# Incorporating Terminology Constraints in Automatic Post-Editing

**David Wan**[1], **Chris Kedzie**[1], **Faisal Ladhak**[1], **Marine Carpuat**[2] **and Kathleen McKeown**[1]
[1] Columbia University, [2] University of Maryland
dw2735@columbia.edu, {kedzie,faisal,kathy}@cs.columbia.edu, marine@cs.umd.edu

## Abstract

Users of machine translation (MT) may want to ensure the use of specific lexical terminologies. While there exist techniques for incorporating terminology constraints during inference for MT, current APE approaches cannot ensure that they will appear in the final translation. In this paper, we present both autoregressive and non-autoregressive models for lexically constrained APE, demonstrating that our approach enables preservation of 95% of the terminologies and also improves translation quality on English-German benchmarks. Even when applied to lexically constrained MT output, our approach is able to improve preservation of the terminologies. However, we show that our models do not learn to copy constraints systematically and suggest a simple data augmentation technique that leads to improved performance and robustness.

## 1 Introduction

Automatic post-editing (APE) aims to improve the quality of the output of an arbitrary machine translation (MT) system by pruning systematic errors and adapting to a domain-specific style and vocabulary (Simard et al., 2007; Chatterjee et al., 2018). Although previous work has shown the usefulness of APE to prune errors by focusing on improving the translation error rate (TER), few have studied the effect of incorporating lexical constraints.

There are several use cases where such a system would be beneficial. For example, content providers meticulously curate lists of terminologies for their domains that indicate preferred translations for technical terms. Lexically constrained APE would also be useful for cross-lingual information retrieval. When displaying snippets from retrieved documents, the query term should appear in the translation output (if it does in the source) as it can make relevance clear to the end user. Here, the query serves as the term.

While recent approaches allow inference time adaptation of NMT systems using these terminologies (Dinu et al., 2019; Post and Vilar, 2018), post-editing translations with a generic APE system may lead to dropped terms. A constraint-aware APE system would allow to fix systematic translation errors, while keeping the terminologies intact.

Inspired by Dinu et al. (2019), we consider a range of representations which augment input sequences with constraint tokens and factors for use in an autoregressive Transformer (AT) APE model. Using this approach, the constraints are explicitly represented in the encoder input sequence, and the model learns to prefer translations that contain the supplied terminologies during decoding. We also explore the use of the Levenshtein Transformer (LevT) (Gu et al., 2019), a non-autoregressive Transformer (NAT) model. The LevT model applies neatly to the APE task since the decoder can be initialized with an incomplete sequence to be refined. Additionally, multiple corrections can be made simultaneously, yielding a decoding speedup over autoregressive models.

We then show that constrained APE improves translation quality and terminology preservation on top of both unconstrained and constrained MT. While both constrained and unconstrained APE models perform similarly on reducing systemic errors in the MT output, they differ in their ability to preserve terminology constraints. When applying unconstrained APE on top of constrained MT, we find a 12.6% relative drop of supplied terminology constraints as compared to a fully constrained MT to APE pipeline.

We experiment extensively with variations of both AT and LevT models, testing on both PBMT and NMT English to German WMT APE tasks (Chatterjee et al., 2018). Under all scenarios, the model performs post-editing while satisfying terminology constraints when supplied.

Our evaluation of both constrained AT and NAT models on PBMT and NMT APE benchmarks shows that both models correctly translate more than 95% of terminology constraints, with the NAT model achieving the highest coverage of terminologies at the expense of post-editing quality.

Finally, using constraints constructed with synonyms and antonyms, we show that our models do not learn to copy constraints systematically, and introduce a simple data augmentation strategy to improve the preservation of unusual constraints.

To summarize, our contributions are as follows:

1. We propose the terminology constrained APE task and evaluate several AT and LevT model variants for incorporating lexical constraints.
2. We empirically show that constrained APE is necessary to preserve terminology constraints in a MT to APE pipeline.
3. We analyze the robustness of the constraint translation behavior and suggest a simple data augmentation technique that both improves translation quality and increases the number of correctly translated terms.

## 2 Related Work

### 2.1 MT with Terminology Constraints

Integrating terminology constraints into translation can be divided into two approaches: constrained decoding and input sequence modification.

Constrained decoding modifies the decoding process to enforce the generation of the specified terminologies. This includes methods that modify beam search, such as grid beam search (Hokamp and Liu, 2017) and dynamic beam allocations (Post and Vilar, 2018). While these approaches are effective in including terminologies, they come with an increase in inference time due to the added overhead in the search algorithm.

