

Machine Translation Testing via Syntactic Tree Pruning

QUANJUN ZHANG, State Key Laboratory for Novel Software Technology, Nanjing University, China
 JUAN ZHAI, Manning College of Information & Computer Sciences, University of Massachusetts Amherst, USA

CHUNRONG FANG*, State Key Laboratory for Novel Software Technology, Nanjing University, China
 JIAWEI LIU, State Key Laboratory for Novel Software Technology, Nanjing University, China
 WEISONG SUN, State Key Laboratory for Novel Software Technology, Nanjing University, China
 HAICHUAN HU, State Key Laboratory for Novel Software Technology, Nanjing University, China
 QINGYU WANG, State Key Laboratory for Novel Software Technology, Nanjing University, China

Machine translation systems have been widely adopted in our daily life, making life easier and more convenient. Unfortunately, erroneous translations may result in severe consequences, such as financial losses. This requires to improve the accuracy and the reliability of machine translation systems. However, it is challenging to test machine translation systems because of the complexity and intractability of the underlying neural models. To tackle these challenges, we propose a novel metamorphic testing approach by syntactic tree pruning (STP) to validate machine translation systems. Our key insight is that a pruned sentence should have similar crucial semantics compared with the original sentence. Specifically, STP (1) proposes a core semantics-preserving pruning strategy by basic sentence structures and dependency relations on the level of syntactic tree representation; (2) generates source sentence pairs based on the metamorphic relation; (3) reports suspicious issues whose translations break the consistency property by a bag-of-words model. We further evaluate STP on two state-of-the-art machine translation systems (i.e., Google Translate and Bing Microsoft Translator) with 1,200 source sentences as inputs. The results show that STP accurately finds 5,073 unique erroneous translations in Google Translate and 5,100 unique erroneous translations in Bing Microsoft Translator (400% more than state-of-the-art techniques), with 64.5% and 65.4% precision, respectively. The reported erroneous translations vary in types and more than 90% of them are not found by state-of-the-art techniques. There are 9,393 erroneous translations unique to STP, which is 711.9% more than state-of-the-art techniques. Moreover, STP is quite effective in detecting translation errors for the original sentences with a recall reaching 74.0%, improving state-of-the-art techniques by 55.1% on average.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**.

Additional Key Words and Phrases: Software testing, Machine translation, Metamorphic testing

*Chunrong Fang is the corresponding author.

Authors' addresses: [Quanjun Zhang](mailto:quanjun.zhang@smail.nju.edu.cn), quanjun.zhang@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093; [Juan Zhai](mailto:juan.zhai@rutgers.edu), juan.zhai@rutgers.edu, Manning College of Information & Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA, 01003; [Chunrong Fang](mailto:fangchunrong@nju.edu.cn), fangchunrong@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093; [Jiawei Liu](mailto:jw.liu@smail.nju.edu.cn), jw.liu@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093; [Weisong Sun](mailto:weisongsun@smail.nju.edu.cn), weisongsun@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093; [Haichuan Hu](mailto:181250046@smail.nju.edu.cn), 181250046@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093; [Qingyu Wang](mailto:wangqingyu2012@gmail.com), wangqingyu2012@gmail.com, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China, 210093.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

1049-331X/2023/0-ART1 \$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

ACM Reference Format:

Quanjun Zhang, Juan Zhai, Chunrong Fang, Jiawei Liu, Weisong Sun, Haichuan Hu, and Qingyu Wang. 2023. Machine Translation Testing via Syntactic Tree Pruning. *ACM Trans. Softw. Eng. Methodol.* 0, 0, Article 1 (2023), 39 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Machine translation aims to translate source content into target languages automatically. Due to the advent of neural machine translation models, the performance of machine translation has been improved significantly. In particular, some advanced machine translation systems are approaching human-level performance in terms of quality score and human parity [19, 62]. More and more people are getting used to employing machine translation systems in their daily lives, such as reading articles from other countries. For example, Google Translate [16] attracts more than 500 million users around the world and translates more than 100 billion words per day [57].

Similar to traditional systems, neural machine translation systems are not perfect and suffer from unreliable results sometimes. For example, they can produce erroneous target outputs when source inputs are adversarially manipulated (e.g., uppercasing some characters or injecting grammatical noise in a sentence) [2, 15]. These adversarial inputs are usually syntactically wrong. Besides, there exist many cases where machine translation systems may return erroneous translations for syntactical and semantical inputs (e.g., an industrial case by WeChat) [76].

However, there are several challenges to test machine translation systems. Firstly, testing traditional systems significantly differs from testing DNN-based systems in general due to the programming paradigm [74]. In the traditional system, the decision logic is manifested in source code. In contrast, the output of DNN-based systems depends mainly on the millions of parameters, which are optimized through training. Secondly, recent testing approaches for DNN-based systems mainly focus on models with a small number of possible outputs (e.g., image classifiers). Rather, it is an intractable problem for machine translation to enumerate all possible outputs [44], making machine translation systems challenging to test. Thirdly, most existing machine translation testing techniques [18, 21, 54] generate test cases (i.e., generated source sentence) by replacing one word in the input (i.e., original source sentence) based on some pre-trained language representation models (i.e., BERT and Spacy). Thus, the testing performance is mainly limited by the maturity of the adopted language model [22] and huge device resources are required in the deployment [51].

To address the above challenging problems, we introduce syntactic tree pruning (STP) testing, a novel and general approach for evaluating machine translation systems. The core idea of STP is inspired by the phenomenon that machine translation systems usually perform significantly better on the simple sentence than on the complex sentence [52]. Thus, people prefer to conduct source sentence simplification to get accurate translation results in practice [12]. The key insight is that eliminating contextual information from a source sentence should not influence the translation results of the trunk [38, 43]. To realize this concept, inspired by the linguistic rhetorical structure theory [38] that specifies each sentence component as either a basic nucleus or a context satellite, STP generates new source sentences by removing words or phrases from an original source sentence without losing core semantics and undermining the sentence validity. In particular, STP first performs a novel semantics-preserving pruning strategy by extracting crucial semantics via the basic sentence structure and designing pruning operators via dependency relations. Then the original and the newly generated source sentences are paired via the defined metamorphic relation. Besides, a bag-of-words model is adopted to measure the consistency of the source sentence pair (i.e., original and generated source sentence), and a suspicious issue will be reported if the translation results have a significant difference in core semantics via a pre-defined threshold value.

We conduct an empirical study to evaluate the effectiveness of STP on two state-of-the-art translators (i.e., Google Translate and Bing Microsoft Translator) with 1,200 English sentences from ten major article categories as inputs [16, 40]. The experimental results show that STP successfully reveals 5,073 and 5,100 erroneous unique translations in Google Translate and Bing Microsoft Translator with high precision (i.e., 64.5% and 65.4%), respectively. The detected erroneous translations are 400% more than state-of-the-art techniques under all dataset categories. We also demonstrate that the precision can be further improved under a large threshold value t . For example, STP with $t = 6$ achieves 80.9% and 84.3% precision for both translators, outperforming the most recent technique CAT by 19.8% and 18.2% with a comparable amount of erroneous issues. The types of reported erroneous translations are diversified, including under-translation, over-translation, word/phrase mistranslation, incorrect modification, and unclear logic. Compared with state-of-the-art techniques, STP is able to report more erroneous translations with higher precision. Due to its conceptual difference, STP reveals many erroneous translations that have not been found by existing techniques. For example, there are 4,700 and 4,692 erroneous translations unique to STP in Google Translate and Bing Microsoft Translator, which are 227% and 221% more than the most recent technique CAT. STP also achieves a higher recall than state-of-the-art techniques by 53.4% and 56.8% on average for original sentence errors when testing Google Translate and Bing Microsoft Translator, respectively. Besides, STP spends 0.09 seconds in sentence generation and 0.02 seconds in erroneous translation detection, achieving comparable efficiency to state-of-the-art techniques. All the reported issues and source code have been released for further research.

The main contributions of this paper are as follows.

- **Novel Technique.** We introduce a novel and widely applicable methodology, *syntactic tree pruning* (STP), to validate machine translation systems by generating sentences with different syntactic structures.
- **Practical Implementation.** We describe a practical implementation that generates new sentences via the proposed core semantic-preserving pruning strategy, and detects erroneous translations by measuring the semantics consistency of the source sentence pair, which is created by the designed metamorphic relation.
- **Extensive Study.** We evaluate STP on 1,200 sentences against five state-of-the-art techniques for Google Translate and Bing Microsoft Translator. To the best of our knowledge, this is the largest empirical evaluation for machine translation testing. The results demonstrate that STP successfully finds 5,073 erroneous translations in Google translate and 5,100 in Bing Microsoft Translator with high precision, most of which cannot be found by state-of-the-art techniques. On average, the recall of original sentence errors reaches 74%, which is 55.1% higher than that of state-of-the-art techniques.
- **Available Artifacts.** We release all experimental data (including the raw data, source code, and result analysis) for replication and future research on machine translation testing [72].

The rest of this paper is organized as follows. Section 2 reviews some background information and presents a motivation example. Section 3 introduces the proposed approach design. Section 4 presents the research questions, and explains the details of the empirical study. Section 5 provides the detailed results of the study and answers the research questions. Section 6 presents some additional discussion, and Section 7 discloses the threats to validity of our experiments. Section 8 discusses some related work. Section 9 presents the conclusions and discusses future work.

2 BACKGROUND & MOTIVATION

2.1 Basic Sentence Structure

In linguistics, given a source sentence, every word serves a specific purpose within the structure [24]. Sentence structure can sometimes be quite complicated according to the grammar rules. In syntax, four types of sentence structures are distinguished: simple sentences, compound sentences, complex sentences and compound-complex sentences [47]. Each sentence is defined by the usage of independent and dependent clauses, conjunctions and subordinators. The type of sentence is determined by how many clauses, or subject-verb groups are included in the sentence.

Although the sentences are diverse in types, a complex linguistic structure sentence can be converted into a set of simple sentences, with each of which presents a simpler and more regular structure that is easier to process by machine translation systems [39]. A simple sentence is a sentence that comprises exactly one independent clause, i.e., a group of words that has both a subject (S) and a verb (V), optionally an indirect or direct object (O), and complement (C). In linguistic typology, most simple sentences are derived based on the five basic clause types: subject-verb (SV), subject-verb-object (SVO), subject-verb-complement (SVC), subject-verb-indirect object-direct object (SVOO), subject-object-complement (SVOc) [36]. However, simple sentences may still include optional constituents that render them overly complex. For instance, the adverbial “in Princeton” in the simple sentence “Albert Einstein died in Princeton.” specifies additional contextual information that can be left out without producing ill-formed output. Rather, the remaining clause “Albert Einstein died.” still carries semantically meaningful information.

The constituents that belong to the clause type are essential components of the corresponding simple sentence, all other constituents are optional and can be discarded without leading to an incoherent or semantically meaningless output. In this paper, given a source sentence, we attempt to identify and simplify the sentence types, and then extract the basic clauses as the crucial semantics.

2.2 Syntactic Tree Representation

A syntactic tree is a tree representation of different syntactic categories of a sentence and helps to understand the syntactic structure of a sentence. There are two common syntactic tree structures: constituency syntactic and dependency syntactic[39], which are illustrated in Figure 1. Given a source sentence, a constituency syntactic tree presents a set of constituency relations, which shows how a word or group of words form different units within a sentence, while a dependency syntactic tree presents a set of relations describing the direct relationships between words rather than how words constitute a sentence. Dependency syntactic trees represent the grammatical relations that hold between constituents. Compared to constituency syntactic trees, dependency syntactic trees are more abstract, as they do not restrict or prescribe a particular word order. They are more specific in terms of semantics, and the notion of relations across words is explicit.

A dependency tree for a sentence is a directed acyclic graph with words as nodes and relations as edges. Each word in the sentence either modifies another word or is modified by a word. The root of the tree is the only entry that is modified but does not modify anything else. The relation that the two words participate in is given as a name on the edge connecting the nodes. More formally, the dependency structure tree can be expressed as follows: given a sentence $S = \{w_0, \dots, w_n\}$, a set of edges or dependencies $E = \{e_1, \dots, e_n\}$ are defined such that each e_i connects two words in the sentence, and w_0 is a root word that only connects a word to another word.

2.3 A Motivating Example

In this section, we will present a real-world erroneous translation example shown in Figure 2 and illustrate how it is detected by syntactic tree pruning. As a non-native English speaker, Echo often

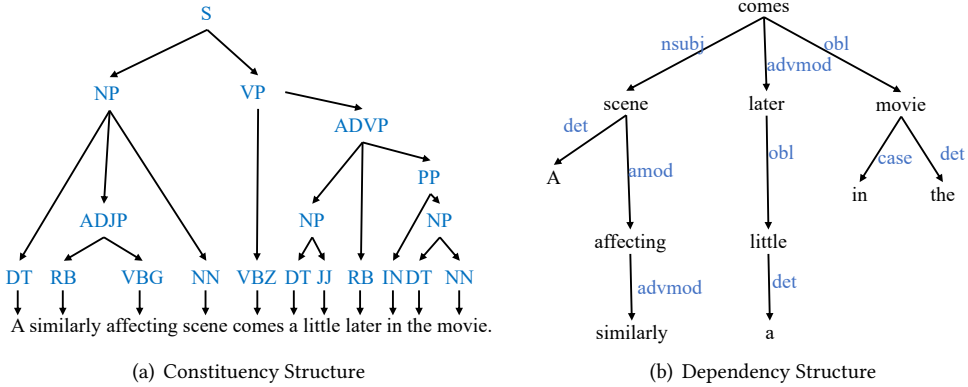


Figure 1. Syntactic tree representation

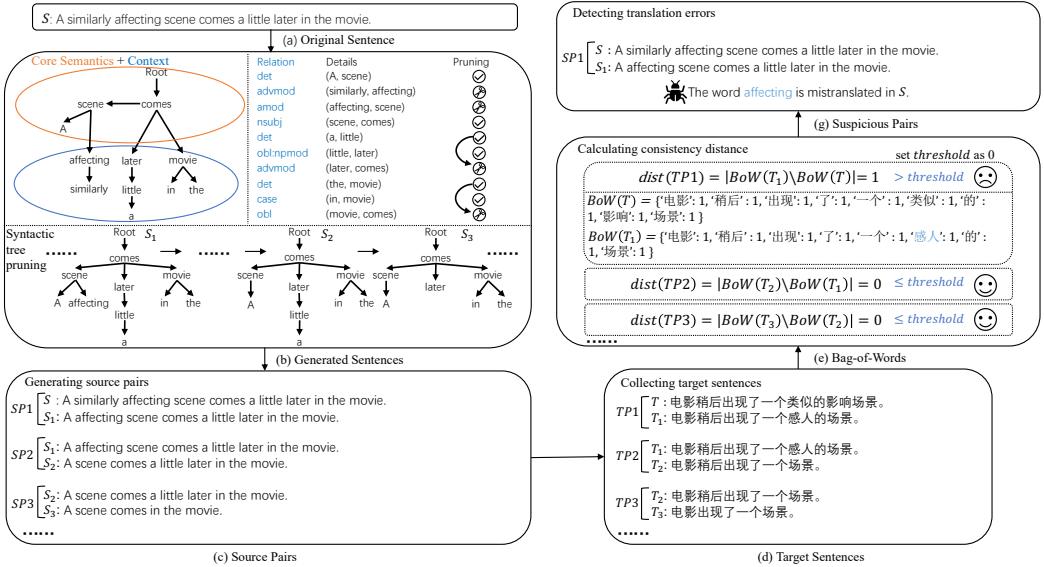


Figure 2. Overview of STP

adopts machine translation systems to read news from other countries. Echo reads a review article about the movie “Single All the Way” from CNN news website¹, and sees the the following English sentence (i.e., S in Figure 2):

A similarly affecting scene comes a little later in the movie.

To figure out its meaning, Echo uses Google Translate, a popular translation service powered by neural machine translation models. Google Translate returns a corresponding Chinese sentence (i.e., T in Figure 2).

¹<https://edition.cnn.com/2021/12/02/entertainment/single-all-the-way-race-deconstructed-newsletter/index.html>. Accessed August, 2022.

电影稍后出现了一个类似的影响场景。

However, Echo finds he misunderstands the article because of the incorrect translation returned by Google Translate. That is a real-world erroneous translation that leads to a confusing, unpleasant reading experience. The correct translation should be:

电影稍后出现了一个类似的感人场景。

It is difficult for most existing techniques to find the translation error. For example, the most recent technique CAT generates new sentences by replacing some words with other semantically similar words (e.g., S' : “comes” → “occurs”, S'' : “comes” → “appears” and S''' : “movie” → “film”). However, CAT fails to detect the erroneous translation as the generated sentences have the same translation as the original sentence. Similar to CAT, SIT also performs word replacement to form a sentence pair (e.g., “scene” → “moment”). The pair is considered to have an erroneous translation if a large difference exists between the structures of translations in the pair. However, Google Translate returns the translations with the same sentence structures and the erroneous translation is also beyond the scope of SIT.

