2025 IEEE 5th International Conference on Software Engineering and Artificial Intelligence

Revisiting Round-Trip Translation with LLMs and Agentic Translation

Yiheng Wang
Information and Computation Science Department
Peking University
Beijing, China
Yiheng.Wang.ai@outlook.com

Zuoquan Lin*
Information and Computation Science Department
Peking University
Beijing, China
linzuoquan@pku.edu.cn
*Corresponding author

2025 IEEE 5th International Conference on Software Engineering and Artificial Intelligence (SEAI) | 979-8-3315-1361-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SEAI65851.2025.11108752

***Abstract*—Round-trip translation (RTT) is a tool for rectifying grammatical errors in text and facilitating automated program repair (APR). When combined with large language models (LLMs), RTT has demonstrated significant efficacy in enhancing APR. We aim to evaluate the performance of LLMs and agentic translation on RTT. We conducted experiments with GPT-4o-mini, finding that LLMs behave well in both contexts and the agentic translation, powerfully on APR through RTT, but lack highlight on natural language RTT. Our findings reveal the exceptional adaptability of LLMs in RTT and the capacity of LLMs and agentic translation to facilitate APR with RTT.**

***Index Terms*—Round-trip translation, Large language models, Agentic translation, Automated program repair**

I. Introduction

Round-trip translation (RTT) refers to translation from the original language to an intermediate language, followed by translating back to the original language. As a powerful translation method, RTT has the property of maintaining factual content while altering the expression style. Researchers exploit this property to apply RTT to address various problems. Multi-turn RTT can cause significant deviations from the original text and might distort factual accuracy during paraphrasing. RTT brings a challenge of balancing the generation of additional text while preserving the information.

LLMs have become the focal point of AI. These large models are mostly built with Transformer, containing huge parameters and training with extensive corpus. After training, LLMs are aligned with human expectations [1, 2]. LLMs exhibit a rapid increase in capacity as their scale grows, enabling them to tackle complex tasks such as mathematics and logical reasoning. The advent of LLMs has attracted significant attention from researchers and has proven effective in advancing science and engineering.

Agentic AI refers to AI systems and models capable of autonomously acting to achieve specified goals without requiring continuous human guidance. The agentic AI system can understand the user's objectives and the context surrounding the objectives. Agentic translation holds potential as a solution for machine translation [3, 4].

With designed prompts, LLMs have demonstrated reliable performance in translation. Some researches employ LLMs to facilitate RTT, drawing our interest in the behavior of LLMs in RTT, especially checking agentic transltion in RTT. However, the uncertainty about evaluating natural language generation veils our investigation, urging us to figure out a feasible way to measure RTT.

RTT has been applied to rectify grammatical errors in natural language. RTT has further found facilitating automated program repair (APR), which makes APR through RTT. The evaluation of APR concerns the program function, and the test is precise. Therefore, APR becomes a promising option for RTT evaluation.

We conduct experiments on natural language RTT to compare LLMs and agentic translation with pre-trained models and self-trained models. Additionally, we investigate the behavior of LLMs and agentic translation in APR through RTT. Our main contributions are summarized as follows:

1) We provide the first empirical analysis of the behavior of LLMs and agentic translation in natural language RTT.
2) We adapt agentic translation for APR and experimentally validate the effectiveness of LLMs and agentic translation in APR through RTT.

II. Related Works

RTT has been applied for various purposes. One common purpose is to facilitate data augmentation and enable the training of translation models on low-resource datasets [5–7]. RTT served as a utilitarian tool for evaluating machine translation quality [8–11]. These researches hold the belief that RTT should result in minimal changes to well-translated text, allowing for text similarity evaluations based on token sequence comparison and text embeddings to reflect translation quality. RTT can aid in grammatical error correction [12–15]. They utilize the property of RTT to synthesize different translation pairs that correspond to the same information. Other works generate data through RTT for purposes [16–20]. RTT was used to generate fake data to commit fact verification on political debates [16]. It was used to generate adversarial examples for neural machine translation [18], and the detection of machine-translated text via RTT-based text similarity was proven to be effective [17]. In education, RTT can provide the wrong options for fill-in-the-blank exercises [19]. RTT on

words from the Internet is less harmful, which helps to mitigate social-engineered attacks [20].

