A Template-based Method for Constrained Neural Machine Translation

Shuo Wang[1] **Peng Li**[2*] **Zhixing Tan**[6] **Zhaopeng Tu**[7] **Maosong Sun**[1,4] **Yang Liu**[1,2,3,4,5*]
[1]Dept. of Comp. Sci. & Tech., Institute for AI, Tsinghua University, Beijing, China
[1]Beijing National Research Center for Information Science and Technology
[2]Institute for AI Industry Research, Tsinghua University, Beijing, China
[3]Beijing Academy of Artificial Intelligence, Beijing, China
[4]International Innovation Center of Tsinghua University, Shanghai, China
[5]Quan Cheng Laboratory [6]Zhongguancun Laboratory, Beijing, P.R.China [7]Tencent AI Lab

Abstract

Machine translation systems are expected to cope with various types of constraints in many practical scenarios. While neural machine translation (NMT) has achieved strong performance in unconstrained cases, it is non-trivial to impose pre-specified constraints into the translation process of NMT models. Although many approaches have been proposed to address this issue, most existing methods can not satisfy the following three desiderata at the same time: (1) high translation quality, (2) high match accuracy, and (3) low latency. In this work, we propose a template-based method that can yield results with high translation quality and match accuracy and the inference speed of our method is comparable with unconstrained NMT models. Our basic idea is to rearrange the generation of constrained and unconstrained tokens through a template. Our method does not require any changes in the model architecture and the decoding algorithm. Experimental results show that the proposed template-based approach can outperform several representative baselines in both lexically and structurally constrained translation tasks. [1]

1 Introduction

Constrained machine translation is of important value for a wide range of practical applications, such as interactive translation with user-specified lexical constraints (Koehn, 2009; Li et al., 2020; Jon et al., 2021), domain adaptation with in-domain dictionaries (Michon et al., 2020; Niehues, 2021), and webpage translation with markup tags as structural constraints (Hashimoto et al., 2019; Hanneman and Dinu, 2020). Developing constrained neural machine translation (NMT) approaches can make NMT models applicable to more real-world scenarios (Bergmanis and Pinnis, 2021).

However, it is challenging to directly impose constraints for NMT models due to their end-to-end nature (Post and Vilar, 2018). In accordance with this problem, a branch of studies modifies the decoding algorithm to take the constraints into account when selecting candidates (Hokamp and Liu, 2017; Hasler et al., 2018; Post and Vilar, 2018; Hu et al., 2019; Hashimoto et al., 2019). Although constrained decoding algorithms can guarantee the presence of constrained tokens, they can significantly slow down the translation process (Wang et al., 2022) and can sometimes result in poor translation quality (Zhang et al., 2021).

Another branch of works constructs synthetic data to help NMT models acquire the ability to translate with constraints (Song et al., 2019; Dinu et al., 2019; Michon et al., 2020). For instance, Hanneman and Dinu (2020) propose to inject markup tags into plain parallel texts to learn structurally constrained NMT models. The major drawback of data augmentation based methods is that they sometimes violate the constraints (Hanneman and Dinu, 2020; Chen et al., 2021), limiting their application in constraint-critical situations.

In this work, we use *free tokens* to denote the tokens that are not covered by the provided constraints. Our motivation is to decompose the whole constrained translation task into the arrangement of constraints and the generation of free tokens. The constraints can be of many types, ranging from phrases in lexically constrained translation to markup tags in structurally constrained translation. Intuitively, only arranging the provided constraints into the proper order is much easier than generating the whole sentence. Therefore, we build a template by abstracting free token fragments into nonterminals, which are used to record the relative position of all the involved fragments. The template can be treated as a plan of the original sentence. The arrangement of constraints can be learned through a *template generation* sub-task.

* Corresponding authors: P.Li (lipeng@air.tsinghua.edu.cn) and Y.Liu (liuyang2011@tsinghua.edu.cn).
[1]The source code is available at https://github.com/THUNLP-MT/Template-NMT.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3665 - 3679
December 7-11, 2022 ©2022 Association for Computational Linguistics

Once the template is generated, we need some derivation rules to convert the nonterminals mentioned above into free tokens. Each derivation rule shows the correspondence between a nonterminal and a free token fragment. These rules can be learned by the NMT model through semi-structured data. We call this sub-task *template derivation*. During inference, the model firstly generates the template and then extends each nonterminal in the template into natural language text. Note that the two proposed sub-tasks can be accomplished through a single decoding pass. Thus the decoding speed of our method is comparable with unconstrained NMT systems. By designing template format, our approach can cope with different types of constraints, such as lexical constraints, XML structural constraints, or Markdown constraints.

Contributions In summary, the contributions of this work can be listed as follows:

- We propose a novel template-based constrained translation framework to disentangle the generation of constraints and free tokens.
- We instantiate the proposed framework with both lexical and structural constraints, demonstrating the flexibility of this framework.
- Experiments show that our method can outperform several strong baselines, achieving high translation quality and match accuracy while maintaining the inference speed.

2 Related Work

2.1 Lexically Constrained Translation

Several researchers direct their attention to modifying the decoding algorithm to impose lexical constraints (Hasler et al., 2018). For instance, Hokamp and Liu (2017) propose grid beam search (GBS) that organizes candidates in a grid, which enumerates the provided constrained tokens at each decoding step. However, the computation complexity of GBS scales linearly with the number of constrained tokens. To reduce the runtime complexity, Post and Vilar (2018) propose dynamic beam allocation (DBA), which divides a fixed size of beam for candidates having met the same number of constraints. Hu et al. (2019) propose to vectorize DBA further. The resulting VDBA algorithm is still significantly slower compared with the vanilla beam search algorithm (Wang et al., 2022).

Another line of studies trains the model to copy the constraints through data augmentation. Song et al. (2019) propose to replace the corresponding source phrases with the target constraints, and Dinu et al. (2019) propose to insert target constraints as inline annotations. Some other works propose to append target constraints to the whole source sentence as side constraints (Chen et al., 2020; Niehues, 2021; Jon et al., 2021). Although these methods introduce little additional computational overhead at inference time, they can not guarantee the appearance of the constraints (Chen et al., 2021). Xiao et al. (2022) transform constrained translation into a bilingual text-infilling task. A limitation of text-infilling is that it can not reorder the constraints, which may negatively affect the translation quality for distinct language pairs.

Recently, some researchers have tried to adapt the architecture of NMT models for this task. Susanto et al. (2020) adopt non-autoregressive translation models (Gu et al., 2019) to insert target constraints. Wang et al. (2022) prepend vectorized keys and values to the attention modules (Vaswani et al., 2017) to integrate constraints. However, their model may still suffer from low match accuracy when decoding without VDBA. In this work, our method can achieve high translation quality and match accuracy without significantly increasing the inference overhead.

2.2 Structurally Constrained Translation

Structurally constrained translation is useful since text data is often wrapped with markup tags on the Web (Hashimoto et al., 2019), which is an essential source of information for humans. Compared with lexically constrained translation, structurally constrained translation is relatively unexplored. Joanis et al. (2013) examine a two-stage method for statistical machine translation systems, which firstly translates the plain text and then injects the tags based on phrase alignments and some carefully designed rules. Moving to the NMT paradigm, large-scale parallel corpora with structurally aligned markup tags are scarce. Hanneman and Dinu (2020) propose to inject tags into plain text to create synthetic data. Hashimoto et al. (2019) collect a parallel dataset consisting of structural text translated by human experts. Zhang et al. (2021) propose a constrained decoding algorithm to translate structured text. However, their method significantly slows down the translation process.