In fact, more and more people are getting used to relying on machine translation systems in their daily lives. The robustness of the machine translation system is particularly crucial in a practical scenario. It is observed that machine translation systems often make mistakes for complex sentences and perform well for simple sentences [52]. Inspired by this, we propose to extract the core semantics of a sentence and leverage the pruned sentence to test machine translation systems. For the example sentence mentioned above, STP first performs syntactic tree scanning to identify “A scene comes a little later” as the core semantics and the remaining words as the context (shown in the upper left part of Figure 2(b)). Then STP generates new sentences by removing context words in turn, which are then used to compare with the original sentence (shown in the upper right part of Figure 2(b)). For example, in Figure 2(c), S_1 can be generated by removing the adjunct “similarly”, and the translation should not influence the trunk of the original sentence. We observe their translations T and T_1 have different meanings, which means the sentence pair $\langle S, S_1 \rangle$ has erroneous translations. In fact, the word “affecting” is mistranslated in the original sentence S and some generated sentences S' , S'' and S''' with the same sentence structure. However, the word is translated correctly if the adjunct “similarly” is removed. Thus, in this work, we attempt to eliminate contextual information from the original source sentence without influencing the core semantics meaning (i.e., basic sentence structure) and propose STP, a novel syntactic tree pruning methodology for testing machine translation systems with arbitrary source sentences.

3 APPROACH AND IMPLEMENTATION

This section introduces syntactic tree pruning testing (STP) and describes our practical implementation. In general, STP takes an unlabeled original source sentence as input at a time and outputs a list of suspicious issues. Figure 2 presents the overview of STP implementation. STP carries out the following steps:

- (1) *Core semantics-preserving pruned sentences generation.* For each unlabelled original source sentence, we generate a list of new source sentences by pruning the context at syntactic tree representation (shown in Figure 2(b)).
- (2) *Metamorphism-based source pair generation.* We pair each source sentence with all the new sentences generated from it to form source sentence pairs (shown in Figure 2(c)).

- (3) *Consistency-based translation error detection.* The core semantics of the translated generated sentences is compared to the core semantics of the translated original sentence via a bag-of-words model (shown in Figure 2(d) and Figure 2(d)(e)). If there is a large difference between the core semantics, STP reports a potential translation issue (shown in Figure 2(g)).

Algorithm 1 presents the process of syntactic tree pruning. The algorithm takes a list of arbitrary sentences as input and returns a list of suspicious pairs as output. The main implementation process of this algorithm is first to establish a syntactic tree representation (line 3-4) for each original sentence by a recent neural network-based parser implemented in Stanford CoreNLP library [17] and then generate a list of new sentences via a novel core semantics-preserving pruning strategy (lines 22-43). Then the source sentence pairs are fed to the machine translation systems based on the metamorphic relation (lines 7 -14), which are used to report suspicious translation errors via consistency detection (lines 17-21).

3.1 Core Semantics-preserving Pruned Sentences Generation

Given an arbitrary sentence in the source language, STP aims to generate new syntactically and semantically valid sentences by removing words or phrases from the original sentence. However, it is not trivial to choose words or phrases from a sentence without missing any crucial semantics and even undermining the sentence validity. A naive implementation may be done by recursively removing a word or phrase until a predefined threshold (e.g., the minimum length of the generated sentence). Although this implementation surely can generate all possible valid pruned sentences, it leads to too many candidate sentences with a low acceptability rate. For example, the sentence in Section 2.3 has nine words and we can generate 604,800 ($10 * 9 * 8 * 7 * 6 * 5 * 4$) new sentences if the predefined threshold is set to 3, as each word removal leads to a new sentence. In the real world, the sentences fed to the machine translation systems tend to be more complex (e.g., more than 19 words on average for the dataset [21]), and this implementation is unaffordable in practice (discussed in Section 6.6).

Determining which word to remove from the original sentence can be aided by understanding the semantics of the word. Inspired by the advance in natural language processing (NLP), several semantics-based tasks (e.g., sentence simplification or compression) may be well suited for this task [30, 52]. Specifically, given a source sentence, we extract semantics relations between words or phrases on its syntactic tree structure. As a part of the grammar, syntax refers to the set of rules in a natural language that governs the structure of a sentence, determining how each component, such as words, phrases, and clauses, forms into their super-ordinate components, until the formation of the sentence [9]. When removing a word from the sentence, the effect of the removal operation on other words should be considered. Thus, the dependency syntactic tree is employed in this paper, as it effectively reflects the changes in how words interact [75].

However, after the given sentence is represented as a dependency syntactic tree, several challenges remain in the pruning process. Now we discuss several problems of generating new sentences by removing irrelevant content while preserving core semantics on the syntactic tree level. Firstly, it is not trivial to define the core semantics and its contextual information for a source sentence. Natural languages often have complex syntactic structures, and it is essential to consider which dependencies can be used as the key semantics of a sentence. Mann et al. [38] describe a rhetorical structure theory (RST), which specifies each sentence component as either a nucleus or a satellite. The nucleus component embodies the central piece of information, whereas the role of the satellite is to specify the nucleus further. In linguistics, most sentences are combined by basic clause patterns (e.g., subject-predicate-object) and adjuncts (e.g., the attributive, adverbial). The former as nucleus

can express the sentence meaning clearly and the latter as the satellite is used to express the meaning more accurately. Then, we define the core semantics and context as follows.

Definition 3.1. Core semantics and context: Given a sentence $S = \{S_e, S_c\}$ where S_e is a set of words denoting the core semantics and S_c is the remaining words denoting the associated context. A $s_e \in S_e$ is a triple $(s, v, o) \in BT$, where $BT = \{SV, SVO, SVC, SVOO, SVOC\}$ represents the set of basic sentence structures. Hence, $s \in S$ denotes a subject, $v \in V$ a verb and $o \in \{O, C, OO, OC\}$ denotes a direct or indirect object, or complement.

For example, in Figure 2(b), the nodes circled in orange represent the core semantics and the nodes circled in blue represent the context. In that way, the source sentence is reduced to the basic structure as the core sentence and augmented with some adjuncts that disclose associated contextual information. On syntactic tree representation, the basic clause patterns are the trunk of the sentence, and the adjuncts are the branches of the sentence.

Secondly, the number of words that can be removed is difficult to determine at each step, as some words are dependent, such as “a little later” in Figure 2. Although we could adopt a strategy of removing one word at each case recursively, this approach would result in a large number of syntactically invalid sentences. For example, we cannot only remove the word “later” from the example sentence “A similarly affecting scene comes a little later in the movie”. When removing such words, we have three possible situations: (1) Unprunable. The word constitutes the five basic clause types. The deletion of these words will result in the absence of core semantics. (2) Prunable. The word does not constitute the five basic clause types and the removal operation of the word does not affect the validity of the sentence. (3) Partially-Prunable. The word does not constitute the five basic clause types and only the removal operation of the word and other words it depends on does not affect the validity of the sentence.

To address the above challenges, we perform a four-step cascaded pruning strategy: sentence type identification, sentence simplification, core semantics extraction and pruned sentence generation. (1) STP performs a syntactic tree scanning and divides the given original sentence into four types, i.e., simple, compound, complex and compound-complex sentences based on the edges connecting the nodes. The type of sentence is determined by how many clauses, or subject-verb groups, are included in the sentence. (2) After determining the sentence type, STP attempts to split the compound and complex sentence into simple sentences. For example, a compound sentence will be split according to the co-dependency relation (i.e., *cc* edge). The sub-trees before and after the edge are divided into new sentences at the same time, and the process is recursive until no compound sentence exists. The noun clause in complex sentences will also be split according to the dependency relation, and the new sentence after splitting only shares conjunctions. The remaining clauses in complex sentences will not affect the core semantics of the sentence, so they will be pruned. (3) For each simple sentence, STP identifies the shortest path among the syntactic tree, which is used to represent the crucial semantics of the original sentence. As most sentences are derived from the five basic sentence structures (i.e., subject-verb, subject-verb-object, subject-verb-complement, subject-verb-object-object, subject-verb-object-complement), the nodes constituting the structures are considered as the shortest path. Thus, although the generated sentence has a different structure from the original one, they share the same basic structure. (4) STP performs a removal operation for each node according to the edges (i.e., dependencies) until the shortest path remains. Among the nodes, there are some constraints between them. In the syntactic tree, when pruning a specific node, its dependent node also needs to be considered. We consider all available dependency types implemented in Stanford’s CoreNLP library² and manually analyze

²English dependencies description. https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf. Accessed August, 2022.

Algorithm 1 Implementation of STP**Input:** *source_sents*: a list of sentences in the source language**Output:** *suspicious_issues*: a list of suspicious pairs

```

1: suspicious_issues  $\leftarrow$  List()
2: for each source_sent  $\in$  source_sents do
3:   dependency_tree  $\leftarrow$  PARSE(source_sent)
4:   head  $\leftarrow$  dependency_tree.head()
5:   gen_sents  $\leftarrow$  PRUNING(source_sent, dependency_tree, head)
6:   target_source_sent  $\leftarrow$  TRANSLATE(source_sent)
7:   for each gen_sent  $\in$  gen_sents do
8:     target_gen_sent  $\leftarrow$  TRANSLATE(gen_sent)
9:     if DISTANCE(target_source_sent, target_gen_sent) > d then
10:      source_pair  $\leftarrow$  {source_sent, gen_sent}
11:      target_pair  $\leftarrow$  {target_source_sent, target_gen_sent}
12:      suspicious_issues.add(source_pair, target_pair)
13:     end if
14:   end for
15: end for
16: return suspicious_issues

17: function DISTANCE(target_source_sent, target_gen_sent)
18:   source_BOW  $\leftarrow$  BAGOFWORDS(target_source_sent)
19:   gen_BOW  $\leftarrow$  BAGOFWORDS(target_gen_sent)
20:   return |source_BOW  $\setminus$  gen_BOW|
21: end function

22: function PRUNING(source_sent, dependency_tree, head)
23:   if len(head.children()) > 0 then
24:     nodes  $\leftarrow$  head.children()
25:     dep_nodes  $\leftarrow$  DEP_PRIOR(dependency_tree) ▷ prioritize the nodes in the dependency tree
26:     relation_set  $\leftarrow$  RELATION() ▷ predefined operation set
27:     for each node  $\in$  dep_nodes do
28:       source_sent  $\leftarrow$  REMOVE(source_sent, relation_set, node)
29:       if len(node.children()) > 0 then
30:         PRUNING(source_sent, dependency_tree, node)
31:       end if
32:       if node.is_removed() then
33:         gen_sents.add(source_sent)
34:       end if
35:     end for
36:   else
37:     source_sent  $\leftarrow$  REMOVE(source_sent, relation_set, head) ▷ determine whether to delete the node according
38:     if head.is_removed() then
39:       gen_sents.add(source_sent)
40:     end if
41:   end if
42:   return gen_sents
43: end function

```

the dependency structure trees on 1000 sentences randomly selected from the news [21]. We then determine which of the above possible situations each dependency type belongs to and design the corresponding pruning operator. Specifically, the unprunable situation maps *up* operator, which denotes the node should be retained, as it belongs to the five basic clause types. The prunable situation maps *pr* operator, which denotes the node can be deleted, as it does not belongs to the

five basic clause types and not affect the validity of the sentence. The partially-prunable situation maps pp , which denotes the node should be deleted with its dependent node simultaneously, as only deleting both of them does not affect the validity of the sentence. With these corresponding operators, we establish a mapping table to determine whether the dependent node needs to be cascaded. Details of the dependencies and the corresponding map table are presented in Table 1. Table 1 contains four main columns, each of which has two sub-columns. The first sub-column lists the dependency types. The second sub-column lists the three types of pruning operators (i.e., up , pr and pp). Through the mapping table, we can determine the least nodes that can be pruned each time. For example, given a simple sentence S in Figure 2, we first identify the crucial semantics (i.e., the circled part) and then iterate over all nodes to determine if they should be deleted based on Table 1.

Table 1. Dependency relation mapping table.

Relation	Type	Relation	Type	Relation	Type	Relation	Type
ROOT	up	iobj	up	predet	pr	poss	pr
dep	pp	nsubj	up	preconj	pr	prt	pp
aux	pp	dobj	up	mwe	pp	compound	pr
auxpass	pp	det	pp	mark	pp	goeswith	pp
cop	up	expl	pp	advmod	pr	ref	pp
ccomp	pp	amod	pr	neg	pr	xsubj	up
xcomp	pp	nmod	pr	tmod	pr	case	pp
obj	up	nummod	pp	punct	up	obl	pr

3.2 Metamorphism-based Source Pair Generation

Once all possible pruned sentences have been generated, they should be translated into the target language for semantics validation (Section 3.3). STP is designed to perform the metamorphic testing for the machine translation systems M . The most significant advantage of metamorphic testing lies in its capability of identifying oracle information for the tests via metamorphic relations (MR). Given a source sentence set S and the pruning operators P , the MR is defined as a set of sentences that can be derived from each sentence $s \in S$ without influencing the crucial information. This MR to test M with additional transformed sentences can be formalized as follows:

$$\forall s \in S \wedge \forall p \in P, \quad M(s) = M(p(s)) \quad (1)$$

Given this MR, we can simply obtain the oracle information by verifying whether it is satisfied in the testing process of M . Under this setting, each generated sentence must be paired with the original sentence, which will be used to check whether the relation is satisfied or violated. If a violation is detected, we can then say that M is faulty. Specifically, each source sentence pair should have two different pieces of text that contain the same phrase. To generate these pairs, we pair each source sentence (i.e., original and generated) with all the generated sentences pruned from the source sentence. For example, as illustrated in Figure 2, source sentence $S1$ is pruned from the original source sentence S by removing the word “similarly”. Meanwhile, new source sentence $S2$ can also be pruned from the generated source sentence $S1$ by removing the phrase “affecting”. Thus, three source sentence pairs will be constructed: (1) the original sentence S and the generated source sentence $S1$; (2) the generated sentence $S1$ and the generated source sentence $S2$; and (3) the generated sentence $S2$ and the generated source sentence $S3$.

3.3 Consistency-based Translation Error Detection

After pruned source sentences have been generated, we feed them and the corresponding original source sentences to the machine translation systems under test. For each original source sentence and all generated source sentences, each machine translation system takes a source sentence and a target language (i.e., Chinese) as inputs, and then returns a translation sentence in a target language.

Once the target sentences are collected, we need to detect erroneous translations. It is non-trivial to detect whether a pruned sentence preserves the core semantics of the original sentence without reporting false positives. Given an original source sentence $E = \{e_1, e_2, \dots, e_n\}$, we generate a corresponding pruned source sentence $E' = \{e'_1, e'_2, \dots, e'_m\}$ by removing several words, and their translations are $T(E)$ and $T(E')$. It is challenging to map the relation between the source language pair (i.e., E and E') to the target language pair (i.e., $T(E)$ and $T(E')$). Machine translation systems may return a sentence with a different structure for imperceptible perturbations due to the brittleness of the neural network. Thus, following existing work [22], we adopt a bag-of-words (BoW) model, a simple representation disregarding grammar and even word order but keeping multiplicity, to capture the inconsistent semantics between the original and pruned sentences. Although simple, it has been proven quite effective in some NLP tasks, such as neural machine translation [22, 61]. For each translated pairs $T(E)$ and $T(E')$, the distance between $T(E)$ and $T(E')$ is calculated by

$$\text{dist}(T(E), T(E')) = |\text{BoW}(T(E')) \setminus \text{BoW}(T(E))| \quad (2)$$

where $\text{BoW}(T(E))$ and $\text{BoW}(T(E'))$ denote the BoW representation of $T(E)$ and $T(E')$. The \setminus operator denotes the set difference (i.e., how many word occurrences in $\text{BoW}(T(E'))$ but not in $\text{BoW}(T(E))$. For example, the distance between “A similarly affecting scene comes a little later in the movie” and “A affecting scene comes a little later in the movie” is 1. Following most existing machine translation testing work [18, 21, 22, 54, 55, 60, 76], we report suspicious issues via a pre-defined threshold value t . If the distance is larger than the threshold value t , the translated pair is considered to break the consistency property and will be reported as a suspicious issue. For example, $t = 0$ represents the distance of all reported issues is larger than the threshold 0, i.e., there exist new word occurrences in the pruned translation but not in the original translation.