Our work was inspired by the approach proposed by Ruiz et al [21], introducing a novel method to implement APR through RTT with LLMs. The approach consists of four steps. Firstly, preprocess the data and design prompts for translation. Next, utilize code LMs and LLMs to perform RTT. Then, postprocessing is applied to ensure that the repaired programs are executable. Finally, evaluate the RTT results.

III. Method

A. Round-trip translation

RTT is a machine translation method that translates a text from its original language to an intermediate one and then back to the original language. Let the original language be A and the intermediate language B. Consider a text in language A, which can be represented as a sequence of tokens: $x = \{x_1, x_2, ..., x_M\}$, where x_i is an individual token, and M is the length of x. The intermediate text in B is $y = \{y_1, y_2, ..., y_N\}$, where y_i is an individual token, and N is the length of the translation y. The translation in B is translated back to language A, resulting in a round-trip translated text denoted as $\tilde{x}_A = \{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_{M'}\}$, where $\tilde{x}_i$ is a token in the round-trip translated text, and $M^{'}$ is the length of the resulting sequence in A. To implement the RTT process, we employ seq2seq models. We denote the translation model from A to B as $M_{A\to B}$ and the translation model from B to A as $M_{B\to A}$. Specifically, the round-trip translation process can be formulated as follows:

$$\begin{aligned} y &= M_{A\to B}(x), \\ \tilde{x} &= M_{B\to A}(y), \\ P(\tilde{x}|x) &= P(\tilde{x}|y)P(y|x), \\ P(y|x) &= \prod_{i=1}^{N} P(y_i|x, y_{<i}), \\ P(\tilde{x}|y) &= \prod_{j=1}^{M^{'}} P(\tilde{x}_j|y, \tilde{x}_{<j}). \end{aligned} \tag{1}$$

RTT creates new text $\tilde{x}$ in the original language with facts unchanged but in a different form if the translation model can preserve the facts in the text. We shall use various models to implement RTT.

We begin by employing the pre-trained model mT5 [22]. mT5 is a multilingual variant of T5, retaining the advantages of T5 while enhancing its capability to process multilingual text-to-text tasks. The model architecture and training procedure of mT5 closely follow those of T5, with the primary distinction being its training on extensive multilingual datasets. We fine-tune mT5 with the data organized as follows. Source: Translate to language B: [language A text] Target: [language B text].

Moreover, we employ self-training with mT5 to see if incorporating additional unlabeled data could improve model performance. Self-training is a semi-supervised learning approach that utilizes unlabeled data to enhance model effectiveness. Our self-training process is composed of two phases. The teacher phase involves fine-tuning mT5 on labeled data to train the initial teacher model. The iteration phase consists of multiple iterations. In each iteration, the teacher model generates pseudo-labels for the unlabeled data. A subset of the pseudo-labeled data, selected based on the highest prediction confidence, is then added to the labeled data. This combined dataset is used for training the student model, which becomes the teacher model in the subsequent iteration. The iterations conclude when certain predefined conditions are met, and the final student model is output.

We adopt LLMs on RTT to observe the performance of the emerging powerful language models. Following the methodology of Ng et al. [3], we utilize both plain prompts and agentic translation techniques to enable LLMs to perform RTT tasks. The plain prompt provides the model with the task description and the input and output formats. The agentic translation method extends this approach by adding two additional steps. The process is illustrated in Figure 1.

(1) *Translate*: A straightforward translation from the source language to the target one.
(2) *Reflect*: After reviewing both the source text and the initial translation, the model carefully reflects on the translation, offering constructive criticisms and suggestions for improvement from specific perspectives.
(3) *Improve*: The model revisits both the source text and the initial translation, taking into account the suggestions and criticisms provided in the previous step, and then revises the translation to enhance its quality.

This reflection agentic workflow for machine translation has demonstrated competitive performance in some cases when compared to leading commercial translation systems. We further investigate the impact of agentic translation on RTT.

