# Lexically Constrained Neural Machine Translation with Explicit Alignment Guidance

**Guanhua Chen**[1], **Yun Chen**[*2], **Victor O.K. Li**[1]

[1] The University of Hong Kong

[2] Shanghai University of Finance and Economics

ghchenhku@connect.hku.hk, yunchen@sufe.edu.cn, vli@eee.hku.hk

## Abstract

Lexically constrained neural machine translation (NMT), which leverages pre-specified translation to constrain NMT, has practical significance in interactive translation and NMT domain adaptation. Previous works either modify the decoding algorithm or train the model on augmented datasets. These methods suffer from either high computational overheads or low copying success rates. In this paper, we investigate ATT-INPUT and ATT-OUTPUT, two alignment-based constrained decoding methods. These two methods revise the target tokens during decoding based on word alignments derived from encoder-decoder attention weights. Our study shows that ATT-INPUT translates better while ATT-OUTPUT is more computationally efficient. Capitalizing on both strengths, we further propose EAM-OUTPUT by introducing an explicit alignment module (EAM) to a pretrained Transformer. It decodes similarly as ATT-OUTPUT, except using alignments derived from the EAM. We leverage the word alignments induced from ATT-INPUT as labels and train the EAM while keeping the parameters of the Transformer frozen. Experiments on WMT16 De-En and WMT16 Ro-En show the effectiveness of our approaches on constrained NMT. In particular, the proposed EAM-OUTPUT method consistently outperforms previous approaches in translation quality, with light computational overheads over unconstrained baseline.

## Introduction

Lexically constrained neural machine translation (NMT) is a task that translates the source sentence to include pre-specified lexical constraints. It is useful in a range of settings, including interactive translation with user-provided lexical constraints (Koehn 2009), domain adaptation where lexical constraints are from pre-specified dictionary (Hasler et al. 2018), and scenarios where some phrases, such as product prices, company names or web URLs, are expected to be translated perfectly without any error. Different from statistical machine translation (Koehn, Och, and Marcu 2003), NMT has no explicit word alignment during model training or decoding. Therefore, it is not trivial to impose lexical constraints into the NMT model.

*Corresponding author



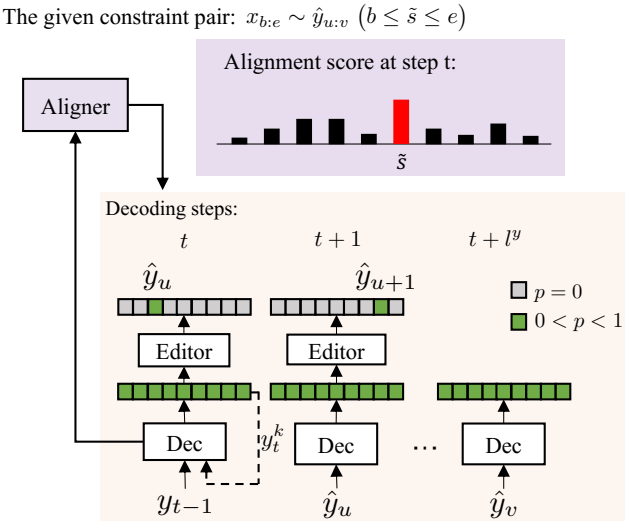


Figure 1: The proposed model structure for constrained decoding. We modify the decoding algorithm by adding an Aligner and Editor. During decoding, the Aligner extracts alignment from attention weights. If the aligned source token is in a given source constraint ($b \le \tilde{s} \le e$), the Editor will revise the decoder output probabilities at the current and the next $l^y - 1$ steps to force the generation of the target constraint $\hat{y}_{u:v}$, where $l^y = v - u + 1$.

Previous works either modify the decoding algorithm or train the model on augmented code-switched data to solve this problem. Some works (Hokamp and Liu 2017; Post and Vilar 2018; Hu et al. 2019) modify the decoding algorithm by enumerating all feasible constraints at each step and selecting hypotheses with constraints according to the translation probabilities. Although these methods can guarantee the presence of the constraints at test time, they significantly slow down the decoding and tend to translate the specified source phrases repeatedly or omit source phrases (Zhang et al. 2019). To improve the decoding efficiency, other works (Hasler et al. 2018; Alkhouli, Bretschner, and Ney 2018; Song et al. 2020) impose lexical constraints with word alignments. The alignments can be extracted with attention weights in vanilla Transformer (Alkhouli,

Bretschner, and Ney 2018) or from the alignment module in an alignment-enhanced Transformer (Song et al. 2020). However, the multi-head attention weights in the Transformer are reported to capture poor alignments (Garg et al. 2019). The alignment-enhanced Transformer (Song et al. 2020) relies on external aligner such as GIZA++ (Brown et al. 1993; Och and Ney 2003) and requires re-training the whole model on the translation and alignment tasks from scratch.

Another line of work trains the NMT model on pseudo code-switched dataset, which is created by replacing the corresponding source phrases with constraints (Song et al. 2019) or by appending constraints right after its correspondents (Dinu et al. 2019). The bilingual dictionary is requisite to provide the alignment between source and target phrases. During inference, the constraints are imposed on the source sentence similarly. These methods can translate the code-switched source sentence efficiently, however the noise in the dictionary may degrade their performance.