The LevT (Gu et al., 2019), which uses a non-autoregressive decoding procedure, can initialize its decoder with a partial or incomplete output sequence. By initializing the decoder output with terminology constraints, Susanto et al. (2020) train a LevT model to perform constrained decoding. Unlike constrained search methods in autoregressive models, this initialization technique does not add any significant overhead to the decoding process. When modified to disallow deletion of terms and insertion between consecutive terminology tokens, LevT is able to retain all terminologies without affecting the performance and speed.

Alternatively, Dinu et al. (2019) propose modifying the encoder input sequence to represent terminology constraints. During training, the model learns to identify constraints in the input sequence, and translate them appropriately during decoding. This approach has the benefit of not adding additional overhead during inference.

### 2.2 Automatic Post-Editing

The APE task has gone through many iterations, since it was originally proposed by Simard et al. (2007). Initially, the task was to improve an unknown phrase-based machine transition (PBMT) system. An additional task to fix errors of an NMT system was introduced at WMT 2018 (Chatterjee et al., 2018).

For the APE tasks, the use of the multi-source variant of the neural encoder-decoder model is the most popular approach (Bojar et al., 2017), with the Multi-source Transformer (MST) instantiation (Junczys-Dowmunt and Grundkiewicz, 2018) achieving state-of-the-art results in 2018. Based on the AT model (Vaswani et al., 2017), the MST model consists of two Transformer encoders and a single decoder. The source sentence and the MT system output are fed separately to the two encoders, where the outputs are concatenated and then fed into the decoder to perform post-editing.

Recent work has explored alternative architectures for APE. The winner of 2019 APE tasks (Lopes et al., 2019), for example, uses a BERT-based encoder and decoder. Gu et al. (2019) both introduce the LevT model and demonstrate its utility on an APE task.

## 3 Constrained APE

The task of APE is to correct systematic errors in an MT system output. An APE model takes as input two sequences: the source language sentence to be translated and the translation of this sentence into the target language by an MT system. The intended output is a corrected version of the MT system's initial translation (Simard et al., 2007).

Constrained APE allows for the specification of terminology constraints: a translation for one or more phrases in the source language input may be pre-specified as additional input. The constrained APE model must use the supplied terminology constraints when performing the APE task.

| | |
|---|---|
| Source ($\mathbf{x}$) | The Gradient tool also provides most of the same **features** as the Gradient panel . |
| Append ($\mathbf{x}^+$) | The$_0$ Gradient$_0$ tool$_0$ also$_0$ provides$_0$ most$_0$ of$_0$ the$_0$ same$_0$ **features**$_1$ **Funktionen**$_2$ as$_0$ the$_0$ Gradient$_0$ panel$_0$ .$_0$ |
| Replace ($\mathbf{x}^-$) | The$_0$ Gradient$_0$ tool$_0$ also$_0$ provides$_0$ most$_0$ of$_0$ the$_0$ same$_0$ **Funktionen**$_2$ as$_0$ the$_0$ Gradient$_0$ panel$_0$ .$_0$ |
| MT ($\boldsymbol{\gamma}$) | Das$_3$ Verlaufswerkzeug$_3$ bietet$_3$ außerdem$_3$ die$_3$ meisten$_3$ der$_3$ gleichen$_3$ Merkmale$_3$ wie$_3$ das$_3$ Verlaufsbedienfeld$_3$ .$_3$ |
| Post-Edit ($\mathbf{y}$) | Das Verlaufswerkzeug bietet fast dieselben **Funktionen** wie das Verlaufsbedienfeld . |

Figure 1: An example of the inputs and output for the constrained APE task. Source is the source sentence. Post-Edit is the corrected MT sentence. We show the *Append* and *Replace* method to incorporate terminologies on the source side. Factors indicated for each word as source word (0), source constraint (1), target constraint (2), and MT word (3). The terminology pair for this example is (*features*, *Funktionen*).

Formally, let $\mathbf{x} = [x_1, \ldots, x_m]$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]$ be the source language sentence and the initial MT translation respectively. The tokens $x_i$ and $\gamma_i$ are drawn from the source and target language vocabularies $\mathcal{S}$ and $\mathcal{T}$ respectively. The target post-edited sentence is a sequence $\mathbf{y} = [y_1, \ldots, y_o]$, with tokens $y_i$ also drawn from $\mathcal{T}$.

