Document-level Translation with LLM Reranking: Team-J at WMT 2024 General Translation Task

*Keito Kudo[1,2], *Hiroyuki Deguchi[3], *Makoto Morishita[4,1], *Ryo Fujii[4], *Takumi Ito[1,5], *Shintaro Ozaki[3], *Koki Natsumi[3], Kai Sato[1], Kazuki Yano[1], Ryosuke Takahashi[1], Subaru Kimura[1], Tomomasa Hara[1], Yusuke Sakai[3], Jun Suzuki[1,2]
[1]Tohoku University [2]RIKEN [3]NAIST [4]Future Corporation [5]Langsmith Inc.

Abstract

We participated in the constrained track for English-Japanese and Japanese-Chinese translations at the WMT 2024 General Machine Translation Task. Our approach was to generate a large number of sentence-level translation candidates and select the most probable translation using minimum Bayes risk (MBR) decoding and document-level large language model (LLM) re-ranking. We first generated hundreds of translation candidates from multiple translation models and retained the top 30 candidates using MBR decoding. In addition, we continually pre-trained LLMs on the target language corpora to leverage document-level information. We utilized LLMs to select the most probable sentence sequentially in context from the beginning of the document.

1 Introduction

This paper details Team-J's system submission for the WMT 2024 Shared Task: General Machine Translation. We participated in the English-Japanese (En→Ja) and Japanese-Chinese (Ja→Zh) translation tasks under the constrained track.

As with last year's competition, the use of publicly available pre-trained models and metrics evaluated in the WMT Metrics shared tasks, such as COMET (Rei et al., 2020), was permitted. Following the Kudo et al.'s (2023) system, we employed multiple machine translation (MT) models to generate numerous candidate sentences for each source text. We then applied minimum Bayes risk (MBR) decoding (Fernandes et al., 2022) using the COMET metric to select the optimal translations.

Additionally, contrary to the previous years, the use of large language models (LLMs) was also permitted this year. Our primary objective was to use these LLMs to achieve consistent document-level machine translation. Specifically, we aimed to develop models based on LLMs and also implemented a reranking system. Figure 1 provides an overview of our system. The following sections describe its components in detail.

*: Equal contributions.

2 Dataset Construction

In this section, we describe the training data, the process of synthetic data generation, and the data cleaning methodologies.

2.1 Provided Data

Since we participated in the constrained track, we solely used the data officially provided by the organizer.

Bitext data. We used all the provided bitext data. For English to Japanese translation, we used JParaCrawl v3.0 (Morishita et al., 2022a), News Commentary v18, Wiki Titles v3, WikiMatrix (Schwenk et al., 2021), Japanese-English Subtitle Corpus (JESC) (Pryzant et al., 2018), The Kyoto Free Translation Task (KFTT) Corpus (Neubig, 2011), and TED Talks (Cettolo et al., 2012). For Japanese to Chinese translation, we used JParaCrawl Chinese (Nagata et al., 2024), News Commentary v18, Linguatools Wiki Titles, WikiMatrix, OPUS, and Neulab TED Talks (Tiedemann, 2012).

Monolingual data. We also used the following provided monolingual data for Japanese and Chinese: News Crawl, News Commentary, Leipzig Corpora (Goldhahn et al., 2012), Common Crawl (Buck et al., 2014), and Extended Common Crawl (Conneau et al., 2020; Wenzek et al., 2020). For the continual pre-training of the language models, we only used the Common Crawl and Extended Common Crawl due to the limited availability of document-level data beyond these two datasets.

Development data. We used NTREX-128 (Federmann et al., 2022), Flores-200 (Team et al., 2022;

Proceedings of the Ninth Conference on Machine Translation, pages 210–226
November 15-16, 2024

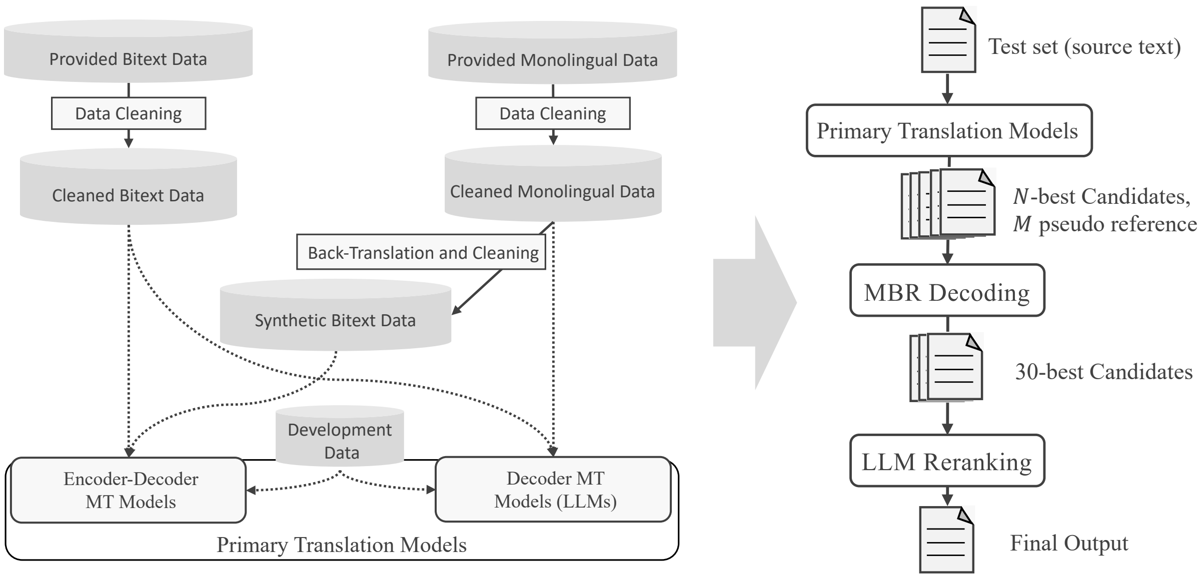

Figure 1: System overview

Goyal et al., 2022; Guzmán et al., 2019) and the past WMT test sets as development data. These datasets were also employed to fine-tune the models.