In this paper, we propose alignment-based constrained decoding methods that can balance between decoding accuracy and efficiency. We improve upon previous work by extracting more accurate alignments and removing the requirement for GIZA++ and training from scratch. As shown in Figure 1, the proposed methods impose lexical constraints with an Aligner and an Editor. At each step, the Aligner determines whether a source constraint is currently being translated, and the Editor revises target tokens with the corresponding target constraint accordingly. We investigate ATT-INPUT and ATT-OUTPUT, two approaches that derive alignments from attention weights and incorporate lexical constraints into vanilla Transformer. We find that ATT-INPUT generates better translations while ATT-OUTPUT has higher decoding efficiency. To capture both advantages, we further propose EAM-OUTPUT which introduces an explicit alignment module (EAM) to a pretrained Transformer. EAM-OUTPUT decodes similarly as ATT-OUTPUT except extracting alignments from the EAM. While keeping the Transformer parameters frozen, the EAM is trained in a self-training manner with alignment labels induced from ATT-INPUT. We examine our approaches on WMT16 De-En and WMT16 Ro-En datasets. The results demonstrate the effectiveness of our proposed methods on balancing translation quality and decoding efficiency.[1]

# Background

## NMT Decoding with Beam Search

Let $\mathrm{X} = \{x_1, ..., x_S\}$, $\hat{\mathrm{Y}} = \{\hat{y}_1, ..., \hat{y}_{\hat{T}}\}$ and $\mathrm{Y} = \{y_1, ..., y_T\}$ be source sentence, reference translation and model translation, respectively. At inference time, given model parameters $\theta$ and source sentence X, the translation with the highest probability is selected as model output:

$$\mathrm{Y} = \underset{\mathrm{Y}}{\operatorname{argmax}} \Big\{ \prod_{t=1}^{\mathrm{T}+1} p_\theta(y_t | y_{0:t-1}, \mathrm{X}) \Big\}. \tag{1}$$

[1]Code is public at https://github.com/ghchen18/cdalign

where the special tokens $y_0$ (i.e. $\langle\text{bos}\rangle$) and $y_{T+1}$ (i.e. $\langle\text{eos}\rangle$) are used to represent the beginning and end of all target sentences. The model usually uses beam search algorithm for decoding. The decoder maintains a beam of size $k$ containing a set of active hypotheses. At each decoding time step $t$, the model predicts a token distribution over the vocabulary $V$ for each hypothesis, producing $k \times V$ candidates. Then it picks the $k$ candidates with the highest overall scores as the new hypotheses for the next time step. The overall score is defined as $\sum_{t'=1}^{t} \log p_\theta(y_{t'} | y_{<t'}, X)$. Any hypothesis that ends with $\langle\text{eos}\rangle$ is restricted from being extended further. Beam search returns the highest scored sequence as the translation $Y$.

In lexically constrained decoding, apart from the source sentence $X$, the model is also provided with $N_c$ constraint pairs $\mathrm{C} = \{(\mathrm{C}_i^x, \mathrm{C}_i^y)\}_{i=1}^{N_c}$, where $\mathrm{C}_i^x = x_{b(i):e(i)}$ and $\mathrm{C}_i^y = \hat{y}_{u(i):v(i)}$ are source and target phrases in the $i^{\text{th}}$ constraint pair respectively. The length of the $i^{\text{th}}$ target constraint is denoted as $l_i^y = v(i) - u(i) + 1$. We modify the beam search decoding algorithm to incorporate these target constraints into the translation.

## Transformer Model

Transformer (Vaswani et al. 2017) has achieved the state-of-the-art performance on machine translation task in recent years. It is an encoder-decoder model that entirely relies on the attention mechanism (Bahdanau, Cho, and Bengio 2015). During decoding, word alignments can be extracted from encoder-decoder attention weights. Given source sentence $X$, we use $H = \{h_1, ..., h_S\}$ to denote the encoder output. At decoding step $t$, the model takes translation prefix $Y = \{y_0, ..., y_{t-1}\}$ as the decoder input and generates an output distribution to predict $y_t$. The $l^{\text{th}}$ decoder layer has $t$ decoder hidden states $Z_t^l = \{z_1^l, ..., z_t^l\}$. We denote the encoder-decoder attention weights of the last position as $w_t^l \in \mathbb{R}^{\mathrm{S}}$, in which the element $[w_t^l]_s$ measures the relevance between decoder hidden state $z_t^l$ and the encoder output $h_s$. We extract the aligned source position $a_t$ for the output target token $y_t$ from the attention weights $w_t^l$ with maximum a posterior strategy:

$$\begin{aligned} w_t^l &= \frac{1}{N} \sum_n \operatorname{softmax}\Big(\frac{(H\boldsymbol{O}_n^K)(z_t^l \boldsymbol{O}_n^Q)^\top}{\sqrt{d_k}}\Big), \\ a_t &= \underset{0<s'<S+1}{\operatorname{argmax}} [w_t^l]_{s'}. \end{aligned} \tag{2}$$

where $\boldsymbol{O}_n^K, \boldsymbol{O}_n^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are the key and query projection matrices for the $n$-th head, $N$ is the number of attention heads and $d_k = d_{\text{model}}/N$.

With the extracted word alignment pair $x_{a_t}$–$y_t$, target constraints can be imposed on the translation (Alkhouli, Bretschner, and Ney 2018). However, previous work (Garg et al. 2019; Song et al. 2020) report that such method is ill-suited for deriving accurate word alignments, thus degrading the performance of alignment-based constrained decoding. As a result, it is necessary to develop novel alignment-based constrained decoding methods and improve upon previous approaches by extracting more accurate alignments and by properly modifying the decoding process.