We are also given a series of $t$ translation constraints, $\mathcal{C} = \left\{\left(\check{\boldsymbol{x}}^{(1)}, \check{\boldsymbol{y}}^{(1)},\right), \ldots, \left(\check{\boldsymbol{x}}^{(t)}, \check{\boldsymbol{y}}^{(t)},\right)\right\}$, where each constraint $\left(\check{\boldsymbol{x}}^{(i)}, \check{\boldsymbol{y}}^{(i)}\right) \in \mathcal{S}^* \times \mathcal{T}^*$ is a tuple of source language phrase, $\check{\boldsymbol{x}}^{(i)} = [\check{x}_1, \ldots, \check{x}_j]$, and its desired translation, $\check{\boldsymbol{y}}^{(i)} = [\check{y}_1, \ldots, \check{y}_k]$, into the target language.

The goal of the constrained APE task is to learn a mapping of $\mathbf{x}, \boldsymbol{\gamma}$, and $\mathcal{C}$ to the target post-edited translation $\mathbf{y}$. Crucially, when a source side constraint $\check{\boldsymbol{x}}^{(i)}$ matches a sub-sequence in $\mathbf{x}$, it is required that the sub-sequence be translated as $\check{\boldsymbol{y}}^{(i)}$. See Figure 1 for an example.

## 4 Models

While there are existing models to address the APE task, and the lexical constrained MT task, it is not clear how to represent lexical constraints for APE models which, unlike MT models, take two sequences as input. We propose several techniques to incorporate constraints as additional inputs to the APE encoder by combining the input sequence modification used in constrained MT (Dinu et al., 2019) with the MST method of (Tebbifakhr et al., 2018). For decoding, we experiment with both the AT and the LevT decoders. The LevT decoder can additionally take advantage of different decoder initialization strategies for constrained decoding.

We first briefly show how we encode terminology constraints in the input sequence, before describing how they are incorporated into the MST and LevT APE models specifically.

| Model | Input | Init. |
|---|---|---|
| MST | $\mathbf{x}, \boldsymbol{\gamma}$ | – |
| MST Append | $\mathbf{x}^+, \boldsymbol{\gamma}$ | – |
| MST Replace | $\mathbf{x}^-, \boldsymbol{\gamma}$ | – |
| LevT | $\mathbf{x}$ | $\boldsymbol{\gamma}$ |
| LevT Append | $\mathbf{x}^+$ | $\boldsymbol{\gamma}$ |
| LevT Replace | $\mathbf{x}^-$ | $\boldsymbol{\gamma}$ |
| MS LevT | $\mathbf{x}, \boldsymbol{\gamma}$ | $\check{\boldsymbol{y}}^1, \ldots, \check{\boldsymbol{y}}^{(t)}$ |

Figure 2: Setup for the models by the input and initialization at inference.

### 4.1 Encoding Lexical Constraints for APE in the Input Sequence

In the APE setting, the input to the model is the source language sentence $\mathbf{x}$ and its initial MT translation $\boldsymbol{\gamma}$. We also need to represent in $\mathbf{x}$ the translation constraints $\mathcal{C}$.

For clarity, we describe the case of representing a single translation constraint $(\check{\boldsymbol{x}}, \check{\boldsymbol{y}})$ where $\check{\boldsymbol{x}} = [\check{x}_1, \ldots, \check{x}_j]$ is a source language constraint and $\check{\boldsymbol{y}} = [\check{y}_1, \ldots, \check{y}_k]$ is its target language translation. Our approach trivially generalizes to multiple constraints. We represent the constraint $(\check{\boldsymbol{x}}, \check{\boldsymbol{y}})$ in $\mathbf{x}$ in one of two ways. Either by appending the target language constraint $\check{\boldsymbol{y}}$ after the occurrence of $\check{\boldsymbol{x}}$ in the input sequence, or by replacing occurrences of $\check{\boldsymbol{x}}$ in $\mathbf{x}$ with $\check{\boldsymbol{y}}$.

For example, if we had the constraint, $(\check{\boldsymbol{x}}, \check{\boldsymbol{y}}) = ([\check{x}_1, \check{x}_2], [\check{y}_1])$ and the source input $\mathbf{x} = [x_1, x_2, x_3, x_4]$, with $[\check{x}_1, \check{x}_2] = [x_2, x_3]$, we would obtain the following input sequences for the append and replace methods:

- *(Append)* $\mathbf{x}^+ = [x_1, x_2, x_3, \check{y}_1, x_4]$
- *(Replace)* $\mathbf{x}^- = [x_1, \check{y}_1, x_4]$ .

To further differentiate the constraint terms from other tokens in the source sentence, a "source factor" is associated with each input token. The source factor is equal to 1 or 2 to indicate a source or target side terminology constraint, while 0 indicates an unconstrained source token. For the above examples, we would obtain the following source factors:

- *(Append)* $\mathbf{s}^+ = [0, 1, 1, 2, 0]$
- *(Replace)* $\mathbf{s}^- = [0, 2, 0]$ .

