Lexical-Constraint-Aware Neural Machine Translation via Data Augmentation

Guanhua Chen[1] , **Yun Chen**[*2] , **Yong Wang**[1] and **Victor O.K. Li**[1]
[1]The University of Hong Kong
[2]Shanghai University of Finance and Economics
{ghchen, wangyong, vli}@eee.hku.hk, yunchen@sufe.edu.cn

Abstract

Leveraging lexical constraint is extremely significant in domain-specific machine translation and interactive machine translation. Previous studies mainly focus on extending beam search algorithm or augmenting the training corpus by replacing source phrases with the corresponding target translation. These methods either suffer from the heavy computation cost during inference or depend on the quality of the bilingual dictionary pre-specified by the user or constructed with statistical machine translation. In response to these problems, we present a conceptually simple and empirically effective data augmentation approach in lexical constrained neural machine translation. Specifically, we construct constraint-aware training data by first randomly sampling the phrases of the reference as constraints, and then packing them together into the source sentence with a separation symbol. Extensive experiments on several language pairs demonstrate that our approach achieves superior translation results over the existing systems, improving translation of constrained sentences without hurting the unconstrained ones.

1 Introduction

Lexically constrained translation [Hokamp and Liu, 2017; Post and Vilar, 2018; Luong *et al.*, 2015; Song *et al.*, 2019], the task of imposing pre-specified words and phrases in the translation output (see Figure 1), has practical significance in many applications. These include domain-specific machine translation, where lexicons can be a domain terminology extracted from an in-domain dictionary [Arthur *et al.*, 2016], and interactive machine translation, where lexicons can be provided by humans after reading a system's initial output [Koehn, 2009; Cheng *et al.*, 2016]. In phrase-based statistical machine translation [Koehn *et al.*, 2003], it is relatively easy to restore these kinds of manual interventions. However, in the paradigm of neural machine translation (NMT) [Bahdanau *et al.*, 2015; Vaswani *et al.*, 2017], the task of lexically constrained translation is not trivial.

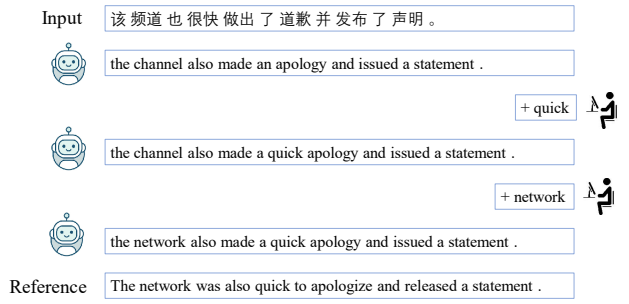


Figure 1: A simple example of lexically constrained machine translation from Chinese to English. The first translation is unconstrained, whereas the second and third have one and two additional constraint imposed.

As a result, a number of authors have explored methods for lexically constrained NMT. These methods can be roughly divided into two broad categories: *hard* and *soft*. In the *hard* category, all constraints are ensured to appear in the output sentence. They achieve this by designing novel decoding algorithms, without modification to the NMT model or the training process. Hokamp and Liu [2017] propose the grid beam search (GBS) decoding algorithm. Post and Vilar [2018] speed up over GBS by presenting the dynamical beam allocation (DBA) algorithm. However, the computation complexity of such decoding algorithms is still much higher compared with conventional beam search.

Another direction achieves lexically constrained translation by modification to the NMT model's training process. These methods are "soft", i.e., they cannot ensure all constraints to appear in the translation output. Luong *et al.* [2015] use placeholder tags to substitute rare words on both source and target sides according to a bilingual dictionary during training. The model then learns to translate constrained words by translating placeholder tags. Song *et al.* [2019]; Dinu *et al.* [2019]; Wang *et al.* [2019] propose a data augmentation method to train the NMT model. They construct synthetic parallel sentences by either replacing the corresponding source words with the constraint or appending the constraint right after the corresponding source words. However, a bilingual dictionary is essential at both training and inference time. Therefore, their performance relies heavily on the

*Corresponding author

quality of the bilingual dictionary, and they can not translate with non-consecutive constraint, i.e., the constraint that corresponds to non-consecutive source words.