2.2 Synthetic Data

We constructed synthetic data to augment the training dataset. We used the synthetic data created by Kudo et al. (2023) for the En→Ja task, and newly created data for the Ja→Zh task as follows. For preprocessing, we tokenized the bitext (Section 2.1) into truecased[1] subwords using a unigram language model with Sentencepiece (Kudo and Richardson, 2018), with "byte_fallback", and "split_digits" options enabled following Touvron et al. (2023); Dubey et al. (2024); Kudo et al. (2023). After that, we created a back translation model (Sennrich et al., 2016), which we call an initial translation model using the training configurations in Table 7 (Appendix C) and trained it on the bitext. Then, we translated the Chinese monolingual data (Section 2.1) with a beam size of 10 and a length penalty of 1.0.

2.3 Data Cleaning

We conducted data cleaning on the corpus. Specifically, we applied several rules to clean and filter out noisy sequences using HojiChar (Shinzato, 2023). HojiChar is a text preprocessing tool that mainly supports monolingual corpus in Japanese and English, with typical filters preinstalled. We first extended HojiChar to make it work with parallel corpus and implemented a variety of rules with careful investigation of the provided data. Table 1 shows the list of data cleaning methods we applied on the bitext and monolingual data. Table 2 shows the amount of data after filtering.

The following provides a detailed explanation of the cleaning rules that were mainly implemented using tools other than HojiChar.

Character count-based filtering. We qualitatively examined the Common Crawl and Extended Common Crawl datasets. Our analysis revealed that shorter sequences tend to be noisy. Therefore, we discarded sequences that were less than or equal to 200 characters for Japanese and 100 characters for Chinese, respectively (see `(26)` in Table 1). This threshold also helps us retain document-level data that is suitable for the continual pre-training of LLMs. To efficiently filter out shorter sequences, we used the `awk` command.

Toxic content cleaning. Qualitative analysis of the Common Crawl data revealed a significant amount of low-quality toxic contents, such as adult material, are included in the corpus. To address this, we applied a toxic content filter to exclude such samples from our training data (`(9)` in Table 1). For the Japanese data, we used filters originally implemented in HojiChar.[2] For the Chinese corpus, we defined a list of toxic words based on

[1] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/recaser/truecase.perl`

[2] `DiscardAdultContentJa` in HojiChar.

Filter & Cleaner	Ja	Zh	En-Ja	Ja-Zh
(1) Discard content having identical source and target			✓	✓
(2) Discard content with invalid unicode characters	✓	✓	✓	✓
(3) Remove non-printable unicode characters	✓	✓	✓	✓
(4) Apply NFKC normalization	✓	✓	✓	✓
(5) Normalize space-like characters to half-width spaces	✓	✓	✓	✓
(6) Restore escaped HTML symbols	✓	✓	✓	✓
(7) Discard content like progress bars	✓	✓	✓	✓
(8) Discard content having many square brackets	✓	✓	✓	✓
(9) Discard content containing keywords for porn contents	✓	✓		
(10) Discard content containing keywords for online bulletin boards	✓	✓		
(11) Discard content containing part of sequences like word lists	✓	✓	✓	✓
(12) Discard content containing having many punctuations	✓	✓	✓	✓
(13) Discard content containing having many numbers	✓	✓	✓	✓
(14) Reduce repeated space and punctuation characters	✓	✓	✓	✓
(15) Discard content having many same consecutive characters	✓	✓	✓	✓
(16) Discard content having many same consecutive N-grams	✓	✓	✓	✓
(17) Discard content having less punctuations	✓	✓		
(18) Discard content having no punctuations in a sliding window of specified length	✓	✓		
(19) Discard content having low compression ratio with zlib compression	✓	✓		
(20) Discard content not in expected languages	✓	✓	✓	✓
(21) Remove ellipsis symbols	✓	✓	✓	✓
(22) Remove open bracket end symbols at the end of the sentence	✓	✓	✓	✓
(23) Remove parentheses with no content inside	✓	✓	✓	✓
(24) Remove Unicode control characters	✓	✓	✓	✓
(25) Remove content starts with "&"	✓	✓	✓	✓
(26) Discard too short content	✓	✓		
(27) Convert traditional Chinese to simplified Chinese				✓
(28) Exact deduplication	✓	✓	✓	✓
(29) Fuzzy deduplication	✓	✓		
(30) Discard too long content			✓	✓
(31) Discard content having too large source/target token ratio			✓	✓
(32) Discard content having too large token/char ratio			✓	✓
(33) Discard semantically irrelevant translations			✓	✓

Table 1: List of data cleaning rules.

those used for the ChineseWebText (Chen et al., 2023) dataset.

Compression rate-based cleaning. We used a cleaning method based on the compression rate to remove non-textual data ((19) in Table 1).[3] Samples with a high compression rate typically contain excessive repetitions, while those with a low compression rate often consist of random strings. Specifically, we calculated the compression rate for each sample and removed those that did not fall within a specified range.

Language detection. To ensure the collection of data in the target language, we used language detection (20) in Table 1. Simple heuristic language detection methods are implemented in Hojichar, such as a method that checks for the presence of *hiragana* or *katakana*. Alongside these simple methods, we also used FastText-based language detection (Joulin et al., 2017b,a).

[3]We referred to `has_good_compression_ratio` in https://github.com/llm-jp/llm-jp-corpus/blob/main/scripts/filters.py

Conversion of traditional Chinese to simplified Chinese. We converted Chinese data written in traditional characters to simplified characters to augment the bitext data ((27) in Table 1). We used `OpenCC`[4] for these conversions.

Deduplication. Duplicate data in training sets can negatively impact the performance of language models (Lee et al., 2022). To mitigate this, we performed exact deduplication using the `sort` command ((28) in Table 1) and fuzzy deduplication using MinHash (Broder, 1997) ((29) in Table 1). We used the `text-dedup` tool (Mou et al., 2023) for implementation.

Bitext similarity cleaning. We performed cleaning based on bitext similarity using LaBSE (Feng et al., 2022) to filter out semantically irrelevant pairs ((33) in Table 1). We set the lenient threshold of 0.5 for bitext and more strict threshold of 0.7 to synthetic data.

[4]https://github.com/BYVoid/OpenCC

	# samples	# tokens
LLMs		
Monolingual Ja	88.4M	35.8B
Monolingual Zh	137.4M	29.9B
Parallel En-Ja	29.8M	4.0B
Parallel Ja-Zh	3.8M	506.3M
Encoder-Decoder		
Synthetic En-Ja	587M	12.9B
Synthetic Ja-Zh	291M	10.3B
Parallel En-Ja	28.2M	730.0M
Parallel Ja-Zh	6.3M	163.6M

Table 2: The amount of training data used for LLMs and Encoder-Decoder MT models. The token count for LLMs is based on the tokenizer of Mistral-7B, and the count for Encoder-Decoder MT models is based on the subwords on the target side.

