Monotonic Simultaneous Translation with Chunk-wise Reordering and Refinement

Hyojung Han[1*†], **Seokchan Ahn**[1*†], **Yoonjung Choi**[1], **Insoo Chung**[1†], **Sangha Kim**[1], **Kyunghyun Cho**[2]
[1]Samsung Research, Seoul, Republic of Korea
[2]New York University, New York, United States
hjhan@cs.umd.edu, seokchaa@uci.edu, yj0807.choi@samsung.com,
insoo.chung@tamu.edu, sangha01.kim@samsung.com, kyunghyun.cho@nyu.edu

Abstract

Recent work in simultaneous machine translation is often trained with conventional full sentence translation corpora, leading to either excessive latency or necessity to anticipate as-yet-unarrived words, when dealing with a language pair whose word orders significantly differ. This is unlike human simultaneous interpreters who produce largely monotonic translations at the expense of the grammaticality of a sentence being translated. In this paper, we thus propose an algorithm to reorder and refine the target side of a full sentence translation corpus, so that the words/phrases between the source and target sentences are aligned largely monotonically, using word alignment and non-autoregressive neural machine translation. We then train a widely used wait-k simultaneous translation model on this reordered-and-refined corpus. The proposed approach improves BLEU scores and resulting translations exhibit enhanced monotonicity with source sentences.

1 Introduction

Simultaneous interpretation is widely used in various scenarios such as cross-lingual communication between international speakers, international summits, and streaming translation of a live video. Simultaneous interpretation has a latency advantage over conventional full-sentence translation, i.e. offline translation, as it requires only partial sequence to start translating. However, as the source and target languages differ in word orders, there is a difficulty in simultaneous interpretation that does not exist in offline translation which translates only after the whole source sentence is received. For example, when dealing with language pairs that significantly differ in word order (e.g., between SOV language and SVO language), an interpreter may not receive sufficient information with partial sequence to start generating a translation that respects the natural order of the target language. One of the approaches to address this problem is to perform *anticipation*[1]. Note that the nature of anticipation relies on interpreters' assumptions and the anticipation may provide incorrect translations. Alternatively, human interpreters strategically resort to producing *monotonic translations* that follow the word order of the source sequence (Cai et al., 2020).

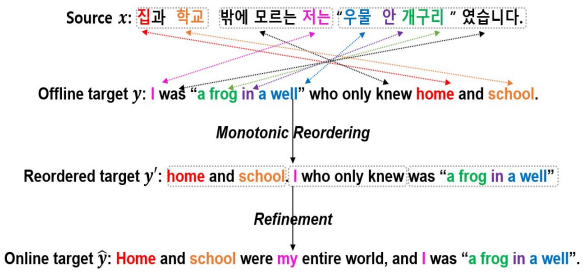

Figure 1: An example illustration of monotonic reordering and refinement for simultaneous translation

To illustrate the differences between the two strategies, the example in Figure 1 may be referred to. Of the two targets, offline target $\mathbf{y}$ respects the target language order and an online target $\hat{\mathbf{y}}$ roughly follows the source word ordering. Successful anticipation in Figure 1's case would be to predict the initial words in $\mathbf{y}$ (I was a "frog in a well") before receiving the full $\mathbf{x}$. This would pose difficulty even to professional translators as all the relevant information is in the latter part of the $\mathbf{x}$ (저는$_{I}$ /"우물$_{a\ well}$ / 안$_{in}$ / 개구리$_{a\ frog}$" 였습니다$_{was}$.). Bartłomiejczyk (2008) reports the success rate of human interpreters' anticipation attempts to be as low as 38.1% even though they make predictions based on pre-acquired domain knowledge. On the other hand, a monotonic approach would be to gen-

* Equal contribution
† Work done at Samsung Research

[1]A simultaneous interpretation strategy where the interpreter says information that is not yet said by the speaker.

Proceedings of the Sixth Conference on Machine Translation (WMT), pages 1110–1123
November 10–11, 2021.

erate an $\hat{y}$ style translation - the grammaticality in the resulting sequence is sacrificed to translate only the received information.

A similar case applies to Simultaneous Machine Translation (SimulMT) models, which start translating before a whole sentence is given. Several studies (Ma et al., 2019; Arivazhagan et al., 2019; Ma et al., 2020) often utilize offline full-sentence translation corpora to train SimulMT models. Offline full-sentence parallel corpora are expected to follow the natural order of languages and mostly contain source to offline-target pairs. Naturally, when SimulMT models are trained on these corpora, the models inevitably learn to perform anticipation. Recent SimulMT studies are focused on reducing anticipation (Zhang et al., 2020a) or performing better anticipation (Zhang et al., 2020b). On the contrary, studies on enabling monotonic translation in SimulMT are scarcely available. Recently, Chen et al. (2020) suggest utilizing pseudo-references for monotonic translation.

In this paper, we propose a paraphrasing method to generate a monotonic parallel corpus to allow a monotonic interpretation strategy in SimulMT. Our method consists of two stages. The method first chunks source and target sequences into segments and monotonically *reorders* the target segments based on source-target word alignment information (Section 3.1). Then, the reordered targets are *refined* to enhance fluency and syntactic correctness (Section 3.2). To show the effectiveness of our method, we train wait-k models (Ma et al., 2019) on the resulting monotonic parallel corpus of reordering-and-refinement. Results show improvements in BLEU scores over baselines and models producing monotonic translations. Our main contributions are as follows:

- We propose a method to reorder and refine the target side in an offline corpus to build a monotonically aligned parallel corpus for SimulMT.
- We investigate the monotonicity in different language pairs, and show monotonicity can be improved after the reordering-and-refinement process.
- We train widely used wait-k models on generated monotonic parallel corpora in multiple language pairs. The results show improvements over baselines in both translation quality and monotonicity.

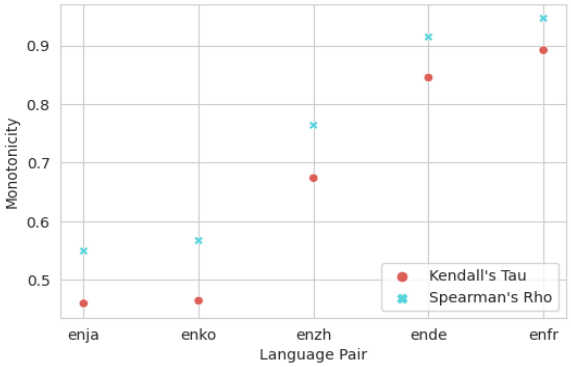

Figure 2: Monotonicity measured on offline trainsets. Utilized data is described in Section 4.1. As Kendall's τ and Spearsman's ρ show similar patterns, we only report Kendall's τ measurements in the rest of the paper.