In this paper, we propose a LExical-Constraint-Aware (LeCA) NMT model by packing constraints and source sentence together without using a bilingual dictionary. During training, we sample pseudo constraints from the reference and construct constraint-aware synthetic parallel corpus by appending the constraints after the source sentence with a separation symbol. The motivation is to make the model learn to utilize the lexicon constraints automatically without the prespecified aligned source words. During inference, the source is similarly modified as a preprocessing step. By training on a mixture of the original and synthetic corpus, the model can perform well on the constrained case while maintaining the performance on the unconstrained case. We evaluate the proposed model on WMT De-En and NIST Zh-En News translation in both directions with different types of lexical constraints. Similar with the previous work [Song *et al.*, 2019; Dinu *et al.*, 2019], our approach can not guarantee all constraints to be generated in the output, but experiments show that the copy success rate is high: it is $96.4\% \sim 99.5\%$ for the WMT De-En task and $89.6\% \sim 98.4\%$ for the NIST Zh-En task. In addition, our model improves over the *code-switching* baselines for all constraint types, and the improvement is more than 3.5 BLEU points for *reference constraint* and *interactive constraint*.[1]

2 Related Work

Recent work on lexically constrained NMT can be loosely clustered into two categories: *hard* and *soft*. The *hard* ensures all constraints to appear at the translation output. In contrast, the *soft* category cannot make such guarantee.

Hard lexically constrained translation. Hokamp and Liu [2017] propose the grid beam search (GBS) algorithm for incorporating lexical constraints at decoding time. The constraints are forced to be present in the translations. Post and Vilar [2018] propose a faster algorithm over grid beam search, namely, dynamical beam allocation (DBA). The decoding complexity is reduced from $\mathcal{O}(|T|kN_C)$ to $\mathcal{O}(|T|k)$ ($|T|$ is the target sentence length, k is beam size, N_C is the number of constraints) by grouping together hypotheses with the same number of constraints into banks and dynamically dividing a fixed-size beam across these banks at each time step. One problem of these methods is that they copy the lexicon constraints in exactly the same form to the output, making it unsuitable to decode with noisy constraints. For example, constraints with incorrect morphological form. Another drawback is that the decoding speed is significantly reduced compared with standard beam search.

Soft lexically constrained translation. Song *et al.* [2019] create a synthetic code-switching corpus to augment the training data for NMT. The code-switching corpus is built by replacing the corresponding source phrase with the target constraint according to a bilingual dictionary. By training on a mixture of original and synthetic parallel corpora, the model learns to translate code-switching source sentences at both training and inference time. Concurrently, Dinu *et al.* [2019] propose a similar approach to translate given terminology constraints. The corresponding target terminology in the dictionary is used to replace the source terminology or appended right after the source one. Although translating fast, these two methods use bilingual dictionary to construct training data, thus their performance relies heavily on the quality of the bilingual dictionary. In addition, at inference time, the model will fail in the cases when constraints do not appear in the bilingual dictionary or when the corresponding source phrases are non-consecutive.

Different from the above approaches, we propose a lexical-constraint-aware Transformer model without using a bilingual dictionary, by simply packing constraints and source sentence together with a separating symbol. Instead of specifying the aligned source words for each constraint, our model learns to utilize the constraint automatically without the explicitly alignment information. This simple approach can carry out constrained NMT task with better performance in terms of a combined metric based on accuracy, copy success rate, and decoding speed.