B. APR with RTT

RTT can retain the information in the original text while altering its structure, word choice, and style of expression. This characteristic makes RTT particularly valuable in various applications. One such application is data augmentation, where RTT generates a modified text. Furthermore, RTT is useful in correcting spelling and grammatical errors and improves the overall clarity and appropriateness of text, making it a suitable tool for correction. APR is another significant application of RTT. LLMs trained on large-scale datasets that include both natural language text and programming language corpus can predict translations based on contextual understanding from the data they have learned. This makes it promising for LLMs to identify and fix bugs in code through RTT. LLMs have demonstrated exceptional capabilities in mathematics, logical reasoning, and programming.

We focus on APR for two main reasons. First, evaluating the performance of each method on APR is straightforward and easily measurable. While data augmentation can be assessed based on the trained model's capacity, it is more burdensome and highly susceptible to random factors. Second, APR remains a challenging problem, as code generated by LLMs is

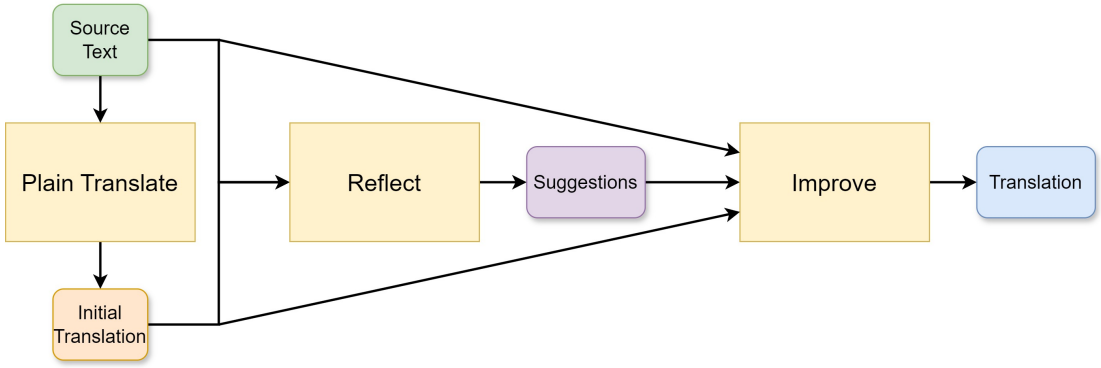

Fig. 1. The illustration of the reflection agentic workflow for machine translation [3].

still prone to have bugs. Therefore, APR needs to make more efforts.

Similar to RTT in natural language, we apply both a direct translation prompt and an agentic translation method to enable LLMs to perform translations in a round-trip manner. In this context, the original language A is a programming language. Following the approach of Ruiz et al. [21], the intermediate language B can either be a programming language or a natural language that describes the input, output, function, and other information about the program. Additionally, we make slight modifications to the prompt to ensure its compatibility with APR. Here are some prompts:

Plain translate prompt

system_message:
You are an expert programmer, specializing in translating {pl1} program into {pl2} program.

prompt:
This is a {pl1} to {pl2} translation. Please provide the {pl2} translation for the {pl1} program without any comments.

Do not generate any additional methods nor repeat the {pl1} program nor give any explanations.
{pl1} program:
{source_text}

{pl2} program:

Reflect prompt

system_message:
You are an expert programmer, specializing in translating {pl1} program into {pl2} program. You will be provided with a {pl1} program and its translation. And your goal is to improve the translation."

prompt:
Your task is to carefully read a source program and its translation from {pl1} to {pl2}, and then give constructive criticisms and helpful suggestions to improve the program.

The source program and initial translation, delimited by XML tags <SOURCE></SOURCE> and <TRANSLATION></TRANSLATION>, are as follows:

<SOURCE>
{source_text}
</SOURCE>

<TRANSLATION>
{program}
</TRANSLATION>

When writing suggestions, pay attention to whether there are ways to improve the program's precision (by correcting logical errors and {pl2} grammar errors to make the program work properly).

Write a list of specific, helpful and constructive suggestions for improving the program. Each suggestion should address one specific part of the program. Output only the suggestions and nothing else.

Improve prompt

system_message:
You are an expert programmer, specializing in translation editing from {pl1} to {pl2}.

prompt:
Your task is to carefully read, then edit, a translation from {pl1} to {pl2}, taking into account a list of expert suggestions and constructive criticisms.

