of reported bugs for different machine translation testing methods.

Translator	TransRepair	SIT	PatInv	Purity	CAT
GT	2,198	5,576	82	3,459	5,109
Transformer	1,957	5,368	99	3,518	4,545

These bugs are reported according to each approach's own test oracles. Thus, they have various bug types, not necessarily inconsistency bugs. To explore the possibility of employing these approach for the same purpose of CAT (i.e., to detect inconsistency bugs), we further conduct a manual inspection on the bugs each approach reported, and report their validity in detecting inconsistency bugs. To this end, the first two authors manually inspected[9] the reported bugs and checked whether they actually exposed inconsistency translation issues. The results of this analysis are recorded in Table 8. In this table, false positive (FP) means that the approach reports the translations as problematic (buggy) but the manual inspection actually finds out that it is correct, i.e., it is not a bug. False negative (FN) means that the approach reports the translations as non-buggy but the manual inspection reveals that it is actually problematic (it is a bug). We observe that CAT achieves a significantly higher F1-score than SIT, PatInv, and Purity. This is not surprising considering that SIT, PatInv, and Purity aim to detect different types of bugs.

[9]The Cohen' Kappa score of SIT / PatInv / Purity is 0.94 / 0.89 / 0.92, respectively.

Table 8: Validity of reported bugs for different translation testing methods.

Approach	TN	FN	FP	TP	Precision	Recall	F1
TransRepair	0.48	0.02	0.15	0.35	0.70	0.95	0.80
SIT	0.28	0.22	0.33	0.17	0.34	0.44	0.38
PatInv	0.27	0.23	0.47	0.03	0.06	0.12	0.08
Purity	0.42	0.08	0.29	0.21	0.42	0.72	0.53
CAT	0.46	0.04	0.15	0.35	0.70	0.90	0.79

6.2 Influence of Mutant Number on Bug Testing and Repair

In our experiment, following the previous work, we set the maximum mutant number to 5 for bug detection, and to 16 for bug repair. Here we investigate the influence of this bound on the number of detected and repaired bugs.

For bug detection, we repeat the experiments with at most 1 or 3 mutants per sentence. Table 9 shows the number of bugs detected by CAT and TransRepair when using different upper bounds (number of mutants) on the four metrics. We find that a higher number of mutants leads to a higher number of detected bugs for both CAT and TransRepair. Notably, CAT detects approximately twice the bugs as those detected by TransRepair for all the three different bounds we examined.

Table 9: Influence of the upper bound of mutant number in bug detection.

Metric	TransRepair			CAT		
	1	3	5	1	3	5
LCS	575	1,385	1,957	1,032	2,707	4,545
ED	576	1,389	1,963	1,036	2,728	4,573
TFIDF	627	1,511	2,146	1,049	2,861	4,798
BLEU	553	1,340	1,897	1,001	2,577	4,325

For bug repair, we repeat the experiments with a maximum number of 4 or 8 mutants. Table 10 records the related results. As can be seen, no matter what upper bound we use, CAT repairs more than three times the number of bugs repaired by TransRepair. Interestingly, we observe that even with 4 mutants, CAT can still fix 143% more bugs than TransRepair with 16 mutants. This is because TransRepair fails to generate mutants for many sentences (as we have observed from Figure 3), thus it is not able to repair any of the bugs falling into these cases.

7 THREATS TO VALIDITY

Threats to external validity mainly lie in the evaluation dataset and the models we used. First, though our approach applies to different datasets and translation models, so far, we follow TransRepair [32] and have only implemented and evaluated it on the same 2,001 sentences and two translation models. So future work is needed to understand the performance of CAT on other datasets and models. Second, though we only use BERT model [5] to extract the context-aware word embeddings, our approach is still compatible with other context-aware embedding models(e.g., Roberta [22]). This is a future work to be explored.

Threats to internal validity mainly lie in our manual assessment. We follow TransRepair [32] and use the same human evaluation criteria. To further reduce the bias when comparing different methods, we randomised the sentences/test inputs to guarantee that the method each sentence belongs to remains unknown to the authors. The high kappa scores indicate that the bias in human evaluation is minor. We also release the details of human labelling on Github (available at https://github.com/zysszy/CAT) to improve the transparency and replicability.

Table 10: Influence of the upper bound of mutant number in bug repair.

Metric	TransRepair			CAT		
	4	8	16	4	8	16
LCS	516	605	684	1,724	2,117	2,399
ED	526	611	689	1,728	2,141	2,399
TFIDF	584	678	748	1,747	2,139	2,415
BLEU	536	635	708	1,667	2,053	2,344

8 CONCLUSION

We propose CAT, an isotopic replacement based approach to improving machine translation. As a special type of replacement, isotopic replacement subtly controls the impact when replacing words. This is achieved by using a neural-based language model to encode the sentence context and designing another neural-network-based algorithm to evaluate context-aware semantic similarity between two words. Such replacement is then used in a similar way as TransRepair to detect and repair translation bugs. The experimental results on Google Translate and Transformer show that CAT successfully detects 129%/129% more bugs and repairs 199%/238% more bugs than TransRepair on Google Translate/Transformer.

For future work, we plan to investigate other oracle approximation techniques to further improve CAT's performance in bug detection and repair.