3 Approach

3.1 Problem Statement

Suppose $\mathrm{X} = (x_1, x_2, \cdots, x_S)$ is the source sentence with length S, $\mathrm{Y} = (y_1, y_2, \cdots, y_T)$ is the target sentence with length T. In conventional neural machine translation, the model is trained with Maximum log-Likelihood Estimation (MLE) method. The conditional probability of Y is calculated as follows:

$$p(\mathrm{Y}|\mathrm{X};\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:S};\theta). \tag{1}$$

When lexical constraints are given, the neural machine translation problem can be defined as

$$p(\mathrm{Y}|\mathrm{X},\mathrm{C};\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:S}, \mathrm{C};\theta), \tag{2}$$

where $\mathrm{C} = (C_1, C_2, \cdots, C_N)$ are the provided lexical constraints which are expected to appear in the translations, and N is the number of constraints in C. Different from Hokamp and Liu [2017] and Post and Vilar [2018], these constraints are not forced to appear in the target sequence, i.e. hard constraints are not considered in our settings. The constraints are given as suggestions in a soft manner. In real applications like domain adaptation via terminology, we go through the source sentence and give constraints according to a terminology dictionary. Therefore, the order of constraints are likely to be the same as their corresponding source phrases in the source sentence, which could be different from the order the constraints appear in the reference. So we say that the input constraints are actually disordered.

The performance of constrained machine translation is evaluated in three aspects.

1. Accuracy. The BLEU [Papineni *et al.*, 2002] is used to evaluate the correctness of translation.

[1] Our code is available at https://github.com/ghchen18/leca.

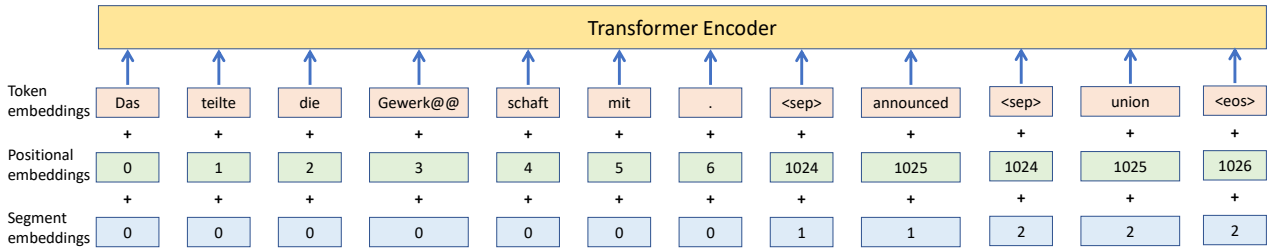


Figure 2: Transformer encoder embedding layer. A special symbol ⟨sep⟩and an additional learned segment embedding are added to distinguish the source and each constraint. The positional index of each constraint starts from a large enough number. The encoder input is the sum of the three embeddings. These modifications help the LeCA better learn to translate with constraints.

2. Coverage. The copy success rate (CSR) is used to check the percentage of constraints that are successfully generated in the translation.
3. Decoding speed. The time-averaged number of generated tokens in inference is used to indicate the time complexity of the approach.

3.2 LExical-Constraint-Aware (LeCA) NMT

Given a triple of source sentence, constraints and target sentence $\langle \mathrm{X}, \mathrm{C}, \mathrm{Y} \rangle$, we define the training loss of the LeCA model as:

$$\begin{aligned} \mathcal{L} &= \sum \log p(\mathrm{Y}|\mathrm{X}, \mathrm{C}; \theta) \\ &= \sum \log p(\mathrm{Y}|\hat{\mathrm{X}}; \theta). \end{aligned} \tag{3}$$

In the above equation, $\hat{\mathrm{X}}$ is a pseudo source sentence, which is constructed by packing the source sentence X and each constraint C_i in the constraint set C together with a separating symbol $\langle sep \rangle$ (See the example in Figure 2):

$$\hat{\mathrm{X}} = \left[\mathrm{X}, \langle \mathrm{sep} \rangle, C_1, \langle \mathrm{sep} \rangle, C_2, ..., \langle \mathrm{sep} \rangle, C_N, \langle \mathrm{eos} \rangle\right], \tag{4}$$

where ⟨eos⟩ is the end of sentence token.

To model such pseudo source sentence $\hat{\mathrm{X}}$, we modify the input representation of the encoder to differentiate the source sentence and each constraint, and add a pointer network to strengthen copying through locating source-side constraints.

