Machine Translation Testing via Pathological Invariance

Shashij Gupta
Department of Computer Science and Engineering
IIT Bombay
India
shashijgupta@cse.iitb.ac.in

Pinjia He
Department of Computer Science
ETH Zurich
Switzerland
pinjia.he@inf.ethz.ch

Clara Meister
Department of Computer Science
ETH Zurich
Switzerland
clara.meister@inf.ethz.ch

Zhendong Su
Department of Computer Science
ETH Zurich
Switzerland
zhendong.su@inf.ethz.ch

ABSTRACT

Machine translation software has become heavily integrated into our daily lives due to the recent improvement in the performance of deep neural networks. However, machine translation software has been shown to regularly return erroneous translations, which can lead to harmful consequences such as economic loss and political conflicts. Additionally, due to the complexity of the underlying neural models, testing machine translation systems presents new challenges. To address this problem, we introduce a novel methodology called *PatInv*. The main intuition behind *PatInv* is that sentences with different meanings should not have the same translation. Under this general idea, we provide two realizations of PatInv that given an arbitrary sentence, generate syntactically similar but semantically different sentences by: (1) replacing one word in the sentence using a masked language model or (2) removing one word or phrase from the sentence based on its constituency structure. We then test whether the returned translations are the same for the original and modified sentences. We have applied PatInv to test Google Translate and Bing Microsoft Translator using 200 English sentences. Two language settings are considered: English→Hindi (En-Hi) and English→Chinese (En-Zh). The results show that PatInv can accurately find 308 erroneous translations in Google Translate and 223 erroneous translations in Bing Microsoft Translator, most of which cannot be found by the state-of-the-art approaches.

CCS CONCEPTS

• **Software and its engineering → Software verification and validation**; • **Computing methodologies → Machine translation**.

KEYWORDS

Testing, Machine translation, Pathological Invariance

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11…$15.00
https://doi.org/10.1145/3368089.3409756

ACM Reference Format:
Shashij Gupta, Pinjia He, Clara Meister, and Zhendong Su. 2020. Machine Translation Testing via Pathological Invariance. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3368089.3409756

1 INTRODUCTION

Due to recent improvements in the quality of translations from machine translation software, many people have started to rely on the technology in their daily lives. For example, people often read political news or articles from other countries and visit websites with content in various languages. According to [73], in 2016, Google Translate [2] had 500 million users and translated more than 100 billion words per day. The improvements that have lead to such widespread usage of machine translation systems are largely due to the advent of deep neural network, which are now frequently the core component of modern machine translation software. Neural machine translation (NMT) models are becoming as good as a human translator. Many recent NMT systems are approaching human-level performance in terms of "quality score" (defined by Google) [77] and "human parity" (defined by Microsoft) [29].

Despite these recent improvements, NMT systems are not as reliable as one might expect. Deep neural networks are brittle; often, when a neural network is evaluated on examples that differ intrinsically from the examples it was trained to model, it does not perform well [6]. NMT models are no exceptions; they can produce erroneous outputs when inputs are adversarially manipulated, such as upper casing some of the letters of the sentence or injecting grammatical errors (e.g., "I are studying"). However, it is not vital for the sentence to be syntactically wrong to fool an NMT model. There are numerous cases found where NMT models return erroneous translations for syntactically and semantically correct inputs, e.g., in WeChat, a messenger app with over one billion monthly active users [88]. When encountered by users, incorrect translations can have severe and harmful consequences such as financial loss, political conflicts, medical misdiagnoses, social issues, or personal safety threats [19, 47, 54, 55]. These side-effects motivate the need to create systems for ensuring the robustness of machine translation software.

Table 1: Examples of the errors detected by PatInv. The first two are from En->Hi and the last two are from En->Zh. The translation is erroneous for both sentences in the first and the third examples; the translation is erroneous for the source sentence in the second example; the translation is erroneous for the modified sentence in the fourth example.

Source Sentence	Modified Sentence	Translation (for both)	Translation Meaning
The situation at the southern border, which started as a crisis, is now a near system wide meltdown.	The situation at the southern border, which started as a crisis, is now a near system wide development.	दक्षिणी सीमा पर स्थिति, जो एक संकट के रूप में शुरू हुई थी, अब एक निकटवर्ती प्रणाली है।	The situation at the southern border, which started as a crisis, is now a nearby system.
I had a story to tell and I wanted to finish it, Draper says.	I had a story to tell and I wanted to finish it, says.	मेरे पास बताने के लिए एक कहानी थी और मैं इसे खत्म करना चाहता था।	I had a story to tell and I wanted to finish it.
The South has emerged as a hub of new auto manufacturing by foreign makers thanks to lower manufacturing costs and less powerful unions.	The South has died as a hub of new auto manufacturing by foreign makers thanks to lower manufacturing costs and less powerful unions.	由于较低的制造成本和较弱的，工会南方已经成为外国制造商新汽车制造的枢纽。	The South has became a hub of new auto manufacturing by foreign makers thanks to lower manufacturing costs and less powerful unions.
The threatened tariffs led to the European Union pledging countertariffs.	The threatened tariffs led to the Union pledging countertariffs.	威胁的关税导致欧盟承诺反关税。	The threatened tariffs led to the European Union pledging countertariffs.

However, it can be very difficult to test NMT models. First, testing deep learning models in general is quite different from testing traditional software, in which systems' core concepts or algorithms manifest in source code. Rather, the output of neural networks depends largely on the millions of parameters it has optimized over during training, making these models essentially black boxes. Second, recent testing approaches for artificial intelligence (AI) software, of which machine translation software is a subset, primarily target models with a small number of potential outputs, such as classifiers. In contrast, simply enumerating the possible outputs of most NMT models is an intractable problem [56], making machine translation systems incredibly difficult to test.

The current standard for automatic evaluation is BLEU [58], which is calculated by comparing the word sequences[1] in the system's output with a set of reference quality translations. One of BLEU's major weaknesses is that it does not truly understand the sentence meanings. Furthermore, it is necessary to provide accurate reference translations to determine such a metric for machine translation systems, which is prohibitive in many situations where such resources are limited.

Clearly, there is need for effective automated systems for testing machine translation software. This paper proposes a novel testing methodology, namely PatInv, whose main intuition is that sentences of different meanings should not have the same translations. We use this intuition to formulate approaches which automatically produce syntactically similar sentences with different meanings. In particular, PatInv generates sentences of different meaning through two approaches: 1) replacing one word in a sentence with a non-synonymous word and 2) removing a meaningful word or phrase from the sentence. The original and the newly generated sentences are fed to the translation system under test; if the translations are exactly the same, we report them as a suspicious issue. Our practical implementation of PatInv uses a masked language model based on BERT [20] to perturb words in a sentence and a constituency parser to identify core words and phrases. Thesaurus [3], WordsAPI [4] and the NLTK library [10] are then used to filter out synonyms and syntactically incorrect sentences.

[1] A maximum length of four words is common.