2 Monotonicity Analysis

In this section, we analyze the degree of word order differences in multiple language pairs, i.e., the monotonicity in different language pairs. To measure the monotonicity, two rank correlation statistics are utilized: Kendall's τ and Spearman's ρ. The analyzed language pairs are: English-{Korean, Japanese, Chinese, German, French}.

According to Polinsky (2012), English is a head-initial language and Korean and Japanese are rigid head-final languages; Korean and Japanese are likely to exhibit extreme word order differences with English. German and Chinese are considered a mixture of head-final and head-initial languages; they are likely to have word differences with English, but not as severe as Korean or Japanese. French is also head-initial, so English and French pair is likely to have similar word order.

Figure 2 show monotonicity measurements between English and five different languages which vary in monotonicity: English-German and English-French pairs show high monotonicity, while English-Japanese and English-Korean pairs show low monotonicity.

Lower monotonicity in language pairs presents higher difficulties for SimulMT tasks. For example, wait-k algorithm only sees $k + t$ source tokens to generate a target token at step t which could lead to unwanted anticipation. To avoid such anticipation, as we mentioned in Section 1, human interpreters often provide a monotonic translation. In the same sense, we conjecture that promoting monotonicity in training corpora is beneficial for translation quality in SimulMT.

3 Monotonic Reordering and Refinement

In this section, we describe our proposed paraphrasing method of chunk-wise reordering and refinement to generate monotonic corpus for SimulMT. Given source $\mathbf{x} = \{x_1, x_2, \cdots, x_{|\mathbf{x}|}\}$ and offline full sentence target $\mathbf{y} = \{y_1, y_2, \cdots, y_{|\mathbf{y}|}\}$, an alignment $\mathbf{a}$ is defined as a set of position pairs of $\mathbf{x}$ and $\mathbf{y}$.

$$\mathbf{a} = \{(s, t) : s \in \{1, \cdots, |\mathbf{x}|\}, t \in \{1, \cdots, |\mathbf{y}|\}\}$$

First, in chunk-wise reordering phase, we generate source chunk set $\mathcal{C}^X$

$$\mathcal{C}^X = \{(x_{1:p_1}), (x_{p_1+1:p_2}), \cdots, (x_{p_{k-1}+1:p_k})\},$$

where $0 < p_1 < p_2 < \cdots < p_k = |\mathbf{x}|$ and reordered target chunk set $\mathcal{C}^Y$

$$\mathcal{C}^Y = \{(y'_{1:q_1}), (y'_{q_1+1:q_2}), \cdots, (y'_{q_{k-1}+1:q_k})\},$$

where $0 < q_1 < q_2 < \cdots < q_k = |\mathbf{y}|$, and $y'_i \in \mathbf{y}'$ is reordered target token from offline target $\mathbf{y}$. The elements of a reordered target chunk $\mathcal{C}^Y_i$ are corresponding target tokens of a source chunk $\mathcal{C}^X_i$ based on given alignment information $\mathbf{a}$. Also, we preserve the original target order within each $\mathcal{C}^Y_i$. For example, offline and reordered target in Figure 1 correspond to $\mathbf{y}$ and $\mathbf{y}'$ respectively, and both sequences are only different in token orders. The number of source chunks in one sentence is the same as the number of reordered target chunks ($|\mathcal{C}^X| = |\mathcal{C}^Y|$), while the number of tokens in $|\mathcal{C}^X_i|$ and $|\mathcal{C}^Y_i|$ could vary. We experiment two chunking methods; fixed-size chunking and alignment-aware adaptive size chunking.

Given chunked sets $\mathcal{C}^X$ and $\mathcal{C}^Y$, we refine reordered target tokens to generate more natural and fluent sentence with a Non-Autoregressive Translation (NAT) model. In the refinement algorithm, final paraphrased sentence $\hat{\mathbf{y}}$ is generated from reordered sequence $\mathbf{y}'$. Furthermore, we incorporate an Autoregressive Translation (AT) model into our refinement process. The more detailed steps for each phase will be explained in the following subsections.

3.1 Chunk-wise Reordering

3.1.1 Fixed-size Chunk Reordering

In the fixed-size chunk reordering method, we simply chunk a sequence of tokens into fixed size segments. The source chunk set $\mathcal{C}^X$ in this chunking method is as follows:

$$\mathcal{C}^X = \{(x_{1:K}), (x_{K+1:2K}), \cdots, (x_{[|\mathbf{x}|/K]:|\mathbf{x}|})\},$$

where $K \in [1, |\mathbf{x}|]$ is chunk size. If $k = 1$, $\mathcal{C}^X$ is identical with $\mathbf{x}$. We conduct subword operation such as sentencepiece or BPE after chunking process in order to avoid subword separation.

3.1.2 Alignment-Aware Chunk Reordering

In the alignment-aware chunking method, we segment a sentence adaptively by leveraging alignment information $\mathbf{a}$, as described in Algorithm 1. The left grid in Figure 3 presents the subword alignment between source and target sentence. We run aligner on subword over word because the alignment performance is consistently better Zenkel et al. (2020) when using GIZA++ (Och and Ney, 2003) , which we use in our experiments. Based on this alignment information, we initialize a list of chunks $\mathcal{C}$. As observable, there are some tokens which have no alignment information. To avoid omission, we assign the same alignment as the previous token; if a token is at the head, it follows the next token's alignment. To ensure subwords can be properly detokenized after reordering, we merge mid-splitted subwords. The middle grid in Figure 3 presents the result of these initialization steps. After initialization, we generate consistent chunks by merging all the inconsistent ones, following the definition of consistency in Zens et al. (2002). In a consistent chunk, tokens are only aligned to each other, not to tokens in other chunks. If any chunk in $\mathcal{C}$ has size smaller than a minimum size threshold δ, we merge a chunk pair that are adjacent in both source and target side and have the shortest target distance between them. If the distances are the same between multiple candidate pairs, we choose the pair of chunks that makes the smallest size after merging. We additionally merge the chunks adjacent to the merged one if they are arranged monotonically. Merging is repeated until all chunks meet the size requirements. An example of final result is the right grid in Figure 3. Phrase extraction method used in statistical machine translation Koehn (2004) also makes phrase level alignments from word alignments using heuristics like ours, but it tends to choose shorter phrases since the number of co-occurrences decrease drastically as the phrase size grows, which makes it difficult to generate larger chunks to prevent hurting grammatical correctness while reordering phase.