The source program, the initial translation, and the expert programmer suggestions are delimited by XML tags <SOURCE></SOURCE>, <TRANSLATION></TRANSLATION> and <SUGGESTIONS></SUGGESTIONS> as follows:

<SOURCE>
{source_text}
</SOURCE>

<TRANSLATION>
{program1}
</TRANSLATION>

<SUGGESTIONS>
{reflection}
</SUGGESTIONS>

Please take into account the expert suggestions when editing the translation. Edit the translation by ensuring precision (by correcting logical errors and {pl2} grammar errors to make the program work properly).

Output only the new translation and nothing else.

IV. Experiment

A. RTT on natural language

We conduct RTT on German-English language pairs. The German-English dataset is sourced from the News Commentary v18.1 of WMT23. We select 25,000 labeled data respectively for German-to-English and English-to-German, along with 100,000 unlabeled English data and 100,000 unlabeled German data. Additionally, 40,000 labeled validation data for both German-to-English and English-to-German are used for model evaluation. We fine-tune the mT5-small model with a learning rate of 2×10^{-5}, a batch size of 32, and the AdamW optimizer. Furthermore, we employ beam search with a beam width of 3 during inference to mitigate repetition in the generated translations.

Given that mT5 can handle multiple languages, we train bidirectional translation models for English-German (ende) and unidirectional translation models, including en2de and de2en. This is done to evaluate whether the bidirectional translation model performs sufficiently well. For self-training, we conduct preliminary experiments to configure the self-training setup. We initialize the student model with pre-trained weights during each iteration and combine pseudo-labeled and labeled data rather than pre-training on pseudo-labeled data and fine-tuning on labeled data. Additionally, we employ GPT-4o-mini for RTT, using a temperature of 0.3 and top_p set to 1.0.

We conduct experiments on German-English pairs, utilizing both pre-trained and fine-tuned unidirectional and bidirectional models and self-trained unidirectional and bidirectional models. We select three automatic machine translation metrics as measures, including BLEU, CHRF, and TER. The statistics are listed in Table I, where ende means bidirectional translation with original language as English, deen means bidirectional translation with original language as German, en2de means two respective unidirectional translation with original language as English, de2en means two respective unidirectional translation with original language as German.

TABLE I
Performance of pre-trained models, self-trained models, and LLMs on German-English multi-turn RTT

Round-trip translation	BLEU (↑)	CHRF (↑)	TER (↓)
Pretrain and Finetune			
ende-turn1	21.99	47.62	64.01
ende-turn2	16.96	41.66	70.04
ende-turn3	14.85	38.80	72.89
deen-turn1	14.33	42.90	72.25
deen-turn2	10.62	37.48	77.15
deen-turn3	9.05	34.77	79.26
en2de-turn1	20.11	44.89	67.66
en2de-turn2	14.61	38.47	74.38
en2de-turn3	12.23	35.10	77.94
de2en-turn1	13.48	41.70	74.05
de2en-turn2	9.45	35.87	78.99
de2en-turn3	7.88	33.21	80.94
Self-training			
ende-turn1	22.89	46.97	62.66
ende-turn2	17.48	41.07	68.86
ende-turn3	15.50	38.38	71.43
deen-turn1	14.14	41.25	71.57
deen-turn2	10.90	36.77	75.87
deen-turn3	9.68	34.74	77.49
en2de-turn1	20.42	44.28	66.64
en2de-turn2	15.40	38.29	73.11
en2de-turn3	13.18	35.30	75.59
de2en-turn1	13.39	40.39	73.55
de2en-turn2	10.20	35.60	78.13
de2en-turn3	8.69	32.85	80.03
Plain-translation			
deen-turn1	33.75	70.24	45.62
deen-turn2	30.47	67.64	49.65
deen-turn3	28.78	66.07	52.25
ende-turn1	50.29	75.41	35.38
ende-turn2	44.39	71.82	40.88
ende-turn3	41.96	70.24	43.39
Agentic translation			
deen-turn1	28.11	66.17	53.15
deen-turn2	22.51	61.52	61.88
deen-turn3	19.83	59.00	66.35
ende-turn1	42.80	71.13	43.36
ende-turn2	34.27	65.58	53.78
ende-turn3	29.27	62.23	60.57

We conclude after observing the experiment results:

(1) All three measures decrease as the RTT turns add, indicating the deviation of RTT from the original text.
(2) Self-training with pretrained models as initialization in each iteration shows stronger defense to the erosion of RTT.
(3) As a multilingual pretrained model, finetuning mT5 with bidirectional translation data has a better performance than that with unidirectional translation data.
(4) Translation with LLMs is better than that with the other models and exhibits resistance compared to that with pretrained and self-trained models.