The source input sequence and source factor sequence are separately embedded and concatenated before they are fed into the encoder. See Figure 1 for examples of the *append* and *replace* methods applied to a source sentence.

We now describe how we use these modified input sequences in the MST and LevT models. See Figure 2 for an overview of the the proposed models and their configuration.

### 4.2 Multi-source Transformer

The input to an APE model is a pair of sequences, the source sentence and the MT output to be post-edited. To accommodate these two sequences, we use the MST model of Tebbifakhr et al. (2018), which uses a separate Transformer to encode each sequence. The outputs of each encoder are concatenated and attended to by the decoder.

We augment the encoder for the source sentence with the *append* and *replace* methods. Figure 1 shows an example of the inputs for the *append* and *replace* methods, $\mathbf{x}^+$ and $\mathbf{x}^-$ respectively. To account for the additional input of MT, $\gamma$, for the source factors, we use 3 for each token in $\gamma$. For Byte-Pair Encoding (BPE) (Sennrich et al., 2016), the corresponding source factor token is applied for all subword units.

We train three variants based on MST: an unconstrained version as the baseline (MST), and two constrained versions using the *append* (MST Append) and *replace* (MST Replace) methods as described in subsection 4.1.

### 4.3 Levenshtein Transformer

The LevT follows the Transformer encoder-decoder architecture. However, instead of a regular Transformer decoder, the model uses three consecutive layers to simulate the edit operations. The first layer predicts whether each token should be deleted or kept. The second layer predicts how many placeholder tokens to insert between every two consecutive tokens. The final layer then predicts the actual target token for each placeholder.

One benefit of using the LevT is its ability to initialize the decoding process with an arbitrary sequence. The first iteration of the decoding process is typically initialized with $\mathbf{y^0} = [<s>, </s>]$, but it is possible to initialize it with MT (i.e. $\gamma$) and allow it to be subsequently refined.

Since LevT retains the single encoder and decoder structure, the changes to incorporate lexical constraints are straightforward; we apply the *append* and *replace* methods to the encoder input.

We also try augmenting the LevT similarly to the MST. Here, we have two encoders for the source, $\mathbf{x}$, and MT, $\gamma$, respectively. During inference, we initialize the decoding string with target-side constraint terms, $\check{\boldsymbol{y}}$, similar to the constrained decoding setup in Susanto et al. (2020).

For multiple constraints, we sort the target side terms $\check{\boldsymbol{y}}^{(i)}$ by the order of the occurrence of $\check{\boldsymbol{x}}^{(i)}$ in the source $\mathbf{x}$. When source and target word order diverge, we hope that the model will learn to reorder constraints correctly, but leave experimentation with constraint ordering for future work.

We train four variants of the LevT model. An unconstrained baseline model (LevT), and two constrained variants, with the same architecture as the base LevT, that incorporate constraints in the source using the *append* $(\mathbf{x}^+)$ (LevT Append) and *replace* $(\mathbf{x}^-)$ (LevT Replace) methods described in subsection 4.1. The decoder initialization for these models is the MT sentence, $\gamma$, that needs to be edited. The final variant has a multi-source encoder, where $\mathbf{x}$ and $\gamma$ are fed into separate encoders. The decoder in this case is initialized with the target sequence of the terminology constraint(s).

## 5 Data

### 5.1 APE Datasets

We use two standard English-to-German APE benchmark datasets, WMT18 PBMT (Chatterjee et al., 2018) and WMT19 NMT (Chatterjee et al., 2019). Both datasets are in the IT domain. Each example from these datasets consists of three sequences: (1) the source sentence $\mathbf{x}$, (2) its MT output $\gamma$, and (3) its post-edited target $\mathbf{y}$.

|  | Dataset | # of Triplets |  |  | Term% | TER | BLEU |
|---|---|---|---|---|---|---|---|
|  |  | Train | Valid | Test |  |  |  |
| PBMT | artificial 4M | 4,390,180 | 1,000 | - | - | - | - |
|  | artificial 500K | 526,368 | - | - | - | - | - |
|  | WMT'18 APE | 24,000 | 2,000 | 2,000 | 88.83 | 24.57 | 62.39 |
| NMT | eSCAPE NMT | 4,999,102 | 1,000 | - | - | - | - |
|  | WMT'19 APE | 13,442 | 1,000 | 1,023 | 89.52 | 16.92 | 74.60 |

Table 1: Statistics for data used. Term%, TER, and BLEU are provided for do-nothing case of test set.