To evaluate the effectiveness of PatInv, we use it to test Google Translate and Bing Microsoft Translator on 200 English sentences from two article categories (politics and business) as input released by He et al. [30]. Without using the optional filtering step (i.e., filtering by sentence embeddings, Section 3.2.1), PatInv successfully reports 452 pathological invariants with 56.6% average precision. When this optional filtering mechanism is employed, PatInv can report 28 pathological invariants with 100% accuracy. Table 1 show some of the erroneous translations found by PatInv. We find that, due to its conceptual novelty, PatInv detects a unique set of errors not found by other approaches. All the reported pathological invariants and source code have been released [65]. The main contributions of this paper are as follows:

- We introduce a novel, widely applicable black-box methodology to validate machine translation software.
- We describe a practical implementation to generate syntactically similar but semantically different sentences using BERT, using various filters to avoid generating invalid test cases.
- We evaluate the model on 200 sentences for Google Translate and Bing Microsoft Translator with two translation settings.
- We successfully find 308 erroneous translations in Google translate and 223 in Bing Microsoft Translator with high accuracy, most of which cannot be found by the state-of-the-art techniques.

2 A MOTIVATING EXAMPLE

During the 2018 Winter Olympic Games, the Norwegian team's cooking facilities intended to order 1500 eggs. The games were held in South Korea and thus they need to place an order in Korean in a local grocery. The chefs turned to Google Translate for help translating their order. To their surprise, a truck load of eggs fell upon their kitchen: Google Translate mistakenly translated 1500 eggs into 15000 eggs in Korean [5].[2]

[2] In theory, this erroneous translation can be detected if PatInv replaces "1500" with "15000." At the moment, this error has already been fixed. In practice, it is possible that BERT does not recommend "15000" as a fill word and thus may not find the translation error.

This is real life translation error which caused inconvenience and could have lead to a huge financial loss. Still, translation errors can have much more serious and harmful consequence [19, 47, 54, 55]. For example, in 2018, due to a machine translation error, Israel's prime minister's compliment to Israel Eurovision winner Netta went from "you are a real darling" to "you are a real cow" [66], leading to embarrassment and misunderstanding. In 2017, a Palestinian man was arrested by police after posting "Good morning" on Facebook in Arabic; the post was wrongly translated to "attack them" in Hebrew and "hurt them" in English [19]. As more and more people have started to rely on machine translation, building robust machine translation software is of significant importance. To enhance the robustness of machine translation software, this paper introduces PatInv, a novel and widely-applicable methodology for testing machine translation.

3 APPROACH AND IMPLEMENTATION

This section discusses the high-level idea of PatInv and provides two implementations (PatInv-Replace and PatInv-Remove). Recall that the main intuition behind PatInv is that sentences with different meanings should not have the same translations. Hence, we find issues such that both sentences have different meanings but result in the same translation by the model under test. The input for PatInv is a list of unlabeled, monolingual sentences, while its output is a list of suspicious issues. For each input sentence, either no issue is detected or a list of suspicious issues is returned. A suspicious issue consists of 1) the original sentence 2) a generated (semantically different) sentence and 3) their shared translation. Three types of translation errors can cause an issue: (1) The original sentence has an erroneous translation (2) the generated sentence has an erroneous translation (3) both the sentences have erroneous translations.

Figure 1 shows the overview of PatInv for both implementations; we use a source sentence from our dataset as the example input. In summary, PatInv carries out the following four steps:

(1) *Generating syntactically-similar sentences.* For each unlabelled sentence, we generate a list of syntactically similar sentences by modifying a word or a phrase in the sentence.
(2) *Filtering via syntactic and semantic information.* We filter out those test cases in which the newly generated sentence has the same meaning as the original sentence.
(3) *Collecting target sentences.* We feed the original and the newly generated sentences to the machine translation system under test and collect their target sentences (i.e., translations).
(4) *Detecting translation errors.* Translations of the newly generated sentences are compared with the translation of the original sentence. If any translation matches that of the original sentence, the pair is reported as a potential issue which may contain an erroneous translation.

3.1 Step 1: Generating Syntactically-Similar Sentences

Given a sentence in a source language (i.e. a language we translate *from*), we need to generate structurally similar sentences that have a different meaning as input for PatInv. Specifically, we want to generate *syntactically* similar but *semantically* different sentences. Several approaches can be taken to produce a list of variations from a single sentence. For example, we can randomly interchange words. However, sentences generated using this method are not necessarily syntactically similar to the original sentence. We have found the following two approaches effective for the task:

(1) Replace a word in the original sentence with a word of the same part-of-speech (e.g., noun or adjective) but different meanings (PatInv-Replace)
(2) Remove a word or phrase from the original sentence sentence. (PatInv-Remove)

3.1.1 PatInv-Replace. In this approach, we replace a single word in our original sentence with another word of the same part-of-speech to generate a new sentence with different meaning. For example, in Fig. 1 we replace the word "completely" with the word "slightly" which leads to a syntactically similar but semantically different sentence. In practice, we replace a given word with k new words to generate k new sentences. If there are m words in a sentence that can be replaced, our method will produce a list of $k \cdot m$ new sentences. We narrow the scope of words that can be replaced to avoid strange linguistic phenomenon, which can lead to false positives later in the testing process. Specifically, only nouns, verbs, adverbs, adjectives, and possessive pronouns are taken as candidates for replacement. Additionally, we elect not to replace *stopwords* (commonly used words, e.g., has, were) as we empirically find that replacing those words often leads to syntactically or semantically incorrect sentences. In our approach, we use the set of stopwords provided by NLTK [10][3].

Now we discuss the problem of selecting candidates to replace a given word in the original sentence. We note that it is important to find words which fit well in the context to ensure that our generated sentence makes sense. This criterion eliminates the method of choosing words just by high word-embedding [60] vector similarity, as standard word-embeddings do not have any contextual information. For example, "back" and "front" have high word-embedding vector similarity but if we replace "back" in "Looking back at the experience" with "front" in "Looking front at the experience," we end up with a sentence unlikely to occur in the English language.

A model that is well suited for this task is the Masked Language Model (MLM) [51], inspired by the Cloze Task [71]. Specifically, given a string containing a single "masked" word as input, an MLM predicts what word belongs in the masked position. The MLM returns a set of words as potential replacements. The key benefit of using an MLM is that it takes into account the context of the sentence when predicting the masked token. If the MLM is good, then it is highly likely that the predicted words will lead to meaningful sentences. Note that we also check whether the words predicted by the MLM belong to NLTK stopwords and likewise elect to not use these words as candidate replacements.

For MLMs, there are several out-of-box options available, which are built on top of language representation models such as ELMo [61], GPT-2 [64], and BERT [20]. While training one's own MLM is also possible, it would require huge amounts of data and vast computational resources to even match the caliber of a system built

