Robust Neural Machine Translation for Abugidas by Glyph Perturbation

Hour Kaing, Chenchen Ding, Hideki Tanaka, Masao Utiyama
Advanced Translation Technology Laboratory,
Advanced Speech Translation Research and Development Promotion Center,
National Institute of Information and Communications Technology,
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{hour_kaing, chenchen.ding, hideki.tanaka, mutiyama}@nict.go.jp

Abstract

Neural machine translation (NMT) systems are vulnerable when trained on limited data. This is a common scenario in low-resource tasks in the real world. To increase robustness, a solution is to intently add realistic noise in the training phase. Noise simulation using text perturbation has been proven to be efficient in writing systems that use Latin letters. In this study, we further explore perturbation techniques on more complex abugida writing systems, for which the visual similarity of complex glyphs is considered to capture the essential nature of these writing systems. Besides the generated noise, we propose a training strategy to improve robustness. We conducted experiments on six languages: Bengali, Hindi, Myanmar, Khmer, Lao, and Thai. By overcoming the introduced noise, we obtained non-degenerate NMT systems with improved robustness for low-resource tasks for abugida glyphs.

1 Introduction

Neural machine translation (NMT) systems have been shown to be vulnerable in noisy settings, where slightly modified inputs cause serious translation failures (Belinkov and Bisk, 2018; Ebrahimi et al., 2018a). Boucher et al. (2022) showed that techniques using pre-trained language models cannot prevent this. This drawback is more disastrous in low-resource scenarios, where the model's robustness becomes a crucial issue.

Several text perturbation techniques have been developed to improve robustness by introducing synthesized textual noise. Typical techniques are DeepWordBug (Gao et al., 2018), TextBugger (Li et al., 2018), and VIPER (Eger et al., 2019). These techniques mostly focus on languages that use alphabetic systems, such as Latin letters. As a more complex writing system, Chinese characters were investigated by Nuo et al. (2020); Zhang et al. (2021). In the present study, we further fill the gap in text perturbation techniques for understudied abugida writing systems, which vary and are used widely in South-East Asia.

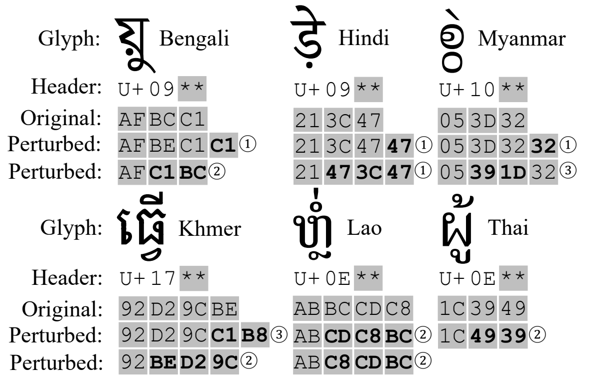

Figure 1: Homoglyph perturbation examples for various abugida systems. The Unicode of each character is listed below the glyph. Perturbed characters are emphasized in bold font. Various patterns cause homoglyphs: 1) repetition, 2) permutation, and 3) decomposition (e.g., `BE` → `C1 B8` in Khmer).

A reasonable perturbation technique should produce meaningful and readable text that is indistinguishable for humans, but disastrous for a system's prediction (Le et al., 2022). Visually similar glyphs or homoglyphs[1] were investigated in Eger et al. (2019); Boucher et al. (2022), and Le et al. (2022) obtained realistic samples from large corpora. As a primary contribution, we further develop these previous studies for abugida writing systems. Some exemplary homoglyphs in various abugida systems are illustrated in Figure 1.

To address noise, we propose a training strategy that leverages adversarial training, subword regularization, and consistency training. We selected six languages that use abugida systems, Bengali, Hindi, Myanmar, Khmer, Lao, and Thai, and experimented on them for low-resource tasks. Overcoming noisy perturbations improved the robustness of NMT systems, with non-degenerate performance.

[1] I.e., glyphs with identical or similar presentations, but different compositions and encodings.