Since the official collections are relatively small, we augment them with large synthetic datasets for pretraining: artificial (Junczys-Dowmunt and Grundkiewicz, 2016) and eSCAPE (Negri et al., 2018). The artificial dataset is generated using round-trip translation of two PBMT systems. It is already cleaned and tokenized. The eSCAPE dataset, containing 7,258,533 triplets, is created using NMT generated output from various parallel corpora. The data for eSCAPE is noisy, and we follow Lee et al. (2019)'s procedure to filter the dataset, which results in around 5 million triplets. We then tokenize the filtered data using Moses (Koehn et al., 2007).[1] For pretraining on the synthetic corpora, we set aside 1,000 randomly sampled triplets as our validation set. Table 1 summarizes the statistics of both the evaluation and pretraining datasets.

For both tasks, we use the same preprocessing steps. After tokenization, we truecase the data using Moses. We then use BPE with 32,000 merge operations on the joined vocabulary of source and target language.

### 5.2 Terminology Dataset

We create terminology sets for each APE dataset using Wiktionary.[2] We follow the procedure of Dinu et al. (2019), finding term translation pairs $(\breve{\mathbf{x}}, \breve{\mathbf{y}})$ in Wiktionary such that $\breve{\mathbf{x}}$ is present in the source sentence $\mathbf{x}$ and $\breve{\mathbf{y}}$ is present in the post-edited target sentence $\mathbf{y}$. We ignore stop words that appear on the source and target side. To include more morphological variations, we include matches on stemmed versions of $\breve{\mathbf{x}}$ and $\breve{\mathbf{y}}$ using Snowball stemming.[3] We recover the unstemmed words from the pairs to be included in the terminology dataset. In order for the model to perform the APE task well when no constraints are supplied, we keep only 25% of matched terminology constraints (i.e. we remove 75% of constraints at random).

We split the terminology dataset into training and test sets so that terminology constraints provided at test time are not seen during training. We only use the training set for the training corpora of APE datasets, and use the test sets of the terminology on the validation and test set of the APE datasets. See Table 2 for statistics of terminology coverage on the training, validation, and test splits.

With the given MT system, we can evaluate on terminology percentage for the do-nothing case, which is shown in Table 1. The original MT model already achieves a high term percentage of around 90% for PBMT and NMT tasks.

## 6 Experiments

We use the FAIRSEQ toolkit (Ott et al., 2019) for implementing the MST and extending the LevT.[4] We evaluate the models on translation error rate (TER) (Snover et al., 2006) and BLEU (Papineni et al., 2002) using the official evaluation script[5] for analyzing the post-editing performance. We also compute the percentage of target language term constraints present in the output (Term %) to measure the performance of the constrained models.

### 6.1 Constrained MT-to-APE Cascades

In our first experiment, we attempt to demonstrate the utility of constrained APE when applied to constrained MT. That is, we have some terminology constraints that we want to preserve throughout the application of MT and subsequent APE. We

[1] `www.statmt.org/moses/`

[2] We use the latest dump as of 06/18/2020

[3] `www.nltk.org/_modules/nltk/stem/snowball.html`

[4] Our code is publicly available at `https://github.com/zerocstaker/constrained_ape`.

[5] `www.dropbox.com/s/5jw5maariwey080/Evaluation_Script.tar.gz?dl=0`

| Dataset | | # of Triplets with Term. | | | Avg # of Term. | | |
|---|---|---|---|---|---|---|---|
| | | Train | Valid | Test | Train | Valid | Test |
| PBMT | artificial 4M | 1,605,075 | 345 | - | 1.25 | 1.25 | - |
| | artificial 500K | 207,225 | - | - | 1.27 | - | - |
| | WMT'18 APE | 6,037 | 834 | 528 | 1.15 | 1.24 | 1.34 |
| NMT | escape NMT | 1,768,587 | 335 | - | 1.28 | 1.30 | - |
| | WMT'19 APE | 3,450 | 262 | 408 | 1.16 | 1.14 | 1.25 |

Table 2: The number of training/validation/test instances that have at least one terminology constraint and the average number of terminology constraints for those instances.

conjecture that unconstrained APE applied on top of constrained MT will potentially discard or re-translate previously translated constraints.

We experiment with all possible pipelines of MT to APE, i.e. the product of $\{\text{MT}, \text{Const. MT}\} \times \{\text{No APE}, \text{APE}, \text{Const. APE}\}$ with six total pipelines possible.

**MT Models** To obtain MT models for this experiment we train both a constrained and unconstrained AT MT model using the default FAIRSEQ Transformer hyperparameters and use the embedding size of 16 for the source factor embedding. We follow the settings of Dinu et al. (2019), training an unconstrained transformer and a constrained model with append method to perform English-to-German translation using the Europarl and News Commentary data, and using the WMT 2013/2017 test set as validation and test set respectively. The preprocessing steps follows that of the APE datasets.