(5) Agentic translation does not behave ideally on RTT, behaving worse than direct translation and more easily affected by RTT.

Language models typically generate without memory, so the deviation of RTT is natural. Initializing a student model with pretrained weights is a common technique when combining pre-training and self-training, and its effectiveness has been revalidated. The third point suggests that a unified translation model performs better in RTT.

While LLMs are competitive at translation, we can still find that most LLMs exhibit a particular strength in processing English due to the unevenness of the training corpus. Agentic translation methods may fall short in RTT performance, yet they still offer valuable suggestions, indicating that the measurement used may not fully reflect the quality of the translation.

Metrics and quality estimation in machine translation are often messed up by inaccurately reflecting translation quality. This particularly holds for automatic metrics, which typically compare translations with reference and measure their similarity. However, since RTT focuses on identifying factually accurate translations rather than stylistically equivalent ones, we found that these metrics are inadequate for evaluating RTT. Consequently, we turn to APR for a more precise assessment of RTT quality. The APR test is more effective as it emphasizes functional correctness rather than stylistic fidelity.

B. APR with RTT

We have chosen two APR benchmarks to observe the effect of LLMs and agentic translation on RTT. All benchmarks contain buggy Python programs and provide test suites to verify whether repaired programs can pass all test cases. The previous works have proven the superiority of LLMs over normal-sized pretrained language models, so we focus on LLMs in this part.

QuixBugs consists of 40 programs from the Quixey Challenge [23], each translated into both Python and Java. Each program contains a one-line defect. We perform experiments and repeat them three times to calculate the average performance, aiming to reduce the randomness.

ConDefects contains 1,254 Java faulty programs and 1,625 Python faulty programs [24], all sourced from the online competition platform AtCoder. For testing purposes, 106 Python programs are selected. Unlike the programs in QuixBugs, which only consist of functions, the programs in ConDefects include input and output parts, making APR more complicated. The programs in QuixBugs are common functions like breadth-first search and greatest common divisor, which are likely familiar to LLMs. As a result, LLMs might provide the correct answer simply by recognizing the function name. The programs in ConDefects are more difficult in algorithms and less common, bringing more challenges for LLMs. Considering this, we allow LLMs to make three attempts, considering a test case passed if any of the three attempts are successful.

We use GPT-4o-mini with the same configuration but adjust the prompts for RTT. We set natural language and Java as intermediate languages, respectively. We use test case pass rate (TCPR), accepted count (AC count), and accepted ratio (AC ratio) to evaluate the effect of each method. TCPR is the ratio of the count of test cases that fixed programs pass and the count of all test cases, calculated as follows:

$$TCPR = \frac{\#\ of\ test\ cases\ that\ programs\ pass}{\#\ of\ all\ test\ cases}, \quad (2)$$

AC count means the number of programs that pass all corresponding test cases. AC ratio is the proportion of AC count to the total program count.

The methods include baseline, plain-NL, plain-PL, NL, and PL. The baseline represents the faulty programs offered by the dataset. Plain-NL and plain-PL are the direct translation with natural language and programming language as intermediate language, respectively. And Ruiz et al. implemented APR with this approach [21]. NL and PL are agentic translations with natural language and programming language as intermediate language,s respectively. The experiment results are listed in Table II and Table III.