3 Primary Translation Models

We developed translation models using two architectures: Encoder-Decoder and Decoder-only (LLMs).

3.1 Encoder-Decoder MT Models

For En→Ja, we used the existing translation models created by Morishita et al. (2022b); Kudo et al. (2023). For Ja→Zh, we newly constructed translation models through pre-training and fine-tuning.

Pre-training. We trained the pre-training model using the pre-training configuration in Table 7 (Appendix C). For the training data, we used the bitext (Section 2.1) and the synthetic data (Section 2.2) after applying data cleaning (Section 2.3). We performed upsampling to achieve a $1 : 4.7$ ratio between the bitext and the synthetic data. Moreover, we applied the tagged back-translation technique (Caswell et al., 2019), adding a special token <BT> at the beginning of the source sentences in the synthetic data and storing this tag in the vocabulary dictionary.

Fine-tuning. After pre-training, we conducted fine-tuning using the development data (Section 2.1) with the fine-tuning configuration in Table 7 (Appendix C).

3.2 LLM-based MT Models

We used the Llama2-13B (Touvron et al., 2023) and Mistral-7B (Jiang et al., 2023), which are permitted for use in the constrained track. These LLMs were used only for the En→Ja direction and not for the Ja→Zh direction. For Mistral-7B, we also prepared a variant with an expanded vocabulary to improve its Japanese generation capability. For more details on vocabulary expansion, please refer to Section B.

Continual pre-training. Although the datasets used for training Llama2 and Mistral are not publicly disclosed, it is generally believed that they are predominantly in English. Consequently, continual pre-training has been conducted to enhance performance on Japanese tasks (Fujii et al., 2024a; Okazaki et al., 2024). This approach has been reported to improve English-Japanese translation performance. To further boost Japanese language capability, we also performed continual pre-training using the cleaned monolingual corpus detailed in Section 2.3. The training configurations are shown in Table 8, 9, and 10.

Supervised fine-tuning After continual pre-training, we conducted supervised fine-tuning for the translation task. In this phase, we used the cleaned bitext corpus and development data described in Section 2. Initially, we fine-tuned the model using the bitext corpus, followed by additional fine-tuning with the development data which is relatively clean. To prepare for the Stepwise MBR-Enhanced LLM decoding detailed in Section 4.2, we used all combinations of the first n sentences from each document as training samples for the development data fine-tuning. Figure 2 shows the prompt template, and Table 8, 9, and 10 shows hyperparameters used in the training process.

Preference learning. To align the translation results with human preferences, we conducted preference learning for Mistral-7B. [5] We used Contrastive Preference Optimization (CPO) (Xu et al., 2024) as the preference learning algorithm. In preliminary experiments, we also tried Direct Preference Optimization (DPO) (Rafailov et al., 2023) as an alternative to CPO. However, despite the decrease in loss during training, we observed that the DPO often resulted in output collapse (complete loss of input-output correspondence) during decoding. Therefore, we selected CPO as our preference learning.

Let $L_{\mathrm{NLL}}(\pi_\theta)$ and $L_{\mathrm{pref}}(\pi_\theta)$ be the negative log-likelihood of π_θ and preference of output given by

[5] Due to computational resource limitations, we applied LoRA fine-tuning (Hu et al., 2022).

次の英語を日本人のネイティブのように日本語に翻訳してください。 原文: {src} 訳文: {tgt}

Figure 2: The general prompt for supervised fine-tuning. {src} denotes the source sentence. {tgt} denotes the target sentence.

π_θ, respectively, that is:

$$\begin{aligned} L_{\mathrm{NLL}}(\pi_\theta) &= -\mathbb{E}_{(s,r)\sim\mathcal{D}}\left[\log \pi_\theta(r \mid s)\right] \\ L_{\mathrm{pref}}(\pi_\theta) &= -\mathbb{E}_{(s,r,y_r)\sim\mathcal{D}}\left[\log \sigma(\beta d)\right] \quad , \\ d &= \log \pi_\theta(r \mid s) - \log \pi_\theta(\hat{y} \mid s) \end{aligned} \tag{1}$$

where σ is the Sigmoid function. Then, CPO minimizes the following objective function during training:

$$\min_\theta \left[L_{\mathrm{pref}}(\pi_\theta) + \alpha L_{\mathrm{NLL}}(\pi_\theta)\right] \quad . \tag{2}$$

Here, $\mathcal{D} = \left\{\left(s^{(i)}, r^{(i)}, \hat{y}^{(i)}\right)\right\}_{i=1}^N$ represents the dataset. π_θ denotes a parameterized policy, and α and β are hyperparameters. We used the development data for training in preference learning. In this context, s corresponds to the source text from the development data, r to the reference text from the development data, and $\hat{y}$ to the output of the model before preference learning.

To prevent the model output from collapsing, we introduced a minor modification to the CPO objective function. Specifically, we implemented a warm-up phase to reduce the impact of the preference learning loss at the beginning of training. This approach is formulated as follows:

$$\min_\theta \left[\min\left(1, \frac{i}{i_{\mathrm{w}}}\right) L_{\mathrm{pref}}(\pi_\theta) + \alpha L_{\mathrm{NLL}}(\pi_\theta)\right] . \tag{3}$$

Here, i represents the number of training steps, and i_{w} denotes the number of warm-up steps for the preference learning loss.

4 Decoding

This year's test set consists of segments with multiple sentences in context. Since most bitext corpora are at the sentence level, translating larger segments in one shot is not preferable. Thus, we initially divided each segment in the test set into individual sentences using spaCy (Honnibal et al., 2020).[6] In case the resulting split was overly short, we combined texts from its adjacent splits.

[6] We used "en_core_web_lg" model for English and "ja_core_news_lg" model for Japanese.

	hypotheses	pseudo-references	
		top-p sampling	epsilon sampling
En→Ja	1272.15	3288.5	3421.99
Ja→Zh	261.84	884.11	3108

Table 3: The average number of hypotheses and pseudo references for each source sentence generated by the Encoder-Decoder MT models. Note that due to errors during decoding, the number of hypotheses and pseudo-references generated for a single source sentence varies.