Input representation. Inspired by BERT [Devlin *et al.*, 2019], we add a learned segment embedding to each token at the encoder embedding layer for the LeCA model. The encoder embedding layer is composed of three components: token embedding, positional embedding and segment embedding, as shown in Figure 2. We differentiate the source sentence and each constraint in three ways. First, we separate them with a special symbol (⟨sep⟩). Second, the positional index of each constraint starts from the same number that is larger than the maximum source sentence length. Finally, we use different segment embedding for the source sentence and each constraint. For a given token, its input embedding is constructed by summing the corresponding token, positional and segment embeddings.

Pointer network. Following Gulcehre *et al.* [2016] and Song *et al.* [2019], we add a pointer network to strengthen copying through locating source-side constraints in the LeCA model (LeCA+Ptr). At decoding time step t, the final token probability $p(y_t|y_{<t}, \hat{\mathrm{X}})$ is a weighted sum of the token distribution in predictive mode and copy mode. The probability distribution over target-side vocabulary in predictive mode p_t^{predict} is calculated using the decoder output softmax. The token probability distribution in copy mode p_t^{copy} is set using averaged multi-head attention weights of the last decoder layer. If a target token y is in the source sentence, the probability $p_t^{\text{copy}}(y)$ is the attention weight for the corresponding source position. Otherwise, the probability is set as zero. Combining p_t^{copy} and p_t^{predict}, we gain a new distribution over the target vocabulary:

$$p(y_t|y_{<t}, \hat{\mathrm{X}}) = (1 - g_t)\, p_t^{\text{copy}} + g_t\, p_t^{\text{predict}}, \tag{5}$$

where $g_t \in [0, 1]$ is a gate that controls the contribution of these two probability distributions at time step t. We calculate g_t with a feed forward neural network from the context vector c_t and the decoder hidden state at the last layer z_t:

$$\begin{aligned} c_t &= \sum_{s=1}^{\hat{S}} \alpha_{t,s} h_s \\ g_t &= \mathrm{FeedForward}(c_t, z_t), \end{aligned} \tag{6}$$

where h_s is the encoder hidden state at position s of the last layer, $\alpha_{t,s}$ is the averaged attention weight at the last decoder layer for source position s at decoding time step t, and $\hat{S}$ is the length of the source sentence $\hat{\mathrm{X}}$.

3.3 Training

In common NMT settings, we are given a set of parallel source-target sentence pairs at training time, without explicitly provided constraints. In order to train the LeCA model, we propose a way to construct constraints for each parallel sentence pair, following previous work [Hokamp and Liu, 2017; Post and Vilar, 2018; Song *et al.*, 2019].

Specifically, we dynamically sample the constraints from the gold reference during training. For simplicity, first we randomly set the number of constrained words k. Then we randomly sample k target words from the reference. We group these words into phrases, and use these phrases as constraints. We exclude high frequency words from sampling given that they are well learned by the model and less likely to be given as constraints in practice. Finally, we shuffle the constraints to simulate training with disordered constraints.

To maintain the translation performance for unconstrained cases, we leave a proportion of source sentences as unconstrained, i.e., we do not sample constraints in such cases. To be consistent with the constrained cases, we still use the segment embedding and pointer network. Apart from the above, we follow the standard MLE training procedure.

4 Experiments

4.1 Setup

Data. We conduct experiments on German-English and Chinese-English translation tasks in both directions. For the De-En task, we use WMT16 news data as training corpus, `newstest2013` as the development set and `newstest2014` as the test set. For the Zh-En task, we use 1.25M parallel sentences extracted from NIST corpora[2] as the training data. The NIST MT04 dataset serves as the development set, and a combination of NIST MT02, 03, 05, 06, 08 dataset serve as the test set. For the NIST development set and test set, we only choose the first sentence out from the four English references. The German and English corpus is tokenized using the Moses tokenizer [Koehn *et al.*, 2007]. Chinese sentences are segmented by an open-source toolkit *jieba*[3]. Byte-pair encoding (BPE) [Sennrich *et al.*, 2016] with 32K joint merge operations for both language pairs.