3.2 Refinement

Reordered target results from previous phase inevitably entail irregularities mainly for two rea-

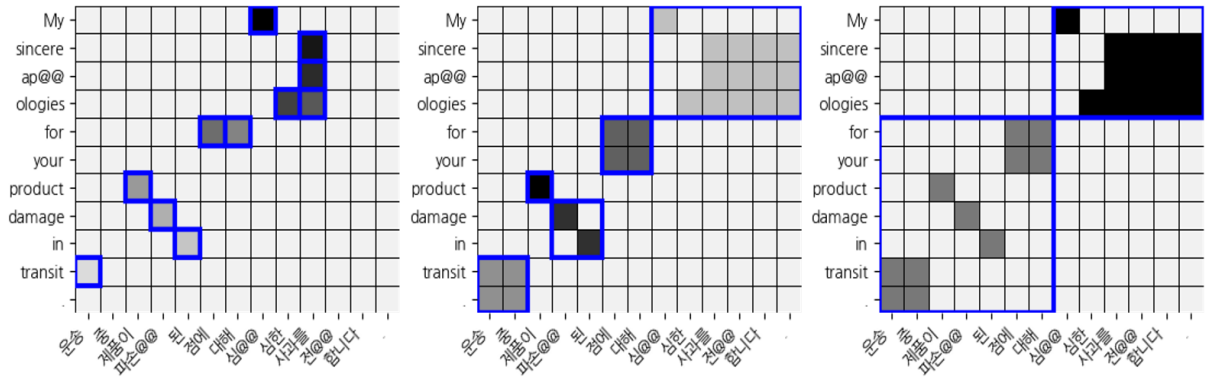

Figure 3: Example of alignment-aware reordering process. **Left**: Original alignments. **Center**: (Initialization) After filling align vacancy, merging mid-split subwords and enforcing the consistency requirement. **Right**: After merging all the chunks shorter than length thresholds.

Algorithm 1: Alignment-Aware reordering

Input: Source sentence $\mathbf{x}$ and target sentence $\mathbf{y}$
Output: Monotonically aligned chunks $\mathcal{C}'$

$\mathbf{a}$ = Alignment between $\mathbf{x}$ and $\mathbf{y}$
$\mathcal{C}$ = Initialize chunks $\mathcal{C}$
$\mathcal{C}$ = Merge all the inconsistent chunks in $\mathcal{C}$
while $|C_i^X| < \delta_{src}$ **or** $|C_i^Y| < \delta_{tgt}$ **for any** $\mathcal{C}_i$ **in** $\mathcal{C}$ **do**
  $\mathcal{C}_k$ = The smaller of the chunks adjacent to $\mathcal{C}_i$
  Merge $\mathcal{C}_i$ and $\mathcal{C}_k$
  Merge monotonic chunks adjacent to $\mathcal{C}_i$
end
$\mathcal{C}'$ = Reorder target side of $\mathcal{C}$ monotonically

Algorithm 2: Chunk-wise Refinement

Input: Source and target chunks $\mathcal{C}^X, \mathcal{C}^Y$
Output: Paraphrased target $\hat{\mathbf{y}}$

$i = 1$
$\hat{Y}_0 = []$
while $i \leq |\mathcal{C}^X|$ **do**
  $X_i = \mathcal{C}^X_{1:i}$ and $Y'_i = [\hat{Y}_{i-1}; \mathcal{C}^Y_i]$
  $\hat{Y}_i = \arg\max_Y \log p^{\mathcal{R}}(Y|X_i, Y'_i)$
  $i = i + 1$
end
Return: $\hat{Y}_{|\mathcal{C}^X|}$

sons. One could be broken connectivity of collocations in segmentation process. The other would be disfluently missing or containing words of endings and preposition as the position of chunk has been changed, thus requiring an addition of new words or clearing unnecessary words. In this part, we focus on refining aforementioned anormalities in order to enhance fluency, while preserving the monotonicity at the same time.

3.2.1 Refinement with NAT

We iteratively decode partial source $\mathcal{C}^X$ with pretrained translation model, given partial reordered target $\mathcal{C}^Y$ as a guidance in order to generate corresponding online target $\hat{Y}$. More specific process is explained in Algorithm 2. As the model refines given $[\hat{Y}_{i-1}; \mathcal{C}_i^Y]$, previous refined output $\hat{Y}_{i-1}$ could be altered as the model re-generates the entire sequence from scratch. Similarly in translation (Arivazhagan et al., 2020; Han et al., 2020b), we set an option of fixed or alterable prefix to force the model whether to generate same target prefix of $\hat{Y}_{i-1}$ or to allow the model to modify the prefix. As we limit the visibility of source information and iteratively generate target tokens with increasing source chunks, we expect the refinement model to generate monotonically aligned and paraphrased targets $\hat{\mathbf{y}}$ with enhanced fluency.

We use NAT architecture as the core refinement model $\mathcal{R}$. In NAT inference, the model's decoder is first given source features and fed an empty target sequence. Then the NAT decoder develops the empty sequence into a translation of the source sequence. This development is often iterative. Note that at every iteration step, the target sequence is refined - closer to the source sequence in meaning and become more fluent. This motivates us to utilize NAT architecture in our refinement process for monotonic-yet-disfluent sequences. In our approach, the NAT model starts refinement iteration with initialized tokens of previous output and

reordered target chunk Y_i', instead of an empty sequence. This target initialization act as a weak supervision to generate monotonically aligned target, which allow model to focus only on the fluency the reordered targets.

3.2.2 Incorporation of AT

Despite the aptness of NAT structure to our refinement phase, NAT model entails a performance degradation compared to AT model in the expense of speedup. Also, there exists repetition problem in NAT (Lee et al., 2018; Gu and Kong, 2021) which is generated in the process of multiple chunk-wise iterative refinement. In order to complement the aforementioned weaknesses of NAT decoding, we incorporate AT into our refinement process with NAT model.The final probability is computed jointly with the probability of AT and NAT model:

$$p^{\mathcal{R}}(Y|X_i, Y_i') \propto p^{AT}(Y|X_i)^{\alpha} \cdot p^{NAT}(Y|X_i, Y_i')^{(1-\alpha)}, \quad (1)$$

where $\alpha \in [0, 1]$ is hyper-parameter deciding the ratio between AT and NAT probability.