4.1 MBR Decoding

We apply minimum Bayes risk (MBR) decoding (Eikema and Aziz, 2020) to select high-quality translations from the set of hypotheses generated by the multiple translation models using MBRS (Deguchi et al., 2024). Let $\mathcal{Y}$ be the output space of translation models. We use the Monte Carlo method to estimate the expected utility (Eikema and Aziz, 2022), as follows:

$$\begin{aligned} y^{\mathrm{MBR}} &= \operatorname*{argmax}_{h\in H} \mathop{\mathbb{E}}_{\hat{r}\in\hat{R}}\left[u(h,\hat{r})\right], \\ &= \operatorname*{argmax}_{h\in H} \frac{1}{|\hat{R}|}\sum_{\hat{r}\in\hat{R}} u(h,\hat{r}), \end{aligned} \tag{4}$$

where y^{MBR} is the selected translation by MBR decoding, $H \subseteq \mathcal{Y}$ is the hypotheses set, and $\hat{R}$ is the multiset (a.k.a bag) of translation samples[7], called "pseudo-references". $u\colon \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ is the utility function that returns scores of the translation quality of the hypothesis under the given pseudo-references, which is formally defined as $h \succeq h' \iff u(h,\hat{r}) \geq u(h',\hat{r})$ where $\succeq$ denotes the preference relation. We employ COMET-22[8] (Rei et al., 2020, 2022) for the utility function u. Therefore, the MBR decoding using COMET-22 is formulated as follows:

$$y^{\mathrm{MBR}} = \operatorname*{argmax}_{h\in\mathcal{H}} \frac{1}{|\hat{R}|}\sum_{\hat{r}\in\hat{R}} \texttt{COMET-22}(s,h,\hat{r}). \tag{5}$$

Note that COMET-22 also takes the source sentence s as input.

[7] The support set is a subset of the output space, i.e., $\mathrm{Supp}(\hat{R}) \subseteq \mathcal{Y}$

[8] https://huggingface.co/Unbabel/wmt22-comet-da

In our system, we select the 30-best translations using MBR decoding instead of selecting the 1-best translation as shown in Equation 4 to determine the final decision using another algorithm than MBR decoding. In other words, MBR decoding is used to prune translation hypotheses. We generate hypotheses for each source sentence using an ensemble of Encoder-Decoder MT models with beam search decoding. In addition, we prepare two types of pseudo-references by decoding with top-p sampling ($p = 0.9$) and epsilon sampling (Freitag et al., 2023) ($\epsilon = 0.02$). The number of hypotheses and pseudo-references used in MBR decoding is presented in Table 3.

4.2 Stepwise MBR-Enhanced LLM Decoding

Algorithm 1: Stepwise MBR-Enhanced LLM decoding

Input: $D_{\text{src}} = \{s_0, s_1, \ldots, s_n\}$
Output: $D_{\text{hyp}} = \{h_0, h_1, \ldots, h_n\}$
1 $D_{tgt} \leftarrow \{\}$;
2 $S_{hist} \leftarrow \{\}$;
3 **for** $i \leftarrow 0$ **to** n **do**
    `// Generate candidates for` s_i
4     $H \leftarrow \text{LLMs}_{\text{MT}}(s_i, S_{hist}, D_{tgt})$;
5     $h_i \leftarrow \text{MBR}(H, H)$;
6     $D_{tgt} \leftarrow D_{\text{hyp}} \cup \{h_i\}$;
7     $S_{hist} \leftarrow S_{\text{hist}} \cup \{s_i\}$;
8 **return** D_{hyp}

During our preliminary experiments with fine-tuned LLMs, we observed frequent issues where some sentences were skipped during decoding. This led to discrepancies in the number of sentences between the source and the translated output. Additionally, we observed samples where the same token was generated repeatedly. To address these issues, we propose a decoding method called `Stepwise MBR-Enhanced LLM Decoding` (Algorithm 4.2). This method translates documents sentence by sentence, considering the overall document context (see Figure 3). This approach resolves the issue of mismatched sentence counts between the source and hypothesis. Furthermore, we applied MBR decoding to achieve high-quality sentence-level translation without repeated tokens or other errors (line 5 of Algorithm 4.2). We used the outputs of four LLMs for this method. Specifically, we used four LLMs with different settings: Mistral-7B with and without vocab expansion and with and without preference learning.

5 LLM Reranking

As mentioned in Section 3 and Section 4, primary translation models decode at the sentence level. To improve the overall document-level consistency of the translation results, we performed reranking using LLMs. We used the top 30 highest-scoring hypotheses from MBR decoding as the candidate pool and reranked them based on context-aware scoring. Specifically, we used the LLMs fine-tuned for the translation task described in Section 3.2 to calculate the likelihood of each hypothesis with context information. We repeated this process to select hypotheses with the highest likelihood scores, resulting in the final translation output. The details are described in Algorithm 2. In our system, we use supervised fine-tuned Mistral-7B as the reranker, and we set the beam size to $b = 2$.

Algorithm 2: LLM Reranking Algorithm

Input: $D_{\text{src}} = \{s_0, s_1, \ldots, s_n\}$
Input: $D_{\text{hyps}} = \{H_0, H_1, \ldots, H_m\}$
Input: b: Beam size
Output: $D_{\text{hyp}} = \{h_0, h_1, \ldots, h_n\}$
$\mathcal{C}_{\text{beam}} \leftarrow \{(\emptyset, -\infty)\}$;
$P \leftarrow \emptyset$;
for $H \in D_{\text{hyps}}$ **do**
    for $(\mathbf{c}, \_) \in \mathcal{C}_{beam}$ **do**
        for $h \in H$ **do**
            $p_h \leftarrow \text{LLM}_{\text{MT}}(D_{\text{src}},\ \mathbf{c} \cup \{h\})$;
            $P \leftarrow P \cup \{(\mathbf{c}, h,\ p_h)\}$;
    $\mathcal{T}_b \leftarrow \text{Top}_b\,(P,\ \text{with respect to } p_h)$;
    $\mathcal{C}_{\text{beam}} \leftarrow \{(\mathbf{c} \cup \{h\},\ p_h) \mid (\mathbf{c}, h,\ p_h) \in \mathcal{T}_b\}$;
$(\mathbf{c}^*, p_c^*) \leftarrow \arg\max_{(\mathbf{c},\ p_c) \in \mathcal{C}_{\text{beam}}} p_c$;
$D_{\text{hyp}} \leftarrow \mathbf{c}^*$;
return D_{hyp}

6 Post processing

Finally, we applied the following postprocessing rules to the selected translations. The rules are designed based on alignment errors commonly seen in the model translations of the development sets.