[3] There is no universally-used list of stopwords available

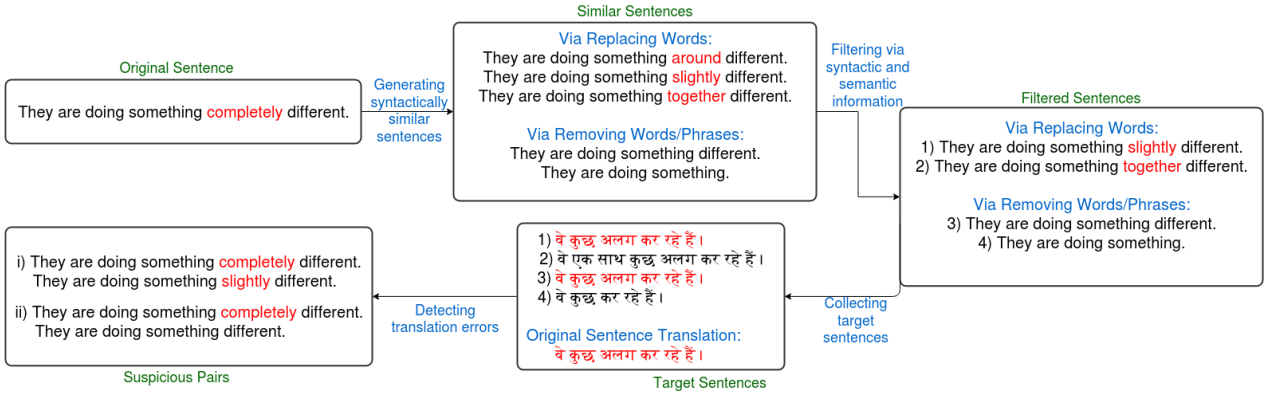


Figure 1: Overview of PatInv (English→Hindi in Google Translate)

on the aforementioned state-of-the-art models, with no guarantee of producing something better. Since a good MLM is critical for our approach, we elect to use the state-of-the-art options. Specifically, in our implementation, we use BERT, a language representation model developed by Google AI. Interestingly, the masked language task was one of the two main tasks used to train the base model, giving us further reason to believe BERT is well-suited for this task.

3.1.2 PatInv-Remove. In this approach, we remove a word or a phrase in our original sentence to generate new sentences. Our main goal is to remove something meaningful from a sentence. For example, Fig. 1 shows the complete removal of a word from the sentence. However, choosing words or phrases to remove is not a trivial problem. A naive implementation of this approach may remove all pairs of consecutive words, then triples of consecutive words, etc. While this implementation surely will not miss any error that can be detected with this approach, it also leads to many false positives.

Identifying meaningful phrases in a sentence can be aided by using its constituency structure, i.e. the syntactic structure of a sentence. Constituency structures show how a word or a group of words form different units in a sentence, such as noun phrases or verb phrases. This structure can be identified using a constituency parser, which derives a representative parse tree by splitting a sentence according to a set of phrase structure rules [17] defined by a context-free grammar. There are several approaches for constituency parsing; currently, the state-of-the-art model uses shift-reduce parsing [91]. This model parses grammar non-terminals from left to right in a stack-like manner to produce a complete set of relations. An implementation of [91], which we use, is available through Stanford's CoreNLP library [49].

Once a sentence is parsed, we identify all noun phrases (NP), verb phrase (VB), prepositional phrase (PP), and adverb phrase (ADVP) as candidates for removal, which can be done by recursively moving through the constituency parse tree. We then form a set of new sentences by removing each word/phrase in this set from the original sentence. Specifically, each removal corresponds to a new sentence.

3.2 Step 2: Filtering by Syntactic and Semantic Information

For our two sentence generation approaches, we apply filters, i.e. remove some of the generated sentences, to reduce potential false positives. Here we explain the two sets of filters.

3.2.1 PatInv-Replace. We find that some sentences generated using the masked language model (MLM) can be too semantically similar to the original sentence or are syntactically incorrect. To address this issue, we introduce three filtering mechanisms:

1) *Filtering out synonyms.* An MLM can predict words that are synonyms of the original word. As we want to generate sentences with meanings different from that of the original sentence, having a word replaced by its synonym (e.g., "good talk" and "nice talk") may lead to false positives. Therefore, we filter out those cases where the predicted word is a synonym of the original word. This is done using Thesaurus.com [3] and WordsAPI [4] dictionary.

Specifically, we crawl Thesaurus.com to generate a list of synonyms for each of the predicted words. We likewise used the api provided by WordsAPI to find synonyms. We note that WordsAPI also includes hierarchical information, such as knowing that a hatchback is a type of car, a finger is a part of a hand, etc. In our implementation, we check whether the predicted word is a synonym or type of the original word. We filter out these cases.

Interestingly, there are a few cases where similar words cannot be detected by either of the services mentioned. For example, "confirmed" and "affirmed" can have the same meaning in certain contexts but neither thesaurus nor WordsAPI tag them as synonyms. Similarly, "say" and "said" are conjugations of the same verb but are not identified as synonyms. In general, we want to eliminate cases where the base of the predicted word is a synonym of or equivalent to the base of the original word. As a solution to this problem, we use NLTK's WordNet Lemmatizer to get the base form of the verb. WordNet Lemmatizer lemmatizes using WordNet's built-in Morphy function. It returns the input word unchanged if it cannot be found in WordNet. A similar situation occurs for words with the same stem, i.e. "select" and "selective"; when both are used as adjectives, replacing one with the other in a sentence will generally not lead to a different semantic meaning.

We used NLTK's snowball stemmer to identify the word stem and elimante cases where the stem of the predicted and original words are the same.

2) *Filtering by constituency structure.* In practice, replacing a word in a sentence with a word of a different part-of-speech can lead to semantically or syntactically incorrect sentences. For example, in figure 1 when "completely" is replaced by "around" we end up with a grammatically incorrect sentence. Such cases can be avoided by ensuring the original and generated sentence have the same constituency structure. For this task, we again use the implementation of [91] available in NLTK's CoreNLPParser to generate a constituency parse. We then calculate the distance between the constituency parse trees of the original and generated sentence. We use tree-edit distance as our distance measurement, as implemented in python's APTED library. Between each original and generated sentence, we ensure tree edit distance is equal to 1, i.e. only one leaf node, the replaced word, has changed. Note that a distance of 1 means the part-of-speech of the replaced word has not changed since part-of-speech tags correspond to nonterminals in the tree.

3) *Filtering by sentence embeddings.*[4] As an additional precaution to ensure generated sentences do not have the same meaning as the original sentence, we ensure the "distance" between sentences is sufficiently large. To mimic a linguistic notion of distance, we take the vector distance between the embedding of the original and generated sentences. There are several out-of-the-box options available for generating sentence embeddings, such as the Universal Sentence Encoder [13] and BERT. While training a sentence encoder would also be possible, we note that these state-of-the-art models are readily available and have proven to perform well on the task.

In PatInv, we used the Universal Sentence Encoder, a model for encoding sentences into vectors based on encoders parameterized by deep neural networks. The Universal Sentence Encoder is freely available on TensorFlow Hub. The similarity between two sentences is then calculated as the cosine similarity of the two sentence embeddings. We filter out the generated sentences with similarity (to its original sentence) greater than a given threshold, which is a tunable parameter.

While there is some redundancy in the above filtering steps, using all three ensures a minimal number of false positives, which is critical for reducing manual effort.

3.2.2 PatInv-Remove. Sentences generated using PatInv-Remove may face similar problems, i.e. they may be nonsensical or retain their original meaning. Interestingly, we find that removing small words in the original sentence often leads to the latter problem. This is the case even when the word is not explicitly a stopword. For example, in our experiments, we remove the word "found" from the sentence, "Whether there is a political gain to be found in dishonoring a servant" and the translation system returns the same results. While it can be argued that there is an erroneous translation here, the sentence after word removal does not have a clearly different meaning. We empirically find that removing words of character length only greater than 6 alleviates this issue. To find a suitable lower bound, we tune this parameter on the Politics dataset with En-Hi language setting from "0" to "10" with step size "1." "6" was found to be the lower bound that led to decent precision and number of pathological invariants. In our experiments, we found that this lower bound also led to high-quality results in other experimental settings.