## Method

We propose alignment-based constrained decoding methods and introduce the Aligner and Editor modules into the Transformer, as shown in Figure 1. At each decoding time step, the Aligner extracts word alignments from attention weights, while the Editor changes the decoder output probabilities to impose the target constraint according to the word alignments. We investigate ATT-INPUT and ATT-OUTPUT, two constrained decoding methods for vanilla Transformer, and EAM-OUTPUT, a method that combines the benefits of ATT-INPUT and ATT-OUTPUT to further improve the performance of constrained decoding task.

### Decoding with Vanilla Transformer

During beam search with vanilla Transformer, the Aligner aligns target token $y_t$ to source token $x_{\tilde{s}}$ at decoding step $t$: $x_{\tilde{s}}$–$y_t$. If $\tilde{s}$ falls into the span of the $i^{\text{th}}$ source constraint, i.e. $\tilde{s} \in [b(i), e(i)]$, the Editor revises the decoder output distribution at time step $r$ ($t \leq r < t + l_i^y$) to incorporate the target constraint $C_i^y = \hat{y}_{u(i):v(i)}$. In particular, to force the generation of constrained token $\hat{y}_{u(i)+r-t}$ at step $r$, the Editor changes the output probabilities by setting the probabilities of all tokens to zero except $\hat{y}_{u(i)+r-t}$. The likelihood of $\hat{y}_{u(i)+r-t}$ is set as the maximum probability among the vocabulary to prevent the corresponding hypothesis from being trimmed in the following decoding time steps. Every constraint pair is allowed to be activated only once for each hypothesis during the whole decoding process. Now, a key question arises: how do we extract the aligned source token $x_{\tilde{s}}$ for target token $y_t$ from the attention weights?

In previous work (Garg et al. 2019; Ding, Xu, and Koehn 2019), attention weights $w_t^l$ from last or penultimate decoder layer are used to align $y_t$ and get alignment pair $x_{a_t}$–$y_t$ following Equation 2. This may be sub-optimal, especially when attention weights are from the bottom decoder layers (Chen et al. 2020b). $w_t^l$ describes the relevance of the decoder hidden state $z_t^l$ and the encoder output $H$. $H$ represents the source tokens $x_{1:S}$. For the top layers, $z_t^l$ represents the $t^{\text{th}}$ *output* token $y_t$. However, for the bottom layers, $z_t^l$ may be more related to the $t^{\text{th}}$ *input* token $y_{t-1}$. So $w_t^l$ describes the relevance of $x_{1:S}$ and $y_t$ or $y_{t-1}$. Therefore we make assumption that we can extract the alignment for $y_t$ either with attention weights $w_t^l$ or $w_{t+1}^l$, depending on the layer the attention weights are from. We thus propose two methods and compare them in the experiments.

**ATT-OUTPUT** When the attention weights are from top decoder layers, $w_t^l$ represent the relevance between source tokens $x_{1:S}$ and target token $y_t$. $y_t$ is the decoder **output** token at step $t$ when $w_t^l$ are computed, thus we name this method ATT-OUTPUT (see the example in Figure 2). The extracted alignment pair is $x_{a_t}$–$y_t$. If $a_t$ locates in the span of the $i^{\text{th}}$ source constraint, then $y_t$ is expected to be the first token of the $i^{\text{th}}$ target constraint $C_i^y$. The Editor revises the output distribution as discussed before to incorporate the constraint $C_i^y$.

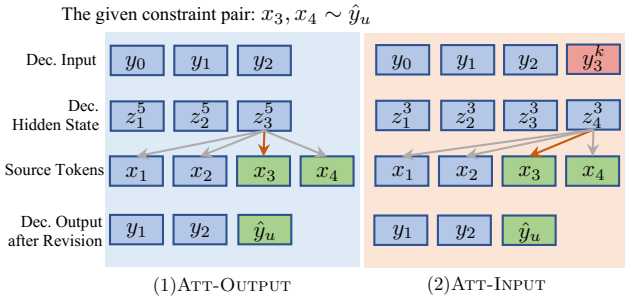


Figure 2: An example illustrating the difference between ATT-OUTPUT and ATT-INPUT. At time step $t = 3$, the decoder input is $y_{0:2}$. (1) ATT-OUTPUT extracts alignment for $y_3$ directly from the attention weights of the fifth decoder layer. (2) For ATT-INPUT, the hypothesis is extended with $k$ predicted target tokens. To get the alignment for each predicted token $y_3^k$, ATT-INPUT runs a second decoder forward process with $\{y_{0:2}, y_3^k\}$ as decoder input and derives alignment from attention weights of the third decoder layer. $\rightarrow$ represents attention weight. We mark the maximum attention weight red.