2 Background

2.1 Abugida Writing System

An abugida is a writing system that combines features of both syllabic and segmental systems. Text is written as a sequence of syllables, which resemble Japanese hiragana, but can be broken down into separate consonants and vowels, as in a segmental system. A typical abugida syllable consists of a base consonant accompanied by a default vowel or additional vowels represented by diacritics. In computer systems, these syllables are rendered into glyphs, which are visual symbols in the rendering process. A glyph represents a letter or certain combinations of multiple letters. For example, in Latin, the letter *a* is a glyph, and combined with a grave accent (diacritic), it becomes another glyph *à*; similarly, in abugidas, as shown in Figure 2, a consonant is represented by a glyph, as in (a), and combined with multiple diacritics to become another glyph, as in (b). As in Figure 1, similar glyphs or homoglyphs commonly occur in the composition of complex diacritics, which have numerous patterns and are difficult to engineer. Therefore, we explore such diacritic composition from human-generated corpora. Hereafter, we use the term *glyph* to refer to a visual symbol and *glyph token* to refer to its corresponding Unicode characters.

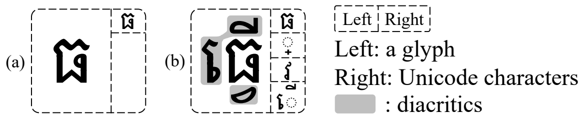

Figure 2: Examples of Khmer glyphs. (a) is a glyph without diacritics and (b) with diacritics.

Even though the issue of homoglyphs for abugidas seem similar to that of the Latin alphabet, the perturbation methods applied to the Latin alphabet cannot be directly extended to abugidas. A crucial reason is the complexity of abugidas glyphs (comprising multiple characters or diacritics) compared to the Latin alphabet (involving single characters). Additionally, such complex glyphs are not predefined in the Unicode table like the Latin alphabet. This work developed a comprehensive process to derive abugida glyphs and identify their potential homoglyphs, thereby enabling us to implement homoglyph perturbations.

2.2 Visual-based Text Perturbation

The objective is to perturb text to cause a system prediction failure while preserving meaning and human readability by replacing characters with other visually similar characters. (Eger et al., 2019; Nuo et al., 2020; Zhang et al., 2021). Eger et al. (2019) replaced each target character with its stylish variants from Unicode data or simply added diacritics above or below the character from a predefined list. Nuo et al. (2020) and Zhang et al. (2021) leveraged a list of handcrafted visually similar glyph characters for replacement. Previous studies represented each character using its glyph image or keywords in the Unicode character description. This work represents each glyph token based on the glyph image and diacritic count as an embedding vector.

3 Proposed Method

3.1 Perturbation for Abugidas

3.1.1 Overall Processing

Given a sentence $\mathbf{x} = (x_1, \ldots, x_n)$, each token x_i has a chance of being replaced with an adversarial candidate $x' \in V$ chosen based on its similarity score w.r.t x_i (Eger et al., 2019), where V is vocabulary that contains all possible tokens, including clean and noisy tokens[2]. A threshold is necessary to prevent undesired x' being assigned to x_i (Ren et al., 2019). The perturbation probability for each targeted token x_i can be formulated as

$$g(x'|x_i) = \begin{cases} \alpha \cdot \dfrac{score(x', x_i, \beta)}{Z(x_i)}, & \text{if } x' \neq x_i \\ 1 - \alpha, & \text{otherwise} \end{cases} \tag{1}$$

$$Z(x_i) = \sum_{x'' \in V \setminus \{x_i\}} score(x'', x_i, \beta) \tag{2}$$

$$score(a, b, \beta) = I(s(a, b) \geq \beta) \cdot s(a, b), \tag{3}$$

where $I(\cdot)$ is an indicator function; α and β control the chance of x_i being perturbed and the similarity threshold, respectively; and $s(a, b)$ is a similarity function between the continuous vectors of two tokens a and b, for example, the cosine similarity $s(a, b) = \cos(v(a), v(b))$, and where $v(\cdot)$ is a vector. The overall perturbation process is illustrated in Figure 3. Next, we present the process for obtaining V from corpora that contain diverse adversarial candidates in Section 3.1.2, and describe how vector $v(\cdot)$ is represented by an image in Section 3.1.3 and by counting diacritics in Section 3.1.4.

[2]As V is fixed in practice, we skip the process if $x_i \notin V$.

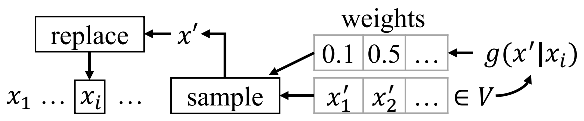

Figure 3: Overall perturbation processing.