Size/L_{avg}	EnKo	EnJa	DeEn	EnZh
Train	3.4M/22	3.9M/12	4M/28	15.9M/27
Valid	800/19	4451/17	3000/25	4000/30
Test	1429/21	1194/17	2169/25	4000/30
AlignAw	3.1M	1.7M	2.2M	6.5M
er	0.85	1.15	0.98	1.12

Table 1: Data statistics and average of En token length L_{avg} of used corpus. AlignAw denotes number of pairs processed with alignment aware refinement. er denotes emission rates used in wait-k decoding. Token lengths and er are measured base on subwords counts.

4 Experiments

4.1 Dataset

In this section, we describe the utilized datasets. Detailed statistics are presented in Table 1. Utilized EnKo trainset and devset are created using in-house translation corpora while test scores are reported on IWSLT17 (Cettolo et al., 2017) EnKo testset. The DeEn trainset of WMT15 translation task (Bojar et al., 2015) is utilized. *newstest2013* is utilized as devset and *newstest2015* is used as testset. The EnJa trainset and validset are respectively the combination trainsets and validsets of KFTT (Neubig, 2011), JESC (Pryzant et al., 2018), TED (Cettolo et al., 2012). The trainset and validset are used as preprocessed and provided by the MTNT authors[2] (Michel and Neubig, 2018). Only the TED portion of testsets is used. For EnZh training UN Corpus v1.0 (Ziemski et al., 2016) is used. Trainset, devset, and testset follow the original splits. Monotonicity of EnFr in Figure 2 is measured on the WMT14 (Bojar et al., 2014) trainset. Additional details regarding utilized tokenization and vocabulary training are listed in Appendix A.

4.2 Metric

All the BLEU scores are cased-BLEU measured using sacreBLEU (Post, 2018). Test scores are measured using models that report best BLEU on their respective devsets. All references and translations of each Korean, Japanese, and Chinese languages are tokenized prior to BLEU evaluation. Tokenizers utilized are mecab-ko[3], KyTea [4], and jieba for Korean, Japanese and Chinese respectively. We report detokenized BLEU on DeEn results. To measure monotoniticy, we use Kendal's τ rank correlation coefficient.

4.3 Implementation Details

The default setting for NMT and SimulMT models follow the base configuration of transformer (Vaswani et al., 2017). SimulMT models are trained using wait-k algorithm, where $k \in \{4, 6, 8, 10, 12\}$, with uni-directional encoder similarly to Han et al. (2020a). The base NMT and SimulMT models are trained up to 300k train steps on a single GPU - each step is performed on a batch of approximately 12288 tokens. For refinement, we utilize NAT models of Levenshtein transformer architecture (Gu et al., 2019) with maximum iteration of 1. The NAT models are trained using sequence-level knowledge distillation (Kim and Rush, 2016) - the references of trainset pairs are replaced with beam search results ($beam = 5$) of NMT teachers. The NAT models follow base configurations and teacher NMT models follow big configuration. Both types of models are trained up to 300k steps on 8 GPUs. In each training step, a 8192 tokens batch is used per GPU. Additional implementation details can be found in the Appendix B.

[2] https://www.cs.cmu.edu/~pmichel1/mtnt/

[3] https://github.com/hephaex/mecab-ko

[4] http://www.phontron.com/kytea/

Wait-k BLEU	k=4	k=6	k=8	k=10	k=12
Offline	10.9	12.1	12.6	12.8	13.4
Fixed + NAT	11.2	12.3	**13.5**	**13.5**	**13.7**
AlignAw + NAT	11.4	**12.8**	12.7	12.9	13.1
AlignAw + NAT + AT ($\alpha = 0.25$)	**11.7**	12.6	12.4	12.7	13.2
AlignAw + NAT + AT ($\alpha = 0.50$)	10.7	12.4	13.0	13.0	13.2
Wait-k Kendal's τ	k=4	k=6	k=8	k=10	k=12
Offline	0.65526	0.60792	0.58440	0.55169	0.53701
Fixed + NAT	0.71822	0.66811	0.63154	0.61244	0.59478
AlignAw + NAT	0.71903	0.69101	0.67149	0.65985	0.64043
AlignAw + NAT + AT ($\alpha = 0.25$)	**0.73215**	0.69386	0.69524	0.67593	0.63335
AlignAw + NAT + AT ($\alpha = 0.50$)	0.73055	0.70106	0.67674	0.65559	0.64042

Table 2: BLEU scores and monotonicity measurements of EnKo wait-k models trained on offline and reordered-and-refined corpora. Note that monotonicity is measured between the model translations and testset references.

4.4 Corpus Generation and Training

We demonstrate the effectiveness of our reordering-and-refinement method by training wait-k models on the resulting datasets. The wait-k models are trained on the combination of the monotonically aligned training pairs and offline trainset. AlignAw + NAT + AT denotes monotonically aligned corpora using alignment-aware reordering and refinement using joint probability of NAT and AT models. And Offline refers to the offline full-sentence corpus.

4.4.1 Reordering

Fixed: For fixed-size reordering, we experiment with chunk sizes $K \in \{4, 6, 8, 10, 12\}$. In wait-$k$ training, k and K are matched. All fixed-size reordered-and-refined corpora have the same size as corresponding offline corpus.

AlignAw: For each corpus, we generate four variations of alignment-aware reordering with source and target minimum chunk size of $2, 3$. Alignment-aware reordering is not applicable on the already-monotonic cases and the sentence pairs which are locally non-monotonic inside a chunk and globally monotonic among chunks within a single pair - typically, the reordering method is applicable to 20% to 50% of offline corpus.

We gather unique pairs from the created four variations to generate the final reordered pairs. The statistics of reordered set for each translation direction is in Table 1. The resulting pairs are refined and combined with corresponding offline corpus to train wait-k models. Here, same set of reordered pairs are utilized for all k settings.

seq-rep-n	Offline	NAT	NAT + AT
1-gram	0.036	0.076	0.072
2-gram	0.008	0.022	0.018
3-gram	0.003	0.009	0.006
4-gram	0.001	0.004	0.002

Table 3: N-gram repetition rate measured on offline and reordered-and-refined EnKo corpora. $\alpha = 0.25$ is set to for NAT + AT

4.4.2 Refinement

NAT: NAT models are utilized to refine the reordered pairs. Both the fixed prefix and alterable prefix refinement is performed and combined. BertScore (Zhang et al., 2020c) is measured and used to discard refinement results that show below average scores. The size of the resulting set is the same as the corresponding offline corpus.