**ATT-INPUT** Different from ATT-OUTPUT, when attention weights $w_{t+1}^l$ are from the bottom decoder layers, $w_{t+1}^l$ represent the relevance between source tokens $x_{1:S}$ and target token $y_t$. Thus we can extract alignment pair $x_{a_{t+1}}$–$y_t$ from $w_{t+1}^l$ using Equation 2. This method is called ATT-INPUT since $w_{t+1}^l$ are computed at step $t+1$ and $y_t$ is the last decoder **input** token at this step. To compute $w_{t+1}^l$, the model has to run one more forward pass with $y_{0:t}$ as the decoder input (see the example in Figure 2). Specifically, the model extends each hypothesis with the top $k$ scored target tokens, producing $k \times k$ candidates. Then it runs one additional forward pass with the $k \times k$ candidates as decoder input and aligns $y_t$ of each candidate to the source token $x_{a_{t+1}}^j (j \in [1, k^2])$. If $x_{a_{t+1}}^j$ is in the span of the $i^{\text{th}}$ source constraint, then $y_t$ is expected to be the first token of $i^{\text{th}}$ target constraint $C_i^y$ and the Editor will revise $y_t$ of the corresponding candidate to be $\hat{y}_{u(i)}$. After that, the beam search algorithm selects the top $k$ scored non-repeating hypotheses from the $k \times k$ candidates as the beam for the next time step. For a hypothesis that is selected for revision with $C_i^y$, the Editor will revise the output distribution as discussed before at step $r$ ($t < r < t + l_i^y$), forcing the translation to contain $C_i^y$. Since ATT-INPUT requires one additional forward pass, it is less computationally efficient compared with ATT-OUTPUT.

### Decoding with Explicit Alignment Guidance

In our primary experiments, we find that ATT-INPUT translates better in terms of quality, while ATT-OUTPUT is more efficient. We also evaluate the word alignments quality and find that ATT-INPUT extracts significantly more accurate word alignments. To capture the advantages of both methods, we propose EAM-OUTPUT, which introduces an explicit alignment module (EAM) to the trained transformer, and further improves translation performance over ATT-

OUTPUT by extracting more accurate word alignments. Different from the joint training scheme in previous methods (Garg et al. 2019; Song et al. 2020), the EAM performs self-training on alignments extracted with ATT-INPUT while keeping the pretrained Transformer frozen.

Specifically, we add an alignment module only at the penultimate layer, the best layer to extract alignments with ATT-OUTPUT (Garg et al. 2019). The alignment module performs multi-head attention similar to the encoder-decoder attention sub-layer. It takes the encoder output $H = \{h_1, ..., h_S\}$ and the current decoder hidden state $z_t^{L-1}$ as input and outputs $g_t$, the alignment score corresponding to the output token $y_t$, using the similar equation as Equation 2.

We train the alignment module on the same training set that the Transformer is trained with and use symmetrized ATT-INPUT alignments as labels. Specifically, given the attention weights $\boldsymbol{W}^l = \{w_2^l;...;w_{\hat{T}+1}^l\} \in \mathbb{R}^{\hat{T}\times S}$ at the third layer[2] for a pair of sentences $X$ and $\hat{Y}$, binary word alignments $\boldsymbol{R}$ can be extracted with:

$$R_{t,s} = \begin{cases} 1 & \text{if } s = \text{argmax}_{s'} W_{t,s'}^l \\ 0 & \text{otherwise} \end{cases} . \quad (3)$$

We extract alignments with source-to-target and target-to-source vanilla Transformers and merge them with the 'grow-diag' heuristic (Koehn et al. 2005) following previous practice (Garg et al. 2019; Zenkel, Wuebker, and DeNero 2019) to get symmetrized ATT-INPUT alignments $\hat{\boldsymbol{R}}$. In this way, we remove the requisite for external aligners such as GIZA++ or Fast-Align (Dyer, Chahuneau, and Smith 2013). While the Transformer is pretrained and fixed, we train the alignment module with the loss function following Garg et al. (2019):

$$\mathcal{L}_a = -\frac{1}{\hat{T}} \sum_{t=1}^{\hat{T}} \sum_{s=1}^{S} \left(\hat{R}_{t,s}^p \odot \log G_{t,s}\right), \quad (4)$$

where $\boldsymbol{G} = \{g_1;...;g_{\hat{T}}\}$ is the alignment score matrix predicted by the EAM module, and $\hat{\boldsymbol{R}}^p$ denotes the normalized reference symmetrized ATT-INPUT alignments.[3] In this way, we transfer the alignment knowledge contained in symmetrized ATT-INPUT alignments into the EAM model and extract better word alignments. With the introduced EAM model, we follow ATT-OUTPUT for constrained decoding, except that we replace attention weights $w_t^l$ with $g_t$ when extracting the alignment pair $x_{a_t}$–$y_t$.

# Experiments

## Setup

**Data** We evaluate our methods on German-English (De-En) and Romanian-English (Ro-En) language pairs. Models are trained on WMT16 De-En and WMT16 Ro-En training set and evaluated on alignment testset and WMT news translation testset. We use `newstest2013` and `newsdev2016` as development sets for De-En and Ro-En respectively. For the alignment testset, we use the hand-aligned, publicly available alignment testset for De-En[4] and Ro-En[5]. For the WMT news translation testset, we use `newstest2016` for both De-En and Ro-En translations. All texts are tokenized using the Moses tokenizer (Koehn et al. 2007). We remove sentences longer than 250 words and sentence pairs with a source/target length ratio exceeding 1.5 in the training set. For all language pairs, we use byte-pair-encoding (Sennrich, Haddow, and Birch 2016, BPE) with 32K joint merge operations.

**Model Configuration** The `base` Transformer model described in Vaswani et al. (2017) with shared embeddings is used in all experiments. Model is implemented on `fairseq` toolkit[6] (Ott et al. 2019). We use Adam (Kingma and Ba 2015) and label smoothing for training. The learning rate is 0.0005 and warmup step is 4000. All the dropout probabilities are set to 0.3. The batch size is 32k tokens. Maximum updates number is 100k for the De-En language pair and 50k for the Ro-En language pair. For training the EAM, the maximum updates number is 10k. Consistent with the observation in Kovaleva et al. (2019), the source punctuation token at the end of sentence may have large attention weights, which interferes with the alignment extraction. We thus mask the attention weights of the punctuation token at the end of the sentence.