The threshold value demonstrates the degree of inconsistency in core semantics between the original and pruned sentences, and controls the trade-off between the precision and the number of reported suspicious issues. When we increase the threshold value t , there exist more new words introduced in the translation of the pruned sentence, leading to more accurate reported issues. Such a threshold setting is also adopted in previous work [18, 21] and proves its effectiveness in detecting translation errors. Specifically, for each original source sentence, STP reports either no issue or a list of suspicious issues. There may exist three possible translation error types in an issue: (1) the original source sentence has an erroneous translation, (2) the generated source sentence has an erroneous translation, and (3) both the source sentences have erroneous translations. A suspicious issue consists of the original source sentence, a generated source sentence, and their translations.

4 EXPERIMENTAL SETUP

4.1 Research Questions

Our main research questions are as follows.

- RQ1.** How precise is STP at finding erroneous issues?
- RQ2.** How many erroneous translations does STP report?
- RQ3.** What type of erroneous translations does STP report?
- RQ4.** How efficient is STP in terms of running time?

Table 2. Statistics of input sentences for evaluation

Corpus	# of words/sentence	Average # of words/sentence	Total # of words	Distinct # of words
Politics	12-69	23.82	2,382	1,089
Business	11-49	25.85	2,585	1,109
Culture	11-49	24.02	2,402	1,160
Sports	11-49	23.83	2,383	1,035
Tech	11-38	21.17	2,117	1,009
Travel	10-63	21.74	2,174	1,018
Health	11-47	22.30	2,230	877
Life	11-46	24.88	2,488	1,112
Legal	11-52	23.74	2,374	967
Opinion	11-48	22.07	2,207	796
Politics*	4-32	19.20	1,918	933
Business*	4-33	19.50	1,949	944

4.2 Machine Translation Systems

Our experiment considers two state-of-the-art industrial machine translators. The former is Google Translate, and the latter is Bing Microsoft Translator [16, 40]. Both of them are widely used machine translation services developed by Google and Microsoft, and have been widely adopted in recent machine translation testing work [18, 21, 22, 54]. Specifically, we invoke the APIs provided by Google Translate and Bing Microsoft Translator to obtain the translation results, which are identical to those returned by their Web interfaces.

4.3 Dataset

Following previous machine translation testing work [18, 21], we collect real-world source sentences from the news. Specifically, we randomly collect the latest articles from CNN (Cable News Network) [10], China Daily [11], BBC (British Broadcasting Corporation) [1] and Reuters [48]. To evaluate whether STP consistently performs well on sentences of different semantic contexts, the articles are extracted from ten categories: Business, Culture, Lifestyle, Sports, Politics, Travel, Technology, Healthy, Opinion and Legal. For each category, we extract its main text contents, and split them into a list of sentences. Then we randomly select 100 syntactically and semantically correct sentences from each category. We also experiment on a widely-adopted dataset from previous machine translation testing work [18, 21, 22] containing 200 sentences. Specifically, the dataset consists of articles from two categories: Politics and Business, where each dataset contains 100 English sentences. In total, we have 1,200 sentences from ten major categories of news sites as the initial sentence corpus for STP and other baselines in the experiment. To the best of our knowledge, this is the largest evaluation dataset for machine translation testing so far.

Statistics of the dataset are illustrated in Table 2. The first column lists the twelve dataset categories, where the first ten rows denote the sentence collected in our work and the last two rows denote the sentences from He et al. [18, 21, 22]. The second column lists the minimum-maximum number of words and the third column lists the average number of words for all sentences in the category. Similarly, the fourth column lists the total number of words that appear in all sentences. The fifth column lists the number of non-repetitive words in all sentences. For example, sentences in Culture dataset contain 11-49 words (the average is 24.02 words) and they contain 2,402 words and 1,160 non-repetitive words in total.

4.4 Labelling

An issue contains a pair of source sentences, where the first one is the original sentence and the second is the generated sentence. To construct the labelling process, we invite 20 graduate students in our experiment. All of them have more than 10 years of English learning experience and are all native speakers of Chinese. Before the formal labelling process, we provide documents and presentation videos as a guide to all participants, to ensure they fully understand the labelling procedure. Specifically, the participants identify whether the source sentences in the reported issue contain any syntax or semantic error. If both source sentences are invalid in the issue, it is labeled as a false positive. Then, for the remaining issues, the participants check whether there is an erroneous translation for the valid source sentence(s). If so, the participants count the reported issue as a true positive. Otherwise, the reported issue is labeled as a false positive. Finally, the participants decide which source sentence(s) in this issue cause(s) the translation error(s).

As we manually label the issues, it is inevitable to introduce subjectivity. To minimize such subjectivity, we assign each issue to two different participants. When there exists a disagreement, all the participants would involve to have an open discussion to resolve it³. Moreover, we mix issues reported by our system and the ones from other systems, and thus the participants are unaware of whether an issue is reported using our system or not. Thus, we believe that the human labelling process can provide reliable ground truth information in our experiment.

4.5 Comparison

To evaluate the performance of STP, following the most recent work [55], we compare it with five state-of-the-art machine translation testing techniques: **SIT**, **TransRepair**, **RTI**, **PatInv** and **CAT**. To our knowledge, our work is the largest evaluation study on machine translation testing so far. We obtain the source code of the first four techniques from the TestTranslation toolkit [20] and CAT from the previous study [55].

Specifically, TransRepair adopts four similarity metrics (i.e., LCS-based metric, ED-based metric, tf-idf-based metric and BLEU-based metric) for measuring inconsistency. Following existing work [22], we select ED-based metric (referred to as TransRepair-ED) in our study because it achieves the highest precision and detects the second largest number of translation errors among four metrics on Google Translate. PatInv has two variants to generate sentences with a different meaning, i.e., replacing one word in a sentence with a non-synonymous word (referred to as PatInv-Replace) and removing a meaningful word or phrase from the sentence (referred to as PatInv-Remove). In this work, we select PatInv-Replace because it performs significantly better than PatInv-Remove in terms of precision and the number of erroneous translations. SIT adopts three different metrics (i.e., edit distance, constituency set distance, and dependency set distance) for evaluating the distance between target sentences. We select the dependency set distance metric (referred to as SIT-Dep) in our study as it has the best performance on top-1 precision for both Google Translate and Bing Microsoft Translator [21]. CAT identifies word replacement with isotopic replacement and adopts the same similarity metrics in TransRepair. Following the original work [55], we use LCS-based metric to represent CAT (referred to as CAT-LCS) in our evaluation.

4.6 Experimental Environments

All experiments (including STP and baselines) are conducted on Ubuntu 16.04 with 16GB RAM and 6 Intel Core i7-10710U CPUs. For sentence parsing, we adopt a shift-reduce parser [79] which is implemented in Stanford CoreNLP toolkit [17]. Our experiments consider the English → Chinese language setting because of the knowledge background of the authors.

³The Cohen's Kappa score is 0.93 on average.

5 EVALUATION

5.1 Precision on Finding Erroneous Issues

STP aims to automatically detect erroneous issues for machine translation systems using arbitrary sentences. Thus, we evaluate the effectiveness of STP from two aspects: (1) how many erroneous issue STP is able to report; and (2) the precision of the erroneous issue reported by STP.

5.1.1 Evaluation Metric. Given a list of unlabeled, monolingual source language sentences, the output of STP is a list of suspicious issues I . A suspicious issue i contains (1) an original sentence, E , in the source language, and its translation sentence, $T(E)$, in target language; (2) a pruned sentence, E' , in the source language, and its translation sentence, $T(E')$, in target language. We define the precision as the percentage of erroneous issues, where there exist translation error(s) in $T(E)$ or $T(E')$. Formally, for a suspicious issue p , we set $error(p)$ to *true*, if $T(E)$ or $T(E')$ has translation error(s), otherwise we set it to *false*. Given a list of suspicious issues, the precision is calculated by the number of erroneous issues divided by the total number of reported suspicious issues:

$$Precision = \frac{\sum_{p \in P} \mathbb{1}\{error(p)\}}{|P|} \quad (3)$$

where $\mathbb{1}$ denotes whether a suspicious issue p contains translation error(s) and $|P|$ denotes the number of suspicious issues returned by STP.

5.1.2 Results. The details of the comparison results are presented in Table 3. The first column lists the twelve dataset categories and two machine translation systems. The second column lists the precision results of STP under different thresholds t . STP reports the suspicious issue if the distance between the translated original sentence and translated pruned sentence is larger than a threshold t . For example, STP with $t = 0$ means there exist new word occurrences in the pruned sentence translation but not in the original sentence translation. Similarly, STP with $t = 12$ means the number of new word occurrences in the pruned sentence translation but not in the original sentence translation is larger than 12. The remaining columns list the precision results of the five compared techniques. We also present the total results calculated by the all twelve dataset categories in the middle part and bottom part of Table 3. Each cell is represented as $x(y)$, where x refers to the precision value (defined in Equation 3) and the y refers to the number of all issues reported by the studied techniques.

Table 3 shows when the threshold value is at its lowest (i.e., $t = 0$), STP is able to report 2,140 and 2,128 erroneous issues with a 64.5% and 65.4% precision on average for Google Translate and Bing Microsoft Translator. For example, when testing Google Translate with Business* dataset, STP reports 1,254 erroneous issues, while 1,094 of them contain translation errors. We also show STP results under different threshold values, which represent the degree of inconsistency in reported issues. In general, we will reasonably achieve less number of issues by setting a larger threshold value, but with a higher precision. For example, we can obtain 118 erroneous issues overall for Google Translate when $t = 10$, which achieves 27% precision improvement against $t = 0$. More importantly, we can achieve 100% precision with Politics* dataset for the both translators when $t > 4$.

We also compare STP under different threshold values against state-of-the-art techniques. For direct comparison, we focus on the top-1 (i.e., the translation that is most likely to contain errors) results of SIT, because SIT returns top- k results for each original sentence. In particular, the top-1 output of SIT contains (1) the original sentence and its translation and (2) the top-1 generated

Table 3. The precision of STP (# of erroneous issues/# of suspicious issues) under different threshold values

	STP												TransRepair	CAT
	$t = 0$	$t = 2$	$t = 4$	$t = 6$	$t = 8$	$t = 10$	$t = 12$	SIT	PathInv	RTI	TransRepair	CAT		
Google Politics	61.0%(1,410/2,310)	68.8%(645/937)	76.7%(207/270)	71.4%(35/49)	77.8%(7/9)	N.A.	N.A.	53.8%(7/13)	64.3%(36/56)	37.1%(49/132)	64.7%(11/17)	62.3%(134/215)		
Google Business	86.1%(2,895/3,364)	85.1%(1,238/1,455)	79.3%(298/376)	76.3%(58/76)	90.0%(18/20)	100.0%(8/8)	100.0%(6/6)	61.1%(11/18)	35.6%(21/59)	39.0%(55/141)	31.6%(12/38)	52.2%(109/209)		
Google Culture	60.5%(2,574/4,255)	59.6%(1,357/2,278)	61.5%(606/985)	62.2%(353/246)	67.5%(27/40)	80.0%(4/5)	N.A.	29.4%(5/17)	34.0%(17/50)	32.6%(61/187)	56.0%(14/25)	37.9%(89/235)		
Google Sport	55.7%(1,315/2,361)	58.6%(675/1,151)	60.7%(2,444/4,002)	80.1%(125/156)	93.4%(71/76)	100.0%(38/38)	100.0%(21/21)	44.0%(11/25)	73.7%(216/293)	48.3%(57/118)	70.2%(33/47)	35.3%(84/238)		
Google Tech	63.1%(1,283/2,033)	74.9%(590/788)	78.0%(206/264)	71.0%(44/62)	92.3%(12/13)	N.A.	N.A.	28.6%(4/14)	55.6%(15/27)	26.9%(21/78)	40.0%(6/15)	87.5%(182/208)		
Google Travel	67.2%(3,087/4,591)	73.2%(1,816/2,480)	80.1%(659/1,073)	86.6%(335/387)	90.6%(125/138)	90.6%(48/53)	95.5%(21/22)	22.2%(2/9)	68.8%(11/16)	30.3%(42/136)	55.6%(10/18)	69.7%(182/261)		
Google Health	51.0%(1,579/3,099)	60.3%(921/1,514)	69.8%(388/556)	84.9%(90/106)	100.0%(7/7)	100.0%(7/7)	N.A.	71.4%(10/14)	37.5%(15/40)	50.0%(29/58)	63.2%(12/19)	72.4%(152/210)		
Google Life	55.8%(1,837/3,295)	62.9%(1,098/1,745)	80.1%(447/558)	92.5%(149/161)	100.0%(26/26)	100.0%(3/3)	N.A.	27.8%(5/18)	46.2%(18/39)	37.0%(44/119)	35.5%(6/11)	63.7%(142/223)		
Google Legal	62.2%(1,848/2,971)	73.0%(1,069/1,464)	79.0%(459/581)	83.3%(169/203)	72.4%(42/58)	54.5%(6/11)	66.7%(2/3)	58.8%(10/17)	29.2%(14/48)	40.2%(39/97)	33.3%(6/18)	72.8%(139/191)		
Google Opinion	61.1%(1,553/2,541)	70.0%(700/1,000)	77.5%(1,222/222)	85.0%(51/60)	100.0%(17/17)	100.0%(2/2)	N.A.	43.8%(7/16)	41.3%(19/46)	38.4%(48/125)	55.6%(10/18)	51.2%(110/214)		
Google Politics*	82.5%(933/1,131)	89.0%(389/437)	97.6%(120/123)	100.0%(38/38)	100.0%(12/12)	100.0%(2/2)	N.A.	82.7%(43/52)	71.3%(77/108)	74.5%(73/98)	77.6%(83/107)	58.4%(125/214)		
Google Business*	87.2%(1,094/1,254)	94.5%(464/491)	97.3%(109/112)	100.0%(14/14)	N.A.	N.A.	N.A.	74.5%(38/51)	79.8%(75/94)	70.7%(58/82)	79.6%(74/93)	75.0%(168/224)		
Google Sum	64.5%(2,144/3,322)	69.6%(1,099/1,577)	74.5%(411/552)	80.9%(16.4)	88.0%(23.5)	91.5%(27.0)	96.2%(31.7)	58.0%(153/264)	61.0%(534/876)	42.0%(576/1,377)	63.2%(282/446)	61.1%(1,611/2,644)		
Bing Politics	72.9%(1,688/2,311)	75.5%(707/937)	75.6%(204/270)	75.5%(37/49)	55.6%(5/9)	N.A.	N.A.	61.5%(8/13)	41.4%(12/29)	40.0%(52/130)	68.8%(11/16)	62.6%(129/206)		
Bing Business	66.6%(2,122/3,199)	72.1%(941/1,300)	73.3%(217/296)	73.0%(27/37)	100.0%(5/5)	N.A.	N.A.	61.1%(11/18)	12.0%(6/50)	36.2%(59/163)	44.0%(11/25)	54.6%(112/205)		
Bing Culture	75.1%(3,122/4,166)	82.8%(1,755/2,122)	84.4%(777/921)	79.9%(270/338)	76.4%(68/89)	76.2%(16/21)	88.9%(8/9)	29.4%(5/17)	15.0%(3/20)	48.9%(86/176)	40.9%(9/22)	62.5%(140/224)		
Bing Sport	51.5%(1,199/2,322)	64.8%(712/1,099)	77.3%(307/397)	88.1%(126/143)	95.2%(59/62)	95.8%(23/24)	100.0%(6/6)	34.6%(9/26)	25.6%(23/90)	55.3%(73/132)	70.0%(21/30)	74.5%(146/196)		
Bing Tech	64.8%(1,333/2,066)	79.9%(609/762)	90.1%(1,922/213)	100.0%(49/49)	100.0%(7/7)	N.A.	N.A.	57.1%(8/14)	44.4%(4/9)	30.0%(21/70)	38.9%(7/18)	81.5%(181/222)		
Bing Travel	69.9%(3,188/4,555)	73.3%(1,800/2,466)	80.7%(620/1,011)	87.3%(261/299)	91.1%(72/79)	100.0%(9/9)	100.0%(2/2)	44.4%(4/9)	56.2%(9/16)	25.4%(29/114)	47.1%(8/17)	69.7%(157/226)		
Bing Health	51.1%(1,555/3,033)	53.8%(792/1,477)	55.2%(267/484)	64.4%(56/87)	50.0%(3/6)	N.A.	N.A.	78.6%(11/14)	32.4%(11/34)	59.7%(37/62)	64.7%(11/17)	70.6%(125/177)		
Bing Life	66.6%(2,122/3,199)	72.4%(1,066/1,477)	83.2%(356/428)	85.4%(76/89)	76.9%(10/13)	100.0%(1/1)	N.A.	50.0%(9/18)	25.9%(7/27)	37.0%(47/127)	50.0%(17/34)	55.0%(116/211)		
Bing Legal	60.7%(1,755/2,888)	70.0%(861/1,233)	84.9%(382/450)	92.8%(193/208)	96.1%(98/102)	93.0%(53/57)	92.2%(47/51)	70.6%(12/17)	36.6%(15/41)	49.5%(54/109)	44.4%(8/18)	79.3%(146/184)		
Bing Opinion	60.1%(1,533/2,555)	60.3%(551/914)	61.7%(1,532/2,448)	73.5%(25/34)	100.0%(7/7)	100.0%(1/1)	100.0%(1/1)	43.8%(7/16)	42.4%(25/59)	40.4%(55/136)	66.3%(8/12)	46.9%(99/211)		
Bing Politics*	80.7%(837/1,033)	86.9%(337/388)	95.8%(92/96)	100.0%(17/17)	100.0%(3/3)	N.A.	N.A.	84.3%(43/51)	60.8%(48/79)	58.0%(65/111)	53.8%(43/80)	63.7%(114/179)		
Bing Business*	67.7%(831/1,222)	76.0%(336/442)	85.5%(65/76)	100.0%(5/5)	N.A.	N.A.	N.A.	75.0%(33/44)	42.2%(43/102)	68.3%(78/114)	67.3%(68/101)	70.9%(139/196)		
Bing Sum	65.4%(2,129/3,256)	71.7%(1,048/1,466)	78.3%(412/529)	84.3%(18.9)	88.2%(22.8)	91.2%(25.8)	92.8%(27.4)	62.3%(160/257)	37.1%(206/556)	45.4%(656/1,444)	56.9%(222/390)	66.1%(1,600/2,422)		

1. ↑ denotes performance improvement against STP with $t = 0$.2. The bold cell denotes optimal precision among STP with $t = 6$, SIT, PathInv, RTI, TransRepair and CAT.