Model. We implement LeCA and all the baselines based on Transformer [Vaswani *et al.*, 2017] using `fairseq`[4] [Ott *et al.*, 2019]. We use the `base` Transformer model described in Vaswani *et al.* [2017] but share all embeddings. The maximum number of constrained phrases is set as 50. We use Adam [Kingma and Ba, 2015] and label smoothing for training. The learning rate is 0.0005 and warmup step is 16000. All the drop-out probabilities are set to 0.3. Maximum update number is 100k for the De-En language pair and 60k for the Zh-En language pair.

Evaluation. In practice, constraints are extracted from an in-domain dictionary or provided by human translator in interactive machine translation. Previous work evaluate lexical constrained translation in a simulation approach, where constraints are sampled from the reference sentence. Existing evaluation approaches construct test set through various means, which can be divided into three main categories:

- *Dictionary constraint* [Song *et al.*, 2019; Dinu *et al.*, 2019; Hasler *et al.*, 2018; Wang *et al.*, 2019], where each constraint and its corresponding source phrase is a pair in the bilingual dictionary;
- *Reference constraint* [Post and Vilar, 2018], where the constraints are randomly sampled from the reference;
- *Interactive constraint* [Hokamp and Liu, 2017; Hasler *et al.*, 2018], where the constraints are those phrases from the reference, that fail to be translated in the unconstrained translation result.

To have comprehensive and systematic evaluations, we test LeCA and all the baselines on four types of test set, i.e. clean test set without constraints and the three constrained test sets above. For *dictionary constraint*, we follow [Song *et al.*, 2019] to extract constraints on the test set. For *reference constraint*, we follow the procedure in Section 3.3 to extract constraints on the test set. The number of constrained words for each sentence pair is sampled from 0 to 4 following the distribution $[0.4, 0.1, 0.2, 0.2, 0.1]$. For *interactive constraint*, we follow Hokamp and Liu [2017] to simulate Pick-Revise for interactive post editing [Cheng *et al.*, 2016], first we translate in a constraint-free setting, then we use the same method to sample constraints from the reference, except that words from the constraint-free hypothesis are excluded. Since different models produce different unconstrained translations, the constraints for different models are different. To eliminate the impact of sampling, we repeat experiments on *interactive constraint* five times and report the averaged scores. All constraints are constructed before applying BPE for all constraint types. We use beam search with a beam size of 10. We report case-sensitive BLEU score using sacreBLEU[5] [Post, 2018]. The copy success rate (CSR) is calculated at token level after removing the BPE symbol. The decoding speed is tested on a single GeForce RTX 2080 Ti GPU and is averaged over five runs. Statistical significance is tested with compare-mt toolkit [Neubig *et al.*, 2019] for 1000 resamples and $p = 0.05$.

Baselines. To make the evaluation convincing, we follow previous work to re-implement three existing methods for comparison. These methods are:

- Grid Beam Search (GBS) [Hokamp and Liu, 2017], which extends beam search to generate pre-specified lexical constraints.
- Code-Switching (CS) [Song *et al.*, 2019], which translates with constraints by training a model on synthetic code-switching corpus.
- Code-Switching with Pointer Network (CS+Ptr) [Song *et al.*, 2019], which investigates incorporating the copy mechanism into the decoder in the code-switching method.

For *reference constraint* and *interactive constraint*, we use `fast_align` [Dyer *et al.*, 2013] to find the corresponding source words. We also evaluate on vanilla Transformer (TRANS) for reference.