- Apply NFKC normalization

```
次の英語を日本人のネイティブのように日本語に翻訳してください。
原文: {src0} {src1} {src2} 訳文: {hyp0} {hyp1}
```

Figure 3: The prompt for stepwise MBR-enhanced LLM decoding from English to Japanese. This is an example for translating {src2}. {src0} and {src1} correspond to S_{hist} in Algorithm 1, and {src2} corresponds to s_i in Algorithm 1. Line breaks are added for readability; there are no them in the actual prompt.

- Append an emoji to the end of the hypotheses if it's present at the end of the source sentence
- Replace Japanese brackets (「」) to its Chinese counterparts (“”) (Ja→Zh only)
- Replace Japanese commas (、) to its Chinese counterparts (,) (Ja→Zh only)
- Remove whitespaces before and after parentheses
- Remove whitespaces before and after commas, periods, exclamations, and question marks
- Fix letter case of alphabets in the hypotheses to match its counterparts in the source sentence
- Fix punctuations in the hypotheses to match their counterparts in the source sentence

7 Post Evaluation

We evaluated the performance of our system using automatic evaluation metrics. Specifically, using this year's test set as the evaluation data, we conducted the evaluation using COMET-22[9] (Rei et al., 2022), MetricX-XL[10] (Juraska et al., 2023), and CometKiwi-XL[11] (Rei et al., 2023) as the evaluation metrics. Note that, since several segments in this year's WMT test set contain multiple sentences, the scores could not be computed at the sentence level.

The results of the post-evaluation from En→Ja are presented in Table 4, while those for the Ja→Zh direction are shown in Table 5. In these tables, "VE" refers to the vocabulary-expanded model, and "CPO" refers to the model where Contrastive Preference Optimization was performed. Additionally, "EncDec" represents outputs from Encoder-Decoder MT models, "MBR (top-p)" refers to the case where MBR decoding was performed using pseudo references generated by top-p sampling, and "MBR (epsilon)" refers to the case where epsilon sampling was used.

[9] https://huggingface.co/Unbabel/wmt22-comet-da

[10] https://huggingface.co/google/metricx-23-xl-v2p0

[11] https://huggingface.co/Unbabel/wmt23-cometkiwi-da-xl

Performance of the LLM-based MT models. Table 4 shows that the translation performance of Llama2-13B is lower than that of Mistral-7B. One potential reason for this is the limited amount of data used for continual pre-training of Llama2-13B due to constraints in computational resources.

Efficiency of vocabulary expansion. Comparing the models with and without vocabulary expansion (`(b)` vs. `(d)`), there is no significant difference in performance. However, as shown in Table 13, the model with vocabulary expansion requires fewer training tokens than the model without it in our settings. The generation speed is also faster for the model with vocabulary expansion compared to the one without it. Thus, we believe vocabulary expansion could be a good option for improved inference efficiency.

CPO is effective but challenging. Comparing the performance before and after preference learning, the model with vocabulary expansion shows improvement across all evaluation metrics (`(d)` vs. `(e)`). On the other hand, the model without vocabulary expansion exhibits a significant decrease in performance for COMET-22 and CometKiwi-XL (`(b)` vs. `(c)`), leading to inconsistent results.

Qualitative analysis of outputs from the model without vocabulary expansion (i.e., `(c)`) revealed instances where decoding of byte-fallbacked text failed, resulting in text being replaced with replacement characters. This may be due to insufficient adjustment of the hyperparameters during CPO training.

Difference in pseudo references for MBR decoding. Comparing settings `(A)` vs. `(B)` and `(C)`, we observe that the performance improves when using MBR decoding compared to the 1-best output from the ensemble of models[12]. The difference in performance with regard to the pseudo-reference generation algorithms (`(i)` vs. `(j)` and `(B)` vs. `(C)`) was not significant.

[12] In the En→Ja, we use results from multiple models with different vocabularies for MBR decoding; hence we cannot compare the performance with the 1-best output from the ensemble of all transformers.

	COMET-22↑	MetricX-XL↓	CometKiwi-XL↑
(a) Llama2-13B	0.820	3.050	0.677
(b) Mistral-7B	0.841	2.806	0.711
(c) Mistral-CPO-7B	0.651	2.254	0.557
(d) Mistral-VE-7B	0.836	2.881	0.695
(e) Mistral-VE-CPO-7B	0.866	2.254	0.732
(f) NT5 (Morishita et al., 2022b)	0.847	2.697	0.718
(g) Stepwise MBR-Enhanced LLM Decoding	0.882	**2.052**	0.729
(i) EncDec → MBR (top-p)	**0.885**	2.263	0.737
(j) EncDec → MBR (epsilon)	0.884	2.264	**0.743**
(k) EncDec → MBR (top-p) → LLM Reranking	0.881	2.269	0.740

Table 4: Results of post evaluation in En→Ja.

	COMET-22↑	MetricX-XL↓	CometKiwi-XL↑
(A) EncDec ensemble	0.818	3.550	0.548
(B) EncDec → MBR (top-p)	**0.841**	**3.168**	**0.570**
(C) EncDec → MBR (epsilon)	**0.841**	3.230	0.566

Table 5: Results of post evaluation in Ja→Zh.

Performance of stepwise MBR-enhanced LLM decoding. `Stepwise MBR-Enhanced LLM Decoding` achieves the highest score on MetricX-XL. Additionally, compared to using a single LLM, the scores of COMET-22 and MetricX-XL improved. This improvement is likely because generating hypotheses at each step with MBR decoding helps eliminate obvious errors, such as repeated tokens.

Effectiveness of LLM reranking. LLM Reranking did not result in any significant improvements according to automatic evaluation metrics. However, we noted improved consistency within segments qualitatively. We intend to evaluate performance through human evaluation as part of future work.

8 Submission System

For the final submission system, we adopted system `(k)` for the En→Ja direction and system `(B)` for the Ja→Zh direction. However, particularly in the En→Ja direction, different systems ranked highest across various automatic evaluation metrics, leaving us uncertain about which system to select even after post-evaluation. Thus, further refinement of automatic evaluation metrics is essential to develop a superior system.