Table 4. The number of detected translation errors

	STP	SIT	PatInv	RTI	TransRepair	CAT
Google Politics	379	11	36	31	11	134
Google Business	683	15	3	46	18	124
Google Culture	552	7	17	40	17	102
Google Sport	320	17	217	42	42	84
Google Tech	363	5	15	12	9	217
Google Travel	604	3	10	32	14	189
Google Health	305	16	15	27	20	171
Google Life	380	7	18	33	16	158
Google Legal	354	13	15	32	7	157
Google Opinion	344	10	19	37	9	100
Google Politics*	372	69	79	62	85	136
Google Business*	417	57	76	51	75	173
Google Sum	5,073	230	520	445	323	1,745
Bing Politics	485	13	13	39	12	134
Bing Business	535	17	6	41	18	130
Bing Culture	579	5	3	65	11	149
Bing Sport	257	17	24	62	31	177
Bing Tech	388	9	4	13	6	225
Bing Travel	718	5	9	26	12	169
Bing Health	312	16	12	39	17	144
Bing Life	395	12	7	49	25	130
Bing Legal	352	19	15	50	11	167
Bing Opinion	422	7	25	39	7	84
Bing Politics*	334	67	50	55	45	132
Bing Business*	323	52	44	61	69	162
Bing Sum	5,100	239	212	539	264	1,803

sentence and its translation. TransRepair reports a list of suspicious sentence pairs and we regard each reported pair as a suspicious issue.

If we want to detect as many erroneous issues as possible, STP with a lowest threshold (i.e., $t = 0$) is able to find 831 ~ 3,185 erroneous issues, improving state-of-the-art techniques by at least 400% on all datasets with competitive precision. For example, when testing Google Translate with Business* dataset, STP reports 1,094 erroneous issues with 87.2% precision, while SIT, PatInv, RTI, TransRepair and CAT only reports 38, 75, 58, 74 and 168 erroneous issues with less than 80% precision, respectively. If we want to get more accurate results, STP is able to outperform state-of-the-art techniques for both translators on average when $t > 0$. For example, compared with SIT, PatInv, RTI, TransRepair and CAT, STP with $t = 6$ improves the precision by 22.9%, 19.9%, 38.9%, 17.7% and 19.8% when testing Google Translate, by 22.0%, 47.2%, 38.9%, 27.4%, 18.2% when testing Bing Microsoft Translator with a comparable amount of erroneous issues. In Table 3, bold cells indicate the optimal precision among STP with $t = 6$, SIT, PatInv, RTI, TransRepair and CAT. In detail, STP achieves better precision performance on 22 testing scenarios, while CAT and SIT outperform on Tech and Health datasets when testing Google Translate and Bing Microsoft Translator, respectively. We believe the results have shown the superiority of STP. As real-world source sentences are almost unlimited (e.g., news network), we could always set a high t to get valuable erroneous issues with high precision.

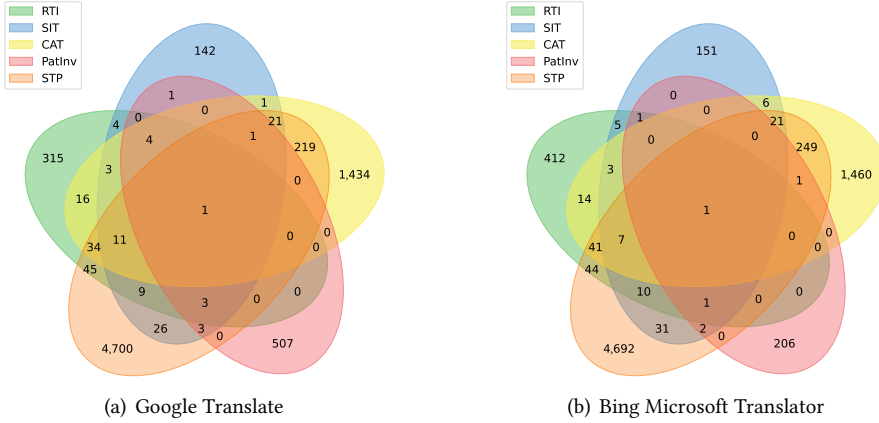


Figure 3. Erroneous translations by different approaches

5.2 Erroneous Translations

We have presented that STP achieves a much better performance in terms of the number of reported issues with comparable precision. In this section, we further analyze how many translation errors STP is able to detect. For all reported erroneous issues in Table 3, we use exact matching to remove duplicated erroneous translations. Thus, we identify all unique translation errors in erroneous issues (i.e., the same translation error will be counted only once no matter how many times it appears).

5.2.1 Number of Erroneous Translations. We present the number of translation errors in Table 4. The first column also lists the twelve dataset categories and two machine translation systems, and the remaining columns show the number of erroneous translations detected by STP and five compared techniques, respectively. The results of total detected erroneous translations under all the twelve dataset categories when testing Google Translate and Bing Microsoft Translator are also listed in the middle part and bottom part of Table 4. The best results are shown in bold. From Table 4, STP detects 257~718 erroneous translations, which is significantly better than other techniques (3~69 for SIT, and 3~217 for PatInv, 12~65 for RTI, 7~85 for TransRepair and 84~225 for CAT). On average, STP outperforms the most recent technique CAT by 290% and 282% for Google Translate and Bing Microsoft Translator, respectively. It is fundamentally difficult for STP to achieve much better performance in terms of the number of reported erroneous translations with comparable precision. We observe that STP can generate a large number of pruned sentences for almost all original sentences by recursively removing contextual words, while existing work can only test a small number of original sentences due to the limitation of adopted DL models. If we intend to have a higher precision by setting a larger threshold value, we will reasonably obtain fewer erroneous translations.

5.2.2 Overlap of Erroneous Translations. We further analyze the overlap of translation errors found by STP and other techniques. We show STP with four best-performing compared techniques due to page limit. Specifically, TransRepair is not included, as CAT is an extension of TransRepair and achieves better performance in terms of precision and the number of erroneous issues. Figure 3 presents the erroneous translations reported by different approaches via Venn diagrams.

Figure 3 shows 7,500 erroneous translations from Google Translate and 7,337 erroneous translations from Bing Microsoft Translator can be detected by the combination of all approaches, resulting in 14,837 erroneous translations in our experiment. Particularly, there are 9,392 erroneous translations unique to STP, while there are 293, 713, 727, 2,894 erroneous translations unique to SIT, PatInv, RTI and CAT, respectively. The improvement reaches 711.93% on average. After inspecting all the erroneous translations, we observe that STP is effective at reporting translation errors for both original and pruned sentences (illustrated in Section 2.3). Meanwhile, the unique errors to RTI are mainly from the extracted phases, which have similar translations in different contexts. The unique errors to CAT mainly come from similar sentences of one number difference (e.g., “good” → “bad”). We also find there are 219 and 249 erroneous translations that can be detected by both STP and CAT. After our careful analysis, the overlapped errors are mostly the original source sentences. The high degree of overlap demonstrates that STP and the most recent technique CAT are quite effective in detecting errors for original sentences. We will discuss it further in Section 5.2.3. For example, in Table 5, we present an example that Google Translate mistranslates where “to tread right now” is not translated. This erroneous translation can be detected by both STP and CAT. STP generates a pruned sentence “It’s a very, very difficult path for banks to tread right now, Knightley said.” by removing a word “central”, while CAT generates a similar sentence “It’s a very, very difficult line for central banks to tread right now, Knightley said.” by replacing “path” with “line”. The translation results of the two generated sentences contain the missing translation text in the original sentence. Based on these results, we believe our approach complements the state-of-the-art approaches.

Table 5. Example of overlapping erroneous translation

Source sentence	It’s a very, very difficult path for central banks to tread right now , Knightley said.
Target sentence	奈特利说，对于中央银行来说，这是一条非常非常困难的道路。(By Google) [Erroneous Translation: “to tread right now” is not translated.]
Target meaning	It’s a very, very difficult path for central banks , Knightley said.

5.2.3 Number of Original Erroneous Translations. Although STP is able to report a large number of erroneous translations for a given sentence corpus, its performance in detecting original erroneous translations is also crucial. For example, as shown in Section 2.3, if the user adopts existing machine translation testing tools to examine the news translated by Google Translate, most of the reported erroneous translations are sentences generated by the machine translation tools. Such a low detection probability of the original sentences may limit the application of existing machine translation testing techniques in practice.

Thus, we further investigate the results of translation errors for the original sentences, which is presented in Table 6. The first column lists the twelve dataset categories and two machine translation systems, and the remaining columns show the number of erroneous translations for the original sentences detected by STP and five compared techniques, respectively. Each cell is represented as $x(y)$, where x is the recall value (i.e., the ratio of reported translation errors only for the original sentences over all the real translation errors) and y is the number of all erroneous translations for the original source sentences. The best results are shown in bold. We also show the total number of original erroneous translations on all twelve dataset categories and its corresponding recall value in the middle part and bottom part of Table 6 for Google translate and Bing Microsoft Translator, respectively.

Table 6. The recall of translation errors for original sentences

	STP	SIT	PatInv	RTI	TransRepair	CAT
Google Politics	69%(54)	6%(5)	3%(2)	19%(15)	5%(4)	50%(39)
Google Business	66%(53)	9%(7)	3%(2)	19%(15)	7%(6)	45%(36)
Google Culture	81%(64)	3%(2)	0%(0)	20%(16)	8%(6)	35%(28)
Google Sport	75%(41)	13%(7)	5%(3)	25%(14)	24%(13)	45%(25)
Google Tech	72%(57)	3%(2)	0%(0)	3%(2)	5%(4)	77%(61)
Google Travel	91%(49)	2%(1)	2%(1)	17%(9)	9%(5)	83%(45)
Google Health	56%(37)	9%(6)	0%(0)	14%(9)	12%(8)	73%(48)
Google Life	68%(44)	3%(2)	0%(0)	12%(8)	8%(5)	66%(43)
Google Legal	62%(41)	9%(6)	3%(2)	23%(15)	5%(3)	65%(43)
Google Opinion	69%(35)	8%(4)	0%(0)	25%(13)	8%(4)	55%(28)
Google Politics*	80%(64)	45%(36)	4%(3)	28%(22)	4%(3)	45%(36)
Google Business*	81%(65)	30%(24)	0%(0)	23%(18)	0%(0)	64%(51)
Google Sum	73%(604)	12%(102)	2%(13)	19%(156)	7%(61)	58%(483)
Bing Politics	82%(69)	8%(7)	1%(1)	15%(13)	4%(3)	42%(35)
Bing Business	87%(69)	11%(9)	0%(0)	11%(9)	11%(9)	35%(28)
Bing Culture	77%(63)	0%(0)	0%(0)	22%(18)	4%(3)	49%(40)
Bing Sport	52%(38)	11%(8)	1%(1)	29%(21)	15%(11)	74%(54)
Bing Tech	73%(63)	2%(2)	0%(0)	2%(2)	5%(4)	74%(64)
Bing Travel	80%(64)	3%(2)	0%(0)	5%(4)	9%(7)	53%(42)
Bing Health	59%(42)	10%(7)	1%(1)	25%(18)	11%(8)	65%(46)
Bing Life	85%(58)	6%(4)	0%(0)	16%(11)	13%(9)	51%(35)
Bing Legal	65%(52)	11%(9)	0%(0)	14%(11)	5%(4)	72%(58)
Bing Opinion	92%(54)	3%(2)	0%(0)	19%(11)	5%(3)	32%(19)
Bing Politics*	78%(58)	42%(31)	3%(2)	28%(21)	3%(2)	41%(30)
Bing Business*	74%(62)	29%(24)	1%(1)	29%(24)	1%(1)	58%(49)
Bing Sum	75%(692)	11%(105)	1%(6)	18%(163)	7%(64)	54%(500)

From Table 6, we find STP achieves a recall of 73% for detecting original erroneous translations when testing Google Translate, which outperforms state-of-the-art techniques by 53.4% on average (61% for SIT, 71% for PatInv, 54% for RTI, 66% for TransRepair and 15% for CAT). Similar comparison performance can be observed for Bing Microsoft Translator (the improvement is 56.8% on average). The performance of detecting original erroneous translations is crucial in practice. Existing works can only test a small number of original sentences, leading to a small recall of errors. STP can generate pruned sentences with different structures for a given original sentence, and detect more original erroneous translations per category. We believe the improvements (especially for original sentences) can make contributions to pushing machine translation testing forward in the SE community.

5.3 Types of Reported Erroneous Translation

In this section, we evaluate the effectiveness of STP by further analyzing the diversity of detected translation errors. In the literature [18, 21, 22], the translation errors are mainly divided into five types: under-translation, over-translation, word/phrase mistranslation, incorrect modification, and unclear logic. We analyze all unique translation errors detected by STP in Section 5.2 and classify each error into at least one type. We report that STP is able to detect all these five types of translation errors.