3.1.2 Vocabulary Construction

This step is similar to a typical vocabulary preparation process that consists of tokenization and unique token extraction to obtain V. Specifically, we categorize each character as a consonant or diacritic based on Unicode Standard data. Then, we base tokenization on the consonant position such that each token starts with a consonant followed by many or zero diacritics. Hence, we extract a list of unique tokens as V.

Because the similarity is mostly around the diacritics, we want to perturb only the diacritic parts of each targeted token. To achieve this, our trick is to replace the consonant counterpart of each token in V with that of the targeted token x_i, which varies every time step i. This trick is based on the assumption that the visual form of the consonant never changes when it is combined with diacritics. However, we discovered one case in Bengali and Hindi in which the base consonant changed its visual form. Hence, we simply skipped the perturbation for such case.

3.1.3 Image-based Glyph Embeddings (IGE)

We convert each glyph image[3] into a linear vector of $m \cdot n$ dimensions by arranging rows in the $m \times n$ matrix, where each entry corresponds to a pixel in the grayscale image. The pixel values range from 0 (representing the empty area) to 255 (representing the visible part of a glyph). Because the image size varies greatly across glyphs, we predetermine the maximum size $m \times n$ based on all glyphs and then render them into the $m \times n$ size. They must align to the left on the horizontal axis and to the middle on the vertical axis. Additionally, we empty the pixels that correspond to the consonant to ensure that the similarity value is not affected by the common pixels of the base consonant. Finally, we use the cosine similarity function for IGE, which is defined as $s(a, b) = \cos(v(a), v(b))$.

3.1.4 Diacritic-Count Embeddings (DGE)

A simpler approach involves counting the diacritics that exist in a glyph token and how many times they occur. Specifically, a glyph token is represented by a frequency vector, where each entry corresponds to a diacritic in the language and the value of each entry is the count of the corresponding diacritic in the glyph token. Additionally, we smooth each frequency value using an exponent γ. For instance, if a language l has a set of diacritics $\{a, b, c\}$ and a glyph token consists of diacritics acc, DGE represents it using a frequency vector $[1, 0, 2]^\gamma$ because a occurs once and c occurs twice. Using DGE, we can identify two glyphs that have similar sets of diacritics, regardless of the order of the diacritics. We set $\gamma = 0.3$ in all experiments and use the inverse Euclidean distance as the similarity function, which is defined as $s(a, b) = (Euclidean(v(a), v(b)) + 1)^{-1}$.

3.2 Robust NMT Training

To generalize a model in the presence of noisy inputs, we propose a training strategy that maximizes the regularization benefit resulting from combining adversarial training (AT) (Eger et al., 2019), subword regularization (SR) (Kudo, 2018), and consistency training (Wang et al., 2021). During training, the inputs are first perturbed by AT, and then various subwords of the perturbed inputs are sampled by SR to generate variants of the original inputs. This ensures that the variants do not resemble the original inputs and thereby maximizes the regularization benefit. Lastly, consistency training is applied to ensure the consistency of the model's predictions between the original inputs and their variants, as explained in Appendix A.

Various perturbation techniques can be employed in this training strategy, such as random character perturbation (RD) (Karpukhin et al., 2019), which consists of four character-operations (insert, delete, substitute, and swap), or our perturbation technique, which uses IGE or DGE. Because our perturbations were used during both training and inference, we prevented the exposure of the test set during training by constructing a perturbation vocabulary for inference from external corpora (V_{test}) and one for training from the training data (V_{train}), while also ignoring all adversarial candidates that exist in V_{train} during inference.

4 Experiments

4.1 Settings

We experimented on six abugida languages: Bengali (`bg`), Hindi (`hi`), Myanmar (`my`), Khmer (`km`),

[3]We used `Pillow9.4.0` and Google Noto Serif fonts with 100px for all languages.