**Evaluation** Our methods are evaluated on alignment extraction task and lexically constrained translation task. The alignment task is evaluated by alignment error rate (AER) introduced in Vilar, Popović, and Ney (2006) on the alignment testset. We run NMT models to force-generate the target side of the testset and measure AER against the human alignment. We extract alignments of two directions for both language pairs and merge them with 'grow-diag' heuristic (Koehn et al. 2005). All AER scores are calculated in raw text format without BPE.

The constrained translation task is evaluated on clean and constrained testsets. Following previous works (Dinu et al. 2019; Hokamp and Liu 2017), we use beam search with beam size of 5. To eliminate the impact of sampling, we repeat experiments on five different sets of constraints and report the averaged scores. We report case-sensitive BLEU score using sacreBLEU[7] (Post 2018). The copying success rate (CSR) is the percentage of constraints that are successfully generated in the translation, and is calculated at word level after removing the BPE symbol. Statistical significance is tested with compare-mt toolkit (Neubig et al. 2019) for

[2] We set $l = 3$ because our primary experiments show that attention weights from the third decoder layer obtain the best alignment performance with ATT-INPUT.

[3] We simply normalize rows corresponding to target tokens that are aligned to at least one source token of $\hat{R}$.

[4] https://www-i6.informatik.rwth-aachen.de/goldAlignment

[5] http://web.eecs.umich.edu/~mihalcea/wpt/index.html#resources

[6] https://github.com/pytorch/fairseq

[7] BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.3

| Layer | ATT-OUTPUT | | ATT-INPUT | |
|---|---|---|---|---|
| | De-En | Ro-En | De-En | Ro-En |
| 1 | 75.7 | 89.5 | 17.0 | 90.5 |
| 2 | 68.5 | 73.9 | 23.7 | 19.6 |
| 3 | 70.5 | 73.6 | **16.3** | **14.3** |
| 4 | 30.7 | 29.7 | 64.4 | 66.6 |
| 5 | **24.9** | **24.2** | 74.7 | 75.1 |
| 6 | 30.5 | 29.2 | 74.9 | 75.9 |

Table 1: AER scores of symmetrized alignments on the development set. Symmetrized GIZA++ alignments are used as reference. The lower AER score the better. The best performance among all layers is underlined and in bold.

1000 resamples and $p = 0.05$. The decoding speed is tested on a single GeForce RTX 2080Ti GPU.

**Baselines** We compare with three existing methods to make the evaluation convincing.

• Vectorized Dynamic Beam Allocation (Hu et al. 2019, VDBA), which improves grid beam search (Hokamp and Liu 2017) and dynamic beam allocation (Post and Vilar 2018) with higher decoding efficiency.

• Add alignment head method (Song et al. 2020, Add-AlignHead), which jointly trains the model on translation and alignment tasks and uses the averaged attention weights over all decoder layers for constrained decoding.

• Code Switching method (Song et al. 2019, Code-Switch), which translates with constraints by training model on synthetic code-switching corpus.

| Method | De-En | Ro-En |
|---|---|---|
| Add-AlignHead | 23.2 | 40.9 |
| ATT-OUTPUT | 29.1 | 43.8 |
| ATT-INPUT | 20.2 | 35.2 |
| EAM-OUTPUT | 22.2 | 39.7 |

Table 2: AER scores of symmetrized alignments on the alignment testset. 'Grow-diag' heuristic is used to extract symmetrized alignments. The lower AER score the better.

## Alignment Extraction Task

To determine the best layer to extract alignments, we enumerate every decoder layer to induce binary alignments on the development set with force decoding using the ATT-OUTPUT and ATT-INPUT methods (see Table 1). The corresponding reference alignments are induced with GIZA++[8]. Alignments extracted with attention weights from the fifth decoder layer get best performance for ATT-OUTPUT, which is in consistent with the observation in Garg et al. (2019). The alignments using attention weights from the third decoder layer get best AER for ATT-INPUT, which outperforms the best alignments with ATT-OUTPUT. In the following experiments, we use attention weights from the fifth layer for the ATT-OUTPUT method and the third layer for the ATT-INPUT method.

ATT-OUTPUT extracts alignments for target token $y_t$ before observing $y_t$. In contrast, for the ATT-INPUT strategy, the alignments are extracted from attention weights after another decoder forward pass where $y_t$ is among the decoder input. This could explain the superior alignment performance of ATT-INPUT over ATT-OUTPUT. Suppose for the same source sentence, two alternative translations diverge at position $t$, with $y_t$ and $y'_t$ which respectively correspond to different source words. Presumably, the source word that is aligned to $y_i$ and $y'_i$ should change correspondingly. However, this is not possible under the ATT-OUTPUT method, because the alignment is extracted before prediction of $y_t$ or $y'_t$.

We further evaluate the alignments extracted with different alignment-based decoding methods on the testset with force decoding and present the AER in Table 2. Among all methods, ATT-INPUT gets the best AER since the other three methods align $y_t$ before observing it. For the other three methods, EAM-OUTPUT extracts the best alignments, indicating that EAM-OUTPUT can potentially improve the alignment-based constrained decoding performance.