Table 7 presents the number of translations that have a specific type of error. The first column lists the twelve dataset categories and two machine translation systems, and the remaining columns

Table 7. The number of translation errors in each category

	under translation	over translation	word/phrase mistranslation	incorrect modification	unclear logic
Google Politics	53%(246)	10%(48)	13%(61)	10%(46)	13%(61)
Google Business	28%(255)	13%(122)	34%(312)	7%(59)	17%(158)
Google Culture	25%(172)	12%(85)	38%(266)	7%(50)	18%(124)
Google Sport	20%(89)	4%(19)	45%(200)	8%(34)	23%(101)
Google Tech	64%(265)	8%(32)	19%(78)	4%(17)	5%(20)
Google Travel	67%(472)	3%(20)	17%(122)	2%(16)	11%(77)
Google Health	38%(149)	20%(79)	28%(109)	8%(32)	7%(27)
Google Life	39%(198)	10%(53)	22%(110)	10%(50)	19%(94)
Google Legal	61%(230)	8%(29)	29%(107)	1%(5)	1%(3)
Google Opinion	41%(159)	34%(132)	10%(39)	11%(42)	3%(13)
Google Politics*	26%(111)	15%(63)	32%(137)	9%(38)	18%(77)
Google Business*	38%(210)	18%(98)	22%(119)	5%(26)	17%(96)
Google Sum	41%(2,556)	12%(780)	27%(1,660)	7%(415)	14%(851)
Bing Politics	36%(221)	19%(119)	28%(170)	9%(57)	8%(47)
Bing Business	46%(297)	16%(102)	21%(136)	3%(17)	15%(100)
Bing Culture	70%(435)	10%(60)	6%(38)	8%(52)	6%(39)
Bing Sport	22%(77)	13%(44)	36%(124)	12%(41)	17%(58)
Bing Tech	68%(307)	12%(56)	8%(36)	3%(15)	9%(40)
Bing Travel	50%(396)	9%(68)	23%(183)	5%(39)	14%(112)
Bing Health	47%(173)	15%(56)	24%(88)	6%(21)	9%(32)
Bing Life	40%(204)	4%(22)	30%(152)	7%(38)	18%(94)
Bing Legal	49%(191)	6%(22)	32%(125)	7%(28)	7%(26)
Bing Opinion	29%(151)	22%(115)	27%(140)	9%(47)	12%(61)
Bing Politics*	31%(143)	10%(46)	33%(152)	8%(35)	18%(82)
Bing Business*	23%(96)	15%(61)	35%(144)	8%(33)	20%(81)
Bing Sum	44%(2,691)	13%(771)	24%(1,488)	7%(423)	13%(772)

list the comparison results for the five error categories. The total results under all the twelve dataset categories are also shown in the middle part and bottom part of Table 7. The comparison result in each cell is represented as $x(y)$, where x is the ratio of translation errors in this error category over the translation errors in all five error categories and y is the number of erroneous translations in this category. We can observe that Google Translate and Bing Microsoft Translator have a very similar distribution of error types overall. For example, under-translation is the most common translation error, with over 2,500 errors for both translators. We think the occurrence is related to our pruning mechanism, as the removed words in the source sentence can be effectively reflected on the translation result. Moreover, different dataset categories have diverse performances regarding the error types. For example, the Travel dataset has more than 700 erroneous translations, while the Legal dataset has only less than 400 errors for both translators. We think the occurrence is closely related to the training data. This indicates there exists an imbalance issue with existing training data in different categories of datasets, which is out of the scope of our work. We highly recommend the researchers conduct thorough evaluations for neural machine translation imbalance issues and explore how machine translation systems perform under different categories of datasets.

We also list examples the five types of erroneous translation detected by STP in Table 8 to Table 12. The first row shows the source sentence that is mistranslated by the translators. The second row shows the translated sentence in the target language and explanation of the error. The third

row shows the meaning of translated sentence in the source language. We present the types of erroneous translation in detail as follows.

5.3.1 Under-translation. Under-translation means that some words in the source language sentence are not translated to the target language sentence. For example, in Table 8, the source sentence emphasizes that there is no limit on the number of things to borrow, while “as much as they need” is not translated by Google Translate⁴. Under-translation often leads to the lack of crucial information and different semantics meanings between the source sentence and the target sentence.

Table 8. Example of under-translation error detected

Source sentence	But parents of undergraduates and graduate students face no such limits, and can borrow <u>as much as they need with the price tag</u> set by schools.
Target sentence	但是，本科生和研究生的父母没有这种限制，他们可以根据学校设定的价格向他们借钱。(By Google) [Erroneous Translation: “as much as they need” is not translated.]
Target meaning	But parents of undergraduates and graduate students face no such limits, and can borrow <u>with the price</u> tag set by schools.

5.3.2 Over-translation. Over-translation means that some words in the target language sentence are not translated from any words in the source sentence or translated multiple times unnecessarily. For example, in Table 9, “the almost anxiety provoking magnitude” is translated twice by Google Translate in the target, while this phrase only appears once in the source sentence⁵. Besides, the “locations” is translated into “business locations” by Google Translate, while “business” does not appear in the original sentence⁵. Over-translation often brings unnecessary information in the target sentence and easily misleads people.

Table 9. Example of over-translation error detected

Source sentence	You should be agonizingly jealous of the sorts of problems we work on and the almost <u>anxiety provoking</u> magnitude of data with which we get to work.
Target sentence	您应该非常嫉妒我们正在处理的各种问题，以及几乎令人不安的令人不安的数据量。(By Google) [Erroneous Translation: “anxiety provoking” is translated twice.]
Target meaning	You should be agonizingly jealous of the sorts of problems we work on and the almost <u>anxiety provoking anxiety provoking</u> magnitude of data.
Source sentence	Imagine if you are a very large retailer that has a large number of <u>locations</u> , said Martin Fleming.
Target sentence	马丁·弗莱明 (Martin Fleming) 说，想像一下，如果您是一家拥有大量营业地点的大型零售商。(By Google) [Erroneous Translation: “location” is translated into “营业地点”，where “营业” does not appear in the source sentence.]
Target meaning	Imagine if you are a very large retailer that has a large number of <u>business locations</u> , said Martin Fleming.

⁴<https://edition.cnn.com/2019/03/18/politics/trump-student-loan-limit-cap/index.html>. Accessed August, 2022.

⁵<https://edition.cnn.com/2019/03/13/tech/amazon-economists/index.html>. Accessed August, 2022.

5.3.3 *Word/phrase Mistranslation.* Word/phrase Mistranslation means that words or phrases in the source sentence are translated incorrectly to the target sentence. For example, in Table 10, “the elite University of California system” is translated to “the elite university system at the University of California”⁶. Besides, “the rugged campaign images” is mistranslated into “the election images”, while the original sentence describes a perfume advertising campaign⁷.

Table 10. Example of word/phrase mistranslation error detected

Source sentence	The elite University of California system has significantly increased its Latino enrollment.
Target sentence	加州大学的精英大学系统极大地增加了拉丁裔的入学率。(By Google) [“The elite University of California system” is incorrectly translated into “加州大学的精英大学系统” which means “ The elite university system at the University of California ”.]
Target meaning	The elite university system at the University of California has significantly increased Latino enrollment.
Source sentence	The actor’s fans propose that the rugged campaign images were a unanimous societal response to the question.
Target sentence	这位演员的粉丝认为，粗犷的竞选形象是对这个问题的一致社会回应。(By Google) [“campaign images” is incorrectly translated into “竞选形象”, and the correct translation should be “活动图片” in the news article.]
Target meaning	The actor’s fans propose that the rugged election images were a unanimous societal response to the question.

5.3.4 *Incorrect Modification.* Incorrect Modification means that some adjuncts modify the wrong elements in the target sentence. For example, in Table 11, “leaders” is modified by “who mimic his own brashness and disregard for political norms” in the source sentence, while the clause modifies another element “those” in the target sentence⁸.

Table 11. Example of incorrect modification error detected

Source sentence	In a world of perceived foes, Trump has often looked to leaders who mimic his own brashness and disregard for political norms as allies .
Target sentence	在一个充满敌意的世界中，特朗普经常寻找模仿自己的傲慢并无视政治规范作为盟友的领导人。(By Google) [Erroneous Translation: “ leaders ” is incorrectly modified by “ allies ”.]
Target meaning	In a world of perceived foes, Trump has often looked to those who mimic his own brashness and disregard for political norms as allies’ leaders .

⁶<https://edition.cnn.com/2019/03/19/politics/college-education-scandal-inequality-higher-education/index.html>. Accessed August, 2022.

⁷<https://edition.cnn.com/style/article/adam-driver-burberry-ltw/index.html>. Accessed August, 2022.

⁸<https://edition.cnn.com/2019/03/19/politics/donald-trump-jair-bolsonaro-brazil-white-house/index.html>. Accessed August, 2022.

5.3.5 *Unclear Logic*. Unclear Logic means that all the tokens or phrases are translated correctly but the sentence logic is not correct. For example, in Table 12, “The UC Latino students” and “Latinos in the community college system” are two objects for comparison⁶. However, Google Translate does not understand the logical relation between them and thinks “the community college system” is the context of the sentence. Although all words are correctly translated, the target sentence has a totally different meaning from the source sentence.

Table 12. Example of unclear logic error detected

Source sentence	The UC Latino students are 25 times more likely to finish their degrees on time than <u>Latinos in the community college system</u> .
Target sentence	在社区大学系统中，加州大学拉丁裔学生按时完成学位的可能性是拉丁裔的25倍。(By Google) [Erroneous Translation: “ <i>in the community college system</i> ” is incorrectly translated as the context in the target sentence.]
Target meaning	<u>In the community college system</u> , the UC Latino students are 25 times more likely to finish their degrees on time than Latinos.

5.4 Efficiency of Our Approach

In this section, we analyze the efficiency of STP in terms of the running time. Following existing studies [18, 21, 22, 55], we run all involved techniques and report their time consumption on the same initial corpus (discussed in Section 4.3) and hardware environment (discussed in Section 4.6) to ensure a fair comparison. It is worth noting that, unlike the fuzzing research [29] that usually sets the same time budget, we execute all techniques without a corresponding time limit. The difference lies in that, fuzzing mainly focuses on test input generation, which can generate unlimited inputs to induce software systems crashes, thus often requiring a time limit to terminate execution. However, machine translation testing mainly relies on metamorphic testing, which designs metamorphic relations (discussed in Section 3.2) and generates a limited number of test inputs to violate the relation, thus resulting in a limited running time overall. Thus, we evaluate the running time of STP on the twelve datasets and two translators, which is a common practice in the machine translation testing community [54]. To mitigate the effect of randomness, following existing studies [21, 22], we repeat STP 10 times in the same experimental setting and report the average running time. Table 13 presents the average comparison results when testing the two translators. The first column lists STP and five compared techniques. The second column lists the number of generated sentences, the average time (in seconds) for each sentence generation, translation collection, and erroneous translation detection. The remaining columns list the detailed running time of the twelve dataset categories and their average time. We also present the average running time in the last row of each technique under the twelve dataset categories. The best results per dataset category are in bold. Each cell is represented as $x(y)$, where x refers to the total running time per dataset category and y refers to the overall running time per sentence.

From Table 13, we can find that STP spends about 6 minutes per dataset category on average. Specifically, the step of collecting translation results via the translator API takes up over 60% of the total running time. In our implementation, we invoke the translator API once for each source sentence and thus the network communication time is included. Table 13 also presents the running time of state-of-the-art techniques using the same experimental settings. We find that STP takes 149.8% and 442.5% more running time than RTI and TransRepair, but it can generate 392.8% and 1886.2% more sentences per dataset category on average. Meanwhile, STP takes 15.8% and 9.3% less running time than SIT and CAT, with 9.5% and 90.1% more sentences than SIT and CAT. We also

find that only PatInv generate more sentence than STP by 441.0%, but it consumes 570.7% more running time than STP.

On average, Table 13 shows STP requires 0.39 seconds for each sentence generation, translation collection, and erroneous translation detection, improving the efficiency of state-of-the-art techniques by 22.0%~46.6%. Specifically, STP consumes 0.09 seconds to generate a sentence, which is 52.6% and 35.7% faster than PatInv and CAT. This improvement is attributed to the fact that, STP, unlike most techniques, does not require the employment of large-scale language models. Among these techniques, TransRepair relies on a dictionary of similar words to directly perform word replacement, thus taking the least time cost (0.01 seconds per sentence). The generation of the dictionary consumes more than 11 hours in our experiment setting, which is not mentioned in Table 13. Compared with STP, CAT directly adopts BERT to perform word replacement and candidate filtering, resulting in more generation time (0.14 seconds per sentence). We find STP takes about 0.02 seconds to detect erroneous translation on average, improving state-of-the-art techniques (except PatInv) by 50%~92.6%, as the bag-of-words metric is very lightweight and fast in practice. PatInv adopts a simple string comparison to compare the translations of the original and generated sentences, so PatInv performs best for erroneous translation detection (0.01 seconds on average). Regarding translation collection, we use the same translator API to get the translation results, so the time cost is similar across different techniques. We also find the translation time of STP is 15.2%~49.1% faster than other techniques. Similar findings can be observed for RTI. This is mainly because most of the source texts that need to be translated are phrases or pruned sentences rather than complete sentences in the dataset. Based on the above observations, it is concluded that STP achieves comparable efficiency to state-of-the-art techniques in terms of running time.

6 DISCUSSION

6.1 Syntactic Tree Pruning

Most existing works [18, 21, 54, 55] usually adopt language models to generate sentences with same structures by word-replacing (e.g., synonym replacement). They can only detect errors related to same structures, while failing to detect errors revealed by different structures (mentioned in Section 2). We propose the crucial semantics invariance metamorphic relation to generate sentences with different structures, which is beyond the scope of existing works. The results demonstrate it is quite effective for STP to trigger machine translation errors with different structures, and achieve a higher recall of erroneous translations for the original sentences. We highlight this direction to generate inputs with different structures to test machine translation systems. Meanwhile, it is more practical to generate sentences using pre-defined operators instead of language models, as recent work shows training and deploying language models are quite costly for developers and users [58]. Besides, the pre-defined operators are designed based on all available dependency types and can generate simple sentences with various structures and are generally applicable to other SE tasks, such as testing question answering systems by asking questions with different structures [4].

6.2 Robust Machine Translation

Compared with traditional software systems directly encoded in source code, it is more difficult to repair machine translation errors because the decision logic of neural machine translation models lies in the complex network structure and a large corpus of data. Even if a fault-triggering test case (e.g., source language sentence) can be identified, how to automatically repair the model without introducing new errors is still a long-standing challenging task. However, we report it is significant to use translation errors to improve machine translation systems.

Table 13. Running time of different approaches

		PO	BU	CU	SP	TE	TR	HE	LI	LE	OP	PO*	BU*	AVE
SIT	Sentence	405	817	578	1259	601	195	525	611	687	710	2084	1941	867.7
	Generation	0.05	0.03	0.03	0.03	0.03	0.05	0.04	0.03	0.03	0.03	0.03	0.02	0.03
	Translation	0.42	0.46	0.44	0.45	0.42	0.44	0.43	0.42	0.42	0.46	0.39	0.37	0.42
	Detection	0.08	0.05	0.06	0.04	0.06	0.02	0.04	0.03	0.04	0.04	0.09	0.03	0.05
	Total	259.3 (0.55)	486.7 (0.54)	349.8 (0.53)	693.2 (0.52)	342.9 (0.51)	143.4 (0.51)	311.0 (0.51)	338.6 (0.48)	374.6 (0.49)	421.2 (0.53)	1089.3 (0.51)	865.1 (0.42)	472.9 (0.50)
PatInv	Sentence	3293	4262	3599	6972	3877	1846	3731	3347	4036	3884	11581	11251	5139.9
	Generation	0.11	0.27	0.21	0.23	0.14	0.11	0.17	0.21	0.2	0.27	0.16	0.16	0.19
	Translation	0.44	0.44	0.37	0.42	0.43	0.48	0.43	0.47	0.45	0.51	0.44	0.44	0.44
	Detection	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Total	1357 (0.56)	2743.1 (0.72)	2017.4 (0.59)	4194.8 (0.66)	1726.5 (0.58)	726.3 (0.60)	2001.9 (0.61)	2018.2 (0.69)	2069.9 (0.66)	2453.3 (0.79)	5552.5 (0.61)	5188.9 (0.61)	2670.8 (0.64)
RTI	Sentence	189	230	228	159	205	219	167	236	192	179	154	155	192.7
	Generation	0.07	0.07	0.07	0.07	0.07	0.06	0.06	0.06	0.06	0.07	0.04	0.05	0.06
	Translation	0.35	0.32	0.30	0.35	0.32	0.35	0.31	0.35	0.37	0.33	0.28	0.29	0.33
	Detection	0.31	0.26	0.23	0.27	0.25	0.27	0.23	0.29	0.32	0.29	0.22	0.24	0.27
	Total	171.2 (0.73)	180 (0.65)	167.2 (0.60)	145.7 (0.69)	162.2 (0.64)	184.0 (0.68)	131.6 (0.60)	200.3 (0.70)	182.6 (0.75)	157.8 (0.69)	110.6 (0.54)	119.8 (0.58)	159.4 (0.66)
TransRepair	Sentence	41	72	46	88	36	30	44	69	35	41	32	40	47.8
	Generation	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Translation	0.46	0.51	0.48	0.53	0.47	0.51	0.43	0.59	0.49	0.46	0.37	0.38	0.48
	Detection	0.05	0.04	0.05	0.05	0.04	0.10	0.05	0.07	0.07	0.07	0.04	0.05	0.05
	Total	67.0 (0.52)	90.9 (0.55)	73.3 (0.54)	104.4 (0.59)	65.2 (0.52)	68.9 (0.62)	64.0 (0.49)	104.0 (0.67)	69.1 (0.57)	68.0 (0.54)	50.2 (0.42)	55.9 (0.44)	73.4 (0.54)
CAT	Sentence	500	500	500	500	500	500	500	500	500	500	498	500	499.8
	Generation	0.15	0.18	0.16	0.15	0.14	0.16	0.13	0.17	0.16	0.13	0.09	0.09	0.14
	Translation	0.59	0.59	0.56	0.58	0.56	0.6	0.53	0.59	0.59	0.51	0.45	0.46	0.55
	Detection	0.04	0.04	0.07	0.04	0.03	0.04	0.03	0.04	0.04	0.04	0.03	0.03	0.04
	Total	448.4 (0.78)	464.2 (0.81)	523.0 (0.79)	583.4 (0.77)	420.8 (0.73)	460.4 (0.80)	402.6 (0.69)	458.1 (0.80)	455.4 (0.79)	390.2 (0.68)	328.6 (0.57)	335.2 (0.58)	439.2 (0.73)
STP	Sentence	919	1038	1155	821	852	1262	937	939	1031	1095	635	716	950.0
	Generation	0.09	0.10	0.11	0.11	0.09	0.08	0.08	0.09	0.08	0.08	0.10	0.09	0.09
	Translation	0.26	0.28	0.27	0.28	0.27	0.36	0.24	0.26	0.31	0.27	0.22	0.22	0.28
	Detection	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.02	0.02
	Total	371.2 (0.38)	457.8 (0.41)	484.6 (0.40)	367.6 (0.41)	350.8 (0.38)	616.2 (0.46)	351.0 (0.34)	377.6 (0.37)	465.4 (0.42)	435.5 (0.37)	238.3 (0.34)	262.9 (0.33)	398.2 (0.39)

In our work, similar to most machine translation testing approaches (e.g., RTI, SIT and PatInv), STP aims to validate machine translation systems without repairing detected errors. However, it is promising to conduct some future work on top of STP to automatically repair the detected errors and improve the robustness of the neural translation model. As an industrial online machine translation service, similar to traditional programming paradigms, it is easy to fix the found issue in a hard-code way without leading to negative effects. Besides, it is possible to design a post-processing strategy to repair erroneous translations for users and developers without additional manual repair efforts [55]. Furthermore, it is more robust to retrain or fine-tune the network with

the source sentence of a translation error and its correct translation. The reported issues may also be useful for some further debugging or repairing work (such as extracting some mistranslation patterns for a given machine translation system).