NAT + AT: NAT and AT models can both be utilized to jointly compute token probability in refinement (Section 3.2.2). The AT models utilized are the baseline wait-k models trained on offline corpora. We experiment with $\alpha \in \{0.25, 0.5\}$. The examples of reordered-and-refined sequences can be found in Appendix E.

5 Results and Analysis

5.1 Experimental Results on EnKo

Table 2 shows BLEU scores and Kendal's τs of wait-k models trained using original offline corpus and variations of reordered-and-refined corpus. We observe that the models trained on monotonically

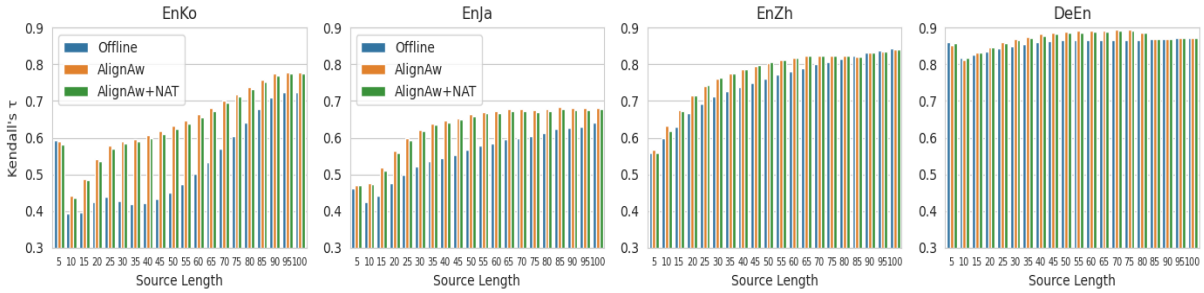

Figure 4: Monotonicity of offline pairs and pairs processed with reordering-and-refinement. AlignAw indicate that targets are alignment-aware reordering (Section 3.1.2, and AlignAw + NAT show monotonicity after NAT refinement (Section 3.2.1) is applied to AlignAw pairs.)

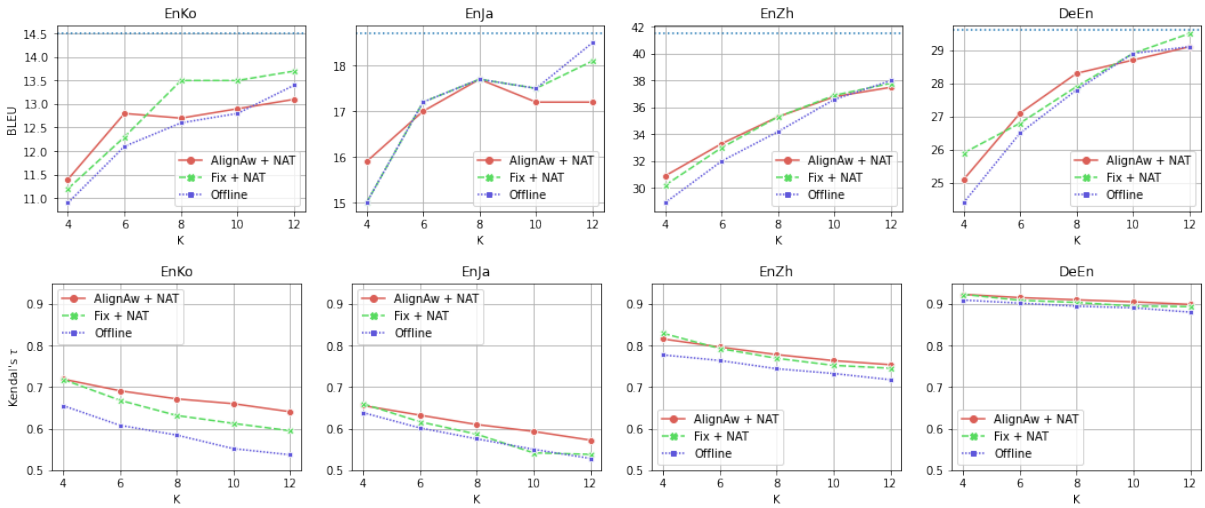

Figure 5: BLEU scores and monotonicity measurements presented by wait-k models trained on offline translation corpora and variations of reordered-and-refined corpora.

reordered-and-refined corpora show higher BLEU scores and monotonicity.

Reordering: Of the variations, corpora including AlignAw chunking process generally show better BLEU scores over Fixed + NAT when $k \leq 6$. This could be the benefit of the semantically plausible way to split sentences provided by AlignAw chunking. On the other hand, models trained with Fixed + NAT corpora show higher BLEU when $k \geq 8$.

Refinement: Experiments on utilizing AT probabilities show degraded BLEU scores in $k \in 6, 8, 10$. On the contrary, the models trained on AlignAw + NAT + AT corpora show enhanced monotonicity. The α value may be adjusted to make trade-off between promoting monotonicity in translation or enhancing translation quality in terms of BLEU.

Repetition Reduction with AT: Following (Welleck et al., 2020), we report n-gram repetition rate, seq-rep-n, on each generated corpus in Table 3. We observe from seq-rep-n in all of the tested n values, that employing AT models in refinement help alleviating the repetition problem of posed by NAT models.

5.2 Language Pairs Comparison

Figure 4 shows the difference in monotonicity between different language pairs: EnKo, EnJa, EnZh, and DeEn. It is observable in Figure 4 that the overall monotonicity in EnKo and EnJa pairs is enhanced after paraphrasing, while monotonicity scores of DeEn remain almost the same, only showing slight improvement. The extent of monotonicity enhancement in EnZh is between that of EnKo/Ja and DeEn. In all language pairs, the enhancements are generally lower in long or very short sequences. In the case of long sequence pairs, a pair may contain multiple sequences and be already aligned at the sequence level, thus resulting in marginal monotonicity enhancement. In the case of shorter length sequences, the whole sentence

may be merged into a single chunk, less benefiting from our process. After the reordered sets are refined, monotonicity marginally decreases. This is expected as forcibly aligned tokens are refined to augment the fluency in the resulting sentence.

To present the effectiveness of generated monotonic corpus in different language pairs, we train wait-k models on EnKo, EnJa, EnZh, and DeEn, and report BLEU and Kendal's τ of the models in Figure 5. The horizontal dotted line presents the BLEU of unidirectional offline model. The important observation we can find is that the monotonicity increment of wait-k model in Figure 5 is proportional to that of generated monotonic corpus in Figure 4, suggesting that promoting monotonicity in training corpus is beneficial for SimulMT models to generate monotonic output, especially in language pairs with differing word orders.