For the constrained MT model we used the *append* input modification method to make the model constraint aware. Terminology constraints are generated according to the method described in subsection 5.2. Dinu et al. (2019) also released their Wiktionary terminology set (Wikt975)[6] and we also show evaluation results using this terminology collection. We report BLEU and terminology coverage (Term %) for our MT models on the WMT 2017 test set in Table 3.

**APE Models** We use the MST and the append method as the unconstrained APE and constrained APE respectively. We evaluate the MT to APE pipelines using the WMT'19 APE test set, replacing the provided MT in the triplet with the outputs from our unconstrained or constrained MT models.

[6]`https://github.com/mtresearcher/terminology_dataset`

| | Our Term. | | Wikt 975 | |
|---|---|---|---|---|
| MT | Term% | BLEU | Term% | BLEU |
| AT | 71.70 | 23.76 | 74.78 | 24.00 |
| AT App. | 93.62 | 24.62 | 93.07 | 24.14 |

Table 3: Translation result of vanilla and lexically constrained translation.

We train the APE models using the eSCAPE corpus, where 1,000 triplets are used as validation set. We use the default FAIRSEQ Transformer hyperparameters. For the constrained APE, we use the embedding size of 16 for the source factor tokens.

## 6.2 Benchmark APE Tasks

The APE models are trained in two step fashion. First, a general APE system is trained using a synthetic dataset until convergence. Then the model is refined on the official dataset. For the PBMT task, we follow the training procedure of Gu et al. (2019). The model is pretrained on the artificial 4M dataset, and fine-tuned on the joined dataset of the 500K artificial dataset and the 10 times up-sampled official PBMT data. For the NMT task, we pretrain on eSCAPE and fine-tune on the official NMT data.

We use the default Transformer parameters for the MST variants, with an embedding size of 16 for source factors of the constrained APE models. For the LevT models, we follow the same setup and hyper-parameters as described in Gu et al. (2019).

We compare our models to the do-nothing case, where the output of the MT, $\gamma$, is treated as the predicted post-edited sentence $\hat{\mathbf{y}}$. The unconstrained variant also serve as a basis for comparing the performance of the constrainted APE models. We also compare our models to the winning system for the tasks, MS_UEdin (Junczys-Dowmunt and Grund-

| Pipeline | Term%↑ | TER↓ | BLEU↑ |
|---|---|---|---|
| MT | 45.33 | 70.78 | 15.28 |
| cMT | 86.33 | 70.24 | 15.47 |
| MT → APE | 55.35 | 59.56 | 22.87 |
| cMT → APE | 77.22 | 59.78 | 23.03 |
| MT → cAPE | 80.18 | **58.70** | **23.95** |
| cMT → cAPE | **88.38** | 59.77 | 23.08 |

Table 4: Result of different combinations of MT and APE systems. Constrained MT and APE are indicated cMT and cAPE respectively.

| Models | Term%↑ | TER↓ | BLEU↑ |
|---|---|---|---|
| Do-nothing | 88.48 | 24.25 | 62.99 |
| MS_UEdin | 88.70 | **18.01** | **72.52** |
| MST | 90.11 | 19.34 | 70.44 |
| MST Append | 95.54 | 18.97 | 70.63 |
| MST Replace | 95.43 | 19.17 | 70.34 |
| LevT | 90.76 | 24.21 | 63.47 |
| LevT App. | 90.98 | 23.88 | 64.97 |
| LevT Rep. | 91.41 | 23.94 | 64.96 |
| MS LevT | **97.50** | 20.39 | 68.57 |

Table 5: Results for PBMT 2018.

| models | Term%↑ | TER↓ | BLEU↑ |
|---|---|---|---|
| Do-nothing | 90.22 | 16.84 | 74.73 |
| Unbabel_BERT | 89.98 | **16.06** | **75.96** |
| MST | 90.66 | 16.46 | 75.61 |
| MST Append | 94.08 | 16.62 | 75.16 |
| MST Replace | 94.08 | 16.56 | 75.39 |
| LevT | 90.41 | 17.28 | 74.17 |
| LevT App. | 91.59 | 17.32 | 74.25 |
| LevT Rep. | 90.61 | 17.14 | 74.46 |
| MS LevT | **98.04** | 17.71 | 73.64 |

Table 6: Results for NMT 2019.

kiewicz, 2018) for PBMT 2018 and Unbabel_BERT Lopes et al. (2019) for NMT 2019.