6.3 Scalability of STP

In our work, we set the source language and target language to English and Chinese, respectively, because of the knowledge background of the authors. Considering the fact that the two chosen languages are Top-2 most spoken languages in the world [34] and the English-to-Chinese setting is widely adopted in most existing machine translation testing studies [18, 21, 54, 60, 76], we think this setting allows us to conduct a comprehensive comparison with selected baseline and provide reliable comparison results in our experiments. Besides, the concept of core syntactic tree methodology is general and can be built on various languages due to two reasons. First, the syntactic tree pruning is based on the basic structure and rhetorical structure theory, which is not limited to English and is a general linguistic theory across a wide array of mainstream languages. Second, the adopted Stanford Parser currently supports six languages (i.e., Arabic, Chinese, English, French, German and Spanish). We also notice there exist other dependency tree parsers targeting more languages (e.g., Japanese⁹). Thus, it is practical and simple to implement STP to other languages with little engineering efforts. About translators, Google Translate and Microsoft Bing Translator are adopted in our work because they are widely used industrial online machine translation services and represent the state-of-the-arts. Other popular translators (e.g., Youdao Translator [67]) can also be integrated into STP easily by standard translation interfaces. Meanwhile, the method to generate new sentences (Section 3.1) and detect translation errors (Section 3.3) is implemented as flexible modules and can be enriched by other methods in the future.

The core concept of the syntactic tree methodology is also general to other NLP testing fields. For example, similar to STP, we can generate complex sentences by context insertion to test machine translation systems based on the hypothesis that adding contextual information into a source sentence should not influence the translation results of the trunk. It is practical to generate sentences by context insertion on top of the existing STP framework with engineering efforts. We can replace the context-removal-based sentence generation module with a context-insertion-based one, as other existing modules are quite suitable for the new context-insertion testing scenario. For example, our translation error detection module equipped with a bag-of-the-words metric is effective in finding inconsistent errors between a simple source sentence and a complex source sentence.

6.4 False Positives

Despite remarkable precision being achieved by STP, there are still some false positives and false negatives in our approach. We conclude them from three main sources. First, a pruned phrase could have a different correct translation compared with the original phrase. For example, “the owner” has several correct meanings (e.g., 拥有者和主人), while “the owner of Carrier” has a specific meaning (i.e., 公司拥有者) in the context “Carrier”. However, we could maintain an alternative translation dictionary to alleviate this kind of false positive. Meanwhile, a filtering mechanism can be introduced to ensure the degree of preserved context between the newly generated and original sentence (in Section 3.2). Second, the dependency parser that we use to parse the sentence could return wrong or inaccurate results. For example, the relation between “that” and “employ” is misidentified as obj for the sentence “It is believed in the field that Amazon employs more PhD economists than any other tech company”, which leads to the generation of invalid sentences.

⁹<https://github.com/ku-nlp/bertknp>. Accessed August, 2022.

Third, several generated sentences are not valid because the pruning strategy we define does not guarantee to take into account all situations in real-world sentences. For example, STP will remove the words “of the WTO” simultaneously because the dependency relation between “of” and “WTO” is “case” (detailed in Table 1) for the sentence “China is not the litmus test of the WTO or the world trade.”. However, STP fails to consider the fact that the “case” relation is further below the conjunction relation between the phrase “the WTO” and “the world trade”.

To address the risks posed by the pruned sentences, we confirm that the erroneous translations labeled as true positives are syntactically and semantically correct (discussed in Section 4.4). As a result, although a translation in a reported issue is erroneous, if its sentence in the source language is invalid, we count the translation as a false positive. Besides, although we conduct a well-designed human labelling experiment, it is inevitable that a few sentences that have very small grammatical errors are not labelled by the participants. In such a case, we consider the grammatical errors are typos that people introduce when typing, and mature translation systems need to handle this situation. Natural language is unstructured and complex in the real world, and it is extremely challenging for rule-based techniques (such as STP) to take all possible situations into consideration. It is promising and valuable to conducting a more in-depth analysis to investigate the impact of different rules on the testing performance. However, considering the numerous designed rules and the extremely required manual inspection efforts in the analysis, it goes beyond the scope of our current work and can be explored in the future. Despite that, STP still achieves state-of-the-art precision for testing machine translation systems. In the future, we attempt to design more mature pruning rules for different types of sentences and analyze the impact of such rules on sentence validity.

6.5 Comparison with PatInv-Remove

In our work, following existing machine translation testing work, five state-of-the-art approaches are selected to compare against STP with 1,200 sentences from 10 major categories of news sites. To the best of our knowledge, the selected baselines are the largest set on machine translation testing in the literature. As discussed in Section 4.5, some baselines may have multiple variants (e.g., word-replacement based one PatInv-replace and word-removal based one PatInv-remove). In such a case, we select the best-performing one among multiple variants according to existing relevant studies. There may exist other possible variants that could have been used in our experiment. For example, we select PatInv-Replace because of its superior performance against PatInv-Remove in terms of detected translation errors and precision. However, different from selected baselines that mainly generate new test cases by replacing words, PatInv-Remove is a word-removal-based approach, which is the closest removal approach to STP. Thus, in this section, we perform an additional evaluation by comparing STP against PatInv-Remove.

The results are presented in Table 14. The first column lists the twelve dataset categories and the total results calculated by all the twelve dataset categories. The remaining columns list the two machine translation systems and two compared approaches. Each cell is represented as $x(y/z)$, where x refers to the precision value, y and z refer to the number of all detected and reported issues by the studied techniques. In particular, due to page limit, we compare PatInv-Remove against STP with the threshold value of 6, which is proven to achieve remarkable precision with a comparable amount of erroneous issues in Section 5.1. The detailed results are presented in our Appendix [72]. From Table 14, we can find STP achieves 62.20%~100.00% precision on Google Translate, improving the metric by 6.70%~61.90% when compared with PatInv-Remove. Besides, STP always detects more translation issues than PatInv-Remove (e.g., 1025 additional issues are detected on Google Translate). When testing Bing Microsoft Translator, STP is able to improve the precision by 34.40% while detecting 949 more translation issues in total. We think the significant improvement of STP

Table 14. Comparison with a state-of-the-art removal-based technique PatInv-Remove

	Google		Bing	
	STP	PatInv-Remove	STP	PatInv-Remove
Politics	71.4%(35/49)	64.7%(33/51)	75.5%(37/49)	69.0%(20/29)
Business	76.3%(58/76)	39.1%(18/46)	73.0%(27/37)	52.0%(13/25)
Culture	62.2%(153/246)	57.1%(24/42)	79.9%(270/338)	12.2%(5/41)
Sport	80.1%(125/156)	55.2%(16/29)	88.1%(126/143)	66.7%(26/39)
Tech	71.0%(44/62)	70.6%(24/34)	100.0%(49/49)	40.7%(11/27)
Travel	86.6%(335/387)	71.0%(22/31)	87.3%(261/299)	62.5%(15/24)
Health	84.9%(90/106)	39.1%(18/46)	64.4%(56/87)	47.7%(21/44)
Life	92.5%(149/161)	42.9%(21/49)	85.4%(76/89)	33.3%(11/33)
Legal	83.3%(169/203)	21.4%(6/28)	92.8%(193/208)	36.0%(9/25)
Opinion	85.0%(51/60)	54.8%(23/42)	73.5%(25/34)	60.0%(24/40)
Politics*	100.0%(38/38)	61.8%(21/34)	100.0%(17/17)	69.2%(18/26)
Business*	100.0%(14/14)	41.7%(10/24)	100.0%(5/5)	58.8%(20/34)
SUM	80.9%(1261/1558)	51.8%(236/456)	84.3%(1142/1355)	49.9%(193/387)

come from several differences with PatInv-Remove: (1) PatInv-Remove assumes that sentences with different meanings should not have the same translation, while STP assumes translation results of the trunk should not be influenced by eliminating contextual information from a source sentence; (2) PatInv-Remove generates new sentences by removing a meaningful word or phrase from the sentence based on constituency trees, while STP generates new sentences by applying a set of core semantics-preserving rules without undermining the basic structure and sentence validity based on dependency trees. Overall, the results demonstrate that, benefiting from the novel metamorphic relation and sentence generation strategy, STP is able to perform better than PatInv-Remove in terms of both the number of detected translation issues and the reported precision.

6.6 Robustness of Pruned Sentences

As mentioned in Section 3.1, a random word detection approach would result in a low acceptability rate, while STP is able to generate a large number of valid sentences based on structure syntactic relations. We conduct a human study to check the validity of sentences generated by random word detection and STP. As recursively removing words usually results in too many sentences, we generate 200 sentences by selecting some sentences and dropping words randomly from the dataset [22]. We then adopt STP to generate 200 sentences randomly from the same dataset. The first two authors manually check whether the generated sentences are grammatically correct and semantically reasonable. Our manual inspection indicates that only three sentences generated by the random word deletion approach are valid, while 191 sentences generated by STP are valid. These results show that STP effectively picks words to drop without breaking the validity of the original sentences.

We notice that although massive manual efforts are devoted to analyzing all available dependency types guided by classical linguistic rhetorical structure theory, the designed pruning strategy suffers from invalid generated sentences, which is a long challenge for existing rule-based techniques [43, 52]. In the literature, a majority of existing machine translation studies adopt a word-replacement-based strategy that mainly replaces a word in the original sentence. In particular, they usually mask out a word and query language models (e.g., BERT) to fill the masked hole with a semantically-similar and syntactically-equivalent word (e.g., smart → cute). Such word-replacement-based approaches design metamorphic relations to generate sentences with the same structures, thus guaranteeing

the validity of generated sentences. However, such same-structure-based metamorphic testing may ignore the errors revealed by sentences with different structures. Machine translation may return a sentence with a different structure for imperceptible perturbations due to the brittleness of the neural network. Thus, STP using sentences with different structures can explore behaviors of machine translation systems more fully, which is beyond the scope of existing same-structure-based works.

6.7 Actionability and Recommendation

In this work, following most existing machine translation testing studies [22, 55], we report suspicious translation issues according to a customized threshold. The experimental results reveal that STP outperforms state-of-the-art approaches on a wide range of threshold values in terms of precision and the number of reported suspicious issues, discussed in Section 5.1. On top of impressive results achieved by STP, we further discuss a crucial question when deploying STP to assist developers to validate machine translation systems in practice, i.e., *how to choose the threshold value under different testing scenarios?*

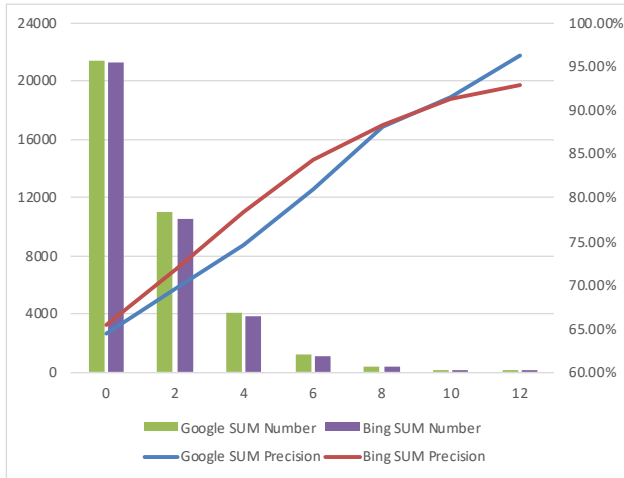


Figure 4. The trade-off between the precision and the number of erroneous issues.

Figure 4 presents the relationship between the number of detected erroneous issues and the corresponding precision under different threshold values. We only present the overall performance of two studied translators across all datasets due to page limit. As can be seen from Figure 4, we observe a trade-off between the precision and the number of reported suspicious issues. In particular, when we increase the threshold value, more translations are regarded as erroneous (i.e., the larger the number of issues reported), while more false positives may be introduced, resulting in a lower precision. Similar performance can also be found in existing studies, such as RTI (Table 2 in [22]) and PatInv-Repalce (Table 10 in [18]). Thus, we recommend different threshold values for developers to employ for testing to improve the actionability of STP in practice, listed as follows: (1) When testing resources are limited, it is obviously recommended that the highest threshold value (i.e., 12) is chosen for developers. In such a scenario, only inspecting fewer reported suspicious issues that have a high probability of being incorrect is more practical and reduces valuable manual effort. (2) When there are sufficient testing resources available, the lowest threshold value (i.e., 0) is recommended for developers, because of the more translation errors STP can detect. On

the one hand, developers need to spend a lot of effort to assess the correctness of the reported suspicious issues manually. On the other hand, developers can check a considerable amount of the reported suspicious issues and reasonably find more translation errors. (3) When a good trade-off between testing resources and testing effectiveness is required, the medium threshold value (i.e., 6) is recommended for developers because of its impressive performance in terms of both precision and the number of translation errors. In such a scenario, developers can detect more translation errors while consuming less manual inspection time than state-of-the-art approaches. Thus, we are confident that STP is effective and easy to use for users and developers in practice.

6.8 Translation Error Detection Metrics

In this work, we adopt a simple but effective bag-of-words strategy to calculate the distance between the original and its pruned sentence translations. The reasons for using the bag-of-words metric are threefold. First, the metric is suitable for our pruning design and error detection scenario. We generate new source sentences by performing some hand-crafted word-removal operations from the original source sentence. In other words, we confirm that the words in the generated source sentence appear in the original sentence. We hope a perfect machine translation can map this relationship into the target sentences, i.e., the original target sentence contains all words that appear in the generated target sentence. In such a case, we adopt a bag-of-words model to denote the set difference, i.e., how many word occurrences are in the generated target sentence but not in the original target sentence. The experimental results also prove its effectiveness in measuring inconsistencies among translated sentences. Second, the bag-of-words metric is already adopted in the literature and proves its supernormal performance in detecting translation errors. For example, similar to our STP finding inconsistent bugs between an original sentence and its pruned sentence, RTI aims to detect the translation errors between an original sentence and its noun phrases. Thus, we are confident that the bag-of-words metric is reasonable to capture the inconsistent core semantics between the translations of the original and pruned sentences. Third, the bag-of-words model is pretty lightweight and fast when deployed in practice. Some existing machine translation approaches tend to design complex strategies to detect translation errors (e.g., subsequence-based metric in TransRepair and structure-based metric in SIT). As discussed in Section 5.4, such mechanisms usually lead to more detection time, hindering the application of such techniques in practice. In daily life, people rely on machine translation systems for education and communication (e.g., reading political news or articles from other countries and visiting websites with content in various languages). Such testing tools can help people to check the translation results in real time, which needs to be done in a very short time. STP adopts a bag-of-words strategy to directly calculate word occurrences with less calculation time while achieving outstanding performance.