Within two paraphrasing methods, Fixed + NAT and AlignAw + NAT , we can see that the monotonicity of Fixed + NAT is always in between that of Offline and AlignAw + NAT in Figure 5, and the gap increases as the k value get higher.

While our methods are effective in EnKo, the performance of suggested method is similar or lower than that of baseline in EnJa. We presume that the ineffectiveness in EnJa is due to its short average sentence length with highest emission rate, as shown in Table 1. A short sentence often cannot preserve semantic properties while being split into chunks and reordered. For example, Fixed-length reordering always chunks all sentences ignores such feature and only increases disfluency in the chunked result. Also, even though AlignAw enforces the consistency requirement on the chunks, adjustment of such requirement like changing minimum chunk size may be required considering the high emission rate.

Based on the highest performance at $k = 8$, there is about 8% BLEU improvement over Offline in EnKo whereas there is about 3% improvement in EnZh and about 2% improvement in DeEn. It is roughly proportional to the monotonicity improvements shown in Figure 4.

5.3 Evaluation on Online References

We test wait-k models on our in-house EnKo online and offline testsets of 150 lines. We choose EnKo because the impact of reordering-and-refinement is the greatest in that pair. The source sentences of both testsets are identical. The online references

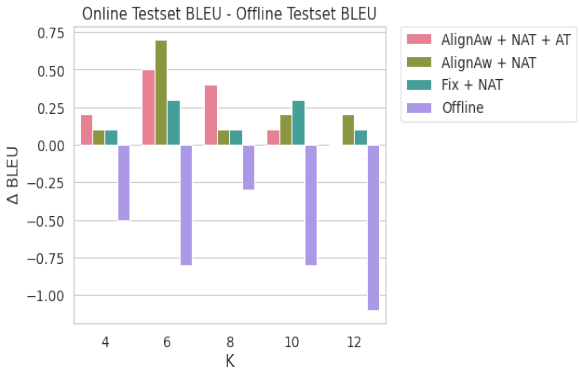

Figure 6: Differences of BLEU score between online and offline of EnKo wait-k models.

are constructed by a professional interpreter under a simulated simultaneous interpretation scenario and the offline references are constructed by the same interpreter assuming a typical translation scenario. In construction of online references interpreter was encouraged to perform monotonic interpretation rather than anticipation. BLEU scores are computed with both online and offline references for each trained model. Figure 6 plot the subtraction of BLEU scores on offline references from BLEU scores on online references. It is noticeable that the wait-k models trained on offline corpus have negative value while all the models trained on generated corpus present positive values, which implies the effectiveness of our approach. Overall, the substantial differences at $k = 6$ may suggest that the chunk size utilized by human interpreter has comparable value.

6 Related Work

Due to word order differences between languages, SimulMT training often face situations where anticipation is required. Note that word order difference is observed to be problematic even for human interpreters (Al-Rubai'i, 2004; Tohyama and Matsubara, 2006). Chen et al. (2020) suggest using pseudo-references which involve utilizing wait-k inference output to limit "future anticipation" in training. Zhang et al. (2020b) utilize NMT teachers to implicitly embed future information in their SimulMT students for better anticipation performance. Zhang et al. (2020a) study adaptive policy to tackle this problem - authors suggest an adaptive SimulMT policy that dictate READ/WRITE actions based on whether "meaningful units" are fully formed with consumed input tokens.

Related work in the broader SimulMT and para-

phrasing domain is presented in Appendix F.

7 Conclusion

Most of SimulMT models are trained on offline translation corpora, which could lead to limitation in translation quality and achievable latency, especially in non-monotonic language pairs. To address this problem, we propose a reordering-and-refinement algorithm to generate monotonically aligned online target with NAT model. We then train widely used wait-k SimulMT models on this newly generated corpus. Resulting models show BLEU score improvement and significant enhancement on monotonicity in multiple language pairs.

References

Alya Al-Rubai'i. 2004. The effect of word order differences on english-into-arabic simultaneous interpreters' performance. *Babel*, 50:246–266.

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027, Brussels, Belgium. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. Association for Computational Linguistics.

Magdalena Bartłomiejczyk. 2008. *Anticipation: A controversial interpreting strategy*, pages 117–126.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Ozan Caglayan, Julia Ive, Veneta Haralampieva, Pranava Madhyastha, Loïc Barrault, and Lucia Specia. 2020. Simultaneous machine translation with visual context. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2350–2361, Online. Association for Computational Linguistics.

Zhongxi Cai, Koichiro Ryu, and Shigeki Matsubara. 2020. What affects the word order of target language in simultaneous interpretation. In *2020 International Conference on Asian Language Processing (IALP)*, pages 135–140.

M. Cettolo, Marcello Federico, L. Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017. Overview of the iwslt 2017 evaluation campaign.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Rakesh Chada. 2020. Simultaneous paraphrasing and translation by fine-tuning transformer models. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 198–203, Online. Association for Computational Linguistics.

Junkun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2020. Improving simultaneous translation with pseudo references. *arXiv preprint arXiv:2010.11247*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, Online. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

Hou Jeung Han, Mohd Abbas Zaidi, Sathish Reddy Indurthi, Nikhil Kumar Lakumarapu, Beomseok Lee, and Sangha Kim. 2020a. End-to-end simultaneous translation system for IWSLT2020 using modality agnostic meta-learning. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 62–68, Online. Association for Computational Linguistics.

Hyojung Han, Sathish Indurthi, Mohd Abbas Zaidi, Nikhil Kumar Lakumarapu, Beomseok Lee, Sangha Kim, Chanwoo Kim, and Inchul Hwang. 2020b. Faster re-translation using non-autoregressive model for simultaneous neural machine translation. *arXiv preprint arXiv:2012.14681*.

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)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Huda Khayrallah, Brian Thompson, Matt Post, and Philipp Koehn. 2020. Simulated multiple reference training improves low-resource machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 82–89, Online. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Machine Translation: From Real Users to Research*, pages 115–124, Berlin, Heidelberg. Springer Berlin Heidelberg.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, page 177–180, USA. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In *International Conference on Learning Representations*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain. Association for Computational Linguistics.

Paul Michel and Graham Neubig. 2018. MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium. Association for Computational Linguistics.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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 *Proceedings of NAACL-HLT 2019: Demonstrations*.