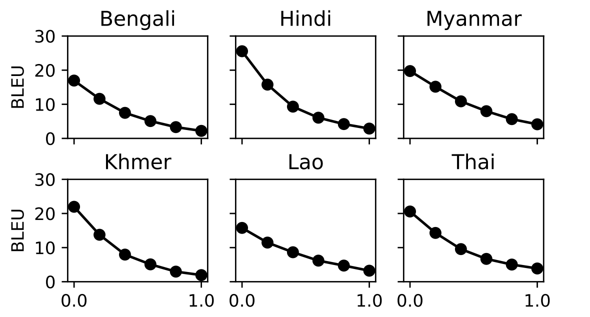

Figure 4: BLEU scores of NMT with α from 0.0 to 1.0, with a step of 0.2 on the x-axis, and β set to 1.0.

Table 1: BLEU results on perturbed inputs. † denotes statistical significance with p-value < 0.01 compared with the second best scores.

	bg	hi	my	km	lo	th
Baseline and comparison methods						
Base	2.2	2.9	4.1	1.9	3.2	3.9
SR	9.3	8.3	14.3	8.4	11.7	8.1
SR_{ct}	9.4	8.3	14.0	8.5	13.7	8.7
RD_{ct}	6.6	9.4	16.7	8.9	13.4	11.9
Proposed robust training with $\beta = 1$, using						
RD	10.2	**12.1**	17.5	13.7	15.5	14.0
DGE	10.9	10.6	**18.2**†	14.8	16.3	14.6
IGE	**12.7**	10.0	16.4	**21.4**	**17.0**†	**18.7**
Improvement of IGE_β with different β						
$\text{IGE}_{0.95}$	-0.2	3.1	-0.7	-0.1	0.3	1.9
$\text{IGE}_{0.90}$	0.4	2.7	-0.4	0.5	-0.1	1.9
$\text{IGE}_{0.85}$	-0.7	1.6	0.7	0.2	0.2	2.1

Lao (lo), and Thai (th). We constructed V_{test} from the cleaned CommonCrawls (Wenzek et al., 2020; Conneau et al., 2020) and evaluated translation performance on the Asian Language Treebank dataset (Riza et al., 2016) from abugida languages to English using SacreBLEU (Post, 2018). Other details are presented in Appendix B.

4.2 Results and Discussion

We evaluated the performance of the vanilla model (Base) with respect to our perturbation technique using IGE with $\beta = 1$. Figure 4 illustrates the performance degradation across all languages. The worst cases achieved a score close to zero; our robust training was investigated for these cases.

Table 1 demonstrates the effectiveness of our approach against perturbation. We trained IGE and DGE with $\alpha = 1$ and $\beta = 1$. First, our robust training using RD outperformed all baselines; in particular, it outperformed SR and RD with consistency training (named SR_{ct} and RD_{ct}, respectively).

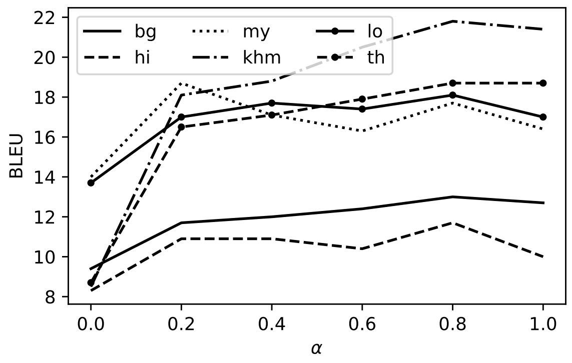

Figure 5: BLEU scores of IGE_α with different α.

This indicates the effect of combining AT with SR. Furthermore, IGE and DGE achieved better results than RD in most cases, signifying the importance of performing our perturbation during training to prevent the models from being affected by similar attacks. Evaluated on clean inputs, our methods were comparable with SR_{ct} and outperformed other baselines because our robust training exploited the effect of SR, as shown in Table 3 in Appendix C.

The last three rows of Table 1 show the further improvement resulting from training IGE using a smaller β. The motivation for this experiment is that our training approach does not benefit from training data that are cleaned or have few homoglyphs. Therefore, using a smaller β to exploit more similar glyphs improved the performance for Hindi and Thai. In addition, because it is possible that attacks with a smaller β, e.g., 0.95, may occur in the real world provided that the readability of the noise is preserved, this setting may be beneficial in such circumstances. However, using a smaller β has the disadvantage that the perturbation would generate some random text, which may make the training of the model more difficult and degrade its performance on clean inputs.

Figure 5 reveals the impact of α on training IGE. It is evident that fine-tuning α is crucial for achieving optimal performance, with $\alpha = 0.8$ emerging as the optimal value for the majority of languages in this study.