[4] optional step to increase precision as discussed in section 4

3.3 Step 3: Collecting Target Sentences

Once modified sentences have been generated, we must then feed them to the translation system under test and retrieve their translations. In the case of Google Translate and Bing Microsoft Translator, which we evaluate PatInv on, we use the Google API and Bing API, which return the same results as their respective web interfaces. Specifically, sentences in a source language are fed to the translation system and their translations, in a chosen target language, are returned. We do this for each of our original sentences and all of the generated sentences.

3.4 Step 4: Detecting Translation Errors

Finally, once all translations are collected, we must search for erroneous translations. This component of our methodology is quite straightforward; as we have generated sentences that differ in meaning from their respective original sentences, if the translations of the original and generated sentences are the same, we have good reason to suspect an erroneous translation. This check can be done via a simple string comparison which checks for equality. Fig. 1 shows an example of such an erroneous translation pair.

4 EVALUATION

In this section, we evaluate PatInv by applying both of its implementations, PatInv-Replace and PatInv-Remove, to Google Translate and Bing Microsoft Translator with real-world sentences as input. Our main research questions are:

- RQ1: How effective is PatInv at finding erroneous translations?
- RQ2: Where do the false positives and false negatives come from?
- RQ3: Can PatInv find erroneous translations that existing approaches cannot find?
- RQ4: How efficient is PatInv?
- RQ5: How does one tune the parameters of PatInv in practice?

In answering these questions, we show that both implementations are effective in finding erroneous translations and most of the erroneous translations found cannot be detected by existing approaches. Additionally, the approach is efficient and is easy to use in practice.

4.1 Experimental Setup

Environment. All experiments are run on DALCO r2264l6g Rackmount Server with 2x12 Core Intel Xeon E5-2650 v4 2.2GHz Processor, 256GB DDR4 2400MHz ECC REG Server Memory and 8x nVidia RTX 2080TI 11GB GPU Processor.

Dataset. To evaluate the performance of PatInv and align with the SOTA, we use the dataset released in our previous paper [30]. Specifically, this dataset contains 200 English sentences crawled

Table 2: Statistics of input sentences for evaluation. Each corpus contains 100 sentences.

Corpus	# of words/ sentence	Average # of words/sentence	# of words	
			Total	Distinct
Politics	4-32	19.2	1,918	933
Business	4-33	19.5	1,949	944

from CNN (Cable News Network).[5] The dataset consists of articles from two categories: Politics and Business, where each dataset contains 100 sentences. More statistics of the released dataset is illustrated in Table 2.

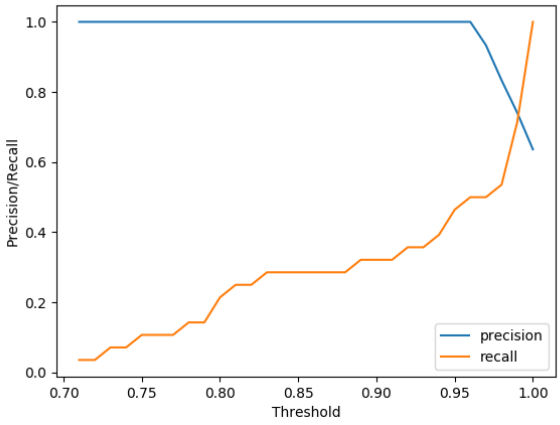
Labeling. The output of our approach is a list of suspicious issues, each of which consists of a pair of sentences where the first is the original sentence and the second is our generated one. We manually label each issue reported by PatInv. Two of the authors inspect all the results separately and discuss the labels for all the reported issues until convergence. Specifically, for PatInv-Replace, we first check whether the generated sentence in the issue contains any syntax or semantic errors. If so, the issue will be labeled as a false positive (examples are in Table 7). Otherwise, we check whether the original sentence and generated sentence are semantically different. If not, the issue will be labeled as a false positive (examples are in Table 6). Otherwise, we label the issue as a true positive. Then, we decide which sentence(s) in this issue contains translation error(s). For PatInv-Remove, the labeling process is similar to PatInv except that we will not label an issue as false positive because of syntax/semantic errors in the generated sentence.

Comparison. We compared PatInv with two state-of-the-art approaches: SIT [30] and TransRepair [69]. Both approaches are based on the intuition that similar source sentences should have similar translations. For SIT, we directly used the source code released by the authors. For TransRepair, the authors did not release their source code due to industrial confidentiality. Thus, we implement TransRepair by carefully following the explanations in their paper and consulting the work's main author for key implemenation details. We re-tune the parameters for both SIT and TransRepair following the same strategies introduced in the papers. In [69], 0.9 is used as the minimum cosine similarity of word embeddings to generate word pairs. In our experiment, we lower the threshold to 0.8 because otherwise, we were unable to reproduce the quantity of pairs reported in the original paper, i.e., using 0.9 as the threshold yielded two magnitudes fewer word pairs than reported by TransRepair. The implementation of PatInv, SIT, and TransRepair are all released for reuse [65].

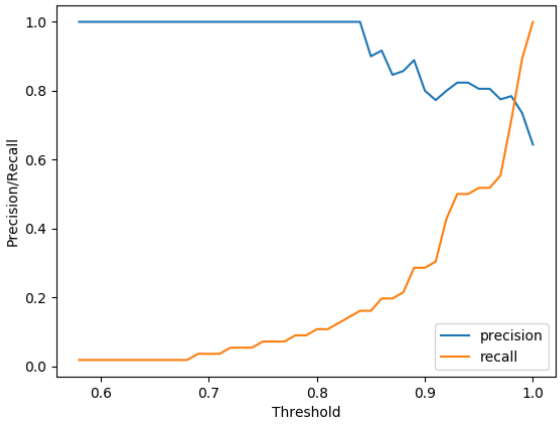
4.2 The Effectiveness of PatInv

Both of our approaches aim to automatically find erroneous translations in machine translation software using unlabelled sentences. Thus the effectiveness of both of our approaches will be determined by (1) How many erroneous translations an approach can find (2) With what precision an approach can find translation errors. In this section, we show the effectiveness of our model by testing Google translate and Bing Microsoft Translator for two translation settings (English→Hindi and English→Chinese). We calculated our results separately for each domain in the evaluation dataset.

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



(a) Politics Dataset



(b) Business Dataset

Figure 2: Precision v/s Recall trade-off threshold curve for En-Hi in Google Translate

4.2.1 Effectiveness of PatInv-Replace. Recall that for PatInv-Replace, we use the similarity between the sentence embedding vectors given by the Universal Sentence Encoder to check whether the original sentence and newly generated have same meaning. For different thresholds we filter out issues in which the pair of sentences have similarity less than the chosen threshold; we calculate the following metrics for the remaining issues:

- **precision**: the number of erroneous issues divided by total number of reported issues.
- **recall**: the number of erroneous issues with similarity less than threshold divided by total number of erroneous issues when there is no threshold.
- **F1-score**: the harmonic mean of precision and recall.

Table 3: Precision, recall and F1-score of PatInv-Replace.

Google Translate

Setting	Dataset	TP	FP	TN	FN	Prec.	Rec.	F1
En-Hi	Politics	28	16	0	0	0.64	1.0	0.78
En-Hi	Business	50	18	13	6	0.74	0.89	0.81
En-Zh	Politics	92	67	0	0	0.58	1.0	0.73
En-Zh	Business	90	54	0	0	0.63	1.0	0.77

Bing Microsoft Translator

Setting	Dataset	TP	FP	TN	FN	Prec.	Rec.	F1
En-Hi	Politics	1	0	12	1	1	0.5	0.67
En-Hi	Business	1	0	12	0	1	1	1
En-Zh	Politics	92	81	0	0	0.53	1	0.69
En-Zh	Business	91	74	0	0	0.55	1	0.71

Table 4: The number of remaining sentences after each step.