## 7 Results and Discussions

### 7.1 Constrained MT-to-APE Cascade

Table 4 shows the result of the various combinations of MT and APE systems. Since the MT system is trained on news/parliamentary proceedings and not on the IT domain of the APE data, the translation quality is relatively low. Nevertheless, the constrained MT can include almost twice as many terminologies as the original model. Both APE systems improve the quality of the MT outputs, with constrained APE performing slightly better. However, constrained APE excels at including terminologies, as it consistently increases the terminology percentage from the previous MT output. When supplied with a constrained MT, the vanilla APE actually decreases the percentage of correct terminologies by 9% (86.33% to 77.22%), whereas the constrained APE model can increase it by 2% (86.33% to 88.38%).

### 7.2 Benchmark APE on PBMT Output

Table 5 shows the results on the PBMT task. All MST variants improve from the do-nothing case, where the output is unchanged, i.e.$\gamma = \hat{y}$. Using either the *append* or *replace* methods shows similar improvements in Term%, increasing about 7% points absolutely over the do nothing case. The terminology aware MST models also see small decreases in TER and small increases in BLEU relative to the unconstrained MST model. These results are encouraging as it shows that introducing terminology constraints does not interfere with the APE system's ability to fix systematic errors.

We were unable to reproduce the result by Gu et al. (2019); we see only small improvements with the LevT models relative to the do-nothing case. Additionally, the *append* and *replace* variants yield only small increases in Term% but are around 4-5% points behind the equivalent MST model. The MS LevT, however, achieves the highest terminology percentage of all models, while slightly underperforming the MST models on TER and BLEU.

None of our proposed models beat the SOTA baseline for this task on TER or BLEU, but our best model on TER, MST Append, is less than 1 percentage point worse in TER. At the same time, MST Append successfully translates 6.8% more terminology constraints than the SOTA baseline.

### 7.3 Benchmark APE on NMT Output

Table 6 shows the result of the NMT task. This is a more difficult post editing task as the machine translated text from NMT systems is of a higher quality than PBMT systems, and the official training corpus is smaller than that of PBMT APE (Chatterjee et al., 2018). As further evidence of the difficulty of this task, the winning system of the WMT2019

| | | |
|---|---|---|
| Original | Source $(\mathbf{x})$ | increasing the **magnification** can also make reshaping easier and more accurate . |
| | MT $(\gamma)$ | durch das Vergrößern der **Vergrößerung** können Sie außerdem das Umformen von Formen und präziser steuern . |
| | Post-Edit $(\mathbf{y})$ | durch das Vergrößern der **Vergrößerung** können Sie außerdem das Umformen von Formen präziser steuern . |
| | MST Append | Durch das Vergrößern der **Vergrößerung** können Sie außerdem das Umformen von Formen erleichtern und präziser steuern . |
| | MS LevT | Durch die zunehmende **Vergrößerung** können Sie außerdem das Umformen von Formen und präziser steuern . |
| Synonym | Post-Edit $(\mathbf{y})$ | durch das Vergrößern der **Magnifizierung** können Sie außerdem das Umformen von Formen präziser steuern . |
| | MST Append | durch das Vergrößern der **Magnifizierung** können Sie außerdem das Umformen von Formen vereinfachen und präziser steuern . |
| | MS LevT | eine Erhöhung der **Magnifizierung** kann außerdem das Umformen von Formen und präziser erleichtern . |
| Antonym | Post-Edit $(\mathbf{y})$ | durch das Vergrößern der **Verkleinerung** können Sie außerdem das Umformen von Formen präziser steuern . |
| | MST Append | durch das Vergrößern der **Vergrößerung** können Sie außerdem das Umformen von Formen vereinfachen und präziser steuern . |
| | MS LevT | durch die zunehmende **Verkleinerung** können Sie außerdem das Umformen von Formen und präziser steuern . |

Figure 3: Example of the outputs by the MST Append and MS LevT when a synonym and an antonym is supplied in place of the original terminology pair (*magnification* - *Vergrößerung*). The synonym *Magnifizierung* (magnification) and antonym *Verkleinerung* (diminishment) is used.

| | |
|---|---|
| Source $(\mathbf{x})$ | if you use the Image Processor , you can **save** the files directly to JPEG format in the size that you want them . |
| Post-Edit $(\mathbf{y})$ | wenn Sie den Bildprozessor verwenden , können Sie die Dateien direkt im JPEG-Format in der gewünschten Größe **speichern** |
| Synonym | wenn Sie den Bildprozessor verwenden , können Sie die Dateien direkt im JPEG-Format in der gewünschten Größe **sichern** |
| Antonym | wenn Sie den Bildprozessor verwenden , können Sie die Dateien direkt im JPEG-Format in der gewünschten Größe **löschen** |

Figure 4: Example of data augmentation. The original term pair is (*save*, *speichern*). We replace the target terminology *speichern* with the synonym *sichern* (to store for future use) or the antonym *löschen* (to delete).

