2019 IEEE/IFIP International Conference on Dependable Systems and Networks Industry Track

Detecting Failures of Neural Machine Translation in the Absence of Reference Translations

Wenyu Wang[‡*], Wujie Zheng[†*], Dian Liu[†], Changrong Zhang[†], Qinsong Zeng[†],
Yuetang Deng[†], Wei Yang[§], Pinjia He[¶], Tao Xie[‡]
[†] Tencent Inc., China
[‡] University of Illinois at Urbana-Champaign, USA
[§] University of Texas at Dallas, USA
[¶] ETH Zurich, Switzerland

***Abstract*—Despite getting widely adopted recently, a Neural Machine Translation (NMT) system is often found to produce translation failures in the outputs. Developers have been relying on in-house system testing for quality assurance of NMT. This testing methodology requires human-constructed reference translations as the ground truth (test oracle) for example natural language inputs. The testing methodology has shown benefits of quickly enhancing an NMT system in early development stages. However, in industrial settings, it is desirable to detect translation failures without reliance on reference translations for enabling further improvements on translation quality in both industrial development and production environments. Aiming for a practical and scalable solution to such demand in the industrial settings, in this paper, we propose a new approach for automatically identifying translation failures without requiring reference translations for a translation task. Our approach focuses on a property of natural language translation that can be checked systematically by using information from both the test inputs (i.e., the texts to be translated) and the test outputs (i.e., the translations under inspection) of the NMT system. Our evaluation conducted on real-world datasets shows that our approach can effectively detect property violations as translation failures. By deploying our approach in the translation service of WeChat (a messenger app with more than one billion monthly active users), we show that our approach is both practical and scalable in the industrial settings.**

***Keywords*—neural machine translation, failure detection, ML quality assurance**

I. Introduction

Neural Machine Translation (NMT) [1] has been widely adopted over recent years due to its simple models and satisfactory effectiveness on various machine translation tasks compared with the traditional Statistical Machine Translation (SMT). Unfortunately, in practice, NMT systems are often found to produce translation failures in the outputs, while various past incidents [2] have shown undesirable consequences brought by translation failures.

Aiming to reduce in-field translation failures for NMT systems before the deployment to serve users, developers have been relying on in-house system testing using parallel corpora, i.e., collections of bilingual text pairs. Each test case used in the system testing corresponds to a text pair, including the original text (i.e., the text to be translated) as a test input and its corresponding human-provided reference translation as an example of valid test outputs. Then developers typically use a special test oracle that (1) calculates the translation quality score (e.g., BLEU score [3]) for measuring lexical similarities between the output and the reference translation, and (2) checks whether the calculated score is greater than a threshold value to determine whether the test case passes.

While reference-based system testing has been shown useful for in-house quality assurance of industrial NMT systems, a strategy of detecting translation failures without reference translations brings further improvements to translation quality when reference translations become unavailable. Besides helping build a much wider scope of test cases, such strategy enables *in-vivo testing* [4], which leverages translation requests from real users in the production environment as test inputs. The strategy not only exposes unseen types or instances of translation failures not revealed by limited in-house system test cases, but also allows real-time monitoring and handling of translation failures in the production environment.

Aiming for a practical and scalable solution to such demand in industrial settings, in this paper, we propose a new approach for automatically identifying translation failures. Our approach focuses on a translation property that generally holds and can be checked systematically: the original text and the translation generally have one-to-one mappings in terms of their constituents, e.g., words/phrases. Any violation of this property in the translation under inspection indicates potential translation failures. With the support of information from both the inputs (i.e., the original texts) and outputs (i.e., the translations under inspection) of the NMT system, our approach includes two algorithms for detecting two specific types of violations of this property, respectively: (1) *under-translation*, where some words/phrases from the original text are missing in the translation, and (2) *over-translation*, where some words/phrases from the original text are unnecessarily translated multiple times. Based on our experience in deploying NMT models in large-scale industrial settings, many translation failures can be reflected by these two types of violations.

We have deployed our approach in both the development and production environments of WeChat, a messenger app with more than **one billion** monthly active users [5]. Our experience [6] in enhancing WeChat's translation service demonstrates that our approach is both practical and scalable. In the

[*] Equal contributions.

978-1-7281-3032-3/19/$31.00 ©2019 IEEE
DOI 10.1109/DSN-Industry.2019.00007

development environment, using our approach, NMT model developers manage to find the NMT system's translations that contain translation failures and are previously not detected by reference-based approaches. In the production environment, our approach processes about **12 million** unique translation tasks every day, enabling developers to collect translation tasks with unseen types or instances of translation failures. Developers are also able to observe the performance of the deployed models in real-world usages and handle translation failures instantly through switching to backup models when necessary. We provide an online demonstration[1] for our approach on tasks of English and Chinese translation.