In this work, our approach can be easily extended for structural constraints, leaving the decoding algorithm unchanged. The template in our approach can be seen as an intermediate plan, which has been investigated in the field of data-to-text generation (Moryossef et al., 2019). Zhang et al. (2019) also explored the idea of disentangling different parts in a sentence using special tokens.

3 Approach

3.1 Template-based Machine Translation

Given a source-language sentence $\mathbf{x} = x_1 \cdots x_I$ and a target-language sentence $\mathbf{y} = y_1 \cdots y_J$, an NMT model is trained to estimate the conditional probability $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$, which can be given by

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{j=1}^{J} P(y_j|\mathbf{x}, \mathbf{y}_{<j};\boldsymbol{\theta}), \quad (1)$$

where $\boldsymbol{\theta}$ is the set of parameters to optimize and $\mathbf{y}_{<j}$ is the partial translation at the j-th step.

In this work, we firstly build a template to simplify the whole sentence. Formally, we use $\mathbf{s}$ and $\mathbf{t}$ to represent the source- and target-side templates, respectively. In the template, free token fragments are abstracted into nonterminals. We use $\mathbf{e}$ and $\mathbf{f}$ to denote the derivation rules of the nonterminals for the source and target template, respectively.

The model is trained on two sub-tasks. Firstly, the model learns to generate the target template $\mathbf{t}$:

$$P(\mathbf{t}|\mathbf{s},\mathbf{e};\boldsymbol{\theta}) = \prod_{j=1}^{T} P(t_j|\mathbf{s},\mathbf{e},\mathbf{t}_{<j};\boldsymbol{\theta}). \quad (2)$$

Secondly, we train the same model to estimate the conditional probability of $\mathbf{f}$:

$$P(\mathbf{f}|\mathbf{s},\mathbf{e},\mathbf{t};\boldsymbol{\theta}) = \prod_{j=1}^{F} P(f_j|\mathbf{s},\mathbf{e},\mathbf{t},\mathbf{f}_{<j};\boldsymbol{\theta}). \quad (3)$$

The target sentence $\mathbf{y}$ can be reconstructed by extending each nonterminal in $\mathbf{t}$ using the corresponding derivation rule in $\mathbf{f}$. We can jointly learn the two sub-tasks in one pass to improve both the training and inference efficiency. Formally, the model is trained to maximize the following joint probability of $\mathbf{t}$ and $\mathbf{f}$ in practice:

$$P(\mathbf{t},\mathbf{f}|\mathbf{s},\mathbf{e};\boldsymbol{\theta}) = P(\mathbf{t}|\mathbf{s},\mathbf{e};\boldsymbol{\theta}) \times P(\mathbf{f}|\mathbf{s},\mathbf{e},\mathbf{t};\boldsymbol{\theta}). \quad (4)$$

3.2 Template for Lexical Constraints

In lexically constrained translation, some source phrases in the input sentence are required to be translated into pre-specified target phrases. For a source sentence $\mathbf{x}$, we use $\left\{\left\langle \mathbf{u}^{(n)}, \mathbf{v}^{(n)} \right\rangle\right\}_{n=1}^{N}$ to denote the given constraint pairs, where $\mathbf{u}^{(n)}$ is the n-th source constraint, and $\mathbf{v}^{(n)}$ is the corresponding target constraint. All the N source constraints can divide $\mathbf{x}$ into $2N+1$ fragments:

$$\mathbf{x} = \mathbf{p}^{(0)}\mathbf{u}^{(1)}\mathbf{p}^{(1)} \cdots \mathbf{u}^{(N)}\mathbf{p}^{(N)}, \quad (5)$$

where $\mathbf{p}^{(n)}$ is the n-th free token fragment. We can set $\mathbf{p}^{(0)}$ to an empty string to represent sentences that start with a constraint, and set $\mathbf{p}^{(N)}$ to an empty string for sentences that end with a constraint. We can also set $\mathbf{p}^{(n)}$ to an empty string for the cases where $\mathbf{u}^{(n)}$ and $\mathbf{u}^{(n+1)}$ are adjacent in $\mathbf{x}$. Similarly, the target sentence can be represented by

$$\mathbf{y} = \mathbf{q}^{(0)}\mathbf{v}^{(i_1)}\mathbf{q}^{(1)} \cdots \mathbf{v}^{(i_N)}\mathbf{q}^{(N)}, \quad (6)$$

where $\mathbf{q}^{(n)}$ is the n-th free token fragment in the target sentence $\mathbf{y}$. We use $i_1, \cdots, i_N$ to denote the order of the constraints in $\mathbf{y}$. The n-th index i_n is not necessarily equal to n, since the order of the constraints in the target sentence $\mathbf{y}$ is often different from that in the source sentence $\mathbf{x}$.

We then abstract each fragment of text into nonterminals to build the template for lexically constrained translation. Concretely, the n-th free token fragment in the source sentence $\mathbf{x}$ is abstracted into X_n, for each $n \in \{0, \cdots, N\}$. The n-th free token fragment in the target sentence is abstracted into Y_n, for each $n \in \{0, \cdots, N\}$. In order to indicate the alignment between corresponding source and target constraints, we abstract $\mathbf{u}_n$ and $\mathbf{v}_n$ into the same nonterminal C_n. Note that X_n and Y_n are **not** linked nonterminals, since fragments of free tokens are not bilingually aligned. The resulting source- and target-side templates are given by

$$\begin{aligned} \mathbf{s} &= \mathrm{X}_0\mathrm{C}_1\mathrm{X}_1 \cdots \mathrm{C}_N\mathrm{X}_N, \\ \mathbf{t} &= \mathrm{Y}_0\mathrm{C}_{i_1}\mathrm{Y}_1 \cdots \mathrm{C}_{i_N}\mathrm{Y}_N. \end{aligned} \quad (7)$$

We need to define some derivation rules to convert the template into a natural language sentence. The derivation of nonterminals can be seen as the inverse of the abstraction process. Thus the derivation of the target-side template $\mathbf{t}$ would be

$$\begin{aligned} \mathrm{C}_n &\rightarrow \mathbf{v}^{(n)} \quad \text{for each } n \in \{1, \cdots, N\}, \\ \mathrm{Y}_n &\rightarrow \mathbf{q}^{(n)} \quad \text{for each } n \in \{0, \cdots, N\}. \end{aligned} \quad (8)$$