M. Polinsky. 2012. *Headedness, again*, pages 348–359. UCLA Department of Linguistics, Los Angeles.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

R. Pryzant, Y. Chung, D. Jurafsky, and D. Britz. 2018. JESC: Japanese-English Subtitle Corpus. *Language Resources and Evaluation Conference (LREC)*.

Hitomi Tohyama and Shigeki Matsubara. 2006. Collection of simultaneous interpreting patterns by using bilingual spoken monologue corpus. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 274–285, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2020. End-to-end neural word alignment outperforms GIZA++. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1605–1617, Online. Association for Computational Linguistics.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence*, pages 18–32, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020a. Learning adaptive segmentation policy for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2280–2289, Online. Association for Computational Linguistics.

Shaolei Zhang, Yang Feng, and Liangyou Li. 2020b. Future-guided incremental transformer for simultaneous translation. *arXiv preprint arXiv:2012.12465*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020c. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020a. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous 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)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.

Renjie Zheng, Mingbo Ma, Baigong Zheng, Kaibo Liu, and Liang Huang. 2020b. Opportunistic decoding with timely correction for simultaneous translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 437–442, Online. Association for Computational Linguistics.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1.0.

A Dataset Details

All utilized texts regarding English-to-Korean and German-to-English directions are first tokenized with Moses (Koehn et al., 2007), then per-language BPE vocabularies are learned on the moses-tokenized trainset. The sizes of the vocabularies are 29k BPE English vocabulary and 44k BPE Korean vocabulary for English-to-Korean and 16k BPE German vocabulary and 16k BPE English vocabulary for German-to-English. The English-to-Japanese texts are first moses-tokenized. And KyTea[5] is applied to additionally tokenize Japanese texts. Separate English and Japanese vocabulary of size 32K is trained on tokenized training data using Sentencepiece (Kudo and Richardson, 2018). For English-to-Chinese training, no moses-tokenization is applied and Chinese sentences are tokenized using Jieba[6]. Separate English and Chinese vocabulary of size 32K is trained on training pairs using sentencepiece. The in-house EnKo data consists mainly of the AIHub EnKo offline translation corpus[7], news domain translation data, in-house proprietary patent data, and translated dialogue data of general domain.

B Additional Implementation Details

Our implementation is based on fairseq (Ott et al., 2019), and all GPUs used are V100s. The alignment information used in reordering process is extracted with GIZA++ (Och and Ney, 2003). The alignment information used to evaluate monotonicity is extracted using fast-align (Dyer et al., 2013). The alignments are measured in subword level. Embedding weights are separately learned for source and target languages, while transposed target language embedding weights also works as linear projection layers at the top of transformer decoders.

C Reporting AlignAw + NAT Scores

The AlignAw + NAT wait-k models are trained on different variations of AlignAw + NAT corpora - AlignAw + NAT corpora generated with prefixes fixed (**b0**), and with alterable prefixes (**b1**), and combination of **b0** and **b1** filtered using BertScore (**b0b1**). The reported AlignAw + NAT testset BLEU scores are of the wait-k models that show highest BLEU score on validset regardless of the dataset variations.

[5] http://www.phontron.com/kytea
[6] https://github.com/fxsjy/jieba
[7] https://aihub.or.kr

EnKo Wait-k	k=4	k=6	k=8	k=10
Offline	10.9/18	12.1/12	12.6/7	12.8/4
Pseudo Refs	11.2/19	11.6/13	12.2/9	13.3/5
Fixed + NAT	11.2/12	12.3/8	13.5/5	13.7/3
AlignAw + NAT	11.4/9	12.8/6	12.7/3	12.9/2

Table 4: BLEU scores/k-AR% of EnKo wait-k models.

D Comparison with Test Time wait-k Refs

In recent work, Chen et al. (2020) propose a method of pseudo-references generated with test time wait-k decoding. We apply their method in EnKo to create pseudo references for $k \in \{4, 6, 8, 10\}$ and train wait-k model. The results are presented in Table 4. Similar to our monotonicity metric, this work also suggest k-anticipation rate (k-AR) as a metric of parallel corpora. We also measure and report our generated corpus with this metric. Compare to Offline and Pseudo Refs , we see that our AlignAw + NAT corpus significantly decrease k-AR and the models trained with AlignAw + NAT also show enhanced BLEU score in general.

E Examples of Paraphrased Targets

Figure 7 presents an example sentence of English to Korean in whole pipeline process. We first represent the source sentence and its two different target sentences, online and offline translation. As results of the reordering phase, for each method (i.e., fixed chunking and AlignAw chunking), we provide only one case: $k = 8$ in fixed chunking and $\delta_{src} = 2$ and $\delta_{tgt} = 2$ in AlignAw chunking. Figure 8 shows the final grid of AlignAw chunking in this example. We conduct the MOS evaluation with the result of refinement phase. MOS is the average of human-evaluated score by professional interpreters. In this evaluation, AlignAW + NAT shows the best performance than others. Moreover, we present the inference outputs of SimulMT models which are trained on generated monotonic corpus. In this case, results of our methods are better than the result of offline model. We also provide DeEn example in Figure 9.

F More Related Work

Simultaneous Translation: A fixed policy is used in (Dalvi et al., 2018) and (Ma et al., 2019) which train SimulMT models according to the predefined policy. In particular, the Wait-k strategy

proposed by (Ma et al., 2019) waits for k sub-words and alternates READ/WRITE based on the emission rate. Due to the deterministic feature of this schedule, the model can be easily implemented and trained. On the downside, anticipation from missing contents often fails to predict correct target tokens, and a fixed schedule could impede the model from speeding up or slowing down flexibly for source inputs. There are several works of SimulMT with many variants of the Wait-k approach. For example, (Caglayan et al., 2020) explores whether additional visual context can complement missing source information. Furthermore, in (Zheng et al., 2020b), the opportunistic decoding technique is introduced which allows partial (certain length of suffix) corrections in a timely fashion. Finally, (Zheng et al., 2020a) extended the wait-k to an adaptive one by composing a set of fixed policies heuristically.