4.2 Results

We present the BLEU scores of the proposed model LeCA and all the baselines in Table 1 for all the language pair-directions and all the constraint types. It is clear that LeCA and LeCA+Ptr outperform all the baselines, especially for *reference constraint* and *interactive constraint*, and LeCA+Ptr obtains the best performance. For *dictionary constraint*, the average performance of LeCA+Ptr is 0.7 BLEU higher compared with CS and 0.6 with CS+Ptr. For *reference constraint*,

[2]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2004T08 and LDC2005T06

[3]https://github.com/fxsjy/jieba

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

[5]BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.3.7

Model	*Without Constraints*				*Dictionary Constraint*				*Reference Constraint*				*Interactive Constraint*			
	De-En	En-De	Zh-En	En-Zh	De-En	En-De	Zh-En	En-Zh	De-En	En-De	Zh-En	En-Zh	De-En	En-De	Zh-En	En-Zh
TRANS	31.2	27.3	23.4	24.5	–	–	–	–	–	–	–	–	–	–	–	–
GBS	31.2	27.3	23.4	24.5	30.2	28.1	22.8	23.0	33.7	32.4	25.1	26.1	37.7	32.3	27.4	28.8
CS	31.0	26.8	23.4	23.8	31.8	27.8	23.9	24.5	31.8	28.5	23.4	24.3	31.5	30.6	24.8	25.7
+ Ptr	30.6	26.3	23.2	23.5	32.0⋆	27.8	24.1	24.6	28.7	28.2	23.3	24.3	31.2	30.9	25.4	26.0
LeCA	30.7⋆	27.1⋆	23.4	24.4⋆	31.5	28.1	24.3	25.3⋆	35.6	33.0⋆	26.4	28.3	38.8	36.4	28.1	30.1⋆
+ Ptr	31.0	27.4	23.7	24.3	32.3	28.8	24.6	25.1	36.1	33.3	26.9	27.8	38.2	36.9	28.9	29.9

Table 1: BLEU results of constrained NMT on four types of test sets. The best performance among each column is underlined. Scores with asterisk indicates no significant difference with LeCA+Ptr results after statistic significance test. LeCA+Ptr gets overall best performance on lexically constrained test sets while keeping the performance on original test sets without constraints.

Model	*Dictionary Constraint*				*Reference Constraint*				*Interactive Constraint*			
	De-En	En-De	Zh-En	En-Zh	De-En	En-De	Zh-En	En-Zh	De-En	En-De	Zh-En	En-Zh
CS	98.4%	97.5%	95.1%	95.7%	86.3%	89.0%	85.9%	87.0%	86.1%	88.8%	87.3%	87.5%
+ Ptr	99.2%	98.3%	96.5%	96.9%	85.1%	87.6%	85.8%	86.9%	85.3%	87.3%	87.7%	87.9%
LeCA	99.4%	99.0%	97.3%	97.6%	97.2%	97.3%	90.7%	92.9%	93.2%	94.0%	82.7%	85.8%
+ Ptr	99.5%	99.4%	98.4%	98.2%	98.3%	98.7%	94.5%	94.6%	96.4%	97.1%	89.6%	89.6%

Table 2: Results of copy success rate (CSR). The best CSR among each column is underlined. Our method is compared with code switching, another "soft" approach. Pointer network improves the CSR for LeCA. LeCA+Ptr gets highest CSR among all models on all test sets.

Model	TRANS	GBS	CS	CS+Ptr	LeCA	LeCA+Ptr
words/sec	2998	11	2474	2328	2390	2078

Table 3: Decoding speed (words/sec) of different models. We evaluate the decoding speed on the test set of WMT De-En translation with beam size of 10. The batch size for GBS is 1 and 100 for others. For TRANS, we report the constraint-free result, while for the other models, we report the result with *dictionary constraint*.

the average performance of LeCA+Ptr is 4.0 BLEU higher compared with CS and 4.9 with CS+Ptr. For *interactive constraint*, the average performance of LeCA+Ptr is 5.3 BLEU higher compared with CS and 5.1 with CS+Ptr. The CS and CS+Ptr methods work well for *dictionary constraint*. However, their performance on the other two types of constraints are comparatively worse. This can be explained since the *dictionary constraint* is more consistent with how the constraints are constructed at training time for CS and CS+Ptr. In contrast, LeCA and LeCA+Ptr are very robust, i.e., they work well across all constraint types.