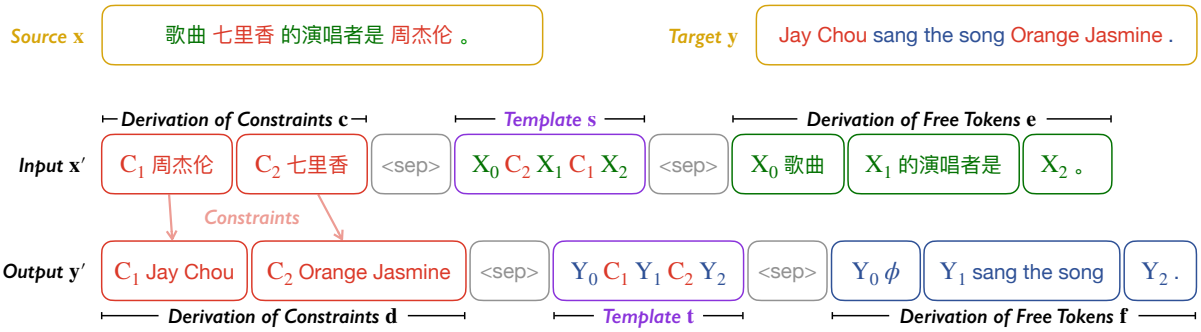

Figure 1: Example for lexically constrained translation. The constraints are ⟨周杰伦, Jay Chou⟩ and ⟨七里香, Orange Jasmine⟩. Note that X_n and Y_n are **not** linked nonterminals, since the source and target free token fragments are not necessarily aligned. The derivation rule $X_0 \rightarrow$ 歌曲 is learned through the concatenation of X_0 and 歌曲 (i.e., X_0 歌曲). "ϕ" denotes an empty string. See Section 3.2 for more details.

The derivation of the source-side template $\mathbf{s}$ can be defined similarly. Note that C_n produces the n-th source constraint $\mathbf{u}_n$ at the source side while producing the target constraint $\mathbf{v}_n$ at the target side. In order to make the derivation rules learnable by NMT models, we propose to use the concatenation of the nonterminal and the corresponding sequence of terminals to denote each derivation rule. For example, we use $Y_n\mathbf{q}^{(n)}$ to represent $Y_n \rightarrow \mathbf{q}^{(n)}$. We use $\mathbf{d}$ and $\mathbf{f}$ to denote the derivation of constraints and free tokens at the target side, respectively:

$$\begin{aligned} \mathbf{d} &= C_1\mathbf{v}^{(1)} \cdots C_N\mathbf{v}^{(N)}, \\ \mathbf{f} &= Y_0\mathbf{q}^{(0)} \cdots Y_N\mathbf{q}^{(N)}. \end{aligned} \tag{9}$$

At the source side, we use $\mathbf{c}$ and $\mathbf{e}$ to denote the derivation of constraints and free tokens, respectively. $\mathbf{c}$ and $\mathbf{e}$ can be defined similarly. Since the constraints are pre-specified by the users, the model only needs to learn the derivation of free tokens. To this end, we place the derivation of constraint-related nonterminals before the template as a conditional prefix. Then the model learns the generation of the template and the derivation of free tokens, step by step.

The final format of the input and output sequences at training time can be given by

$$\begin{aligned} \mathbf{x}' &= \mathbf{c}\ \texttt{<sep>}\ \mathbf{s}\ \texttt{<sep>}\ \mathbf{e}, \\ \mathbf{y}' &= \mathbf{d}\ \texttt{<sep>}\ \mathbf{t}\ \texttt{<sep>}\ \mathbf{f}, \end{aligned} \tag{10}$$

respectively. We use the delimiter <sep> to separate the template and the derivations. Figure 1 gives an example of both $\mathbf{x}'$ and $\mathbf{y}'$. At inference time, we feed $\mathbf{x}'$ to the encoder, and provide "$\mathbf{d}$ <sep>" to the decoder as the constrained prefix. Then the model generates the remaining part of $\mathbf{y}'$ (i.e., "$\mathbf{t}$ <sep> $\mathbf{f}$").

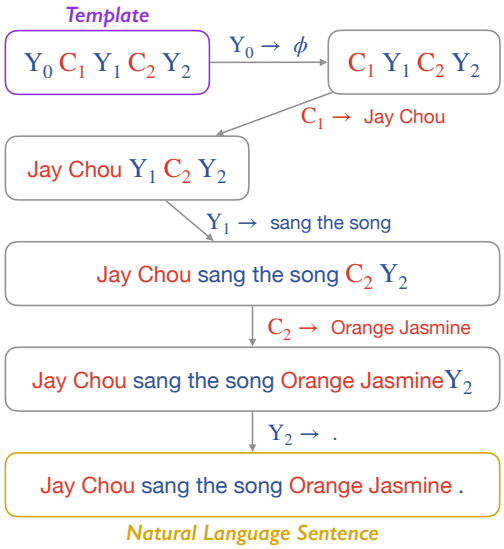

Figure 2: The template can be converted into a natural language sentence by replacing the nonterminals according to the corresponding derivation rules.

Figure 2 explains the way we convert the output sequence into a natural language sentence. The conversion from the template to the target-language sentence can be done through a simple script, and the computational cost caused by the conversion is negligible, compared with the model inference.

Note that we also abstract the constraints when building the template. The reason is that the model only needs to generate the order of constraints in this way, rather than copy all the specific tokens, which may suffer from copy failure (Chen et al., 2021). The formal representation for our lexically constrained model is slightly different from that defined in Eq. (4), which should be changed into

$$\begin{aligned} &P(\mathbf{t}, \mathbf{f}|\mathbf{c}, \mathbf{s}, \mathbf{e}, \mathbf{d}; \boldsymbol{\theta}) \\ =&P(\mathbf{t}|\mathbf{c}, \mathbf{s}, \mathbf{e}, \mathbf{d}; \boldsymbol{\theta}) \times P(\mathbf{f}|\mathbf{c}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{t}; \boldsymbol{\theta}). \end{aligned} \tag{11}$$

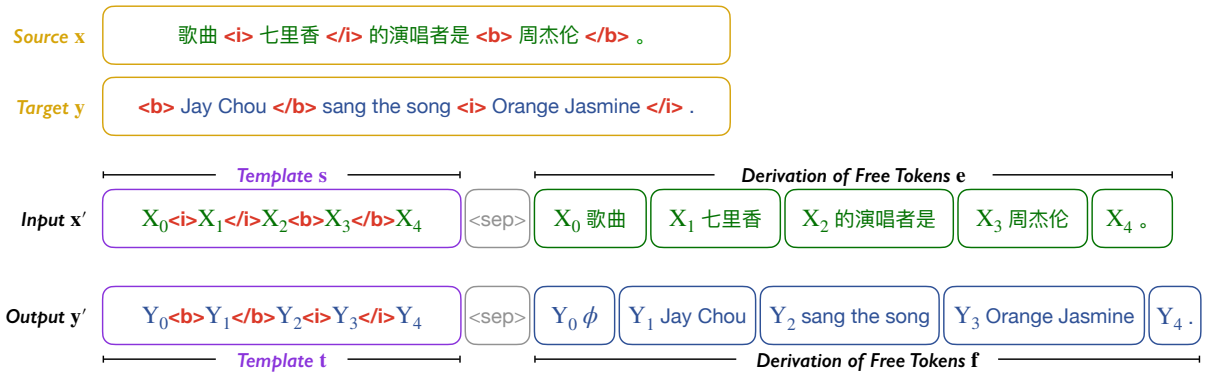

Figure 3: Example for structurally constrained translation. The markup tags are reserved in the template, while free tokens are abstracted. Note that X_n and Y_n are **not** linked nonterminals. See Section 3.3 for more details.