9 Negative Results and Discarded Trials

Poor performance of LLMs for Japanese-to-Chinese translation. We conducted continual pre-training and supervised fine-tuning of LLMs for Ja→Zh translation. However, the translation performance did not meet our expectations, leading us to exclude it from the submission system (see Table 5 for post evaluation results). This shortfall likely resulted from our computational resource constraints, which limited continual pre-training to Chinese datasets only. For further details, please refer to Section A.

Use of LLM outputs as candidates for MBR decoding. We also explored the inclusion of LLM outputs in the candidate pool for MBR Decoding. However, we observed a decrease in translation quality when these outputs were included, leading us to exclude this approach from the final system. This decline in quality can be attributed to two main factors: i). a substantial difference in the distribution between the outputs generated by LLMs and the pseudo references produced by Encoder-Decoder MT models, and ii). inadequate tuning of hyperparameters during decoding with LLMs.

10 Conclusion

This paper described our systems for the constrained track of the WMT 2024 Shared Task: General Machine Translation. We developed translation systems for En→Ja and Ja→Zh. To achieve consistent document-level machine translation, we concentrated on investigating the application of LLMs, which have become available for use this year, employing methods such as LLM Reranking and Stepwise MBR-Enhanced LLM Decoding.

Our submitted system consists of the following steps: i) First, we generate translations using multiple Encoder-Decoder MT models. ii) Next, we narrow down the generated candidates by selecting the optimal translation through MBR decoding. iii) Finally, we apply LLM reranking to incorporate contextual information in order to determine the final output (only for En→Ja). The results from the post-evaluation did not provide quantitative confirmation of the final submission system's effectiveness. However, we did observe a qualitative improvement in consistency within the documents. We hope for future research on better automatic evaluation metrics that can assess these document-level translation performances.

Acknowledgments

We would like to thank the member of the Tohoku NLP Group and NAIST NLP Laboratory for their cooperation in conducting this research. We would like to especially thank Mengyu Ye, Yunmeng Li, Qin Dai, Zhang Ying, and Suchun Xie for their advice on constructing clean Chinese datasets.

This work was supported by Moonshot R&D Grant Number JPMJMS2011 (fundamental research) and JST BOOST, Japan Grant Number JPMJBS2421 and JPMJBS2423.

Contributions

Keito Kudo conducted the cleaning of the monolingual data, trained and decoded LLM-based MT models, developed the Ja→Zh Encoder-Decoder MT models, and performed post-evaluations.
Hiroyuki Deguchi conducted MBR decoding.
Makoto Morishita cleaned the monolingual and bitext data, pre-trained and fine-tuned the Ja→Zh translation model.
Ryo Fujii designed and implemented rules for filtering and post-processing, and performed qualitative evaluation of the resulting translations.
Takumi Ito designed and customized Hojichar for data cleaning, and designed and implemented Section 4.2.
Shintaro Ozaki, **Koki Natsumi** conducted pre-training and fine-tuning of the Ja→Zh Encoder-Decoder MT models, along with back-translation.
Kai Sato, **Kazuki Yano** implemented filters for data cleaning.
Ryosuke Takahashi implemented filters for data cleaning, conducted preference learning, and prepared scripts for decoding.
Subaru Kimura conducted the cleaning of the monolingual data, implemented checkpoint averaging, fine-tuned the LLM-based MT models, and implemented post-processing.
Tomomasa Hara implemented filters for the cleaning of the monolingual data and performed hyperparameter tuning for LLM-based MT models.
Yusuke Sakai managed the training process for the Ja→Zh Encoder-Decoder MT models.
Jun Suzuki provided the primary computational budget and overall project advice and carried out the decoding of NT5.

References

A.Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3579–3584, Reykjavik, Iceland. European Language Resources Association (ELRA).

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy. European Association for Machine Translation.

Jianghao Chen, Pu Jian, Tengxiao Xi, Dongyi Yi, Qianlong Du, Chenglin Ding, Guibo Zhu, Chengqing Zong, Jinqiao Wang, and Jiajun Zhang. 2023. Chinesewebtext: Large-scale high-quality chinese web text extracted with effective evaluation model. *Preprint*, arXiv:2311.01149.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mbrs: A library for minimum bayes risk decoding. *Preprint*, arXiv:2408.04167.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Bryan Eikema and Wilker Aziz. 2020. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bryan Eikema and Wilker Aziz. 2022. Sampling-based approximations to minimum Bayes risk decoding for neural machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10978–10993, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Christian Federmann, Tom Kocmi, and Ying Xin. 2022. NTREX-128 – News Test References for MT Evaluation of 128 Languages. In *Proceedings of the First Workshop on Scaling Up Multilingual Evaluation*, pages 21–24, Online. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT Sentence Embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Patrick Fernandes, António Farinhas, Ricardo Rei, José G. C. de Souza, Perez Ogayo, Graham Neubig, and Andre Martins. 2022. Quality-aware decoding for neural machine translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States. Association for Computational Linguistics.

Markus Freitag, Behrooz Ghorbani, and Patrick Fernandes. 2023. Epsilon sampling rocks: Investigating sampling strategies for minimum Bayes risk decoding for machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9198–9209, Singapore. Association for Computational Linguistics.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024a. Continual Pre-Training for Cross-Lingual LLM Adaptation: Enhancing Japanese Language Capabilities. *Preprint*, arXiv:2404.17790.

Kazuki Fujii, Taishi Nakamura, and Rio Yokota. 2024b. llm-recipes.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 Evaluation Benchmark for Low-Resource and Multilingual Machine Translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

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 *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 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jegou, and Tomas Mikolov. 2017a. FastText.zip: Compressing text classification models.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017b. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Juraj Juraska, Mara Finkelstein, Daniel Deutsch, Aditya Siddhant, Mehdi Mirzazadeh, and Markus Freitag. 2023. MetricX-23: The Google Submission to the WMT 2023 Metrics Shared Task. In *Proceedings of the Eighth Conference on Machine Translation*, pages 756–767, Singapore. Association for Computational Linguistics.

Seungduk Kim, Seungtaek Choi, and Myeongho Jeong. 2024. Efficient and effective vocabulary expansion towards multilingual large language models. *CoRR*, abs/2402.14714.

Keito Kudo, Takumi Ito, Makoto Morishita, and Jun Suzuki. 2023. SKIM at WMT 2023 general translation task. In *Proceedings of the Eighth Conference on Machine Translation*, pages 128–136, Singapore. 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 (EMNLP)*, pages 66–71. Association for Computational Linguistics.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Makoto Morishita, Katsuki Chousa, Jun Suzuki, and Masaaki Nagata. 2022a. JParaCrawl v3.0: A Large-scale English-Japanese Parallel Corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6704–6710, Marseille, France. European Language Resources Association.