## Constrained NMT Task

**Constraints** In practice, the constraints are provided by human translators through user feedback or by looking up the domain-specified dictionaries. Following previous work (Hokamp and Liu 2017; Post and Vilar 2018), in the experiment we simulate the practical scenario by sampling constraints from phrase pairs that are extracted from source and reference sentences using human-annotated alignments for the alignment testset and GIZA++ alignments for WMT news testset. Constraints are assumed to be not easily translated; thus they exclude the top-100 highest frequency tokens and tokens that have been successfully translated with greedy search. The number of constraints in each sentence is up to 3. For each constraint, the length of constrained phrase is sampled among 1 to 3. We sample uniformly in the sentences before applying BPE. In practical applications like domain adaptation via terminology dictionaries, constraints are given in the same order as their corresponding source phrases in the source sentence, which could be different from the order of constraints in the translation. We shuffle the sampled constraints to simulate this scenario.

**Results** The BLEU and CSR results on constrained decoding task are shown in Table 3 and Table 4, respectively. ATT-INPUT performs better than ATT-OUTPUT in terms of BLEU and CSR. The averaged BLEU over all language pairs of ATT-INPUT are 2.8 and 2.9 higher than that of ATT-OUTPUT on the alignment and news translation testsets, respectively. However, ATT-OUTPUT decodes faster than ATT-INPUT, which can be attributed to ATT-INPUT

[8]https://github.com/moses-smt/mgiza

| Model | Alignment testset | | | | | WMT news translation testset | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | De→En | En→De | Ro→En | En→Ro | Avg. | De→En | En→De | Ro→En | En→Ro | Avg. | Time |
| Vanilla Transformer | 31.0 | 24.1 | 21.0 | 16.7 | 23.2 | 36.2 | 31.7 | 33.2 | 26.6 | 31.9 | 716 |
| VDBA | 34.2⋆ | 25.4 | 24.2⋆ | 19.2⋆ | 25.8 | 41.5 | 32.7 | 37.5 | 30.5 | 35.6 | 1590 |
| Add-AlignHead | 33.3 | 27.0 | 24.0⋆ | 17.6 | 25.5 | 41.0 | 36.1 | 36.4 | 30.5 | 36.0 | 1145 |
| Code-Switch | 33.1 | 27.4⋆ | 22.7 | 17.2 | 25.1 | 40.8 | 36.1 | 36.7 | 30.6 | 36.1 | 759 |
| ATT-OUTPUT | 29.6 | 23.6 | 22.2 | 17.4 | 23.2 | 38.5 | 33.7 | 35.3 | 29.0 | 34.1 | 997 |
| ATT-INPUT | 33.7 | 26.9 | 23.6 | **19.7**⋆ | 26.0 | 41.6 | 36.9 | 37.6 | 31.7 | 37.0 | 1442 |
| EAM-OUTPUT | **34.7** | **27.7** | **24.2** | 19.6 | 26.6 | **42.6** | **37.8** | **38.0** | **32.1** | 37.6 | 968 |

Table 3: BLEU scores of constrained decoding on the testsets of both alignment task and news translation task. The best result among each column is underlined and in bold. The 'Vanilla Transformer' row presents unconstrained decoding results with vanilla Transformer. Scores with asterisk indicate no significant difference with EAM-OUTPUT results after statistic significance test. Time (Second) is the total decoding time on De-En news translation testset when $\mathrm{batchsize} = 1$.

| Model | De→En | En→De | Ro→En | En→Ro |
|---|---|---|---|---|
| VDBA | 99.3% | 99.7% | 99.0% | 99.0% |
| Add-AlignHead | 90.3% | 89.8% | 80.8% | 84.4% |
| Code-Switch | 91.2% | 90.2% | 81.2% | 66.2% |
| ATT-OUTPUT | 92.9% | 85.9% | 83.6% | 82.1% |
| ATT-INPUT | 95.2% | 95.1% | 93.9% | 97.3% |
| EAM-OUTPUT | 97.4% | 96.1% | 93.7% | 96.7% |

Table 4: Copying success rate (CSR) on the alignment testset. CSR is calculated on word level without BPE.

requiring an additional forward pass to extract alignments (discussed in Section ). EAM-OUTPUT gets the highest averaged BLEU score in both alignment and news translation testsets. It significantly outperforms Add-AlignHead and Code-Switch in terms of BLEU and CSR. When comparing with ATT-INPUT, it obtains slightly higher BLEU and similar CSR, with significantly faster decoding speed. VDBA gets the highest CSR score, but its time complexity is also the highest and the averaged BLEU is lower than EAM-INPUT. In summary, EAM-OUTPUT achieves the best overall performance when considering BLEU, CSR and decoding speed.

We also test EAM-INPUT, the method that adds the EAM to the third decoder layer and decodes similarly as ATT-INPUT, except replacing $w^l_{t+1}$ with $g_{t+1}$, the output of EAM when the decoder input is $y_{0:t}$. We find that although EAM-INPUT extracts significantly better alignments than EAM-OUTPUT, they have similar BLEU scores. For the alignment task, observing the ground truth $\hat{y}_t$ can help to extract more accurate alignment for $\hat{y}_t$. However, for constrained decoding, since EAM-INPUT only observes $y_t$ predicted by the model, its alignment extraction ability may degrade. Considering the computational overhead brought by EAM-INPUT, we recommend EAM-OUTPUT for constrained decoding.

| Metric | no-cons | gt-cons | o2m-cons |
|---|---|---|---|
| BLEU | 30.5 | 31.2 | 30.9 |
| CSR | 67.9% | 96.4% | 81.2% |

Table 5: Constrained decoding on the De-En WSD testset. No-cons, gt-cons and o2m-cons represent decoding given no, ground truth and one-to-many constraints respectively.