3.3 Template for Structural Constraints

The major challenge of structured text translation is to maintain the correctness of the structure, which is often indicated by markup tags (Hashimoto et al., 2019). The proposed framework can also deal with structurally constrained translation. Similarly, we replace free token fragments with nonterminals to build the template, where the markup tags are reserved. Figure 3 shows an example. Formally, given a sentence pair $\langle \mathbf{x}, \mathbf{y} \rangle$ with N markup tags, the source- and target-side templates are given by

$$\begin{aligned} \mathbf{s} &= \mathrm{X}_0\texttt{<tag}_1\texttt{>}\mathrm{X}_1 \cdots \texttt{<tag}_N\texttt{>}\mathrm{X}_N, \\ \mathbf{t} &= \mathrm{Y}_0\texttt{<tag}_{i_1}\texttt{>}\mathrm{Y}_1 \cdots \texttt{<tag}_{i_N}\texttt{>}\mathrm{Y}_N, \end{aligned} \tag{12}$$

respectively. The order of markup tags at the target side (i.e., $i_1 \cdots i_N$) may be different from that at the source side (i.e., $1 \cdots N$).

For each $n \in \{0, \cdots, N\}$, X_n can be derived into the n-th source-side free token fragment $\mathbf{p}^{(n)}$, and Y_n can be extended into the target-side free token fragment $\mathbf{q}^{(n)}$. X_n and Y_n are **not** linked. The derivation sequences can be defined as

$$\begin{aligned} \mathbf{e} &= \mathrm{X}_0\mathbf{p}^{(0)} \cdots \mathrm{X}_N\mathbf{p}^{(N)}, \\ \mathbf{f} &= \mathrm{Y}_0\mathbf{q}^{(0)} \cdots \mathrm{Y}_N\mathbf{q}^{(N)}. \end{aligned} \tag{13}$$

The format of the input and output would be

$$\begin{aligned} \mathbf{x}' &= \mathbf{s}\ \texttt{<sep>}\ \mathbf{e}, \\ \mathbf{y}' &= \mathbf{t}\ \texttt{<sep>}\ \mathbf{f}, \end{aligned} \tag{14}$$

respectively. Figure 3 illustrates an example for both $\mathbf{x}'$ and $\mathbf{y}'$. The formal representation of our structurally constrained model is the same as Eq. (4). The model arranges the markup tags when generating $\mathbf{t}$ and completes the whole sentence when generating $\mathbf{f}$, which is consistent with our motivation to decompose the whole task into constraint arrangement and free token generation.

4 Lexically Constrained Translation

4.1 Setup

Parallel Data We conduct experiments on two language pairs, including English-Chinese and English-German. For English-Chinese, we use the dataset of WMT17 as the training corpus, consisting of 20.6M sentence pairs. For English-German, the training data is from WMT20, containing 41.0M sentence pairs. We provide more details of data preprocessing in Appendix. Following recent studies on lexically constrained translation (Chen et al., 2021; Wang et al., 2022), we evaluate our method on human-annotated alignment test sets. For English-Chinese, both the validation and test sets are from Liu et al. (2005). For English-German, the test set is from Zenkel et al. (2020). We use `newstest2013` as the validation set, whose word alignment is annotated by `fast-align`[2]. The training sets are filtered to exclude test and validation sentences.

Lexical Constraints Following some recent works (Song et al., 2019; Chen et al., 2020, 2021; Wang et al., 2022), we simulate real-world lexically constrained translation scenarios by sampling constraints from the phrase table that are extracted from parallel sentence pairs based on word alignment. The script used to create the constraints is publicly available.[3] Specifically, the number of constraints for each sentence pair ranges between 0 and 3, and the length of each constraint ranges between 1 and 3 tokens. We use `fast-align` to build the alignment of the training data.

[2] https://github.com/clab/fast_align

[3] https://github.com/ghchen18/cdalign/blob/main/scripts/extract_phrase.py

Model Configuration We adopt Transformer (Vaswani et al., 2017) as our NMT model, which is optimized by Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. Please refer to Appendix for more details on the model configuration and the training process.

Baselines We compare our approach with the following six representative baselines:

- Placeholder (Crego et al., 2016): replacing constrained terms with placeholders;
- VDBA (Hu et al., 2019): modifying beam search to incorporate target-side constraints;
- Replace (Song et al., 2019): replacing source text with the corresponding target constraints;
- CDAlign (Chen et al., 2021): inserting target constraints based on word alignment;
- AttnVector (Wang et al., 2022): using attention keys and values to model constraints;
- TextInfill (Xiao et al., 2022): filling free tokens through a bilingual text-infilling task.

Evaluation Metrics We follow Alam et al. (2021a) to use the following four metrics to make a thorough comparison of the involved methods:

- BLEU (Papineni et al., 2001): measuring the translation quality of the whole sentence;
- Exact Match: indicating the accuracy that the source constraints in the input sentences are translated into the provided target constraints;
- Window Overlap: quantifying the overlap ratio between the hypothesis and the reference windows for each matched target constraint, indicating if this constraint is placed in a suitable context. The window size is set to 2.
- 1-TERm: modifying TER (Snover et al., 2006) by setting the edit cost of constrained tokens to 2 and the cost of free tokens to 1.

We use sacreBLEU[4] (Post, 2018) to estimate the BLEU score, and adapt the scripts released by Alam et al. (2021a) for the other three metrics.

[4]English-Chinese: nrefs:1 | case:mixed | eff:no | tok:zh | smooth:exp | version:2.0.0. English-German: nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp | version:2.0.0.

4.2 Main Results

Template Accuracy We firstly examine the performance of the model in the template generation sub-task before investigating the translation performance. We compare the target-side template extracted from the reference sentence and the one generated by the model to calculate the accuracy of template generation. Formally, if the reference template $\mathbf{t}$ is $\mathrm{Y}_0\mathrm{C}_{i_1}\mathrm{Y}_1 \cdots \mathrm{C}_{i_N}\mathrm{Y}_N$, the generated template $\hat{\mathbf{t}}$ is correct if

- $\hat{\mathbf{t}} = \mathrm{Y}_0\mathrm{C}_{j_1}\mathrm{Y}_1 \cdots \mathrm{C}_{j_N}\mathrm{Y}_N$;
- the set $\{j_1, \cdots, j_N\}$ equals $\{i_1, \cdots, i_N\}$.

In other words, the model must generate all the nonterminals to guarantee the presence of the provided constraints. However, the order of constraint-related nonterminals can be flexible since there often exist various suitable orders for the provided constraints. In both English-Chinese and English-German, the template accuracy of our model is 100%. An interesting finding is that our model learns to reorder the constraints according to the style of the target language. We provide an example of constraint reordering in Table 1.

When generating the free token derivation $\mathbf{f}$, the model can recall all the nonterminals (i.e., Y_n) presented in the template $\mathbf{t}$ in English-Chinese. In English-German, however, the model omits one free token nonterminal, of which the frequency is 0.2%. We use empty strings for the omitted nonterminals when reconstructing the output sentence.