II. Background

A. Translation-Property Violation

Under-translation. For a specific translation task, if the translation misses one or more words/phrases from the text to be translated, then the translation is considered under-translated. Table I shows one example of under-translation: the underlined Chinese word corresponding to 'mother' is missing in the English translation.

Over-translation. For a specific translation task, if any word/phrase is unnecessarily translated multiple times, then the translation is considered over-translated. Table III shows an example of over-translation, where the underlined Chinese phrase representing 'can never be changed' appears four times in the translation, while there is only one occurrence (as indicated by italicized words) in the original sentence. Such repetition is redundant since it does not change the meaning of the translation.

These two specific types of violation can directly affect the user experience and even lead to misunderstanding of the translation. Also according to our observation, these two specific types of violation can indicate more types of translation failures, including wrong names and incorrect interpretation of numbers.

B. Translation Quality Scores

There are various translation quality scores that can be used to measure the overall quality of translations with respect to reference translation(s). The BLEU (BiLingual Evaluation Understudy) score [3] is one of the quality scores most widely used for machine translation. In particular, a higher BLEU score indicates that the translation under inspection is closer to the reference translation(s), being considered of higher translation quality. The range of the BLEU score is $[0, 1]$, which is often presented as a percentage value (i.e., $[0, 100]$).

III. Detection Algorithms for Under- and Over-translation

A. Building Mappings Between Bilingual Words and Phrases

Aiming to support two detection algorithms, based on a massive training dataset for translation tasks (i.e., parallel corpora), we aim to build the mapping from each word/phrase in the source language to a set of words/phrases representing corresponding valid translations in the destination language. There are two steps to achieve this goal: *phrase identification* and *mapping learning*.

Phrase identification. The purpose of identifying phrases from texts (of the target language) is to handle the situations where a phrase's meaning cannot be conveyed by the combination of its constituent words' meaning. Aiming for efficiency and robustness, we assume that each phrase can be represented by a pair of keywords, which are two words conveying major semantics in the phrase. Under such assumption, we can enumerate all pairs of words appearing in each sentence of the target language from the parallel corpora, count the occurrences of these word pairs, and pick the word pairs with frequent occurrences (judged based on a predefined threshold c_{ph}) to regard them as potential keyword pairs. *Mapping learning* (as introduced next) further helps identify real keyword pairs. Another observation is that phrases are usually short, and thus a limit should be imposed on the distance of two words under consideration to form a phrase. We denote the maximum distance of two words under consideration as d_{ph}. As a concrete example, when $d_{ph} = 1$, for a 4-word sentence $w_1w_2w_3w_4$, we obtain 5 unique word pairs: $\langle w_1, w_2\rangle, \langle w_1, w_3\rangle, \langle w_2, w_3\rangle, \langle w_2, w_4\rangle, \langle w_3, w_4\rangle$.

Mapping learning. We employ the recommendation algorithm of Item-based Collaborative Filtering [7] to build the translation mappings for words/phrases. Originally used in scenarios such as product recommendation [8], the algorithm predicts each user's potentially interested items by looking at the user's rating history and searching for items similar to the user's highly rated items. The algorithm exploits a key insight that similar items should have similar ratings from massive users. To reduce our mapping problem to the recommendation problem, we view each training translation task as a user, each word/phrase appearing in either the original text (of the source language) or reference translation (of the destination language) as an item with a positive rating, and all other words/phrases as items with negative ratings. Under such settings, if two words have similar meaning in two languages, then they should have similar ratings (i.e., occurrence patterns in training translation tasks). Specifically, we define the user rating matrix M as follows:

$$M_{k,w} = \begin{cases} 1 & \text{if } w \text{ appears in } P_s^k \text{ or } P_d^k \\ 0 & \text{otherwise} \end{cases}$$

where w is a word/phrase, P_s^k and P_d^k denote the original text and the reference translation in the training translation task numbered k, respectively. Then, we calculate the relevance/similarity between w_s (a word/phrase in the source language) and w_d (a word/phrase in the destination language) using the Cosine similarity [9]:

$$R_{w_s,w_d} = \frac{\overrightarrow{M_{\cdot,w_s}} \cdot \overrightarrow{M_{\cdot,w_d}}}{||\overrightarrow{M_{\cdot,w_s}}||_2 \cdot ||\overrightarrow{M_{\cdot,w_d}}||_2} = \frac{\sum_k M_{k,w_s} M_{k,w_d}}{\sqrt{\sum_k M_{k,w_s}^2}\sqrt{\sum_k M_{k,w_d}^2}}$$

[1] https://bit.ly/2P4hEB4

TABLE I: Example translation with under-translation violation

Chinese (original)	English (translated)	English (reference)
三姑给你的红包给你妈妈了。	Third Aunt gave you a red envelope.	Third Aunt gave your red envelope to your *mother*.