(a) PatInv-Replace

Step No.	Dataset	No. of sentences
1	Politics	15675
1	Business	16091
2	Politics	11639
2	Business	12604

(b) PatInv-Remove

Step No.	Dataset	No. of sentences
1	Politics	2047
1	Business	2091
2	Politics	1619
2	Business	1660

(c) State-of-the-art approaches

Toolname	Dataset	No. of sentences
SIT	Politics	2,068
SIT	Business	1,957
TransRepair	Politics	209
TransRepair	Business	196

Results. Table 4 (a) shows the number of total sentences (original sentences + newly generated sentences) after step 1 and step 2 of PatInv-Replace. For each translation setting, we count the total number of issues reported for each dataset. We manually check and verify each issue, reporting how many of these issues are erroneous. Ultimately, we were able to achieve precision comparable with those of the state-of-the-art methods.

We categorize erroneous issues by which of the two sentences (original or generated) contain a translation error. We categorize non-erroneous issues into two categories: (1) Issues in which both sentences have the same meaning. (2) Issues in which the second sentence is semantically or syntactically incorrect.

For each dataset (business and politics), we tune PatInv's available parameter, i.e., the threshold similarity for sentence embeddings, and report the precision, recall and F1-score corresponding each threshold. For the threshold with the best F1-score, we report True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN). Table 3 shows results for Google Translate and Bing Microsoft Translator. The results shows that we were able to achieve high recall with precision comparable to the state-of-the-art approaches as discussed in section 4.4. We note that reducing the threshold does not necessarily increase the precision as a lower threshold does not eliminate semantic or syntactic errors; this was specifically apparent in the En-Zh results for the Bing-Business dataset, as shown in Table 10. We additionally find that we can achieve 100% precision given a low enough threshold. We were able to report 28 erroneous issues in total in this case. The precision-recall trade-off curve is illustrated in Fig. 2.

Table 5: Result of PatInv-Remove for En-Hi.

Translator	Dataset	TP	FP	Precision
Google	Politics	3	2	0.60
Google	Business	3	1	0.75
Bing	Politics	0	1	0
Bing	Business	2	3	0.4

4.2.2 Effectiveness of PatInv-Remove. **Evaluation Metric.** Likewise, the output of this approach consists of a list of issues where the first sentence is the original sentence and the second sentence is a generated one. The generated sentence is obtained by removing something meaningful (a word or a phrase) from the sentence. As in our first approach, an issue is only reported when both the original and generated sentences have the same translations. We note that there are two potential cases for each issue: (1) something meaningful was removed from the original sentence, in which case the issue is a true positive (TP) (2) nothing important or meaningful was removed from the sentence, in which case the issue is a false positive (FP).

Results. Table 4(b) shows the number of total sentences (original sentences + generated sentences) after step 1 and step 2 of PatInv-Remove. For each translation pair, we count the total number of issues reported for each dataset and manually verified how many of these issues were erroneous. Table 5 shows results for Google Translate and Bing Microsoft Translator for English→Hindi. The results for English→Chinese are shown in Table 10, where they are compared with state-of-the-art approaches in section 4.4. The results shows that we were able to achieve precision comparable to state-of-the-art approaches. For each erroneous issue, we label which sentence among the pair has an erroneous translation.

4.3 False Positives and False Negatives

Despite applying multiple filters, there were still some false positives in both of our approaches. Specifically, we encountered five sources of false positives. 1) *The generated sentence is syntactically or semantically incorrect* as shown in Table 7. Note that our approach minimizes these false positives; we have used a masked language model built on top of BERT, a high-performing language representation model. Employing this strategy should lead to very few syntactically or semantically incorrect sentences. 2) *The replaced word is a synonym of the original word.* Despite combining multiple large online databases to filter out synonyms, such cases still occurred. The main source of such false positives was when

Table 6: False Positives for PatInv-Replace due to same meaning.

Source	And this would take us closer to our goal of a truly European banking sector.
Newly Generated	And this would allow us closer to our goal of a truly European banking sector.
Translation (En→Hi)	और यह हमें वास्तव में यूरोपीय बैंकिंग क्षेत्र के हमारे लक्ष्य के करीब ले जाएगा।
Translation Meaning	And this would take us closer to our goal of a truly European banking sector.
Source	Nielsen said the most serious cyber threats are those aimed at the heart of democracy.
Newly Generated	Nielsen said the most serious cyber threats are those posed at the heart of democracy.
Translation (En→Hi)	नीलसन ने कहा कि सबसे गंभीर साइबर खतरे लोकतंत्र के केंद्र में हैं।
Translation Meaning	Nielsen said that the most serious cyber threats are at the heart of democracy.

Table 7: Syntactically/semantically wrong sentences generated by PatInv-Replace.

Source	They left out that the pilots were not trained to handle it.
Newly Generated	They leaves out that the pilots were not trained to handle it.
Translation (En→Hi)	उन्होंने कहा कि पायलटों को इसे संभालने के लिए प्रशिक्षित नहीं किया गया था।
Translation Meaning	They said that the pilots were not trained to handle it.
Source	The key to accepting praise at work is to show you received it and appreciate it.
Newly Generated	The key to accepting praise at work is to take you received it and appreciate it.
Translation (En→Hi)	काम पर प्रशंसा स्वीकार करने की कुंजी यह है कि आप इसे प्राप्त करें और इसकी सराहना करें।
Translation Meaning	The key to accepting praise at work is to receive it and appreciate it.

Table 8: False Positives generated by PatInv-Remove.

Source	First, proper training for pilots who are flying new aircraft is crucially important.
Newly Generated	First, proper training for pilots who are flying new aircraft is crucially .
Translation (En→Hi)	पहला, नए विमान उड़ाने वाले पायलटों के लिए उचित प्रशिक्षण महत्वपूर्ण है ।
Translation Meaning	First, proper training for pilots who are flying new aircraft is important.
Source	Three months later, Holmes was indicted on federal wire fraud charges and stepped down from Theranos.
Newly Generated	Three later, Holmes was indicted on federal wire fraud charges and stepped down from Theranos.
Translation (En→Zh)	三个月后，霍姆斯因联邦电汇欺诈指控被起诉，并辞去了 Theranos 的辞职。
Translation Meaning	Three months later, Holmes was indicted on federal wire fraud charges and stepped down from Theranos.

our databases did not identify words as synonymous due to differences in tense or plurality from the relevant word registered in the database. While we identified word stems with available tools, we note that this approach does not always work as expected. For example, NLTK's snowball stemmer returns the stems "univers" for "universities" and "colleg" for "college," which then cannot be used with a synonym database as they are not standalone words. 3) *The replaced word in a sentence does not carry much meaning.* We find that replacing some words does not change the meaning of the sentence because the replaced word was not vital for the meanning of the sentence. Examples of such cases has been provided in Table 6. To reduce such cases, we remove generated sentences where sentence embedding similarity is high, implying semantic similarity between the original and generated sentences. 4) *Something meaningful is not removed from the sentence.* We find that only removing words and phrases with length > 6 helps to minimize these cases. 5) *The translation system successfully "predicts" the removed word/phrase.* In particular, it is possible that something meaningful is removed from the original sentence and a syntactically- or semantically-incorrect sentence is generated, but the translation system still returns the removed word translation for the generated sentence. Examples are illustrated in Table. 8.