Translation Performance Table 2 shows the results of lexically constrained translation, demonstrating that all the investigated methods can recall more provided constraints than the unconstrained Transformer model. Our approach can improve the BLEU score over the involved baselines. This improvement potentially comes from two aspects: (1) our system outputs can match more pre-specified constraints compared to some baselines, such as AttnVector (Wang et al., 2022) (100% vs. 93.8%) ; (2) our method can place more constraints in appropriate context, which can be measured by window overlap. The exact match accuracy of VDBA (Hu et al., 2019) is lower than 100% due to the out-of-vocabulary problem in English-Chinese.

TextInfill (Xiao et al., 2022) and our approach can achieve 100% exact match accuracy in both the two language pairs. However, TextInfill can only place the constraints in the pre-specified order,

Constraints	⟨slowing down,减弱⟩; ⟨price hike,价格上涨⟩
Source	Analysts are concerned that since there is no sign yet of any slowing down of this price hike , the prospect of the British real estate market as where it is heading now is far from optimistic.
Reference	分析家担心, 由于目前还看不见 价格上涨 趋势有 减弱 的迹象, 照此发展下去, 英国房地产市场前景堪忧。
Input (enc)	C_1 slowing down C_2 price hike <sep> X_0 C_1 X_1 C_2 X_2 <sep> X_0 Analysts are concerned that since there is no sign yet of any X_1 of this X_2 , the prospect of the British real estate market as where it is heading now is far from optimistic.
Prefix (dec)	C_1 减弱 C_2 价格上涨 <sep>
Output	Y_0 C_2 Y_1 C_1 Y_2 <sep> Y_0 分析师们担心, 由于目前还没有迹象显示 Y_1 会 Y_2 , 英国房地产市场的前景远不乐观。
Result	分析师们担心, 由于目前还没有迹象显示 价格上涨 会 减弱 , 英国房地产市场的前景远不乐观。

Table 1: An example of our method. We replace the nonterminals in the template using the derivation rules to reconstruct the final result (i.e., "**Result**"). Surprisingly, we find that our model can automatically sort the provided constraints when generating the template. In this example, C_1 is before C_2 in the source-side template. But in the target-side template generated by our model, C_2 is before C_1, which is more suitable for the target language.

Method	**BLEU**	**Exact Match**	**Window Overlap**	**1-TERm**	**BLEU**	**Exact Match**	**Window Overlap**	**1-TERm**
Direction	*English-Chinese*				*English-German*			
Vanilla	42.7	10.1	4.8	35.7	24.8	10.0	8.1	39.2
Placeholder	46.6	99.4	33.9	41.5	27.2	**100.0**	29.4	44.6
VDBA	45.8	99.6	33.4	41.7	29.0	**100.0**	31.1	45.1
Replace	46.4	93.8	35.5	40.7	31.1	96.6	35.7	<u>48.3</u>
CDAlign	46.2	92.1	31.7	41.6	29.7	95.9	32.3	46.3
AttnVector	<u>46.9</u>	93.8	<u>35.8</u>	<u>42.4</u>	<u>31.3</u>	97.5	<u>37.2</u>	47.9
TextInfill	45.6	**100.0**	32.8	39.9	30.7	**100.0**	35.5	47.1
Ours	**47.5**	**100.0**	**36.9**	**43.1**	**32.3**	**100.0**	**38.5**	**49.8**

Table 2: Results of the lexically constrained translation task for both English-Chinese and English-German. For clarity, we highlight the **highest** score in bold and the <u>second-highest</u> score with underlines.

while our approach can automatically reorder the constraints. As a result, the window overlap score of our approach is higher than TextInfill. Please refer to Table 8 in Appendix for more translation examples of both our method and some baselines

4.3 Unconstrained Translation

A concern for lexically constrained translation methods is that they may cause poor translation quality in unconstrained translation scenarios. We thus evaluate our approach in the standard translation task, where the model is only provided with the source sentence $\mathbf{x}$. Under this circumstance, the input and output can be given by

$$\begin{aligned} \mathbf{x}' &= \phi \texttt{ <sep> } X_0 \texttt{ <sep> } X_0\mathbf{x}, \\ \mathbf{y}' &= \phi \texttt{ <sep> } Y_0 \texttt{ <sep> } Y_0\mathbf{y}, \end{aligned} \tag{15}$$

respectively. The BLEU scores of our method are 42.6 and 25.0 for English-Chinese and English-German, respectively. The performance of our method is comparable with the vanilla model, which can dispel the concern that our approach may worsen the unconstrained translation quality.

4.4 Inference Speed

Methods	**Speed**
Vanilla	3392 tokens per second
Ours	3390 tokens per second

Table 3: Inference speed of our method and the vanilla model on the English-Chinese validation set.

Table 3 shows the decoding speed. Since we did not change the model architecture and the decoding algorithm, the speed of our method is close to the vanilla Transformer model (Vaswani et al., 2017). Although our speed is almost the same as the vanilla model, our inference time is a bit longer, given the fact that the output sequence $\mathbf{y}'$ is longer than the original target-language sentence $\mathbf{y}$.

Method	BLEU	Structure Accuracy		BLEU	Structure Accuracy	
		Correct	Match		Correct	Match
Direction	*English-French*			*English-Russian*		
Remove	31.4	n/a	n/a	21.0	n/a	n/a
Split-Inject	66.1	**100.00**	**100.00**	43.1	**100.00**	**99.85**
XML	65.3	99.55	99.30	44.9	99.45	98.90
Ours	**67.3**	**100.00**	**100.00**	**45.8**	**100.00**	99.80
Direction	*English-Chinese*			*English-German*		
Remove	31.5	n/a	n/a	25.7	n/a	n/a
Split-Inject	57.0	**100.00**	99.30	50.7	**100.00**	**99.80**
XML	61.2	99.85	99.75	52.7	99.80	99.20
Ours	**61.5**	**100.00**	**99.80**	**53.6**	**100.00**	**99.80**

Table 4: Results of the structurally constrained translation task. We highlight the **highest** score in bold and the second-highest score with underlines.

4.5 Effect of Data Scale

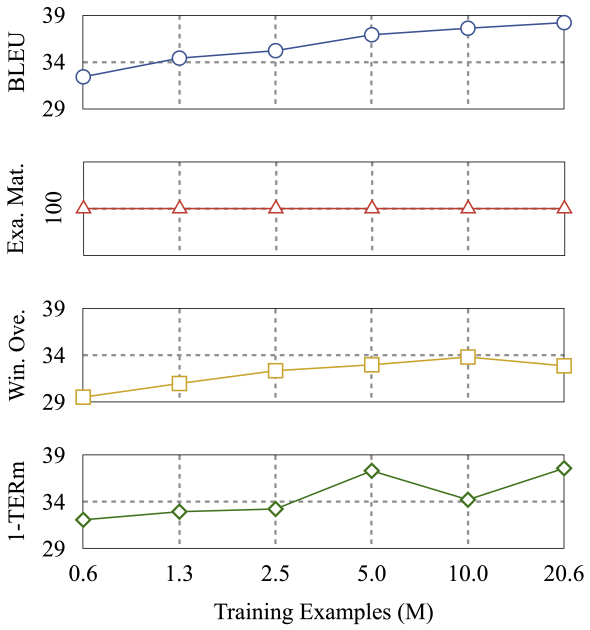

Figure 4: Effect of data scale. The results are reported on the English-Chinese validation set.