Upon the proposal by (Cho and Esipova, 2016), various adaptive policies have been suggested by several works including (Gu et al., 2017; Zheng et al., 2019a,b; Arivazhagan et al., 2019; Ma et al., 2020). SimulMT proposed by (Cho and Esipova, 2016) use greedy decoding with heuristic waiting criteria to decide whether the model should read or emit, while (Gu et al., 2017) utilize a pre-trained model with a reinforcement learning agent that maximizes quality and minimizes latency. Advancing this work, (Alinejad et al., 2018) proposes to add a new action PREDICT that anticipate future source words. Recently, (Arivazhagan et al., 2019) use hard attention to schedule the policy and introduced new differentiable average lagging metrics which can be integrated into training losses, and (Ma et al., 2020) incorporate this work into the multi-headed Transformer model. Furthermore, (Zhang et al., 2020a) proposes an adaptive policy which learns to segment source input considering possible target output. Other researches including (Zheng et al., 2019a) use separately trained oracles in the supervision of extracted action sequence.

Paraphrase: Translation is one of more common approaches for paraphrase generation. Mallinson et al. (2017) explore pivoting (translating a source sequence to a pivot language, then to a target language) to generate paraphrases and assess correlation between original and paraphrased sentences. Back-translation has also been explored for paraphrase generation (Wieting et al., 2017; Iyyer et al., 2018). Other techniques, such as translating with oversampling strategy have also bee studied (Chada, 2020).

On the other hand, various NMT research employ paraphrased data to overcome data limitation. Edunov et al. (2018) show that source-paraphrased corpus generated with back-translation can significantly improve BLEU scores in NMT tasks. Similarly, Khayrallah et al. (2020) directly implements paraphrasers in NMT training to improve translation quality.

Src-Tgt Pair			
Source		So *I* want to talk about that magic that happens when we actually represent something .	
Online		그래서 *제가* 얘기하고_싶은 것은 마@@ 법에 대한 이야기@@ 에요 . 우리가 실제로 어떤 것을 표현할 때 일어나는 그 마@@ 법 말입니다 .	
Offline		그래서 *나는* 우리가 실제로 어떤 것을 표현할 때 일어나는 마@@ 법에 대해 이야기@@_하려고 한다 .	
Reordering			
Fix	k=6	So *I* want to talk about ‖ that magic that happens when we ‖ actually represent something .	
	k=6	그래서 *나는* 대해_이야기@@ 하려고 ||| 우리가 것을 때 일어나는 마@@ ||| 실제로 것을 표현할 .	
AlignAw	src min 2	So *I* ‖ want to talk about ‖ that magic that happens when we actually represent something .	
	tgt min 2	그래서 *나는* ||| 대해 이야기@@_하려고 한다 . ||| 우리가 실제로 어떤 것을 표현할 때 일어나는 마@@ 법에	
Refinement			MOS(5/5)
Fix + NAT (K=6)	fixed prefix	그래서 *나는* 이 것에 대해 이야기@@ 하고자 할 때 일어나는 마@@ 법에 대해 말하고 싶습니다 . 우리가	2
	alterable	그래서 *나는* 그 마@@ 법에 대해 ~~그 마@@ 법에 대해~~ 이야기@@ 하고자 합니다 . 우리가 실제로 무언가를 나타@@ 낼 때 말이죠 .	2.75
AlignAw + NAT	fixed prefix	그래서 *나는* 이것에 대해 이야기@@ 하려고 한다_. 우리가 실제로 어떤 것을 표현할 때 일어나는 그 마@@ 법에 대해 말입니다	3.5
	alterable	그래서 *나는* 이것에 대해 이야기@@ 하려고 한다 . 우리가 실제로 어떤 것을 표현할 때 일어나는 마@@ 법에 대해 말입니다	4
SimulMT Wait-6			
Offline		그래서 *나는* ~~아 일아~~ ~~일어@@ 나면~~ 실제로 우리가 무언@@ 가를 대표@@ 할 때 일어나는 마@@ 법에 대해 이야기@@ 하고 싶습니다 .	2.5
Fix + NAT		그래서 *나는* 그 마@@ 법에 대해 이야기@@ 하고 싶습니다 . 우리가 실제로 무언@@ 가를 대표@@ 할 때 일어나는 마@@ 법에 대해 말하고 싶습니다 .	3.5
AlignAw + NAT		그래서 *저는* 그 마@@ 법에 대해 이야기@@ 하고 싶습니다 . 그래서 저는 실제로 어떤 것을 대변@@ 할 때 일어나는 것입니다 .	4
AlignAw + NAT + AT		그래서 *저는* 그 마@@ 법에 대해 이야기@@ 하고 싶습니다 . 우리가 실제로 무언@@ 가를 나타@@ 낼 때 일어나는 마@@ 법에 대해 말하고 싶습니다 .	4.5

Figure 7: Example sentence of EnKo in whole pipeline process from inputs to SimulMT results.

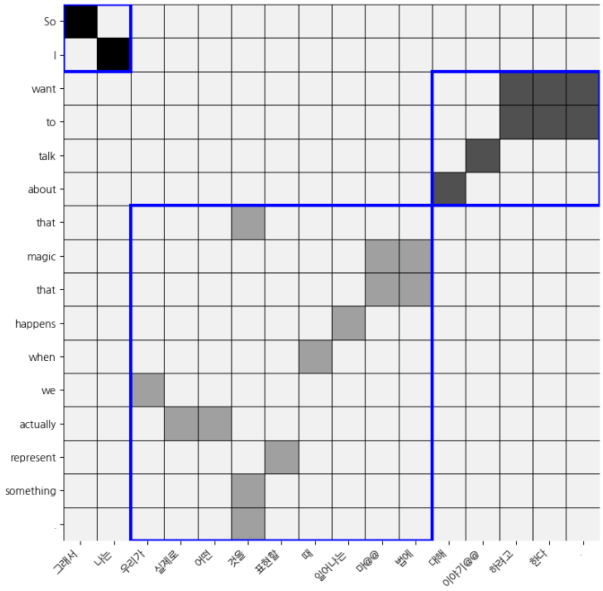

Figure 8: Alignment-aware result of the EnKo pipeline example in Figure 7

Src-Tgt Pair		
Source	Wie bei den meisten Kreb@@ ser@@ krank@@ ungen ist der genaue Auslö@@ ser meist schwer zu bestimmen .	
Offline	With most can@@ c@@ ers , it is hard to know the exact cause .	
SimulMT Wait-6		MOS(5/5)
Offline	As with most can@@ c@@ ers , the exact ~~extent of the action~~ is usually difficult to determine .	2
Fix + NAT	As with most can@@ c@@ ers , the exact ~~exact~~ trigger for the disease is usually difficult to determine .	4
AlignAw + NAT	As with most can@@ c@@ ers , the exact ~~spread~~ of the disease is usually difficult to determine .	3

Figure 9: Example outputs of DeEn wait-k models trained on reordered-and-refined corpora