**Decoding with One-to-many Constraints** In practical scenarios when constraints are derived automatically from dictionaries, each source phrase may have multiple target translation candidates. We extend the EAM-OUTPUT method to work under such setting and test its performance on De-En word sense disambiguation (WSD) testset[9] (Rios Gonzales, Mascarell, and Sennrich 2017). Each source sentence in the testset has an ambiguous phrase and each ambiguous phrase is provided with multiple translation candidates. During inference, if the aligned source token locates in the source phrase span, we enumerate all target constraint candidates and append each after the hypothesis to get a batch of new hypotheses. The batch size is the same as the constraint candidates number. Then the model runs another decoder forward pass and selects the constraint with the highest length-averaged log-probability as the target constraint. EAM-OUTPUT trained on WMT16 De-En training set is used in this experiment. According to the results shown in Table 5, EAM-OUTPUT improves over unconstrained baseline given one-to-many constraints, demonstrating its ability to select the right target constraint from multiple candidates. We also present the upper bound when

[9]https://github.com/ZurichNLP/ContraWSD

| Item | Translations |
|---|---|
| Input | Dies bestätigt auch Peter Arnold vom Landratsamt Offenburg. |
| Without constraint | Peter Arnold of the Landratsamt Offenburg also confirms this. |
| +(Landratsamt,District Office) | This is also confirmed by Peter Arnold of the District Office Offenburg. |
| Reference | This was also confirmed by Peter Arnold from the Offenburg District Office. |
| Input | Bekannt ist der Büchner-Preisträger vor allem als Prosaautor, Theatertexte sind in seinem Werk rar. |
| Without constraint | The bookmaker-prizewinner is known mainly as a prosaautor, theatre texts are rar in his work. |
| +(Büchner, Büchner)+(rar, rare) +(Prosaautor, prose writer) | The Büchner-laureate is known above all as prose writer, theatre texts are rare in his work. |
| Reference | The Büchner prizewinner is known primarily as a writer of prose, with theatre texts something of a rarity for him. |

Table 6: Lexically constrained translation examples for German to English with the EAM-OUTPUT method.

giving the ground truth target constraints (gt-cons) for reference. More explorations in decoding with one-to-many constraints are left as future work.

**Case Study** We present two examples of translations with EAM-OUTPUT in Table 6. The vanilla Transformer just copies some German phrases and fails to translate them, such as 'Landratsamt', 'Prosaautor' and 'rar'. It mistranslates the person name 'Büchner' which should be copied. The EAM-OUTPUT method successfully translates these phrases given the constrained pairs. These examples and manual inspections of others indicate that our method successfully incorporates the given constrained pairs into the translations.

## Related Work

Hokamp and Liu (2017) propose Grid Beam Search (GBS), a lexically constrained decoding algorithm with only target constraints provided. The constraints are enforced in output translations by enumerating constraints at each decoding step. The beam size varies with the number of constraints and can hardly scale to batch decoding. Post and Vilar (2018) propose dynamic beam allocation (DBA) algorithm, which dynamically allocates the slots in a beam with fixed size and improves the efficiency of GBS. Hu et al. (2019) further improve DBA by batch decoding and trie representations. However, Zhang et al. (2019) report that these methods either translate the specified source phrases repeatedly or omit some source phrases. Hasler et al. (2018) propose to conduct alignment-based constrained decoding with finite-state acceptors, but they fail to extract accurate alignments from attention weights. Song et al. (2020) add additional attention modules on each decoder layer and jointly train the model on the translation and alignment tasks (Garg et al. 2019). It trains an alignment-enhanced Transformer from scratch and requires alignment labels from external aligners such as GIZA++; thus its training cost is high. Recently, Susanto, Chollampatt, and Tan (2020) propose to impose target constraints with Levenshtein Transformer (Gu, Wang, and Zhao 2019) in a non-autoregressive manner. Starting from the given constraints, the model inserts tokens at every time step. They assume the order of given constraints are the same with their order in the reference, which may not be the case when constraints are provided by dictionaries.

Another line of works (Song et al. 2019; Dinu et al. 2019; Chen et al. 2020a) create a synthetic code-switching corpus to augment the training data. The code-switching corpus is built by replacing or appending the target constraint with the corresponding source phrase according to a bilingual dictionary (Song et al. 2019; Dinu et al. 2019) or by appending the target constraints after the source sentence (Chen et al. 2020a). By training on a mixture of original and synthetic parallel corpora, the model learns to translate code-switching source sentences. Although translating fast, Song et al. (2019); Dinu et al. (2019) rely on the quality of the bilingual dictionary and all these works cannot guarantee the presence of target constraints in translation.

## Conclusion

Lexically constrained machine translation aims to incorporate external target constraints into NMT model and has many practical applications, such as interactive translation and NMT domain adaptation. In this paper, we investigate ATT-INPUT and ATT-OUTPUT, two alignment-based constrained decoding methods for vanilla Transformer and propose EAM-OUTPUT, a novel self-training and alignment based constrained decoding approach. EAM-OUTPUT extends standard Transformer by introducing an alignment module in a plug and play manner. We train this module with self-training while keeping the underlying pretrained Transformer frozen. Experiments on alignment and translation tasks have proven the effectiveness of our methods. In particular, EAM-OUTPUT improves over all baselines on balancing among translation quality, constraints copying success rate and decoding efficiency.