TABLE II: Example translation lists for the under-translated word in Table I

origin	error_rate	trans 1	trans 2	trans 3	trans 4	trans 5	trans 6	trans 7	trans 8	trans 9	trans 10
妈妈	0.04116	mother	mom	mum	mama	mommy	moms	mothers	mummy	my	her

We then build the mappings based on the relevance between words/phrases in two languages. Specifically, for each word w_s in the source language, we consider a total of c_{tr} words in the destination language with the highest relevance to w_s to be its valid translations, where c_{tr} is a predefined threshold value.

B. Detection Algorithm for Under-translation

Under-translation detection can now be achieved by checking the existence of word/phrase translations in the whole translation text. Specifically, for each word/phrase in the original text, we obtain the list of its corresponding word/phrase translations, and check whether any of these word/phrase translations appears somewhere in the translation text. We use a real-world example (shown in Table I) to illustrate this process. The underlined Chinese word (corresponding to 'mother' in English) is missing in the English translation. Our algorithm first produces the contents in Table II as part of the mappings constructed in the previous steps. In the table, 'origin' indicates the words to be translated and 'trans k' denotes the k-th most relevant translation. Our algorithm then goes through each translation for the Chinese word and checks whether it appears in the translation being examined. The algorithm subsequently finds that none of translations in Table II appears in the translation being examined (as shown in Table I) and marks the translation with a potential under-translation violation.

However, there might be some words/phrases of the source language incurring implicit translations, i.e., the translations of these words/phrases usually do not show up in the translations of the whole sentences. Examples include *the* and *to* in English. Our techniques described earlier can likely produce many false positives due to such phenomenon. We introduce *error rate filtering* to address such limitation. Specifically, we calculate the error rate e_{w_s} for each word/phrase w_s of the source language on the training dataset using $e_{w_s} = N^e_{w_s}/N_{w_s}$, where $N^e_{w_s}$ denotes the number of sentence pairs that are considered under-translated on w_s, and N_{w_s} is the number of translation sentence pairs involving w_s. Then, we disregard any under-translation caused by missing w_s on translations under examination if we find that $e_{w_s} > e_{th}$, where e_{th} is a predefined threshold value. Table II shows the technique being used in the example translation, where 'error_rate' shows the error rate for the missing Chinese word. Our algorithm confirms that the error rate is within e_{th} (usually set to be 0.2 in our production environment) for the missing Chinese word before concluding that the translation has a potential under-translation violation.

TABLE III: Example translation with over-translation violation

English (original)	Chinese (translated)
U have to admit that something *can never be changed*, live with it or break with it!	你必须承认，有些东西是永远无法改变的，无法改变的，无法改变的，无法改变的！

C. Detection Algorithm for Over-translation

Our over-translation detection algorithm is based on frequencies of words appearing in the translation. Specifically, we count the occurrences for each word appearing in the translation under inspection, and mark the word as over-translated if more than one occurrence is found and the occurrences are near to each other. However, there are two issues with this approach. First, particles such as *have*, *the*, and *to* in English tend to have multiple occurrences and no actual meaning in many well-formed expressions. These words could confuse over-translation detection and cause false positives. Second, some words/phrases might have multiple occurrences in the original text, and they are probably also supposed to appear multiple times in the translation.

To address the first issue, we remove all such common words (i.e., stop words) from the translation under inspection before conducting over-translation detection. To address the second issue, our algorithm leverages the word/phrase translation mappings introduced in the previous steps in the other direction, aiming to find out a set of words/phrases (of the source language) that can all be translated to each word/phrase w_d in the translation under inspection. Our algorithm then counts the number of words/phrases in the original text that fall in the set (denoted as $count_s(w_d)$) as well as the number of occurrences of w_d (denoted as $count(w_d)$). Finally, we consider a w_d to be over-translated if we find $count_s(w_d) < count(w_d)$.

Table III shows an example with an over-translation violation to illustrate our algorithm. The Chinese translation contains 3 extra repetitions for the translation of "can never be changed". Our algorithm first discovers that there are 4 instances of "无法" and "改变". Then, our algorithm looks up the word/phrase translation mappings and finds that "can never" and "changed" have such translations. Subsequently, our algorithm finds that these two English words/phrases appear only once in the original text, fewer than the occurrences in the translation under inspection. Finally our algorithm marks the translation with an over-translation violation.