The experimental results demonstrate the remarkable fault detection performance of the adopted bag-of-the-words when the sentence pair is made up of an original sentence and its simplified sentence [22]. We further explore some other metrics that consider sentence structure (e.g., constituency tree and dependency tree) from related studies, which may be more semantic-aware. Existing studies usually generate a sentence with the same structure as the original sentence and calculate the edit distance or relation distance (e.g., the number of each phrasal type or the number of each type of dependency relations) between two similar tree structures of the translations. However, STP generates a sentence with a different structure from the original sentence and it is improper to apply such metrics directly. Inspired by the bag-of-words strategy that considers the word frequency in the pruned sentence translation relative to the original sentence translation, we propose a constituency tree-aware bag-of-words metric and a dependency tree-aware bag-of-words metric. In particular, the former evaluates the distance between two sets of constituency relations

by considering the set difference in the count of each phrasal type based on the intuition that the constituents of core semantics should stay the same with the original and pruned sentence where only context information differs. Similarly, the latter evaluates the distance between two lists of dependencies by considering the set difference in the count of each type of dependency relations based on the intuition that the relationships between words that constituent core semantics will ideally remain unchanged when context information is removed. We also consider the tree edit distance to detect suspicious issues. In particular, we parse the original translated and generated translated sentences into their constituency parse trees and then calculate the edit distance (i.e., Levenshtein distance) between the two trees. The tree edit distance determines how closely they match each other by calculating the minimum number of character edits (deletions, insertions, and substitutions) needed to transform one tree into the other.

The results are presented in Table 15. The first column lists the testing scenarios (i.e., twelve dataset categories \times two machine translation systems). The remaining columns list the precision results under different threshold values and evaluation metrics. We only present the results with the threshold value as 0, 6 and 12 (recommended in Section 6.7) due to page limit (details in our Appendix [72]). From Table 15, we find that the constituency tree-aware bag-of-words metric and dependency tree-aware bag-of-words metric achieve better precision performance slightly against the original bag-of-words metric. For example, the constituency tree-aware bag-of-words metric improves the precision by 0.5% and 1.6% with $t = 0$ when testing Google Translate and Bing Microsoft Translator. We also find that the original bag-of-words metric detects much more erroneous issues against other metrics. For example, the original bag-of-words metric finds 1261 erroneous issues with $t = 6$ when testing Google Translate, 1126 and 695 more than other metrics. Besides, we find that the tree edit metric does not achieve a better performance against our bag-of-words metric. The precision does not increase and the number of reported issues does not decrease when we set a larger threshold value. In other words, the threshold has no effect on the results when we consider the edit distance on the constituency parser tree. The possible reason lies in the difference in the design of our STP and existing studies. In particular, existing studies usually generate a sentence with the same structure as the original sentence and calculate the edit distance or relation distance (e.g., the number of each phrasal type or the number of each type of dependency relations) between two similar tree structures of the translations. However, STP generates a sentence with a different structure from the original sentence and it is improper to apply such metrics directly. Overall, the original bag-of-words metric is effective in detecting more translation errors with considerable precision, highlighting its substantial benefit in our work. In addition to the mentioned metrics, the results also demonstrate promising results with other tree-aware [21, 22] or semantic-aware metrics [54, 55], which is interesting to further explore in the future. Besides, it is recommended to consider recent neural models (e.g., BERT) as the metrics to detect translation errors in our work. However, whether to directly adopt existing trained models or to retrain using a large amount of labeled data, as well as how to measure the consistency of core semantics in sentence pairs, requires further exploration in the future, which goes beyond the scope of our work.

7 THREATS TO VALIDITY

To facilitate the investigation of potential threats and to support future work, we have made the relevant materials (including source code, and translation results) available at our project website [72]. Despite that, our study still faces some threats to validity, listed as follows.

The selection of the dataset might be biased. With respect to the representativeness of the dataset, all of them are crawled from the news networks. To mitigate the threat that the used dataset may not be representative of all real-world sentences, we adopt the dataset, which has been widely used

Table 15. Comparison results with different metrics

Category	t=0			t=6			t=12		
	STP	STP_edit	STP_con	STP	STP_edit	STP_con	STP	STP_edit	STP_con
GO	61.0%(1410/2310)	57.6%(1717/2982)	66.1%(926/1400)	71.4%(35/49)	60.2%(1571/2610)	11.1%(1/9)	N.A.	63.7%(1362/2137)	N.A.
GO BU	86.1%(2895/3364)	85.8%(3369/3925)	85.1%(1825/2145)	76.3%(58/76)	86.1%(3095/3595)	100.0%(17/17)	100.0%(6/6)	86.0%(2643/3072)	100.0%(6/6)
GO CU	60.5%(2574/4255)	59.1%(2882/4879)	59.1%(1657/2805)	62.2%(153/246)	60.4%(2688/4454)	84.6%(11/13)	N.A.	61.9%(2378/3844)	N.A.
GO SP	55.7%(1315/2361)	53.7%(1480/2758)	54.5%(806/1479)	80.1%(1017/1807)	55.0%(1386/2532)	70.6%(12/17)	100.0%(21/21)	57.0%(1236/2167)	100.0%(3/3)
GO TE	63.1%(1283/2033)	57.3%(1525/2663)	63.5%(805/1267)	71.0%(44/62)	59.8%(1414/2363)	75.0%(12/16)	N.A.	62.3%(1241/1993)	N.A.
GO TR	67.2%(3087/4591)	65.4%(3453/5278)	67.3%(2012/2988)	66.8%(2513/3762)	66.7%(3225/4833)	93.1%(27/29)	95.5%(21/22)	68.7%(2886/4199)	100.0%(5/5)
GO HE	51.0%(1579/3099)	48.4%(1727/3569)	51.5%(1056/2051)	84.9%(90/106)	49.8%(1617/3247)	75.0%(3/4)	N.A.	52.6%(1453/2764)	N.A.
GO LI	55.8%(1837/3295)	53.7%(2013/3746)	56.6%(1319/2330)	58.4%(1525/2613)	55.3%(1892/3424)	90.9%(20/22)	N.A.	57.4%(1722/3017)	N.A.
GO LE	62.2%(1848/2971)	56.7%(2010/3544)	63.6%(1120/1761)	63.7%(1569/2463)	59.0%(1893/3210)	83.3%(10/12)	66.7%(2/3)	62.7%(1718/2738)	N.A.
GO OP	61.1%(1553/2541)	58.2%(1797/3088)	64.3%(1048/1630)	61.6%(1204/1953)	60.2%(1660/2757)	100.0%(48/48)	N.A.	64.1%(1442/2251)	N.A.
GO PO*	82.5%(933/1131)	79.2%(1072/1353)	82.5%(570/691)	83.3%(707/849)	82.4%(985/1195)	100.0%(3/3)	N.A.	86.7%(800/923)	N.A.
GO BU*	87.2%(1094/1254)	84.4%(1310/1553)	89.1%(770/864)	88.6%(906/1022)	86.6%(1180/1363)	100.0%(1/1)	N.A.	89.1%(994/1116)	N.A.
GO SUM	64.50% (21408/33205)	61.90% (24355/39338)	65.00% (13914/21411)	65.20% (17291/26520)	63.50% (22606/35573)	83.90% (135/161)	81.10% (566/698)	96.20% (50/52)	100.00% (9/9)
BI PO	72.9%(1685/2310)	69.1%(2060/2982)	71.1%(996/1400)	70.9%(1257/1772)	70.9%(1851/2610)	11.1%(1/9)	60.9%(14/23)	73.8%(1578/2137)	N.A.
BI BU	66.6%(2129/3198)	62.5%(2460/3933)	71.3%(1364/1913)	67.3%(1700/2525)	63.9%(27/37)	100.0%(6/6)	62.2%(28/45)	64.6%(1965/3043)	N.A.
BI CU	75.1%(3127/4166)	70.9%(3464/4885)	77.2%(2295/2971)	74.8%(2604/3482)	79.9%(270/338)	96.6%(85/88)	88.6%(209/236)	76.9%(2980/3877)	N.A.
BI SP	51.5%(1197/2326)	47.8%(1314/2751)	55.4%(730/1317)	55.5%(926/1668)	50.3%(1244/2471)	62.5%(5/8)	78.7%(37/47)	100.0%(6/6)	N.A.
BI TE	64.8%(1338/2064)	60.7%(1615/2662)	64.3%(868/1349)	66.9%(1112/1661)	62.9%(1494/2377)	80.0%(12/15)	90.3%(65/72)	N.A.	N.A.
BI TR	69.9%(3185/4558)	69.1%(3651/5283)	73.6%(2263/3074)	72.1%(2720/3771)	87.3%(261/299)	92.7%(38/41)	91.2%(196/215)	100.0%(2/2)	N.A.
BI HE	51.1%(1550/3033)	48.6%(1738/3573)	50.0%(1047/2093)	50.5%(1288/2548)	64.4%(56/87)	50.4%(1620/3213)	86.4%(38/44)	N.A.	N.A.
BI LI	66.6%(2124/3190)	63.3%(2360/3731)	69.6%(1486/2136)	68.0%(1725/2536)	65.4%(76/89)	65.5%(2222/3391)	100.0%(3/3)	68.9%(1995/2896)	N.A.
BI LE	60.7%(1750/2881)	56.1%(1992/3551)	60.0%(940/1567)	58.2%(1305/2242)	92.8%(193/208)	57.8%(1844/3189)	84.4%(92/109)	92.2%(47/51)	84.6%(22/26)
BI OP	60.1%(1536/2556)	59.1%(1824/3084)	60.5%(1059/1750)	61.1%(1203/1970)	73.5%(25/34)	59.7%(1641/2747)	48.3%(23/47)	100.0%(1/1)	N.A.
BI PO*	80.7%(837/1037)	76.7%(1041/1357)	80.2%(542/676)	78.7%(661/840)	100.0%(17/17)	78.2%(938/1199)	N.A.	80.2%(736/918)	N.A.
BI BU*	67.7%(831/1227)	64.9%(1005/1548)	64.9%(519/800)	64.2%(623/970)	100.0%(5/5)	65.9%(889/1348)	85.7%(6/7)	66.4%(700/1054)	N.A.
BI SUM	65.40% (21289/32546)	62.30% (24524/39340)	67.00% (14109/21046)	65.90% (17124/25985)	64.10% (22703/35418)	84.90% (242/285)	84.30% (735/872)	92.80% (64/69)	85.20% (23/27)
									87.10% (74/85)

in recent studies [18, 21, 22]. Meanwhile, we randomly extract 1,000 sentences from ten different categories with varied complexity. As such, we believe the selection strategy may not be a key point to our user study.

The second threat to validity is that the adopted dependency parser might return inaccurate even wrong dependencies. Although, the most recent neural network-based parser made available by Stanford CoreNLP is adopted in our implementation, which parses about 100 sentences per second. However, the parser still has several wrong dependencies. For example, for the sentence “It is believed in the field that Amazon employs more PhD economists than any other tech company.”, the relation between “that” and “employs” is recognized as obj, which should be mark. Thus, we use the Universal Dependencies as our annotation scheme, which has evolved based on the Stanford Dependencies. Meanwhile, two authors manually check all the generated sentences to ensure the syntax and semantics legitimacy.

The third threat to validity lies in the selection of baselines. Recently, quite a few techniques have been proposed to validate machine translation systems. We first consider all available machine translation testing techniques (i.e., [18, 21, 22, 25, 54, 55]) according to [55] when we conducted the work. Among them, the implementation of [25] is not public (last accessed in February 2022) and we fail to reproduce this work after contacting the authors. The remaining techniques are included in our work, and the included techniques are the same as the most recent work [55]. Several variants may exist for the included techniques, and we select the best one according to the existing work (shown in Section 4.5).

The final threat to validity comes from the human labelling. In the phase of label construction, following existing studies [22, 25], 20 participants with both English and Chinese language backgrounds are responsible for deciding whether the reported issue contains a real erroneous translation. To alleviate the influence of potential bias (e.g., imprecise labelling results) introduced by particulars, an introduction to human labelling in the form of both documents and presentation videos is given to all participants, to ensure they fully understand the experimental procedure. To further reduce the bias when comparing different approaches, we blend all issues reported by STP and baselines to confirm the approach each issue belongs to remains unknown to the particulars. We then guarantee that each reported issue is checked by two independent participants to reduce human impact further. Besides, we invite participants to have an additional discussion to resolve the issues with inconsistent labels. The high kappa scores also indicate that the bias in human labelling may not be a key point in our experiments.

8 RELATED WORK

Our work aims to validate the robustness of machine translation systems via a novel metamorphic testing approach. We divide the related work of our paper into three parts: robust machine translation, machine translation testing and metamorphic testing.

8.1 Robust Machine Translation

Artificial intelligence (AI) systems have been widely adopted in our daily lives thanks to the success of deep learning. Despite much recent research, AI systems are not as robust as we might hope sometimes. For example, recent research has reported a variety of adversarial examples could mislead AI systems, resulting in fatal accidents, such as autonomous cars [13, 69] and object classifiers [63, 64]. Thus, a huge body of research effort has been dedicated for promoting the robustness of such AI software, focusing on testing [14, 27], debugging [37, 59] and training [32, 45].

In the NLP field, various tasks have achieved outstanding performance, such as text summarization [66], question answering [4] and incomplete utterance rewriting [35]. However, these NLP

systems still often fail catastrophically when the given inputs are adversarially perturbed slightly, which has encouraged researchers to investigate the robustness of NLP systems. For example, Jia et al. [26] propose an adversarial evaluation scheme for the Stanford Question Answering Dataset (SQuAD) and find no published open-source model is robust for paragraphs that contain adversarially inserted sentences. Besides, Mudrakarta et al. [41] analyze the robustness of three question answering models (i.e., mages, tables, and passages of text) by perturbing questions to craft a variety of adversarial examples. However, testing machine translation is more difficult, as one source sentence could have several correct target sentences, while the output of these systems is unique (e.g., the output of reading comprehension could be a specific person name).

As a typical NLP task, machine translation aims to automatically translate text from a source language to text in a target language. It is recently common practice to improve the robustness of machine translation software via adversarial machine learning, which aims to mislead machine translation systems with adversarial examples. In general, the generation of these adversarial examples falls into two categories: black-box and white-box manners. The white-box manner usually applies small perturbations that are jointly learned together with the NMT model, where complete knowledge of network structure and parameters of the machine translation model are available. For example, Chen et al. [8] propose a gradient-based method to craft adversarial examples guided by the translation loss of clean inputs. On the contrary, the black-box manner perturbs or paraphrases sentences without accessing the implementation of machine translation systems. For example, Zhang et al. [73] build black-box adversarial examples based on the round-trip translation, which assumes a practical adversarial example can naturally lead to a semantics destroying round-trip translation result. Such work can easily lead to invalid sentences, while this paper aims to generate syntactically and semantically correct test cases.

8.2 Machine Translation Testing

Machine translation testing aims to generate translation error-triggering sentences with syntactically and semantically correct [18, 21, 22, 54, 55, 60, 76]. Pesu et al. [46] present the first machine translation testing approach based on metamorphic testing without human intervention or reference translation. In particular, they design a novel metamorphic relation with the help of multiple intermediate languages, and adopt English as the source language and consider eight target languages (e.g., Chinese and Japanese). Furthermore, they extend the above work by introducing an additional metamorphic relation, which assumes that a small change to the source sentence should not have an impact on the overall structure of the target sentence [53].