Table 9: False Negatives generated by PatInv.

Source	The key to accepting praise at work is to show you received it and appreciate it.
Newly Generated	The key to accepting praise at work is to think you received it and appreciate it.
similarity	0.993
Source	Actress Jennifer Lawrence is also expected to star as Holmes in a movie based on Bad Blood.
Newly Generated	Actress Jennifer Lawrence is also expected to star as Holmes in a movie titled on Bad Blood.
similarity	0.990

These false positives can be further reduced to make our approaches more efficient. For example, syntactically incorrect sentences can be eliminated by using a good grammar checker. To eliminate cases of semantically incorrect sentences, we can train a corresponding classifier. We note that there were also some false negatives produced in "filtering by sentence embeddings" step of PatInv-Replace, which are shown in Table 9.

4.4 Errors Reported by Different Approaches

In this section, we compare PatInv with SIT and TransRepair in terms of precision and overlap of erroneous translations found. The comparison is done with En-Zh language setting because SIT only provides En-Zh implementation.

Precision. First, we compare the precision of PatInv with SIT [30] and TransRepair [69] because precision is used as a core evaluation metric in both papers. The results are presented in Table 10. PatInv-Remove achieves comparable precision compared with existing approaches. When the threshold for filtering by sentence semantics is 1.0 (i.e., no filtering in this step), PatInv-Replace obtains more erroneous issues with slightly lower precision. When we decrease the threshold (more generated sentences are filtered out), the precision of PatInv-Replace will increase accordingly. For

Table 10: Precision and the number of erroneous issues using different threshold values (En-Zh)

	PatInv-Replace				PatInv-Remove	SIT	TransRepair
	1.00	0.99	0.95	0.90			
Google-Politics	57.9% (92)	68.5% (61)	72.2% (13)	75.0% (3)	42.9% (6)	65.3% (34)	64.2% (45)
Google-Business	62.5% (90)	70.0% (56)	67.4% (31)	76.7% (23)	76.9% (10)	64.7% (33)	61.1% (22)
Bing-Politics	53.1% (92)	60.0% (63)	61.1% (22)	66.7% (6)	61.5% (16)	70.5% (36)	70.5% (24)
Bing-Business	55.2% (91)	61.8% (60)	53.5% (23)	65.4% (17)	58.4% (14)	62.7% (32)	55.0% (22)

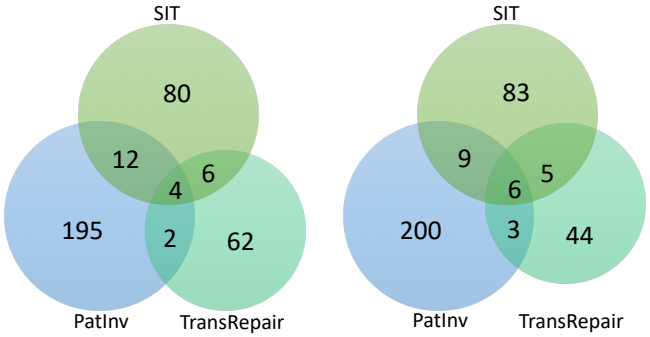


Figure 3: Erroneous Translations reported by PatInv, SIT and TransRepair (En-Zh)

example, when the threshold is 0.99, PatInv-Replace finds more erroneous issues with comparable precision. Thus, consider the performance of PatInv-Replace and PatInv-Remove, the effectiveness of PatInv on accurately finding erroneous translations is comparable to the state-of-the-art approaches.

Overlap of erroneous translations. Fig. 3 presents the overlap of the erroneous translations reported by PatInv, SIT and TransRepair. This Venn diagram shows that most of the erroneous translations reported by PatInv cannot be found by existing approaches. Specifically, only 8.4% (18/213) and 8.2% (18/218) erroneous translations reported by PatInv can be found by either of the existing approaches. Thus, we believe PatInv complements with SIT and TransRepair.

4.5 Running Time of Our Approach

In this section, we present the running time of PatInv. For each experimental setting, we run PatInv 10 times and report the average timing. Results are shown in Table 11. Note that the running time of PatInv-Replace's "filtering by sentence embeddings" step is not counted in the total time of PatInv-Replace because it is an optional step. Compared with other steps, this optional step is relatively time-consuming. For example, our implementation takes around 27 seconds for calculating the similarity of one sentence pair due to expensive API calls.

As we can observe from Table 11, most of the time is consumed in the filtering step and the translation step for PatInv-Replace. The filtering step of PatInv-Replace takes time because of the constituency parser used (8.1 seconds per hundred sentences). The translation step takes time because for each sentence in our experiments, we invoke the translators' API, which include both the translation time and the network communication. Compared with existing approaches, PatInv-Replace takes more time because PatInv-Replace generates much more sentences (i.e., 31,766 for PatInv-Replace, 4,025 for SIT, and 405 for TransRepair) and thus PatInv-Replace needs to run the constituency parser and the translator's API much more times, leading to longer running time. However, we believe PatInv is efficient for practical usage because it is designed to be an offline approach. Moreover, with this longer running time, PatInv-Replace has reported more erroneous translations (in Table 10). PatInv-Remove and TransRepair are much faster because they generate much less sentences (Table 4).

4.6 Fine-Tuning with Errors Reported by PatInv

In this section, we explore whether the reported erroneous translations can be utilized as a fine-tuning set for the NMT model to quickly fix translation errors. Fine-tuning is a common practice in NMT [18, 68]. It is often used when developers intend to adapt a trained NMT model to a new domain of data (e.g., from text on "politics" to text on "business"). To simulate this situation, we train a transformer network with global attention [74] on the WMT'18 Zh-En (Chinese-to-English) corpus [11], which contains ~ 20M sentence pairs. We use the fairseq framework [1] to create the model. We note that transformers are currently the state-of-the-art models for NMT. We reverse the original direction of translation so that we can properly compare with our experiments on the Google and Bing translation systems.

We test this model with PatInv-Replace using the Politics dataset in Table 2. PatInv-Replace successfully finds 26 erroneous translations. We manually label them with correct translations and use them as the dataset to fine-tune our NMT model. Specifically, we train the model for another 6 epochs with only the fixed translations. As the dataset was small, training took $<$ 10 mins. After this fine-tuning, all the 26 erroneous translations are fixed. Meanwhile, the BLEU score on a held out test set remains the same. Thus, the erroneous translations reported by PatInv can be used to improve the robustness of machine translation software. Note that although error fixing for machine translation is an interesting and important topic, it is not the focus of this paper. Thus, we regard it as a promising future direction.