ACKNOWLEDGMENTS

Zeyu Sun, Yingfei Xiong, and Lu Zhang are sponsored by the National Key Research and Development Program of China under Grant No. 2019YFE0198100, the Innovation and Technology Commission of HKSAR under Grant No. MHP/055/19, and National Natural Science Foundation of China under Grant No. 61922003. Jie M. Zhang and Mark Harman are supported by the ERC advanced grant with No. 741278. Mike Papadakis is supported by the Luxembourg National Research Fund (FNR) through the CORE project C17/IS/11686509/CODEMATES.

REFERENCES

[1] [n.d.]. The worst translation mistake in history. https://pangeanic.co.uk/knowledge/the-worst-translation-mistake-in-history/
[2] Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proc. ICLR.*

[3] Jialun Cao, Meiziniu Li, Yeting Li, Ming Wen, and Shing-Chi Cheung. 2020. SemMT: A Semantic-based Testing Approach for Machine Translation Systems. *CoRR* abs/2012.01815 (2020). arXiv:2012.01815 https://arxiv.org/abs/2012.01815

[4] Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust Neural Machine Translation with Doubly Adversarial Inputs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 4324–4333. https://www.aclweb.org/anthology/P19-1425/

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *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)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). arXiv:2010.11929 https://arxiv.org/abs/2010.11929

[7] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 31–36. https://doi.org/10.18653/v1/P18-2006

[8] Silvia P Gennari, Maryellen C MacDonald, Bradley R Postle, and Mark S Seidenberg. 2007. Context-dependent interpretation of words: Evidence for interactive neural processes. *Neuroimage* 35, 3 (2007), 1278–1286.

[9] Carlo Giglio and Richard Caulk. 1965. Article 17 of the Treaty of Uccialli. *Journal of African History* (1965), 221–231.

[10] Google. 2021. Google Translate. http://translate.google.com.

[11] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based Adversarial Attacks against Text Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 5747–5757. https://aclanthology.org/2021.emnlp-main.464

[12] Shashij Gupta, Pinjia He, Clara Meister, and Zhendong Su. 2020. Machine translation testing via pathological invariance. In *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*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 863–875. https://doi.org/10.1145/3368089.3409756

[13] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567* (2018).

[14] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiaodong Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *24th International Symposium on High-Performance Computer Architecture (HPCA 2018), February 24-28, Vienna, Austria*.

[15] Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-invariant testing for machine translation. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 961–973.

[16] Pinjia He, Clara Meister, and Zhendong Su. 2021. Testing Machine Translation via Referential Transparency. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 410–422.

[17] Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef van Genabith. 2018. How Robust Are Character-Based Word Embeddings in Tagging and MT Against Wrod Scramlbing or Randdm Nouse?. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas, AMTA 2018, Boston, MA, USA, March 17-21, 2018 - Volume 1: Research Papers*. 68–80. https://aclanthology.info/papers/W18-1807/w18-1807

[18] Yue Jia and Mark Harman. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37, 5 (September–October 2011), 649 – 678.

[19] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 8018–8025. https://aaai.org/ojs/index.php/AAAI/article/view/6311

[20] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 6193–6202. https://doi.org/10.18653/v1/2020.emnlp-main.500

[21] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for End-to-End Aspect-based Sentiment Analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*. 34–41.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[23] Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[24] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, Qun Liu and David Schlangen (Eds.). Association for Computational Linguistics, 119–126. https://doi.org/10.18653/v1/2020.emnlp-demos.16

[25] Mike Papadakis, Marinos Kintis, Jie Zhang, Yue Jia, Yves Le Traon, and Mark Harman. 2019. Mutation testing advances: an analysis and survey. In *Advances in Computers*. Vol. 112. Elsevier, 275–378.

[26] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[27] Daniel Pesu, Zhi Quan Zhou, Jingfeng Zhen, and Dave Towey. 2018. A Monte Carlo Method for Metamorphic Testing of Machine Translation Services. In *3rd IEEE/ACM International Workshop on Metamorphic Testing, MET 2018, Gothenburg, Sweden, May 27, 2018*. ACM, 38–45. http://ieeexplore.ieee.org/document/8457612

[28] Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021. Turn the Combination Lock: Learnable Textual Backdoor Attacks via Word Substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 4873–4883. https://doi.org/10.18653/v1/2021.acl-long.377

[29] SpaCy. 2019. SpaCy. https://spacy.io/.

[30] Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*.

[31] Liqun Sun and Zhi Quan Zhou. 2018. Metamorphic testing for machine translations: MT4MT. In *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE, 96–100.

[32] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic testing and improvement of machine translation. In *ICSE*. 974–985. https://doi.org/10.1145/3377811.3380420

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 6000–6010.

[34] Wikipedia. 2014. Wikipedia. https://dumps.wikimedia.org/.

[35] WMT. 2018. News-Commentary. http://data.statmt.org/wmt18/translation-task/.

[36] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[37] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level Textual Adversarial Attacking as Combinatorial Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 6066–6080. https://doi.org/10.18653/v1/2020.acl-main.540

[38] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine Learning Testing: Survey, Landscapes and Horizons. *arXiv preprint arXiv:1906.10742* (2019).

[39] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating Natural Adversarial Examples. *CoRR* abs/1710.11342 (2017). arXiv:1710.11342 http://arxiv.org/abs/1710.11342

[40] Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2019. BERT-based lexical substitution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3368–3373.