5 Related Work

5.1 Text Perturbation

Text perturbation has been extensively studied in the literature, with two scenarios: white-box and black-box. In the white-box scenario, the model's gradients are leveraged (Li et al., 2018; Ebrahimi et al., 2018b), whereas in the black-box scenario, only the model's input and output are known (Li et al., 2018; Ebrahimi et al., 2018b). Various perturbation operations have been proposed, such as randomly characters perturbation (Karpukhin et al., 2019), perturbation based on the keyboard layout and natural typos (Belinkov and Bisk, 2018), extraction of visually similar glyphs of characters (Eger et al., 2019), and similar embedding subwords (Park et al., 2020). Our study explores visually similar glyphs beyond characters.

5.2 Consistency Training

In various studies, researchers have used consistency training in various ways to enhance the performance of natural language processing (NLP) models. Previously, Wang et al. (2021) used consistency training to improve subword tokenization in multilingual models. Xie et al. (2020) and Kambhatla et al. (2022) improved data augmentation techniques for NMT using consistency training. Furthermore, Park et al. (2022) used consistency training on virtual noise to improve the performance of text classification and natural language inference tasks. In this study, we adopted consistency training to regularize our training on the joint sampling of adversarial text and subwords to enhance the robustness of the NMT model against perturbations.

6 Conclusion

In this study, we presented a perturbation approach that leverages visual similarity and introduced a training strategy to maintain the performance of the NMT model. We exposed the vulnerability of the vanilla NMT model through experiments that perturbed test data using homoglyphs, and demonstrated the importance of robust training against text perturbation. The findings of this study can aid future research effort in evaluating the generalization capabilities of NMT models, particularly for low-resource settings and understudied languages.

Limitations

More abugida writing systems should be experimented with. The methods used in this study require eight GPUs, which may not be available to all researchers.

Ethics Statement

Like previous approaches in the NLP text perturbation literature, our approach could be unintentionally used by malicious actors to attack textual machine learning systems. To mitigate this, we implemented precautionary measures. First, access to our perturbation is limited to our private API, with appropriate security measures. Second, we proposed a technique that enhances the model's robustness against our perturbation, as shown in Section 3.2. Thus, we believe that this study will contribute to enhancing the robustness of NLP tasks for low-resource languages. All datasets used in this study are either open-source or released by the original authors.

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proc. of ICLR*.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *Proc. of SP)*, pages 1987–2004. IEEE.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proc. of ACL*, pages 8440–8451.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *Proc. of COLING*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. Hotflip: White-box adversarial examples for text classification. In *Proc. of ACL*.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding nlp systems. In *Proc. of NAACL*, pages 1634–1647.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proc. of SPW)*, pages 50–56. IEEE.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. In *Proc. of EMNLP*, page 6098–6111.

Nishant Kambhatla, Logan Born, and Anoop Sarkar. 2022. Cipherdaug: Ciphertext based data augmentation for neural machine translation. In *Proc. of NAACL*, pages 201—218.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proc. of W-NUT*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proc. of ACL*, pages 66–75.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Proc. of ACL*, pages 2953–2965.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. In *Proc. of NDSS*.

Cheng Nuo, Guo-Qin Chang, Haichang Gao, Ge Pei, and Yang Zhang. 2020. Wordchange: Adversarial examples generation approach for chinese text classification. *IEEE Access*, 8:79561–79572.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL-HLT*.

Jungsoo Park, Gyuwan Kim, and Jaewoo Kang. 2022. Consistency training with virtual adversarial discrete perturbation. In *Proc. of NAACL*, pages 5646–5656.

Jungsoo Park, Mujeen Sung, Jinhyuk Lee, and Jaewoo Kang. 2020. Adversarial subword regularization for robust neural machine translation. In *Proc. of EMNLP*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proc. of WMT*, pages 186–191.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proc. of ACL*, pages 1085–1097.

Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Rapid Sun, Vichet Chea, Sethserey Sam, Sopheap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian language treebank. In *Proc. of O-COCOSDA*, pages 1–6.

Xinyi Wang, Sebastian Ruder, and Graham Neubig. 2021. Multi-view subword regularization. In *Proc. of NAACL*, pages 473–482.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proc. of LREC*, pages 4003–4012.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Proc. of NeurIPS*, volume 33, pages 6256–6268.