Makoto Morishita, Keito Kudo, Yui Oka, Katsuki Chousa, Shun Kiyono, Sho Takase, and Jun Suzuki. 2022b. NT5 at WMT 2022 General Translation Task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 318–325, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Chenghao Mou, Chris Ha, Kenneth Enevoldsen, and Peiyuan Liu. 2023. Chenghaomou/text-dedup: Reference snapshot.

Masaaki Nagata, Makoto Morishita, Katsuki Chousa, and Norihito Yasuda. 2024. A Japanese-Chinese Parallel Corpus Using Crowdsourcing for Web Mining. *Preprint*, arXiv:2405.09017.

Graham Neubig. 2011. The Kyoto Free Translation Task. http://www.phontron.com/kftt.

Naoaki Okazaki, Kakeru Hattori, Hirai Shota, Hiroki Iida, Masanari Ohi, Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Rio Yokota, and Sakae Mizuki. 2024. Building a Large Japanese Web Corpus for Large Language Models. *Preprint*, arXiv:2404.17733.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Reid Pryzant, Youngjoo Chung, Dan Jurafsky, and Denny Britz. 2018. JESC: Japanese-English Subtitle Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Ricardo Rei, Nuno M. Guerreiro, JosÃ© Pombal, Daan van Stigt, Marcos Treviso, Luisa Coheur, José G. C. de Souza, and André Martins. 2023. Scaling up CometKiwi: Unbabel-IST 2023 submission for the quality estimation shared task. In *Proceedings of the Eighth Conference on Machine Translation*, pages 841–848, Singapore. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021. WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96. Association for Computational Linguistics.

Kenta Shinzato. 2023. HojiChar: The text processing pipeline.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation. *Preprint*, arXiv:2207.04672.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

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 *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive Preference Optimization: Pushing the Boundaries of LLM Performance in Machine Translation. In *Forty-first International Conference on Machine Learning*.

请像中国本地人一样将以下日语翻译成中文。
原文: {src} 译文: {tgt}

Figure 4: The general prompt for supervised fine-tuning. {src} denotes the source sentence. {tgt} denotes the target sentence. Line breaks are added for readability; there are no them in the actual prompt.

	COMET-22↑	MetricX↓	CometKiwi↑
Llama2-13B	0.754	4.763	0.503
Mistral-7B	0.795	4.410	0.547
EncDec ensemble	0.818	3.550	0.548

Table 6: Post evaluation results of the LLM trained for Ja→Zh translation. Compared to the ensemble of Encoder-Decoder MT models, the performance of the LLM for Ja→Zh translation was not sufficient.

A Japanese-Chinese LLM

Training configurations. We trained LLMs for Ja→Zh translation, although these were not included in the final system. Due to time and computational resource constraints, we only conducted continual pre-training and supervised fine-tuning on Chinese monolingual corpora. During supervised fine-tuning, we used the template shown in Figure 4. Table 12, 13 lists the hyperparameters used for training in the Ja→Zh direction.

Post evaluation. We conducted evaluations for the LLMs trained for the Ja→Zh translation. Table 6 presents the results. The performance of the LLMs in the Ja→Zh translation was insufficient compared to the ensemble of Encoder-Decoder MT models. This is likely because we were limited to continual pre-training using only Chinese corpora due to computational resource constraints.

B Vocabulary Expansion for LLM

As described in Section 3.2, we aimed to improve the Japanese language generation capability of Mistral-7B by expanding the model' s vocabulary. Here, we provide details on the vocabulary expansion.

Construction of additional vocabulary. We first constructed a Japanese vocabulary using the unigram algorithm of the `Sentencepiece` tool (Kudo and Richardson, 2018). This vocabulary was trained on a subset of 30,000,000 samples from the Japanese Monolingual Corpus. We set the vocabulary size to 27,000. During vocabulary training, we enabled the options "byte_fallback" and "split_digits".

Vocabulary initialization. We initialized the embeddings for the additional vocabulary using the weighted average of the original Mistral embeddings. The weights were determined based on the similarity scores between the new and original Mistral vocabularies, computed by LaBSE (Feng et al., 2022). The process is described by the following equation:

$$\begin{aligned}\mathbf{v}_{\text{new}} &= \sum_{i=1}^{N} \left(\frac{\exp(w_i)}{\sum_{j=1}^{N} \exp(w_j)} \right) v_i \\ &= \sum_{i=1}^{N} \text{softmax}(w_i) v_i\end{aligned} \tag{6}$$

Here, $\mathbf{v}_{\text{new}}$ represents the embedding for the additional vocabulary, w_i is the similarity score between the additional vocabulary and vocabulary entry i as calculated by LaBSE, $\mathbf{v}_i$ is the vector of the existing vocabulary entry i, and n is the size of the original vocabulary. This method was also used to initialize the language modeling head.

Given our focus on the English-to-Japanese translation task, vocabularies other than English and Japanese are considered less critical. Therefore, we replaced any vocabulary not identified as Japanese, English, or special tokens with the new additional vocabulary. The determination of the language for each token followed these rules:

Japanese: Tokens consisting of *hiragana*, *katakana*, common-use *kanji*, symbols, JIS level 1 *kanji*, and ASCII characters

English: Tokens consisting solely of ASCII characters

Special tokens: Tokens split by byte fallback, as well as bos, eos tokens, etc.

Consequently, we expanded the vocabulary to 51,200.

Vocabulary warmup training. To address inconsistencies introduced by adding new vocabulary, prior research has proposed gradually training the model while fixing specific parameters after adding the vocabulary (Kim et al., 2024). We adopted a similar method to resolve these inconsistencies. Initially, we fixed the parameters of all transformer layers except for the embedding layer and the language modeling head and conducted the training. The hyperparameters used during this initial training phase are detailed in Table 11.

C Training Hyperparameters

The hyperparameters during the training of each model are shown in Table 7- 13.