Table 11: Running time of PatInv

		Google Politics Hindi	Google Business Hindi	Google Politics Chinese	Google Business Chinese	Bing Politics Hindi	Bing Business Hindi	Bing Politics Chinese	Bing Business Chinese
Initialization	PatInv-Replace	29.62	29.44	27.49	27.58	29.62	29.44	27.49	27.58
Pair Generation		129.94	120.92	129.94	120.92	129.94	120.92	129.94	120.92
Filtering		2231.55	2313.04	2231.55	2313.04	2231.55	2313.04	2231.55	2313.04
Translation		1600.47	1726.74	2530.22	2735.07	1775.17	1915.8	3068.59	3310.73
Detection		0.002	0.0022	0.0019	0.0023	0.0017	0.0025	0.0019	0.0022
Total		**3991.29**	**4190.14**	**4919.2**	**5196.61**	**4166.29**	**4379.2**	**5457.57**	**5772.27**
Initialization	PatInv-Remove	3.03	3.06	3.03	3.06	3.03	3.06	3.03	3.06
Pair Generation		15.20	16.14	15.20	16.14	15.20	16.14	15.20	16.14
Filtering		0.00002	0.00002	0.00002	0.00002	0.00002	0.00002	0.00002	0.00002
Translation		220.78	228.95	246.15	252.98	344.34	360.11	425.14	435.76
Detection		0.0019	0.002	0.0019	0.0022	0.0023	0.0021	0.0018	0.0021
Total		**239.01**	**248.15**	**264.38**	**272.18**	**362.57**	**379.31**	**443.37**	**454.96**
SIT		NA	NA	**1360.91**	**1285.73**	NA	NA	**2303.93**	**2214.46**
TransRepair		NA	NA	**18.37**	**15.11**	NA	NA	**69.92**	**66.01**

5 RELATED WORK

5.1 Robust AI Software

Throughout recent years, there has been much work and development in the area of artificial intelligence (AI). AI is used everywhere today, such as in autonomous cars and face recognition. However, modern AI models are not as robust as we might hope. In particular, they can return incorrect results (e.g., wrong classification labels) that lead to fatal accidents [37, 39, 92]. Recent research has reported a variety of adversarial examples that can fool AI software, such as autonomous cars [8, 21, 27, 83], 3D object classifiers [80, 82], and speech recognition services [12, 22]. To promote robustness, lines of research have focused on testing, [23, 26, 31, 36, 44, 59, 62, 72, 79, 85, 86], debugging [45], detecting adversarial examples [46, 70, 75, 81], or training networks in a robust way [35, 42, 48, 57]. However, none of these papers explore machine translation, which is the main focus of our paper.

5.2 Robust NLP Systems

The recent advances in the robustness of computer vision systems have encouraged researchers to investigate the robustness of NLP systems performing tasks such as sentiment analysis [7, 32, 40, 63], textual entailment [32] and toxic content detection [40]. However, all of these studies work on classification problems which have a much smaller output space than machine translation. Robustness of more complex NLP system has also been explored in a a few recent studies. In particular, Jia and Liang [34] proposed an adversarial evaluation scheme for the Stanford Question Answering Dataset (SQuAD), which has been widely used for the evaluation of reading comprehension systems. They found that appending a specially designed sentence to the paragraph can mislead the well-trained reading comprehension systems. Additionally, Mudrakarta et al. [52] successfully attacked question answering models. In particular, they leverage the finding that deep networks often ignore important question terms and thus they perturb the questions accordingly. Compared with machine translation, these systems are easier to verify because the ground truth is unique. For example, the output of reading comprehension could be a specific person name. For the task of translation, there could be multiple correct translations for a given sentence, which makes detecting incorrect outputs incredibly nuanced.

5.3 Robust Machine Translation

Machine translation strives to translate text in a source language to text in a target language automatically. There are primarily two lines of work on the robustness of machine translation software: (1) adversarial machine learning and (2) machine translation testing.

Adversarial machine learning aims at fooling machine translation models with adversarial examples. Most of the existing techniques generate such examples by white-box methods [14, 50, 87], in which complete knowledge of network structure and parameters of the machine translation model is available. Unlike them, PatInv takes a black-box approach. Existing black-box techniques [9, 24, 33] perturb or paraphrase sentences that can easily lead to invalid sentences (e.g., syntactic or semantic errors). In comparison, this paper aims at generating syntactically- and semantically-correct test cases (i.e., source sentences).

Machine translation testing aims at finding syntactically- and semantically-correct sentences that trigger translation errors. [76, 89] proposed two specialized methods to detect two kinds of translation errors respectively (under-translation and over-translation). In contrast, PatInv is a general methodology that targets general translation errors. He et al. [30] and Sun et al. [69] developed metamorphic testing techniques for general translation errors based on the assumption that similar sentences should have similar translations (evaluated by sentence structures [30] or existing distance metrics [69]). Sun et al. [69] also further proposed an automated repair approach. *PatInv* complements these approaches because it is based on a brand new concept named pathological invariance: sentences of different meanings should not have the same translation. Our evaluation has also shown that PatInv can report many

erroneous translations that the state-of-the-art approaches [30, 69] cannot report. Thus, we believe PatInv can complement these approaches. In addition, all existing works were evaluated by one language setting (En-Zh), while PatInv is evaluated by two language settings (En-Hi and En-Zh).

5.4 Metamorphic Testing

Metamorphic testing techniques find software bugs by detecting the violation of necessary functionalities (i.e., metamorphic relations) of software under test [15, 16, 67]. As an effective approach for addressing the test oracle and test case generation problems, metamorphic testing has been adopted in the testing phase of a variety of software systems, such as compilers [38, 41], scientific libraries [84], and database systems [43]. In addition, metamorphic testing has also been used in AI software testing because of its ability to test "non-testable" programmings, such as statistical classifiers [53, 78], search engines [90], and autonomous cars [72, 86]. PatInv is a novel metamorphic testing approach for machine translation software.

6 CONCLUSION

In this paper, we present PatInv, a novel, effective, and widely applicable methodology for finding translation errors in machine translation systems. In contrast to existing approaches that rely on invariances between translations (e.g., structural invariance [30]), PatInv is based on pathological invariance: sentences of different meanings have identical translation. Because of this significant conceptual difference, our implementations of PatInv have reported many erroneous translations that cannot be found by existing approaches. In our evaluation, PatInv successfully finds 100 erroneous translations for En-Hi and 431 erroneous translations for En-Zh respectively in Google Translate and Bing Microsoft Translator. In particular, among the errors reported for En-Zh, 395 out of 431 are unique to PatInv. In addition, PatInv performs accurately and is especially effective in finding word/phrase mistranslation errors and under-translation errors. Thus, PatInv complements with existing approaches. In the future, we will provide more implementations on top of PatInv's core idea, for example, generating new sentences by inserting words to detect over-translation errors. We will also apply the general idea to validate other text generation tasks, such as speech recognition and code summarization.

ACKNOWLEDGMENTS

We thank the anonymous ESEC/FSE reviewers for their valuable feedback on the earlier draft of this paper. In addition, the tool implementation benefited tremendously from Stanford NLP Group's language parsers [28] and Hugging Face's BERT implementation in PyTorch [25]. Pinjia He is the corresponding author.