After that, more metamorphism-based machine translation testing approaches get published in software engineering top conferences and journals, demonstrating the usefulness and potential of metamorphic testing for applications in the machine translation domain. He et al. [21] propose a novel structure-invariant testing technique (i.e., SIT) based on the hypothesis that similar source sentences should typically exhibit similar translation results. In particular, they generate similar source sentences by leveraging BERT to replace one word in an original sentence with semantically-similar and syntactically-equivalent words. They then report suspicious issues if the distance between the structure (e.g., constituency and dependency tree) of the translated original sentence and generated sentence is larger than a threshold. Furthermore, He et al. [22] propose a novel referential transparency test technique (i.e., RTI) based on the hypothesis that a piece of text should have similar translations in different contexts. In particular, they extract noun phrases from the original sentence as its referential transparency by analyzing the constituent structure. They then employ a BoW model to calculate the distance between the translation results of the original sentence and extracted phrases and report a suspicious issue if the distance is larger than a pre-defined threshold. Similarly, Ji et al. [25] introduce a constituency-invariant testing technique,

which indicates the constituency structure of a sentence should be similar to the sentence derived from it. We do not include this work in our paper because the core implementation about sentence generation is not publicly available. Gupta et al. [18] propose a novel pathological invariance metamorphic testing approach (i.e., PatInv) based on the hypothesis that the sentences of different meanings should not have the same translation. In particular, they generate syntactically similar but semantically different sentences by (1) replacing one word in an original sentence with a non-synonymous word via BERT (2) or removing a meaningful word or phrase from an original sentence via its constituency structure. They then suspect an erroneous translation if the translation results of the original and generated sentences are the same via a simple string comparison. Recently, Cao et al. [3] propose a semantic-based machine translation testing approach (i.e., SemMT) based on the hypothesis that regular expressions can effectively extract the semantics concerning logical relations and quantifiers in sentences. In particular, they first conduct the round-trip translation to collect the intermediate and translated sentences and transform the sentences into regular expressions with existing tools. They then capture semantic similarities over regular expressions by a set of regex-related metrics and report suspicious issues if the similarity is higher than a predefined threshold. However, instead of arbitrary sentence semantics, SemMT focuses on the semantics of quantifiers and logical relations, which is out of the scope of our work. Sun et al. [54] propose a novel approach to test machine translation (i.e., TransRepair) by combining mutation with metamorphic testing to detect inconsistency bugs and attempt to automatically repair reported errors in a black-box or grey-box manner. In particular, they conduct context-similar word replacement to generate a mutated sentence and assume that translation results from both the original sentence and its mutant should have a certain degree of consistency modulo the replaced word. Furthermore, Sun et al. [55] propose a novel word-replacement-based approach (i.e., CAT) to test machine translation on the top of TransRepair based on the insight that controlling the semantic difference between the replaced words is crucial in the impact of word replacement. In particular, they employ BERT to encode the sentence context during replacement and calculate context-aware semantic similarity to identify an isotopic replacement on the top of TransRepair.

The key idea of STP is conceptually different from most existing approaches [18, 21, 54], which replace a word or extract a phrase to generate new sentences. In contrast, STP assumes that the translation of a source sentence should be similar to the newly generated sentence generated by eliminating contextual information.

8.3 Metamorphic Testing

Metamorphic testing [6, 7, 49] is a well-known technique to check the functional correctness of various systems in the absence of an ideal oracle. Its key idea is to detect violations of domain-specific metamorphic relations among the inputs and outputs of multiple executions of the program under test. As an effective method to alleviate the oracle problem, metamorphic testing has seen successful applications in a variety of domains, including compilers [28, 31], database systems [33] and scientific libraries [70]. In addition, the effectiveness of metamorphic testing has also been shown repeatedly in AI systems testing because of its ability to test non-testable applications, such as search engines [77, 78], autonomous driving systems [56, 71] and classifiers [42, 65]. For example, Chen et al. [5] propose three novel metamorphic relations for testing question answering software by checking its behaviors on multiple recursively asked questions that are related to the same knowledge. To further eliminate false positives, Shen et al. [50] propose a precise metamorphic testing approach for question answering softwares, involving five sentence-level metamorphic relations (e.g., inserting a redundant sentence as a clause of the original question). Yu et al. propose [68] the first metamorphic approach to test image captioning systems, which attempt to generate a brief depiction of the salient objects in an image. They assume that the object names should

exhibit directional changes after object insertion and design two metamorphic relations (i.e., object appearance and singular-plural form). Besides, Huang et al. [23] propose a novel approach to evaluate the NLP test cases generated by metamorphic testing approaches based on similarity consistency and language naturalness. STP is a novel metamorphic testing approach for machine translation software based on the insight that the new sentence generated by eliminating contextual information should retain the core semantics of the original sentence.

9 CONCLUSION

In this paper, we present a widely applicable methodology, syntactic tree pruning, to detect erroneous translations for machine translation systems. In contrast to existing approaches, which rely on perturbing a word or extracting specific phrases (i.e., noun phrases) in natural sentences, STP assumes the pruned sentence should retain the core semantics of the original sentence. In particular, given an arbitrary sentence, STP generates new sentences via a novel core semantics-preserving pruning strategy on the syntactic tree level, and then pairs the generated and original sentence by the designed metamorphic relation. STP reports suspicious issues if the translation results break the consistency property in semantics via a bag-of-words model. Benefitting from the distinct concept, STP has reported a diversity of erroneous translations, most of which can not be found by existing approaches. As a result, STP successfully detect 5,073 translation errors for Google Translate and 5,100 translation errors for Microsoft Translator, respectively, with comparable precision (i.e., 64.5% and 65.4%). STP also achieve a recall of 74% when detecting erroneous translation for the 1,200 original sentences, improving state-of-the-art techniques by 55.1% on average.

In the future, we will generalize the core concept of STP to other implementation scenarios, such as inserting an unimportant word into a source sentence without changing crucial information. Meanwhile, it would be interesting to conduct a series of research on automated program repair for machine translation systems with the reported translation results.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments. This work is supported partially by the National Natural Science Foundation of China (61932012, 62141215, 62372228) and the China Scholarship Council (202306190117).

REFERENCES

- [1] BBC. 2022. The British Broadcasting Corporation (BBC) News Homepage. Site: <https://www.bbc.com/>. Accessed August, 2022.
- [2] Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. 1–13.
- [3] Jialun Cao, Meiziniu Li, Yeting Li, Ming Wen, Shing-Chi Cheung, and Haiming Chen. 2022. SemMT: A Semantic-based Testing Approach for Machine Translation Systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–36.
- [4] Danqi Chen and Wen-tau Yih. 2020. Open-domain question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'20)*. 34–37.
- [5] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. 2021. Testing Your Question Answering Software Via Asking Recursively. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE'21)*. 104–116.
- [6] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. 2020. Metamorphic Testing: A New Approach For Generating Next Test Cases. *arXiv preprint arXiv:2002.12543* (2020).
- [7] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, TH Tse, and Zhi Quan Zhou. 2018. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–27.
- [8] Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust Neural Machine Translation with Doubly Adversarial Inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 4324–4333.
- [9] Noam Chomsky. 2002. *Syntactic Structures*. Walter de Gruyter.
- [10] CNN. 2022. The Cable News Network (CNN) News Homepage. Site: <https://edition.cnn.com/>. Accessed August, 2022.

- [11] China Daily. 2022. China Daily News Homepage. Site: <https://www.chinadaily.com.cn/>.
- [12] IBM Cloud Docs. 2016. Machine Translation Tips. Site: https://cloud.ibm.com/docs/GlobalizationPipeline?topic=GlobalizationPipeline-globalizationpipeline_tips&locale=en. Accessed August, 2022.
- [13] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. 2019. Benchmarking Adversarial Robustness. *arXiv preprint arXiv:1912.11852* (2019).
- [14] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19)*. 477–487.
- [15] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On Adversarial Examples for Character-Level Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING'18)*. 653–663.
- [16] Google. 2022. Google Translate. Site: <https://translate.google.com>. Accessed August, 2022.
- [17] Stanford NLP Group. 2022. CoreNLP. Site: <https://stanfordnlp.github.io/CoreNLP>. Accessed August, 2022.
- [18] Shashij Gupta, Pinjia He, Clara Meister, and Zhendong Su. 2020. Machine Translation Testing Via Pathological Invariance. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. 863–875.
- [19] 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).
- [20] Pinjia He. 2022. Machine Translation Testing Toolkit. Site: <https://github.com/RobustNLP/TestTranslation>. Accessed August, 2022.
- [21] Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-invariant Testing for Machine Translation. In *Proceedings of the 42nd IEEE/ACM International Conference on Software Engineering (ICSE'20)*. 961–973.
- [22] Pinjia He, Clara Meister, and Zhendong Su. 2021. Testing Machine Translation via Referential Transparency. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE'21)*. 961–973.
- [23] Jen-tse Huang, Jianping Zhang, Wenxuan Wang, Pinjia He, Yuxin Su, and Michael R. Lyu. 2022. AEON: A Method for Automatic Evaluation of NLP Test Cases. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'22)*. 202–214.
- [24] Rodney Huddleston. 1984. *Introduction to the Grammar of English*. Cambridge University Press.
- [25] Pin Ji, Yang Feng, Jia Liu, Zhihong Zhao, and Baowen Xu. 2021. Automated Testing for Machine Translation via Constituency Invariance. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE'21)*. 468–479.
- [26] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*. 2021–2031.
- [27] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE'19)*. IEEE, 1039–1049.
- [28] Vu Le, Mehrdad Afshari, and Zhendong Su. 2014. Compiler Validation Via Equivalence Modulo Inputs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'14)*. 216–226.
- [29] Shaohua Li and Zhendong Su. 2023. Accelerating Fuzzing through Prefix-Guided Execution. *Proceedings of the ACM on Programming Languages* 7, OOPSLA1 (2023), 1–27.
- [30] Zuchao Li, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, Zhuosheng Zhang, and Hai Zhao. 2020. Explicit Sentence Compression for Neural Machine Translation. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'20)*, Vol. 34. 8311–8318.
- [31] Christopher Lidbury, Andrei Lascu, Nathan Chong, and Alastair F Donaldson. 2015. Many-core Compiler Fuzzing. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'15)*. 65–76.
- [32] Ji Lin, Chuang Gan, and Song Han. 2018. Defensive Quantization: When Efficiency Meets Robustness. In *International Conference on Learning Representations*.
- [33] Mikael Lindvall, Dharmalingam Ganesan, Ragnar Árdal, and Robert E Wiegand. 2015. Metamorphic Model-based Testing Applied on NASA DAT—an Experience Report. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE'15)*, Vol. 2. 129–138.
- [34] Lingua. 2022. The 20 Most Spoken Languages in the World in 2022. Site: <https://lingua.edu/the-20-most-spoken-languages-in-the-world-in-2022/>. Accessed August, 2022.
- [35] Qian Liu, Bei Chen, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020. Incomplete Utterance Rewriting as Semantic Segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 2846–2857.
- [36] John Lyons and Lyons John. 1995. *Linguistic Semantics: An Introduction*. Cambridge University Press.

- [37] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*. 175–186.
- [38] William C Mann and Sandra A Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text-interdisciplinary Journal for the Study of Discourse (Text & Talk)* 8, 3 (1988), 243–281.
- [39] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNlp Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL'14)*. 55–60.
- [40] Microsoft. 2022. Bing Microsoft Translator. Site: <https://www.bing.com/translator>. Accessed August, 2022.
- [41] Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhare. 2018. Did the Model Understand the Question?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 1896–1906.
- [42] Christian Murphy, Gail E. Kaiser, Lifeng Hu, and Leon Wu. 2008. Properties of Machine Learning Applications for Use in Metamorphic Testing. In *Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE'08)*. 867–872.
- [43] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2019. Transforming Complex Sentences into a Semantic Hierarchy. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 3415–3427.
- [44] Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*. 3956–3965.
- [45] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP'16)*. IEEE, 582–597.
- [46] Daniel Pesu, Zhi Quan Zhou, Jingfeng Zhen, and Dave Towey. 2018. A Monte Carlo method for metamorphic testing of machine translation services. In *2018 IEEE/ACM 3rd International Workshop on Metamorphic Testing (MET'18)*. IEEE, 38–45.
- [47] Randolph Quirk. 2010. *A Comprehensive Grammar of the English Language*. Pearson Education India.
- [48] Reuters. 2022. Reuters News Homepage. Site: <https://www.reuters.com/>. Accessed August, 2022.
- [49] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on software engineering (TSE)* 42, 9 (2016), 805–824.
- [50] Qingchao Shen, Junjie Chen, Jie Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. 2022. Natural Test Generation for Precise Testing of Question Answering Software. In *IEEE/ACM Automated Software Engineering (ASE'22)*.
- [51] Jieke Shi, Zhou Yang, Bowen Xu, Hong Jin Kang, and David Lo. 2022. Compressing Pre-trained Models of Code into 3 MB. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE'22)*. 1–12.
- [52] Punaardeep Sikka, Manmeet Singh, Allen Pink, and Vijay Mago. 2020. A Survey on Text Simplification. *arXiv preprint arXiv:2008.08612* (2020).
- [53] Liqun Sun and Zhi Quan Zhou. 2018. Metamorphic testing for machine translations: MT4MT. In *2018 25th Australasian Software Engineering Conference (ASWEC'18)*. IEEE, 96–100.
- [54] Zeyu Sun, Jie M Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic testing and improvement of machine translation. In *Proceedings of the 42nd IEEE/ACM International Conference on Software Engineering (ICSE'20)*. 974–985.
- [55] Zeyu Sun, Jie M Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. 2022. Improving Machine Translation Systems Via Isotopic Replacement. In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering (ICSE'22)*.
- [56] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. In *Proceedings of the 40th IEEE/ACM International Conference on Software Engineering (ICSE'18)*. 303–314.
- [57] Barak Turovsky. 2016. Ten Years of Google Translate. <https://blog.google/products/translate/ten-years-of-google-translate/>.
- [58] Deze Wang, Zhouyang Jia, Shanshan Li, Yue Yu, Yun Xiong, Wei Dong, and Xiangke Liao. 2022. Bridging Pre-trained Models and Downstream Tasks for Source Code Understanding. In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering (ICSE'22)*. 287–298.
- [59] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE'19)*. IEEE, 1245–1256.

- [60] Wenyu Wang, Wujie Zheng, Dian Liu, Changrong Zhang, Qinsong Zeng, Yuetang Deng, Wei Yang, Pinjia He, and Tao Xie. 2019. Detecting Failures of Neural Machine Translation in the Absence of Reference Translations. In *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Industry Track (DSN'19)*. 1–4.
- [61] Lei Wu, Steven CH Hoi, and Nenghai Yu. 2010. Semantics-preserving Bag-of-words Models and Applications. *IEEE Transactions on Image Processing (TIP)* 19, 7 (2010), 1908–1920.
- [62] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).
- [63] Chong Xiang, Charles R Qi, and Bo Li. 2019. Generating 3d adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 9136–9144.
- [64] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. 2019. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 6898–6907.
- [65] Xiaoyuan Xie, Joshua WK Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. 2011. Testing And Validating Machine Learning Classifiers By Metamorphic Testing. *Journal of Systems and Software (JSS)* 84, 4 (2011), 544–558.
- [66] Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 5021–5031.
- [67] Youdao. 2022. Youdao Translator. Site: <http://www.youdao.com>. Accessed August, 2022.
- [68] Boxi Yu, Zhiqing Zhong, Xinran Qin, Jiayi Yao, Yuancheng Wang, and Pinjia He. 2022. Automated testing of image captioning systems. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'22)*. 467–479.
- [69] Fuyuan Zhang, Sankalan Pal Chowdhury, and Maria Christakis. 2019. DeepSearch: Simple and Effective Blackbox Fuzzing of Deep Neural Networks. *arXiv preprint arXiv:1910.06296* (2019).
- [70] Jie Zhang, Junjie Chen, Dan Hao, Yingfei Xiong, Bing Xie, Lu Zhang, and Hong Mei. 2014. Search-based Inference of Polynomial Metamorphic Relations. In *Proceedings of the 29th IEEE/ACM International Conference on Automated Software Engineering (ASE'14)*. 701–712.
- [71] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. Deeproad: Gan-based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE'18)*. 132–142.
- [72] Quanjun Zhang and Haichuan Hu. 2023. STP Reproduction Artifacts. Site: <https://github.com/iSEngLab/STP>. Accessed December, 2023.
- [73] Xinze Zhang, Junzhe Zhang, Zhenhua Chen, and Kun He. 2021. Crafting adversarial examples for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL'21)*. 1967–1977.
- [74] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An Empirical Study on Tensorflow Program Bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'18)*. 129–140.
- [75] Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*. 2205–2215.
- [76] Wujie Zheng, Wenyu Wang, Dian Liu, Changrong Zhang, Qinsong Zeng, Yuetang Deng, Wei Yang, Pinjia He, and Tao Xie. 2019. Testing Untestable Neural Machine Translation: An Industrial Case. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering: Companion Proceedings (ICSE-Companion'19)*. 314–315.
- [77] Zhi Quan Zhou, Shaowen Xiang, and Tsong Yueh Chen. 2015. Metamorphic Testing for Software Quality Assessment: A Study of Search Engines. *IEEE Transactions on Software Engineering (TSE)* 42, 3 (2015), 264–284.
- [78] Zhi Quan Zhou, Shujia Zhang, Markus Hagenbuchner, TH Tse, Fei-Ching Kuo, and Tsong Yueh Chen. 2012. Automated Functional Testing of Online Search Services. *Software Testing, Verification and Reliability (STVR)* 22, 4 (2012), 221–243.
- [79] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*. 434–443.