TABLE IV: Overview of the evaluation datasets

Name	Lang	Type	$\#_{all}$	$\#_{ws}$	$\#_U$	$\#_O$
ench_news	en-cn	News	200	7497	54	4
chen_news	cn-en	News	200	7418	31	8
ench_oral	en-cn	Oral	300	3237	42	1
chen_oral	cn-en	Oral	300	2918	37	5

TABLE V: Detection results on the evaluation datasets

	under-translation			over-translation		
	P	R	F	P	R	F
ench_news	0.51	0.69	0.59	0.38	0.75	0.50
chen_news	0.43	0.65	0.52	0.73	1.00	0.84
ench_oral	0.52	0.40	0.45	0.33	1.00	0.50
chen_oral	0.30	0.49	0.37	0.80	0.80	0.80

IV. Evaluation

A. Evaluation Setup

We evaluate our approach on four manually-labeled datasets consisting of real-world translation tasks and corresponding translation failures. We build the word/phrase mappings using 16 million sentence pairs crawled from various sources (e.g., example word/phrase usages in dictionaries) on the Internet.

As shown in Table IV, each evaluation dataset contains original sentences randomly sampled from larger datasets (crawled online and different from the training datasets), translations under inspection (that contain translation failures), and labels manually added by us indicating the existence of the two types of violations. $\#_{all}$ denotes the total number of sentence pairs, $\#_{ws}$ denotes the total number of words of the source language; $\#_U$ and $\#_O$ denote the numbers of sentence pairs flagged with under- and over-translation violations, respectively. 'en-cn' and 'cn-en' indicate translating from English to Chinese and from Chinese to English, respectively.

B. Algorithm Effectiveness

Tables V shows the result summary of under-translation detection and over-translation detection. The precision, recall, and F-measure are abbreviated as 'P', 'R', and 'F', respectively. We empirically set $c_{tr} = 10$ and $c_{ph} = 10$ for all the experiments, $e_{th} = 0.15$ for the experiments on the 'ench_news' and 'chen_news' datasets, and $e_{th} = 0.25$ for the experiments on the 'ench_oral' and 'chen_oral' datasets. As shown in Table V, our approach achieves reasonable F-measures on all the datasets, with recalls generally contributing more to the scores.

C. Experience of Deployment

We have deployed our approach in both the development and production environments of WeChat. In the development environment, the NMT model developers manage to find the NMT system's translations that contain translation failures and are previously not detected by reference-based approaches. In the production environment, our approach processes about 12 million unique translation tasks every day, enabling the developers to collect translation tasks with unseen types or instances of translation failures. On top of that, the developers are able to observe the performance of the deployed models in real-world usages and handle translation failures instantly through switching to backup models when necessary.

Our approach also helps collect 130,000 English and 180,000 Chinese meaningful words/phrases (i.e., words or phrases with low error rates when learning from the training data). These words are used as in-house test cases for testing and improving WeChat's continuously-improved machine translation model. Using our approach also helps diagnose issues in other competing machine translation systems released by other providers. In practice, our approach is able to detect potential defects lying in the design, implementation, or training data in multiple machine translation systems.

V. Conclusion

To address the demand in industrial settings, in this paper, we have presented our approach for automatically identifying translation failures without requiring reference translations, with focus on two specific types of translation-property violations. Our evaluation conducted on real-world datasets has shown that our approach can effectively detect property violations as translation failures. By deploying our approach in the translation service of WeChat, a messenger app with more than one billion monthly active users, we show that our approach is both practical and scalable in the industrial settings.

Acknowledgment

The authors from Illinois CS were supported in part by NSF under grants no. CNS-1513939, CNS-1564274, CCF-1816615, and by a 3M Foundation Fellowship.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] A. Okrent. (2016) 9 little translation mistakes that caused big problems. [Online]. Available: http://mentalfloss.com/article/48795/9-little-translation-mistakes-caused-big-problems

[3] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annual Meeting on Association for Computational Linguistics*, 2002, pp. 311–318.

[4] C. Murphy, G. Kaiser, I. Vo, and M. Chu, "Quality assurance of software applications using the in vivo testing approach," in *Proc. International Conference on Software Testing Verification and Validation*, 2009, pp. 111–120.

[5] R. Hollander. (2018) WeChat has hit 1 billion monthly active users. [Online]. Available: https://www.businessinsider.com/wechat-has-hit-1-billion-monthly-active-users-2018-3

[6] W. Zheng, W. Wang, D. Liu, C. Zhang, Q. Zeng, Y. Deng, W. Yang, P. He, and T. Xie, "Testing untestable neural machine translation: An industrial case," in *Proc. 41st International Conference on Software Engineering: Companion, Poster*, 2019.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th International Conference on World Wide Web*, 2001, pp. 285–295.

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[9] P. Dangeti, *Statistics for Machine Learning*. Packt Publishing Ltd, 2017.