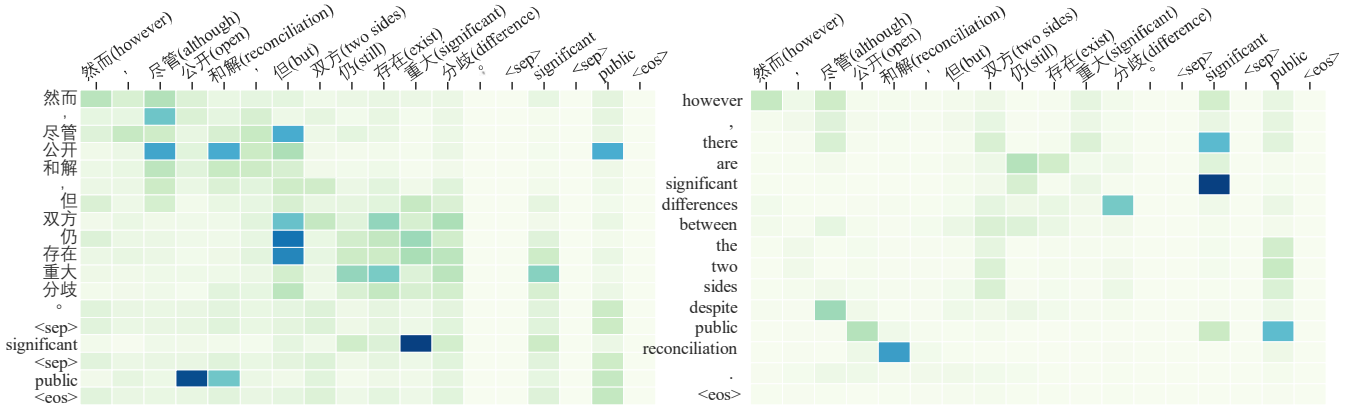
We also observe that all models preserve their strength on translating unconstrained source sentences. As shown in the column *without constraint*, all models perform similarly as the vanilla Transformer in a constraint-free setting.

In Table 2, we compare the copy success rate of our methods with the baseline CS methods. We do not list the CSR of GBS because it imposes hard constraints and the CSR is always 100%. Although our methods do not guarantee all constraints to be generated in the output, we observe that the CSR is high. The average CSR of LeCA+Ptr is 98.9% for *dictionary constraint*, 96.53% for *reference constraint* and 93.18% for *interactive constraint*, significantly improving over the CS and CS+Ptr methods. We also observe that applying the pointer network with LeCA can lead to higher CSR, indicating that the pointer network can strengthen copying through locating source-side constraints.

Table 3 compares the decoding speed of different models. It shows that LeCA and LeCA+Ptr are slightly slower than CS and CS+Ptr. The reason could be that LeCA methods introduce an additional segment embedding and the source sentence in LeCA methods is longer than that in CS methods (the source phrases are kept in LeCA methods, but not in CS methods). However, given the translation quality improvement and the robustness of LeCA methods, such computation overhead is acceptable.

4.3 Analysis

Ablation study. We conduct ablation study on WMT En→De translation with *reference constraint* to study the effect of different model components. For the LeCA+Ptr model, we gradually remove the pointer network (− Ptr), the segment embedding (− segment emb.) and replace the proposed novel positional embedding with consecutive positional embedding that does not differentiate source sentence and constraints (− novel positional emb.). The results are shown in Table 5. We note that pointer network helps to improve the copy success rate. Segment embeddings can improve the performance in terms of BLEU. The novel positional embedding that separately indexes source sentence and



(a) Last encoder layer self attention weights

(b) Last decoder layer encoder-decoder attention weights

Figure 3: Attention map in LeCA model. We masked the attention weights for ⟨sep⟩, ⟨eos⟩and period punctuation in this attention map.

Item	Translations
Input	但是在其启蒙教练张亚杰反复做工作后，张彬彬又回到了厦门队。
w/o cons.	however , after his mongolian coach zhang yajie repeatedly worked , zhang binbin returned to the xiamen team .
+ enlightening	however , after his enlightening coach zhang yajie repeatedly worked , zhang binbin returned to the xiamen team .
+ enlightening + efforts	but after his enlightening coach zhang yajie made repeated efforts , zhang binbin returned to the xiamen team .
ref.	But after the repeated efforts of her enlightening teacher zhang yajie, Zhang Binbin returned to Xiamen team .
input	然而，尽管公开和解，但双方仍存在重大分歧。
w/o cons.	however , despite open reconciliation , major differences still exist between the two sides .
+ significant + public	however , there are significant differences between the two sides despite public reconciliation .
ref.	still , despite the public display of reconciliation , the two still have major differences .