## Acknowledgements

We thank the anonymous reviewers for their insightful feedback on this work. Yun Chen is partially supported by the Fundamental Research Funds for the Central Universities.

## References

Alkhouli, T.; Bretschner, G.; and Ney, H. 2018. On The Alignment Problem In Multi-Head Attention-Based Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation*, 177–185.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Brown, P. F.; Pietra, V. J. D.; Pietra, S. A. D.; and Mercer, R. L. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.* 19(2): 263–311. ISSN 0891-2017.

Chen, G.; Chen, Y.; Wang, Y.; and Li, V. O. 2020a. Lexical-Constraint-Aware Neural Machine Translation via Data Augmentation. In *Proceedings of IJCAI*, 3587–3593.

Chen, Y.; Liu, Y.; Chen, G.; Jiang, X.; and Liu, Q. 2020b. Accurate Word Alignment Induction from Neural Machine Translation. In *Proceedings of EMNLP*, 566–576.

Ding, S.; Xu, H.; and Koehn, P. 2019. Saliency-driven Word Alignment Interpretation for Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, 1–12. Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/W19-5201.

Dinu, G.; Mathur, P.; Federico, M.; and Al-Onaizan, Y. 2019. Training Neural Machine Translation to Apply Terminology Constraints. In *Proceedings of ACL*, 3063–3068.

Dyer, C.; Chahuneau, V.; and Smith, N. A. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of NAACL*, 644–648. Atlanta, Georgia: Association for Computational Linguistics.

Garg, S.; Peitz, S.; Nallasamy, U.; and Paulik, M. 2019. Jointly Learning to Align and Translate with Transformer Models. In *Proceedings of EMNLP*, 4453–4462.

Gu, J.; Wang, C.; and Zhao, J. 2019. Levenshtein Transformer. In *Proceedings of NeurIPS*, 11179–11189.

Hasler, E.; de Gispert, A.; Iglesias, G.; and Byrne, B. 2018. Neural Machine Translation Decoding with Terminology Constraints. In *Proceedings of NAACL*, 506–512.

Hokamp, C.; and Liu, Q. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of ACL*, 1535–1546.

Hu, J. E.; Khayrallah, H.; Culkin, R.; Xia, P.; Chen, T.; Post, M.; and Van Durme, B. 2019. Improved Lexically Constrained Decoding for Translation and Monolingual Rewriting. In *Proceedings of NAACL*, 839–850. Minneapolis, Minnesota.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*.

Koehn, P. 2009. A process study of computer-aided translation. *Machine Translation* 23: 241–263.

Koehn, P.; Axelrod, A.; Mayne, A. B.; Callison-Burch, C.; Osborne, M.; and Talbot, D. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *International Workshop on Spoken Language Translation (IWSLT) 2005*.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; Dyer, C.; Bojar, O.; Constantin, A.; and Herbst, E. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL*, 177–180.

Koehn, P.; Och, F. J.; and Marcu, D. 2003. Statistical Phrase-Based Translation. In *Proceedings of NAACL*, 127–133.

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 2019. Revealing the Dark Secrets of BERT. In *Proceedings of EMNLP*, 4365–4374. Hong Kong, China.

Neubig, G.; Dou, Z.-Y.; Hu, J.; Michel, P.; Pruthi, D.; and Wang, X. 2019. compare-mt: A Tool for Holistic Comparison of Language Generation Systems. In *Proceedings of NAACL*.

Och, F. J.; and Ney, H. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1): 19–51.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL*.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation*, 186–191.

Post, M.; and Vilar, D. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of NAACL*, 1314–1324.

Rios Gonzales, A.; Mascarell, L.; and Sennrich, R. 2017. Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings. In *Proceedings of the Second Conference on Machine Translation*, 11–19.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*.

Song, K.; Wang, K.; Yu, H.; Zhang, Y.; Huang, Z.; Luo, W.; Duan, X.; and Zhang, M. 2020. Alignment-Enhanced Transformer for Constraining NMT with Pre-Specified Translations. In *Proceedings of AAAI*.

Song, K.; Zhang, Y.; Yu, H.; Luo, W.; Wang, K.; and Zhang, M. 2019. Code-Switching for Enhancing NMT with Pre-Specified Translation. In *Proceedings of NAACL*, 449–459.

Susanto, R. H.; Chollampatt, S.; and Tan, L. 2020. Lexically Constrained Neural Machine Translation with Levenshtein Transformer. In *Proceedings of ACL*, 3536–3543.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. In *Proceedings of NeurIPS*, 5998–6008.

Vilar, D.; Popović, M.; and Ney, H. 2006. AER: Do we need to "improve" our alignments? In *International Workshop on Spoken Language Translation (IWSLT) 2006*.

Zenkel, T.; Wuebker, J.; and DeNero, J. 2019. Adding Interpretable Attention to Neural Translation Models Improves Word Alignment. *arXiv CoRR* abs/1901.11359.

Zhang, J.; Luan, H.; Sun, M.; Zhai, F.; Xu, J.; and Liu, Y. 2019. Neural Machine Translation with Explicit Phrase Alignment. ArXiv preprint 1911.11520.