TABLE II
TCPR, AC COUNT AND AC RATIO OF REPAIRED PROGRAMS IN QUIXBUGS

Method	TCPR	AC count	AC ratio
Baseline	30.43%	0	0
Plain-NL	81.52%	25.67	64.18%
Plain-PL	51.81%	14.33	35.83%
NL	**87.32%**	**29.33**	**73.33%**
PL	59.78%	18	45.00%

TABLE III
TCPR, AC COUNT AND AC RATIO OF REPAIRED PROGRAMS IN CONDEFECTS

Method	TCPR	AC count	AC ratio
baseline	46.28%	0	0
Plain-NL	44.78%	29	27.36%
Plain-PL	38.30%	20	18.87%
NL	**54.04%**	**32**	**30.19%**
PL	46.69%	25	23.58%

We conclude three facts:

(1) Although plain-NL and plain-PL are occasionally inferior to baseline in TCPR, plain-NL, plain-PL, NL, and PL show the ability to fix bugs existing in programs.
(2) LLMs with natural language as intermediate language behave better than them with programming language as intermediate language.
(3) Agentic translation significantly improves the bug-fixing performance.

For the first point, we conclude by comparing the other methods with the baseline. While plain-NL and plain-PL achieve worse TCPR in ConDefects, methods with LLMs behave much better on TCPR, AC count, and AC ratio in the rest cases. Thus, LLMs only with prompts inducing are capable of solving program repair.

Next, plain-NL and NL showed consistently better performance on all three measures. NL proved to be the best method among the methods. When we observe the examples from experiments, we find that plain-PL and PL are more likely to keep the existing bugs unchanged in the RTT process, and they are also easily affected by the intermediate programming language, obscure the language information, and inject more bugs.

Finally, agentic translation shows their superiority after observing the measures of NL, PL, plain-NL, and plain-PL. The designed reflection agentic workflow takes progress on TCPR, AC count, and AC ratio. While LLMs usually provide no more than ten suggestions, program repair needs some crucial criticisms to point out and fix the bugs. So agentic translation proves its effectiveness in RTT with APR, unlike in natural language RTT.

V. Conclusion

We evaluate the performance of LLMs and agentic translation on RTT by the experiments on natural language RTT and APR with RTT to judge whether LLMs still have magic in RTT and agentic translation are potential in RTT. The experiment results indicate that LLMs maintain their effectiveness in natural language RTT and capacity in APR. While agentic translation fails to mitigate the erosion of multi-turn RTT, it provides notable enhancements in APR with RTT. The reason for this phenomenon is the characteristics of agentic translation that reflect that LLMs only generate limited suggestions and lead to increased variability and alterations.

References

[1] Z. Wang, B. Bi, S. K. Pentyala, K. Ramnath, S. Chaudhuri, S. Mehrotra, Zixu, Zhu, X.-B. Mao, S. Asur, Na, and Cheng, "A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more," 2024. [Online]. Available: https://arxiv.org/abs/2407.16216

[2] Y. Zhang, S. Ren, J. Wang, J. Lu, C. Wu, M. He, X. Liu, R. Wu, J. Zhao, C. Zhan, D. Du, Z. Zhan, R. K. Singla, and B. Shen, "Aligning large language models with humans: A comprehensive survey of chatgpt's aptitude in pharmacology," *Drugs*, vol. 85, no. 2, pp. 231–254, Feb 2025. [Online]. Available: https://doi.org/10.1007/s40265-024-02124-2

[3] andrewng, "traslation-agent," 2024. [Online]. Available: https://github.com/andrewyng/translation-agent

[4] M. Wu, Y. Yuan, G. Haffari, and L. Wang, "(perhaps) beyond human translation: Harnessing multi-agent collaboration for translating ultra-long literary texts," 2024. [Online]. Available: https://arxiv.org/abs/2405.11804

[5] B. Ahmadnia and B. Dorr, "Bilingual low-resource neural machine translation with round-tripping: The case of Persian-Spanish," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, R. Mitkov and G. Angelova, Eds. Varna, Bulgaria: INCOMA Ltd., Sep. 2019, pp. 18–24. [Online]. Available: https://aclanthology.org/R19-1003/

[6] B. Ahmadnia and B. J. Dorr, "Augmenting neural machine translation through round-trip training approach," *Open Computer Science*, vol. 9, no. 1, pp. 268–278, 2019. [Online]. Available: https://doi.org/10.1515/comp-2019-0019

[7] Z. Guo, Z. Huang, K. Q. Zhu, G. Chen, K. Zhang, B. Chen, and F. Huang, "Automatically paraphrasing via sentence reconstruction and round-trip translation," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 3815–3821, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2021/525