Initial Translation Model	
Subword Size	32,000
Architecture	Transformer (big) with 6 layers, Encoder and Decoder FFN size of 8,192
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.98,$ $\epsilon = 1 \times 10^{-8},$ $\mathrm{weight\_decay} = 0.0$
Learning Rate Schedule	Inverse square root decay, Cosine
Warmup Steps	4,000
Max Learning Rate	0.001
Dropout	0.1
Gradient Clip	1.0
Batch Size	1,048,576 tokens
Max Number of Updates	50,000 steps
Averaging	Save a checkpoint every 500 steps and average the last ten
Implementation	`fairseq` (Ott et al., 2019)
Pre-training Configuration	
Subword Size	16,000
Architecture 1	Transformer (big) with 9 layers, Encoder FFN size of 16,384, and Decoder FFN size of 4,096
Architecture 2	Transformer (big) with 9 layers, Encoder and Decoder FFN size of 8,192
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.98,$ $\epsilon = 1 \times 10^{-8},$ $\mathrm{weight\_decay} = 0.0$
Learning Rate Schedule	Inverse square root decay, Cosine
Warmup Steps	4,000
Max Learning Rate	0.001
Dropout	0.1
Gradient Clip	0.1
Batch Size	1,048,576 tokens
Max Number of Updates	50,000 steps
Averaging	Save a checkpoint every 500 steps and average the last ten
Implementation	`fairseq` (Ott et al., 2019)
Fine-tuning Configuration	
Learning Rate Schedule	Fixed
Warmup Steps	N/A
Max Learning Rate	1×10^{-5}
Dropout	0.2
Gradient Clip	1.0
Batch Size	14,400 tokens
Max Number of Updates	1,000 steps
Averaging	Save a checkpoint every ten steps and average the last ten

Table 7: List of hyper-parameters. We used the initial translation model to generate synthetic data, the pre-training configuration to build the models described in Section 3.1, and the fine-tuning configuration to develop the models for submission. We created two models for pre-training and fine-tuning, labeled as "Architecture 1" or "Architecture 2," and used them for ensembling. The hyperparameters listed in the fine-tuning configuration represent only the differences from the pre-training configuration.

Llama2-13B Pretraining	
Vocab Size	32,000
Train Steps	10,000
Batch Size	1,572,864 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	250
Max Learning Rate	2×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 100 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Llama2-13B Supervised Finetuning	
Vocab Size	32,000
Train Steps	3,500
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	175
Max Learning Rate	3×10^{-6}
Min Learning Rate	3×10^{-7}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 100 steps and average the last three
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)

Table 8: A list of hyperparameters used when training Llama2-13B on the En→Ja task.

Mistral-7B Pretraining	
Vocab Size	32,000
Train Steps	20,000
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	500
Max Learning Rate	2×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 200 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Mistral-7B Supervised Finetuning	
Vocab Size	32,000
Train Steps	3,100
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	155
Max Learning Rate	1×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 200 steps and average the last three
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Mistral-7B Preference Learning	
Vocab Size	32,000
Train Steps	250
Batch Size	144 samples
Learning Rate Schedule	Constant
Learning Rate	1×10^{-5}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, weight_decay $= 0.1$
Gradient Clip	1.0
CPO β	0.1
CPO α	1.5
i_w (See Section 3.2)	740
Lora r	16
Lora α	32
Lora Dropout	0.1
Lora Target Layetr	All linear layer
Implementation	`Transformers` (Wolf et al., 2020), `TRL` (von Werra et al., 2020)

Table 9: A list of hyperparameters used when training Mistral-7B on the En→Ja task.

Mistral-7B (vocab expanded) Pretraining	
Vocab Size	51,200
Train Steps	12,283
Batch Size	1,376,256 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	300
Max Learning Rate	2×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.95,$ $\epsilon = 1 \times 10^{-6},$ weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 200 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Mistral-7B (vocab expanded) Supervised Finetuning	
Vocab Size	51,200
Train Steps	2,000
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	100
Max Learning Rate	1×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.95,$ $\epsilon = 1 \times 10^{-6},$ weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 200 steps and average the last two
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Mistral-7B (vocab expanded) Preference Learning	
Vocab Size	51,200
Train Steps	250
Batch Size	144 samples
Learning Rate Schedule	Fixed
Learning Rate	1×10^{-5}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.999,$ $\epsilon = 1 \times 10^{-8},$ weight_decay $= 0.1$
Gradient Clip	1.0
CPO β	0.1
CPO α	1.5
i_w (See Section 3.2)	740
Lora r	16
Lora α	32
Lora Dropout	0.1
Lora Target Layer	All linear layers
Implementation	`Transformers` (Wolf et al., 2020), `TRL` (von Werra et al., 2020)

Table 10: A list of hyperparameters used when training Mistral-7B with vocabulary expansion on the En→Ja task.

Mistral-7B (vocab extended) Vocabulary Warmup	
Vocab Size	51,200
Train Steps	1800
Batch Size	1,376,256 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	50
Max Learning Rate	2×10^{-4}
Min Learning Rate	6.6×10^{-7}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.95,$ $\epsilon = 1 \times 10^{-6},$ weight_decay $= 0.1$
Gradient Clip	1.0
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)

Table 11: A list of hyperparameters used when training Mistral-7B with vocabulary expansion for vocabulary warmup on the En→Ja task.

Llama2-13B Pretraining	
Vocab Size	32,000
Train Steps	10,000
Batch Size	1,572,864 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	250
Max Learning Rate	2×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.95,$ $\epsilon = 1 \times 10^{-6},$ weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 100 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Llama2-13B Supervised Finetuning	
Vocab Size	32,000
Train Steps	500
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	25
Max Learning Rate	3×10^{-6}
Min Learning Rate	3×10^{-7}
Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.95,$ $\epsilon = 1 \times 10^{-6},$ weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 25 steps and average the last three
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)

Table 12: A list of hyperparameters used when training Llama2-13B on the Ja→Zh task.

Mistral-7B Pretraining	
Vocab Size	32,000
Train Steps	20,000
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	500
Max Learning Rate	2×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 200 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)
Mistral-7B Supervised Finetuning	
Vocab Size	32,000
Train Steps	420
Batch Size	1,310,720 tokens
Learning Rate Schedule	Cosine (Loshchilov and Hutter, 2017)
Warmup Steps	25
Max Learning Rate	1×10^{-5}
Min Learning Rate	1×10^{-6}
Optimizer	Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-6}$, weight_decay $= 0.1$
Gradient Clip	1.0
Averaging	Save a checkpoint every 10 steps and average the last five
Implementation	`Transformers` (Wolf et al., 2020), `llm-recipies` (Fujii et al., 2024b)

Table 13: A list of hyperparameters used when training Mistral-7B on the Ja→Zh task.