Table 4: Lexically constrained translation examples with the LeCA model. w/o cons. denotes translations without constraints, ref. is the reference sentence.

Model	refer. cons.	CSR
LeCA+Ptr	33.3	98.7%
− Ptr	33.0	97.3%
− segment emb.	31.7	97.4%
− novel positional emb.	25.3	75.5%

Table 5: Results of ablation study in En-De test set.

each constrain is the key for high copy success rate. When combining novel positional embedding, segment embedding and pointer network, our model obtains the best performance.

Case study. We include two examples of translations with the method LeCA in Table 4. It shows that our method successfully incorporates constraints into the translation output. Since our methods learn to incorporate the constraints without explicit alignment, we suspect that the encoder has the ability to align each target constraint to its corresponding source phrase. Therefore, we analyze the attention maps of our methods and show one example in Figure 3. This example as well as manual inspection of others confirm our suspicion. In addition, we also find that when producing constrained words, the decoder tends to attend to the corresponding constraint at the source side, instead of the aligned source phrase. We leave more in-depth analysis as future work.

5 Conclusion

In this paper, we propose a novel lexical-constrained-aware (LeCA) NMT model that can incorporate lexical constraints in the translation accurately and efficiently. Our model modifies the input representation of the Transformer encoder and augments training data by appending constraints after the source sentence with a separation symbol. Experiments on WMT De-En and NIST Zh-En show that our model significantly improves the performance over baselines, with minimal additional decoding time. As LeCA can generate various translations given different constraints, we see the exploration of diverse neural machine translation as a clear avenue for future work.

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 and the funds of Beijing Advanced Innovation Center for Language Resources (No. TYZ19005).

References

[Arthur *et al.*, 2016] Philip Arthur, Graham Neubig, and Satoshi Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of EMNLP*, pages 1557–1567, 2016.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, 2015.

[Cheng *et al.*, 2016] Shanbo Cheng, Shujian Huang, Huadong Chen, Xin-Yu Dai, and Jiajun Chen. PRIMT: A pick-revise framework for interactive machine translation. In *Proceedings of NAACL*, pages 1240–1249, 2016.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186, 2019.

[Dinu *et al.*, 2019] Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. Training neural machine translation to apply terminology constraints. In *Proceedings of ACL*, pages 3063–3068, 2019.

[Dyer *et al.*, 2013] Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of NAACL*, 2013.

[Gulcehre *et al.*, 2016] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. In *Proceedings of ACL*, pages 140–149, 2016.

[Hasler *et al.*, 2018] Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. Neural machine translation decoding with terminology constraints. In *Proceedings of NAACL*, pages 506–512, 2018.

[Hokamp and Liu, 2017] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of ACL*, pages 1535–1546, 2017.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[Koehn *et al.*, 2003] Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 127–133, 2003.

[Koehn *et al.*, 2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*, pages 177–180, 2007.

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

[Luong *et al.*, 2015] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*, pages 11–19, 2015.

[Neubig *et al.*, 2019] Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. compare-mt: A tool for holistic comparison of language generation systems. In *Proceedings of NAACL*, 2019.

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

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 2002.

[Post and Vilar, 2018] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of NAACL*, pages 1314–1324, 2018.

[Post, 2018] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation*, pages 186–191, 2018.

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

[Song *et al.*, 2019] Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. Code-switching for enhancing NMT with pre-specified translation. In *Proceedings of NAACL*, pages 449–459, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2019] Tao Wang, Shaohui Kuang, Deyi Xiong, and António Branco. Merging external bilingual pairs into neural machine translation. *ArXiv*, abs/1912.00567, 2019.