REFERENCES

[1] [n.d.]. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. https://github.com//pytorch/fairseq
[2] [n.d.]. Google Translate. https://translate.google.com
[3] [n.d.]. Thesaurus. https://www.thesaurus.com/
[4] [n.d.]. WordsAPI. https://www.wordsapi.com/
[5] 2018. 15,000 Eggs Delivered to Norwegian Olympic Team After Google Translate Error. https://www.nbcwashington.com/news/national-international/google-translate-fail-norway-olympic-team-gets-15k-eggs-delivered/2034392/
[6] 2018. Greedy, Brittle, Opaque, and Shallow: The Downsides to Deep Learning. https://www.wired.com/story/greedy-brittle-opaque-and-shallow-the-downsides-to-deep-learning/
[7] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating Natural Language Adversarial Examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
[8] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.
[9] 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)*.
[10] Edward Loper Bird, Steven and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.
[11] Ond rej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*. Association for Computational Linguistics, Belgium, Brussels, 272–307. http://www.aclweb.org/anthology/W18-6401
[12] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*.
[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).
[14] Akshay Chaturvedi, Abijith KP, and Utpal Garain. 2019. Exploring the Robustness of NMT Systems to Nonsensical Inputs. *arXiv preprint arXiv:1908.01165* (2019).
[15] Tsong Y. Chen, Shing C. Cheung, and Shiu Ming Yiu. 1998. *Metamorphic testing: a new approach for generating next test cases*. Technical Report. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong.
[16] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. 2018. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys (CSUR)* 51 (2018). Issue 1.
[17] N. Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.
[18] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.
[19] Gareth Davies. 2017. Palestinian man is arrested by police after posting 'Good morning' in Arabic on Facebook which was wrongly translated as 'attack them'. https://www.dailymail.co.uk/news/article-5005489/Good-morning-Facebook-post-leads-arrest-Palestinian.html
[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).
[21] 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).
[22] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. 2019. SirenAttack: Generating Adversarial Audio for End-to-End Acoustic Systems. *arXiv preprint arXiv:1901.07846* (2019).
[23] 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 *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.
[24] 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)*.
[25] Hugging Face. [n.d.]. Transformers: State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch. https://github.com/huggingface/transformers
[26] Alessio Gambi, Marc Mueller, and Gordon Fraser. 2019. Automatically testing self-driving cars with search-based procedural content generation. In *Proc. of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*.
[27] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.
[28] Stanford NLP Group. [n.d.]. Stanford CoreNLP –Natural language software. https://stanfordnlp.github.io/CoreNLP/
[29] 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).
[30] Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-Invariant Testing for Machine Translation. In *Proc. of the 42nd International Conference on Software Engineering (ICSE)*.
[31] J. Henriksson, C. Berger, M. Borg, L. Tornberg, C. Englund, S. R. Sathyamoorthy, and S. Ursing. 2019. Towards Structured Evaluation of Deep Neural Network Supervisors. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*.
[32] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.
[33] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.
[34] 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)*.
[35] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. 2018. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373* (2018).
[36] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*.
[37] Fred. Lambert. 2016. Understanding the fatal Tesla accident on Autopilot and the NHTSA probe. https://electrek.co/2016/07/01/understanding-fatal-tesla-accident-autopilot-nhtsa-probe/
[38] Vu Le, Mehrdad Afshari, and Zhendong Su. 2014. Compiler Validation via Equivalence Modulo Inputs. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*.
[39] Sam Levin. 2018. Tesla fatal crash: 'autopilot' mode sped up car before driver killed, report finds. https://www.theguardian.com/technology/2018/jun/07/tesla-fatal-crash-silicon-valley-autopilot-mode-report
[40] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*.
[41] Christopher Lidbury, Andrei Lascu, Nathan Chong, and Alastair F. Donaldson. 2015. Many-Core Compiler Fuzzing. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*.
[42] Ji Lin, Chuang Gan, and Song Han. 2019. Defensive Quantization: When Efficiency Meets Robustness. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.
[43] 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 International Conference on Software Engineering (ICSE)*.
[44] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-Granularity Testing Criteria for Deep Learning Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE)*.
[45] 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 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*.
[46] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*.
[47] Fiona Macdonald. 2015. The Greatest Mistranslations Ever. http://www.bbc.com/culture/story/20150202-the-greatest-mistranslations-ever
[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.
[49] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.
[50] Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.
[51] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints* (2013).
[52] Pramod K. Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the Model Understand the Question?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.
[53] 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 and Knowledge Engineering (SEKE)*.
[54] Arika Okrent. 2016. 9 Little Translation Mistakes That Caused Big Problems. http://mentalfloss.com/article/48795/9-little-translation-mistakes-caused-big-problems
[55] Thuy Ong. 2017. Facebook apologizes after wrong translation sees Palestinian man arrested for posting 'good morning'. https://www.theverge.com/us-world/2017/10/24/16533496/facebook-apology-wrong-translation-palestinian-arrested-post-good-morning
[56] Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. arXiv:cs.CL/1803.00047
[57] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. In *IEEE Symposium on Security and Privacy*.
[58] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*.
[59] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*.
[60] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[61] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv e-prints* (2018).
[62] Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, and Lin Tan. 2019. CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE.
[63] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating Adversarial Misspellings with Robust Word Recognition. In *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.
[64] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.
[65] RobustNLP. 2020. A toolkit for testing machine translation. https://github.com/RobustNLP/TestTranslation
[66] Benny Royston. 2018. Israel Eurovision winner Netta called 'a real cow' by Prime Minister in auto-translate fail. https://metro.co.uk/2018/05/13/israel-eurovision-winner-netta-called-a-real-cow-by-prime-minister-in-auto-translate-fail-7541925/
[67] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering (TSE)* 42 (2016). Issue 9.
[68] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.
[69] Zeyu Sun, Jie M Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic Testing and Improvement of Machine Translation. In *Proc. of the 42nd International Conference on Software Engineering (ICSE)*.
[70] Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. 2018. Attacks Meet Interpretability: Attribute-steered Detection of Adversarial Samples. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*.
[71] Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Bulletin* 30, 4 (1953), 415–433.
[72] 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 International Conference on Software Engineering (ICSE)*.
[73] Barak Turovsky. 2016. Ten years of Google Translate. https://blog.google/products/translate/ten-years-of-google-translate/
[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Illia Kaiser, Lukasz abd Polosukhin. 2017. Attention is All you Need. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*.
[75] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*.
[76] 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 *Proc. of the*

49th IEEE/IFIP International Conference on Dependable Systems and Networks (industry track).

[77] 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).

[78] 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 (2011). Issue 4.

[79] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: A coverage-guided fuzz testing framework for deep neural networks. In *ISSTA 2019 - Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis.*

[80] Chong Xiong, Charles R. Qi, and Bo Li. 2019. Generating 3D Adversarial Point Clouds. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[81] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS).*

[82] Dawei Yang, Chaowei Xiao, Bo Li, Jia Deng, and Mingyan Liu. 2019. Realistic Adversarial Examples in 3D Meshes. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[83] 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).

[84] 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 ACM/IEEE International Conference on Automated Software Engineering (ASE).*

[85] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* (2019).

[86] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. Deeproad: Gan-Based Metamorphic Autonomous Driving System Testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE).*

[87] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations (ICLR).*

[88] Wujie Zheng, Wenyu Wang, Dian Liu, Changrong Zhang, Qinsong Zeng, Yuetang Deng, Wei Yang, Pinjia He, and Tao Xie. 2018. Testing Untestable Neural Machine Translation: An Industrial Case. *arXiv preprint arXiv:1807.02340* (2018).

[89] 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 *Proc. of the 41st International Conference on Software Engineering: Companion Proceedings.*

[90] Zhi Quan Zhou, Shaowen Xiang, and Tsong Yueh Chen. 2016. Metamorphic Testing for Software Quality Assessment: A Study of Search Engines. *IEEE Transactions on Software Engineering (TSE)* 42 (2016). Issue 3.

[91] 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 (Volume 1: Long Papers).* Association for Computational Linguistics.

[92] Chris. Ziegler. 2016. A Google self-driving car caused a crash for the first time. https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report