Zihan Zhang, Mingxuan Liu, Chao Zhang, Yiming Zhang, Zhou Li, Qi Li, Haixin Duan, and Donghong Sun. 2021. Argot: generating adversarial readable chinese texts. In *Proc. of IJCAI*, pages 2533–2539.

A Consistency Training

The purpose of consistency training is to ensure that a model's prediction for a sequence $\mathbf{x}$ remains consistent with its prediction for the variant sequence $\mathbf{x}'$ (Wang et al., 2021). Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, its objective function can be expressed as

$$\mathcal{L}(\theta) = \sum[-\frac{1}{2}\log p_\theta(\mathbf{y}_i|\mathbf{x}_i) - \frac{1}{2}\log p_\theta(\mathbf{y}_i|\mathbf{x}'_i) + \lambda D(p_\theta(\mathbf{y}_i|\mathbf{x}_i)||p_\theta(\mathbf{y}_i|\mathbf{x}'_i))], \quad (4)$$

where θ is a set of model parameters and $D(\cdot||\cdot)$ is a non-negative distance metric between two distributions that are controlled by the hyperparameter λ. Following Wang et al. (2021), we use Kullback–Leibler divergence for $D(\cdot||\cdot)$ and set $\lambda = 0.2$.

B Other Settings

B.1 Dataset

The ALT data were released under `CC-BY-4.0`[4]. The terms of use of CommonCrawls can be found on its official website[5]. We used these data and followed their intended use for this study. For the translation, we split the data into training, validation, and test sets following the ALT standard[6]. We tokenized the training, validation, and test sets using SentencePiece based on the unigram language model with a joint vocabulary of 4k.

B.2 Implementation

We used the transformer architecture for all the models and implemented them using Fairseq (Ott et al., 2019) in our experiments. We trained all the models on the eight GPUs (Tesla V100 SXM2 with 32 GB memory) and the number of parameters was approximately 54 million. We mostly based the

[4]https://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/ALT-Parallel-Corpus-20191206/README.txt

[5]https://commoncrawl.org/terms-of-use/

[6]https://www2.nict.go.jp/astrec-att/member/mutiyama/ALT

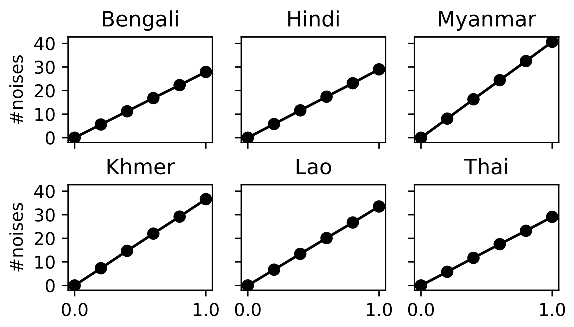

Figure 6: Average amount of noise per sentence on the test set with various values of α.

Table 2: chrF++ results on perturbed inputs.

	bg	hi	my	km	lo	th
Baseline and comparison methods						
Base	24.0	22.2	28.2	22.5	22.4	23.9
SR	35.5	32.6	40.4	34.9	35.4	31.2
SR$_{ct}$	35.2	35.6	42.9	32.0	37.7	32.8
RD$_{ct}$	32.7	38.0	43.7	33.8	35.2	33.8
Proposed robust training with $\beta = 1$, using						
RD	35.5	**41.4**	**45.8**	41.0	39.9	40.3
DGE	36.8	39.3	44.1	43.1	41.4	40.4
IGE	**39.3**	38.8	44.6	**47.6**	**42.4**	**44.5**
Improvement of IGE$_\beta$ with different β						
IGE$_{0.95}$	−0.3	2.9	−0.6	−0.5	0.0	1.2
IGE$_{0.90}$	0.1	2.6	−1.3	0.0	−0.4	1.6
IGE$_{0.85}$	−1.5	1.0	−1.3	−0.2	−0.4	1.8

configuration on Guzmán et al. (2019), which was specifically designed for the Indic low-resource NMT setting. However, we further fine-tuned the number of epochs and found that increasing the number of epochs to 1k achieved improvements across all models.

The hyperparameters for SR, that is, the n-best size l and the distribution smoothness μ, were also fine-tuned and the best setting was $l = \infty$ and $\mu = 0.2$. In addition, for RD, the perturbation probability (which is equivalent to our perturbation hyperparameter α) was set to 1.