We vary the amounts of training data to investigate the effect of data scale on our approach. Figure 4 shows the results. The BLEU score increases with the data size, while the window overlap score reaches the highest value when using 10.0M training examples. When using all the training data, the 1 - TERm metric achieves the best value. We find that the exact match accuracy of our method is maintained at 100%, even with only 0.6M training examples. This trend implies that our method can be applied in some low-resource scenarios.

4.6 More Analysis

Due to space limitation, we place a more detailed analysis of our approach in Appendix, including the effect of the alignment model, the performance on more language pairs, and the domain robustness of our model, which is evaluated on the WMT21 terminology translation task (Alam et al., 2021b) that lies in the COVID-19 domain.

5 Structurally Constrained Translation

5.1 Setup

Data We conduct our experiments on the dataset released by Hashimoto et al. (2019), which supports the translation from English to seven other languages. We select four languages, including French, Russian, Chinese, and German. For each language pair, the training set contains roughly 100K sentence pairs. We report the results on the validation sets since the test sets are not open-sourced. We follow Hashimoto et al. (2019) to use `SentencePiece`[5] to preprocess the data, which supports user-defined special symbols. The model type of `SentencePiece` is set to `unigram`, and the vocabulary size is set to 9000. For English-Chinese, we over-sample the English sentences when learning the joint tokenizer, since Chinese has more unique characters than English (Hashimoto et al., 2019). We did not perform over-sampling for other language pairs. We register the XML tags and URL placeholders as user-defined special symbols. In addition, we also register `&`, `<`, and `>` as special tokens, following Hashimoto et al. (2019).

[5] https://github.com/google/sentencepiece

Model Configuration Since the data scale for structurally constrained translation is much smaller than lexically constrained translation, we follow Hashimoto et al. (2019) to set the width of the model to 256 and the depth of the model to 6. See Section B.1 in Appendix for more details.

Baselines We compare our approach with the following three baselines:

- Remove: removing the markup tags and only translating the plain text;
- Split-Inject (Al-Anzi et al., 1997): splitting the input sentence based on the markup tags and then translating each text fragment independently, and finally injecting the tags;
- XML (Hashimoto et al., 2019): directly learning the NMT model end-to-end using parallel sentences with XML tags.

Evaluation Metrics We follow Hashimoto et al. (2019) to use the following metrics:

- BLEU: considering the structure when estimating BLEU score (Papineni et al., 2001);
- Structure Accuracy: utilizing the `etree` package to check if the system output is a valid XML structure (i.e., Correct), and if the output structure exactly matches the structure of the given reference (i.e., Match).

All the metrics are calculated using the evaluation script released by Hashimoto et al. (2019).

5.2 Main Results

Template Accuracy We firstly examine the accuracy of the generated templates. A generated template is correct if

- the template is a valid XML structure;
- the template recalls all the markup tags of the input sentence.

The template accuracy of our method is 100% in all the four language pairs. Similar to lexically constrained translation, the model may omit some free token nonterminals (i.e., Y_n) when generating the derivation $\mathbf{f}$, of which the ratios are 0.4%, 0.6%, 0.1%, 0.9% in English-French, English-Russian, English-Chinese, English-German, respectively. We use empty strings for the omitted nonterminals when reconstructing the output sentence.

Translation Performance Table 4 shows the results of all the involved methods. Our approach can improve the BLEU score over the three baselines, and the structure correctness is 100%. Although Split-Inject can also guarantee the correctness of the output, its BLEU score is much lower, which is potentially caused by the reason that some fragments are translated without essential context. The structure match accuracy with respect to the given reference is not necessarily 100%, since the order of markup tags can be diverse due to the variety of natural language. See Table 9 in Appendix for some translation examples.

6 Conclusion

In this work, we propose a template-based framework for constrained translation and apply the framework to two specific tasks, which are lexically and structurally constrained translation. Our motivation is to decompose the generation of the whole sequence into the arrangement of constraints and the generation of free tokens, which can be learned through a sequence-to-sequence framework. Experiments demonstrate that the proposed method can achieve high translation quality and match accuracy simultaneously and our inference speed is comparable with unconstrained NMT baselines.

Limitations

A limitation of this work is that our method can not cope with one-to-many constraints (e.g., ⟨bank, 河岸|银行⟩). Moreover, we only validate the proposed template-based framework in machine translation tasks. However, constrained sequence generation is vital in many other NLP tasks, such as table-to-text generation (Parikh et al., 2020), text summarization (Liu et al., 2018), and text generation (Dathathri et al., 2020). In the future, we will apply the proposed method to more constrained sequence generation tasks.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61925601, No. 62006138), the National Social Science Fund of China (No. 20&ZD279), Beijing Academy of Artificial Intelligence (BAAI), a grant from the Guoqiang Institute, Tsinghua University, and the Tencent AI Lab Rhino Bird Focused Research Program (No. JR202031). We thank all the reviewers for their valuable and insightful comments.

References

F Al-Anzi, K Al-Zame, M Husain, and H Al-Mutairi. 1997. Automatic english/arabic html home page translation tool. In *Proc. 1st Workshop Technol. Arabizing Internet*.

Md Mahfuz Ibn Alam, Antonios Anastasopoulos, Laurent Besacier, James Cross, Matthias Gallé, Philipp Koehn, and Vassilina Nikoulina. 2021a. On the evaluation of machine translation for terminology consistency. *CoRR*, abs/2106.11891.

Md Mahfuz Ibn Alam, Ivana Kvapilíková, Antonios Anastasopoulos, Laurent Besacier, Georgiana Dinu, Marcello Federico, Matthias Gallé, Kweonwoo Jung, Philipp Koehn, and Vassilina Nikoulina. 2021b. Findings of the WMT shared task on machine translation using terminologies. In *Proceedings of the Sixth Conference on Machine Translation*, pages 652–663.

Toms Bergmanis and Mārcis Pinnis. 2021. Facilitating terminology translation with target lemma annotations. In *Proceedings of EACL 2021*.

Guanhua Chen, Yun Chen, and Victor O.K. Li. 2021. Lexically constrained neural machine translation with explicit alignment guidance. In *Proceedings of AAAI 2021*.

Guanhua Chen, Yun Chen, Yong Wang, and Victor O.K. Li. 2020. Lexical-constraint-aware neural machine translation via data augmentation. In *Proceedings of IJCAI 2020*.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurélien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. Systran's pure neural machine translation systems. *CoRR*, abs/1610.05540.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proceedings of ICLR 2020*.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of ACL 2019*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Proceedings of NeurIPS 2019*.

Greg Hanneman and Georgiana Dinu. 2020. How should markup tags be translated? In *Proceedings of the Fifth Conference on Machine Translation*.

Kazuma Hashimoto, Raffaella Buschiazzo, James Bradbury, Teresa Marshall, Richard Socher, and Caiming Xiong. 2019. A high-quality multilingual dataset for structured documentation translation. In *Proceedings of the Fourth Conference on Machine Translation*.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of NAACL 2018*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of ACL 2017*.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of NAACL 2019*.