[8] T. Y. Zhuo, Q. Xu, X. He, and T. Cohn, "Rethinking round-trip translation for machine translation evaluation," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 319–337. [Online]. Available: https://aclanthology.org/2023.findings-acl.22/

[9] N. Crone, A. Power, and J. Weldon, "Quality estimation using round-trip translation with sentence embeddings," 2021. [Online]. Available: https://arxiv.org/abs/2111.00554

[10] J. Moon, H. Cho, and E. L. Park, "Revisiting round-trip translation for quality estimation," in *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, A. Martins, H. Moniz, S. Fumega, B. Martins, F. Batista, L. Coheur, C. Parra, I. Trancoso, M. Turchi, A. Bisazza, J. Moorkens, A. Guerberof, M. Nurminen, L. Marg, and M. L. Forcada, Eds. Lisboa, Portugal: European Association for Machine Translation, Nov. 2020, pp. 91–104. [Online]. Available: https://aclanthology.org/2020.eamt-1.11/

[11] M. Aiken and M. Park, "The efficacy of round-trip translation for mt evaluation," in *Translation Journal*, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:20235137

[12] Y. Kementchedjhieva and A. Søgaard, "Grammatical error correction through round-trip machine translation," in *Findings of the Association for Computational Linguistics: EACL 2023*, A. Vlachos and I. Augenstein, Eds. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2208–2215. [Online]. Available: https://aclanthology.org/2023.findings-eacl.165/

[13] E. Voita, R. Sennrich, and I. Titov, "Context-aware monolingual repair for neural machine translation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng,

and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 877–886. [Online]. Available: https://aclanthology.org/D19-1081/

[14] M. Hermet and A. Désilets, "Using first and second language models to correct preposition errors in second language authoring," in *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, and C. Leacock, Eds. Boulder, Colorado: Association for Computational Linguistics, Jun. 2009, pp. 64–72. [Online]. Available: https://aclanthology.org/W09-2110/

[15] N. Madnani, J. Tetreault, and M. Chodorow, "Exploring grammatical error correction with not-so-crummy machine translation," in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, J. Tetreault, J. Burstein, and C. Leacock, Eds. Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 44–53. [Online]. Available: https://aclanthology.org/W12-2005/

[16] H. Ishikawa, R. Kobayashi, Y. Gato, and T. Akiba, "Pseudo fake data generation using round-trip translation for fact verification on political debates," in *2023 10th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)*, 2023, pp. 1–5.

[17] H.-Q. Nguyen-Son, T. Thao, S. Hidano, I. Gupta, and S. Kiyomoto, "Machine translated text detection through text similarity with round-trip translation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, Jun. 2021, pp. 5792–5797. [Online]. Available: https://aclanthology.org/2021.naacl-main.462/

[18] S. Lai, Z. Yang, F. Meng, X. Zhang, Y. Chen, J. Xu, and J. Zhou, "Generating authentic adversarial examples beyond meaning-preserving with doubly round-trip translation," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 4256–4266. [Online]. Available: https://aclanthology.org/2022.naacl-main.316/

[19] S. Panda, F. Palma Gomez, M. Flor, and A. Rozovskaya, "Automatic generation of distractors for fill-in-the-blank exercises with round-trip neural machine translation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, S. Louvan, A. Madotto, and B. Madureira, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 391–401. [Online]. Available: https://aclanthology.org/2022.acl-srw.31/

[20] C. Yung, H. M. Dolatabadi, S. Erfani, and C. Leckie, "Round trip translation defence against large language model jailbreaking attacks," 2024. [Online]. Available: https://arxiv.org/abs/2402.13517

[21] F. V. Ruiz, A. Grishina, M. Hort, and L. Moonen, "A novel approach for automatic program repair using round-trip translation with large language models," 2024. [Online]. Available: https://arxiv.org/abs/2401.07994

[22] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 483–498. [Online]. Available: https://aclanthology.org/2021.naacl-main.41

[23] jkoppel, "Quixbugs," 2017. [Online]. Available: https://github.com/jkoppel/QuixBugs

[24] appmlk, "Condefects," 2023. [Online]. Available: https://github.com/appmlk/ConDefects