C Additional Results and Discussion

The impact of our perturbation technique with $\beta = 1$ on the dataset was measured by the amount of noise introduced, as shown in Figure 6. Figure 6 shows the per-sentence amount of homoglyph noise that was added during inference. The results plotted in Figure 4 show a strong correlation between α and the amount of noise per sentence, which explains why a larger α degrades the translation performance more. Table 2 further shows the chrF++ results on noisy inputs.

Table 3: BLEU results on clean inputs.

	bg	hi	my	km	lo	th
Baseline and comparison methods						
Base	17.0	25.6	19.7	22.0	15.8	20.6
SR	17.2	26.5	19.8	22.3	17.2	20.7
SR$_{ct}$	19.6	**28.3**	**22.3**	**24.1**	**18.9**	22.9
RD$_{ct}$	17.6	25.5	21.0	23.8	18.2	22.3
Proposed robust training with $\beta = 1$, using						
RD	**19.7**	27.9	**22.3**	23.9	18.7	**23.2**
DGE	19.1	28.2	21.4	23.9	**18.9**	22.8
IGE	19.5	28.1	22.1	23.6	18.5	23.0
Improvement of IGE$_\beta$ with different β						
IGE$_{0.95}$	−1.0	−1.3	−1.1	0.3	−0.4	−1.0
IGE$_{0.90}$	−0.5	−1.0	−1.4	−0.3	−0.8	−0.7
IGE$_{0.85}$	−1.0	−0.8	−1.8	0.2	−0.1	−0.6

Table 4: chrF++ results on clean inputs.

	bg	hi	my	km	lo	th
Baseline and comparison methods						
Base	43.8	52	45.6	47.9	41.7	46.2
SR	44.0	52.8	45.6	48.3	42.6	46.1
SR$_{ct}$	46.5	**54.4**	**48.3**	**49.9**	44.3	48.1
RD$_{ct}$	44.5	51.8	46.4	49.6	43.7	47.9
Proposed robust training with $\beta = 1$, using						
RD	**46.9**	53.9	48.2	**49.9**	44.2	**48.4**
DGE	46.4	54.3	47.2	49.7	**44.4**	47.9
IGE	46.4	**54.4**	47.7	49.7	43.8	48.1
Improvement of IGE$_\beta$ with different β						
IGE$_{0.95}$	−0.6	−1.3	−1.2	−0.1	0.0	−0.6
IGE$_{0.90}$	−0.3	−1.0	−1.3	−0.5	−0.2	−0.4
IGE$_{0.85}$	−0.6	−0.8	−1.4	−0.1	0.2	−0.4

Table 3 and 4 summarizes the performance of all NMT models on clean inputs. The results show that our models achieved performance comparable with that of the state-of-the-art SR$_{ct}$ for all languages. Additionally, the comparison of RD$_{ct}$ with our robust training using RD reveals that combining AT with SR affected the performance on the clean inputs in addition to the noisy inputs. Nonetheless, using a smaller β slightly degraded the performance on clean inputs.

Table 5 highlights the perturbed examples obtained by IGE using $\beta = 1$, 0.95, and 0.9. The examples with $\beta = 1$ are identical to their original clean examples. For $\beta = 0.95$ and 0.9, diacritics with minor visibility were added, mostly above and below the original glyphs. Even though these

Table 5: Perturbed examples.