Eric Joanis, Darlene Stewart, Samuel Larkin, and Roland Kuhn. 2013. Transferring markup tags in statistical machine translation: a two-stream approach. In *Proceedings of the 2nd Workshop on Post-editing Technology and Practice*.

Josef Jon, João Paulo Aires, Dusan Varis, and Ondřej Bojar. 2021. End-to-end lexically constrained machine translation for morphologically rich languages. In *Proceedings of ACL-IJCNLP 2021*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn. 2009. A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007*.

Huayang Li, Guoping Huang, Deng Cai, and Lemao Liu. 2020. Neural machine translation with noisy lexical constraints. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1864–1874.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *Proceedings of AAAI 2018*.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL 2005*.

Elise Michon, Josep Crego, and Jean Senellart. 2020. Integrating domain terminology into neural machine translation. In *Proceedings of COLING 2020*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of NAACL 2019*.

Mathias Müller, Annette Rios, and Rico Sennrich. 2020. Domain robustness in neural machine translation. In *Proceedings of AMTA 2020*.

Jan Niehues. 2021. Continuous learning in neural machine translation using bilingual dictionaries. In *Proceedings of EACL 2021*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2001*.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP 2020*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of NAACL 2018*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA 2006*, pages 223–231.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing NMT with pre-specified translation. In *Proceedings of NAACL 2019*.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with Levenshtein transformer. In *Proceedings of ACL 2020*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS 2017*.

Shuo Wang, Zhixing Tan, and Yang Liu. 2022. Integrating vectorized lexical constraints for neural machine translation. In *Proceedings of ACL 2022*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proceedings of ICLR 2019*.

Yanling Xiao, Lemao Liu, Guoping Huang, Qu Cui, Shujian Huang, Shuming Shi, and Jiajun Chen. 2022. Bitiimt: A bilingual text infilling method for interactive machine translation. In *Proceedings of ACL 2022*.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2020. End-to-end neural word alignment outperforms GIZA++. In *Proceedings of ACL 2020*.

Hao Zhang, Richard Sproat, Axel H. Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark. 2019. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2).

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, and Yang Liu. 2021. Neural machine translation with explicit phrase alignment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1001–1010.

A Supplementary Material for Lexically Constrained Translation

A.1 More Details on Data

For the lexically constrained translation task, Chinese sentences are segmented by `Jieba`[6], while English and German sentences are tokenized using Moses (Koehn et al., 2007). The tokenized sentences are then processed by BPE (Sennrich et al., 2016) with 32K merge operations for both the two language pairs. We detokenize the model outputs before calculating the sacreBLEU.

A.2 More Details on Model

We adopt Transformer (Vaswani et al., 2017) as our NMT model. For English-Chinese, we use the base model, whose depth is 6, and the width is 512. For English-German, we use the big model, whose depth is 6, and the width is 1024. The base and big models are optimized using the corresponding learning schedules introduced in Vaswani et al. (2017). We train base models for 200K iterations using 4 NVIDIA V100 GPUs and train big models for 300K iterations using 8 NVIDIA V100 GPUs. Each mini-batch contains approximately 32K tokens in total. All the models are optimized using Adam (Kingma and Ba, 2015), with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. In all experiments, both the dropout rate and the label smoothing penalty are set to 0.1. The beam size is set to 4.

A.3 Effect of Alignment Model

In this work, we use an alignment model to produce word alignments for the training set, which is then used for phrase table extraction. By default, we use all the parallel data in the training set to train the alignment model, using the `fast-align` toolkit. To better understand the effect of the alignment model, we replace the default alignment model with a weaker one that is trained using only 0.1M sentence pairs. Table 5 shows the result, from which we find that using the weaker word alignment can negatively affect the BLEU score. However, the exact match accuracy is still 100%, and changes in the other two metrics are modest.

A.4 Domain Robustness

Domain robustness is about the generalization of machine learning models to unseen test domains (Müller et al., 2020). In our experiments, all the involved models are trained in the news domain. We evaluate the domain robustness of these methods on the WMT21 terminology translation task (Alam et al., 2021b)[7] , which lies in the COVID-19 domain. Since this task does not support English-German translation, we only conduct this experiment on English-Chinese. In this test set, the maximum number of constraints is 12. We thus modify the phrase extraction script to increase the maximum number of constraints from 3 to 12, and then re-train both the baselines and our models. Note that we only change the number of constraints, while the training domain is still news. Since the open-sourced implementation of AttnVector (Wang et al., 2022)[8] does not support more than 3 constraints, we omit this baseline in this experiment. The test set of the WMT21 terminology translation task also contains some constraints that consist of more than one target term (i.e., one-to-many constraints). We only select the one that appear in the reference as our constraint. We leave it to future work to extend the current framework for one-to-many constraints.

Table 6 provides the results on the COVID-19 domain, where our approach performs best across all the four evaluation metrics. VDBA (Hu et al., 2019) and our method can both maintain the exact match accuracy, while the other three baselines

# Sent.	BLEU	Exact Match	Window Overlap	1-TERm
0.1M	37.5	100.0	32.7	37.5
20.6M	38.2	100.0	32.9	37.6

Table 5: Effect of the alignment model on the English-Chinese validation set. "# Sent." means the number of sentence pairs used to train the alignment model.

Method	BLEU	Exa. Mat.	Win. Ove.	1 - T.m
Vanilla	37.7	58.1	19.4	37.9
Placeholder	38.5	98.9	24.4	38.8
VDBA	38.0	**100.0**	24.3	39.1
Replace	38.4	87.3	24.5	39.7
CDAlign	38.6	89.3	24.0	40.5
TextInfill	38.7	97.0	23.2	38.4
Ours	**39.6**	**100.0**	**26.3**	**41.3**

Table 6: Results on the English-Chinese test set of the WMT21 terminology translation.

[6] https://github.com/fxshy/jieba

[7] https://www.statmt.org/wmt21/terminology-task.html

[8] https://github.com/shuo-git/VecConstNMT

Method	BLEU	Exa. Mat.	Win. Ove.	1 - T.m
Chinese-English				
Vanilla	23.3	17.6	10.4	36.6
AttnVector	25.9	95.5	35.5	42.1
TextInfill	25.0	**100.0**	33.3	39.0
Ours	**26.7**	**100.0**	**37.3**	**45.1**
German-English				
Vanilla	32.4	9.5	7.3	45.8
AttnVector	37.8	91.4	36.4	53.3
TextInfill	37.2	**100.0**	37.1	51.4
Ours	**38.8**	**100.0**	**39.7**	**53.4**

Table 7: Results of the lexically constrained translation task in Chinese-English and German-English.

achieve much lower exact match accuracy due to the domain shift. However, the BLEU score of VDBA is lower than other constrained translation approaches, while our method can also achieve the best BLEU score. The exact match accuracy of TextInfill (Xiao et al., 2022) is lower than 100% because sometimes the model can not generate all the slots within the length limitation. The results indicate that our approach can better cope with constraints coming from unseen domains.