bg	clean:	এই ফ্লুটি খুবই সংক্রামক কিন্তু এটি মানুষের দেহে ছড়াতে পারে না।
	β=1:	এই ফ্লুটি খুবই সংক্রামক কিন্তু এটি মানুষের দেহে ছড়াতে পারে না।
	β=.95:	এই ফ্লুটি খুবই সংক্রামক কিন্তু এটি মাঁনুষের দেহে ছড়াঁতে পঁ◌ারে নঁ◌া।
	β=.9:	এই ফ্লুটি খুঁবই সঁংক্রামক কিন্তু এটি মঁ◌ানূষের দেহে ছড়ঁ◌াতে পাঁরে নঁ◌া।
hi	clean:	फ्लू बेहद संक्रामक है लेकिन इसे मनुष्यों में नहीं हो सकता है।
	β=1:	फ्लू बेहद संक्रामक है लेकिन इसे मनुष्यों में नहीं हो सकता है।
	β=.95:	फ्लू बेहद सोंक्रामक है लेकिंन इसे मऩुष्यों में नहीं हॉ सकतो है।
	β=.9:	फ्लू बेहद सोंक्रामक है लेकिंन इसे मऩुष्यों में नहाँ हौं सकतौं है।
my	clean:	တုတ်ကွေး သည် အလွန် အလွယ်တကူ ကူးစက် သော်လည်း လူသားများ သို့ မထုတ်လွှင့်နိုင်ပါ ။
	β=1:	တုတ်ကွေး သည် အလွန် အလွယ်တကူ ကူးစက် သော်လည်း လူသားများ သို့ မထုတ်လွှင့်နိုင်ပါ ။
	β=.95:	တုံတ်ကွေ့း သည် အလွဂ်န် အလွငယ်တကို ကူ့စက် သော်လည်း လုူသားများ့ သို့ မထုံတ်လွှင့်နှိုင်ပီ ။
	β=.9:	တွုတ်ကွျေ့း သည် အလွှ်န် အလွွယ်တကှု ကူ့စက် သော်လည်ိး လုိသံားမျ်ား သို့ မထွုတ်လွှင့်နှိုင်ပိ ။
km	clean:	មេរោគនេះងាយឆ្លងតែមិនឆ្លងដល់មនុស្សឡើយ។
	β=1:	មេរោគនេះងាយឆ្លងតែមិនឆ្លងដល់មនុស្សឡើយ។
	β=.95:	ម៉េរោគនេះង៉ាយឆ្លុងតែ៉មិនឆ្លងដល់មនុ៉ស្ស៍ឡើយ។
	β=.9:	ម៉េ៊រោិគនេ៍ះង៉ាយឆ្លុ៍ងតែិមីនឆ្លុងដល់មនិស្ស្សៀយ។
lo	clean:	ໄຂ້ຫວັດໃຫຍ່ແມ່ນ ພະຍາດຕິດຕໍ່ຂະໜາດສູງ ແຕ່ບໍ່ສາມາດຕິດຕໍ່ຫາຄົນໄດ້.
	β=1:	ໄຂ້ຫວັດໃຫຍ່ແມ່ນ ພະຍາດຕິດຕໍ່ຂະໜາດສູງ ແຕ່ບໍ່ສາມາດຕິດຕໍ່ຫາຄົນໄດ້.
	β=.95:	ໄຂ້ຫວັ່ດໃຫຍ່ແມ່ນ ພ່ະຍ໋າດຕິດຕໍ່ຂະໜ່າດສູ່ງ ແຕ່ບໍ່ສໍາມ່າດຕິດຕໍ່ຫ໋າຄົນໄດ້.
	β=.9:	ໄຂ້ຫວັ໊ດໃຫຍ່ແມ່ນ ພ່ະຍ່າດຕຶດຕໍ່ຂີະໜ້າດສູງ ແຕ່ບໍ່ສໍາມຶາດຕຶດຕໍ່ຫໍາຄົນໄດ້.
th	clean:	สนามแข่งม้าแร็นด์วิคถูกปิดและคาดว่าจะยังคงปิดอยู่ต่อไปอีกถึง 2 เดือน
	β=1:	สนามแข่งม้าแร็นด์วิคถูกปิดและคาดว่าจะยังคงปิดอยู่ต่อไปอีกถึง 2 เดือน
	β=.95:	สนำมแข่งม้ำแร็นด์วิคถูกปิดและคำดวาจ่ะยังคงปิดอยู่ต่อไปอีกถึง 2 เดือน
	β=.9:	สน่ำมแข่งม่าแร็นด์วีคถู๋กปีดและค่าดว์า่จ์ะยั่งคงปีดอยู่ต่อไปอิกถึง 2 เดือน

noisy diacritics are noticeable, the readability of the examples is probably preserved because humans have a strong recognition capability. A native Khmer speaker, who was asked to read the Khmer examples, could understand the one obtained with $\beta = 0.9$ without seeing the highlights or the corresponding original example. However, we observed that the reading speed was slower than usual because the speaker was looking at the context to understand the perturbed glyphs. More extensive assessments with native speakers are required in our future study to better understand the potential glyph attacks using smaller values of β.