A.5 X-English Translation

We also conduct experiments on X-English translation directions (i.e., Chinese-English and German-English). Due to the limitation of computational resources, we only train the two most recent baselines: AttnVector (Wang et al., 2022) and TextInfill (Xiao et al., 2022). Moreover, AttnVector and TextInfill achieve the best BLEU score and exact match accuracy, excluding our approach, respectively. As shown in Table 7, we find that our approach performs well in both Chinese-English and German-English, achieving 100% exact match accuracy and a better BLEU score.

A.6 Case Study

As mentioned in Section 4.2, our approach outperforms the baselines in the lexically constrained translation task. To better understand the difference between our approach and some representative baselines, we list some examples in Table 8.

B Supplementary Material for Structurally Constrained Translation

B.1 More Details on Model

All the models are trained for 40K iterations in all the four translation directions. We adopt the cosine learning rate schedule presented in Wu et al. (2019), but we set the maximum learning rate to 7×10^{-4} and the warmup step to 8K. The period of the cosine function is set to 32K, which means that the learning rate decays into the minimum value at the end of the training. Both the dropout rate and the label smoothing penalty are set to 0.2. Each mini-batch consists of approximately 32k tokens in total. We use Adam (Kingma and Ba, 2015) for model optimization, with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We also set the weight decay coefficient to 10^{-3}. Both the baseline models and our models are trained using the same hyperparameters.

B.2 Case Study

We list some translation examples in Table 9 to provide a detailed understanding of our work. The examples demonstrate that our approach can effectively cope with structured inputs.

Constraints	⟨guests ,来宾 ⟩; ⟨culinary culture ,食品文化 ⟩; ⟨Chinese-style ,中式 ⟩
Source	Wang Kaiwen , Chinese ambassador to Latvia , introduced to the guests a few major styles of cooking in Chinese gourmet foods and expressed his hope that through tasting Chinese-style gourmet foods more will be learned about China and Chinese culinary culture.
Reference	中国驻拉脱维亚大使王开文向 来宾 们介绍了中国美食的几大菜系, 表示希望通过品尝 中式 美味食品更多了解中国和中国 食品文化 。
AttnVector	中国驻拉托维亚大使王开文向 来宾 介绍了中国美食食品的几种主要烹饪方式, 并表示希望通过品尝 中式 美食, 更多地了解中国和中国的文化。
TextInfill	中国驻拉脱维亚大使王开文向 来宾 介绍了几种主要的中国美食 食品文化 , 并表示希望通过品尝 中式 美食, 能够了解更多关于中国和中国烹饪文化的知识。
Ours	中国驻拉脱维亚大使王开文向 来宾 介绍了中国美食的几种主要烹饪风格, 并表示希望通过品尝 中式 美食, 更多地了解中国和中国的 食品文化 。
Constraints	⟨Italian engineer,义大利工程师⟩; ⟨Gidzenko,吉曾柯⟩; ⟨Shuttleworth,夏特沃斯⟩
Source	Returning together with Shuttleworth to earth are the Russian spacecraft commander Gidzenko and the Italian engineer Vittori who entered space with him.
Reference	与 夏特沃斯 一同返回地球的, 是这次和他一起进入太空的俄罗斯太空船指挥官 吉曾柯 与 义大利工程师 维托利。
AttnVector	吉曾柯 和 义大利工程师 维托利与 夏特沃斯 一同返回地球, 他们一同进入太空。
TextInfill	俄罗斯太空船指挥官吉登科(Gidzenko)和 义大利工程师 吉曾柯 (Vittori)与 夏特沃斯 一起重返地球。
Ours	与 夏特沃斯 一起返回地球的是俄罗斯航天器指挥官 吉曾柯 和与他一同进入太空的 义大利工程师 维托里。

Table 8: Examples for lexically constrained translation. For clarity, we only list the results of two representative baselines. We choose AttnVector (Wang et al., 2022) and TextInfill (Xiao et al., 2022) since they achieve the best BLEU score and the highest exact match accuracy, respectively, excluding our approach. In the first example, AttnVector omits the target constraint 食品文化 in its output, while both TextInfill and our approach can generate all the three constraints. In the second example, TextInfill places the constraint 吉曾柯 in the wrong context, while our approach outputs a better result.

Source	... <ph> Each dashboard can have up to <ph> 3 </ph> filters. Contact <ph> Salesforce </ph> to increase the filter options limit in <ph> Salesforce Classic </ph> . A maximum of <ph> 50 </ph> filter options is possible. </ph>
Reference	... <ph> Chaque tableau de bord peut inclure jusqu'á <ph> 3 </ph> filtres. Pour augmenter les limitations des options de filtrage dans <ph> Salesforce Classic </ph> , contactez <ph> Salesforce </ph> . <ph> 50 </ph> options defiltre sont possibles au maximum. </ph>
Split-Inject	... <ph> Chaque tableau de bord peut avoir jusqu'á <ph> 3 </ph> filtres. Contact <ph> Salesforce </ph> pour accroître la limitation des options de filtrage <ph> Salesforce Classic </ph> . maximum d'un maximum <ph> 50 </ph> Les options de filtrage sont possibles. </ph> L
XML	... <ph> Chaque tableau de bord peut avoir jusqu'á <ph> 3 </ph> filtres. Pour augmenter la limitation en options de filtrage dans <ph> Salesforce Classic </ph> , chaque filtre peut inclure jusqu'á <ph> 50 </ph> options de filtrage. </ph>
Ours	... <ph> Chaque tableau de bord peut avoir jusqu'á <ph> 3 </ph> filtres. Contactez <ph> Salesforce </ph> pour augmenter les options de limitation de filtrage dans <ph> Salesforce Classic </ph> . Un maximum de <ph> 50 </ph> options de filtrage est possible. </ph>
Source	Each <ph> Event Monitoring app </ph> user needs an <ph> Event Monitoring Analytics Apps </ph> permission set license. The <ph> Event Monitoring Analytics Apps </ph> permission set license enables the following permissions.
Reference	Chaque utilisateur de l' <ph> application Event Monitoring </ph> doit disposer d'une licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> . La licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> accorde les autorisations ci-dessous.
Split-Inject	Chaque <ph> Application Event Monitoring </ph> utilisateur doit avoir un utilisateur <ph> Applications Event Monitoring Analytics </ph> Licence d'ensemble d'autorisations. <ph> Applications Event Monitoring Analytics </ph> La licence d'ensemble d'autorisations active les autorisations ci-dessous.
XML	Chaque utilisateur de l' <ph> application Event Monitoring </ph> doit disposer d'une licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> . La licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> active les autorisations ci-dessous.
Ours	Chaque utilisateur de l' <ph> application Event Monitoring </ph> doit disposer d'une licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> . La licence d'ensemble d'autorisations <ph> Event Monitoring Analytics Apps </ph> active les autorisations suivantes.

Table 9: Examples for structurally constrained translation. We only highlight some text fragments wrapped by markup tags to show the difference between the involved methods. In the first example, XML (Hashimoto et al., 2019) omits the fragment <ph> Salesforce </ph>, while Split-Inject and our method recall all the markup tags of the source sentence. In the second example, the colored contents are mistranslated by Split-Inject, which is potentially caused by the lack of context when translating these fragments.