APE shared task is able to achieve a mere 0.78 point decrease in TER.

The two terminology-aware MST models (*append* and *replace*) are able to improve Term% over the baseline, at the cost of a slight increase in TER and decrease in BLEU, but both are better than doing nothing. The LevT and its variants perform worse than doing nothing in terms of TER and BLEU, but has a small gain in Term%. The MS LevT again achieves the highest Term% but does worse than the do-nothing case on TER and BLEU.

## 8 Analyzing Constraint Translation Behavior

Terminology constrained APE aims to add some degree of user control over the APE process without destabilizing the general post-editing behavior of the decoder. However, the imposition of rare or unusual terminology constraints will necessarily be in conflict with the decoder language model, which will give higher probabilities to terminology translations found frequently in the training data.

In practice, a user may specify a terminology constraint that is not well represented in the training distribution. For example, a user may want a product description translated using location specific brand names or marketing copy. Ideally, a terminology constrained model would reliably produce these terms and use them appropriately even if they do not rank highly by the decoder.

Additionally, it is desirable that the addition of terminology constraints does not lead to large changes in the model's output. Since terminology constraints are only bound to a word or phrase, the model should only need to make minimal changes between the unconstrained and constrained output. Large changes in output may make it harder for a user to anticipate the effects of a constraint which may make constrained APE less useful in practice.

| | WMT'19 APE | | | Augmentation | | |
|---|---|---|---|---|---|---|
| | Term%↑ | TER↓ | BLEU↑ | Term%↑ | TER↓ | BLEU↑ |
| Do-nothing | 90.22 | 16.84 | 74.73 | 1.66 | 24.77 | 62.56 |
| MST Append | 94.08 | 16.62 | 75.16 | 7.47 | 24.92 | 61.80 |
| MST Append + *pretrain* | 94.08 | 16.46 | 75.25 | 18.67 | 23.70 | 64.38 |
| MST Append + *pretrain* + *ft* | 93.85 | **16.29** | **75.38** | 43.15 | **21.85** | **67.41** |
| MS LevT | 98.04 | 17.71 | 73.64 | 43.57 | 33.07 | 54.33 |
| MS LevT + *pretrain* | **99.09** | 17.18 | 74.22 | 52.70 | 29.79 | 60.24 |
| MS LevT + *pretrain* + *ft* | 98.41 | 17.00 | 74.66 | **63.07** | 29.66 | 60.47 |

Table 7: Results with data augmentation for the official APE data, as well on the augmented dataset consisting of synonyms and antonyms generated from Wiktionary. The size of the additional data for test set is 236.

| | Synonym | | | Antonyms | | |
|---|---|---|---|---|---|---|
| | Term%↑ | TER↓ | BLEU↑ | Term%↑ | TER↓ | BLEU↑ |
| MST Append | 7.33 | **1.31** | **97.80** | 8.88 | **1.06** | **98.01** |
| MST Append + *pretrain* | 16.75 | 2.81 | 94.19 | 28.88 | 3.81 | 92.94 |
| MST Append + *pretrain* + *ft* | 38.74 | 5.48 | 88.86 | 66.66 | 6.46 | 87.50 |
| MS LevT | 41.36 | 19.57 | 70.60 | 57.77 | 17.56 | 74.25 |
| MS LevT + *pretrain* | 47.12 | 18.33 | 72.27 | 82.22 | 13.28 | 79.42 |
| MS LevT + *pretrain* + *ft* | **43.97** | 18.25 | 73.25 | **77.78** | 11.99 | 80.41 |

Table 8: Structural change from the output of constrained models using the correct terminology. We split the dataset by synonyms and anotnyms, consisting of 191 and 45 samples respectively.

We refer to this behavior as *systematic copying*, i.e. the model should behave in a transparent and stable way, enforcing terminology constraints even when they strongly disagree with the decoder language model, while only making minimally necessary changes in the output to do so.

By harvesting terminology constraints from the training data, we run the risk that the model simply learns to draw some translation hints from the supplied terminology constraints, but does not actually learn this systematic copying behavior. That is, it never truly sees an out-of-sample constraint that is extremely unlikely from the perspective of the decoder language model.

To test whether our proposed models indeed learn this systematic copying behavior we perform a qualitative experiment, comparing model outputs when supplying different constraints for a source word, by varying whether the target language constraint was (a) the original target language constraint specified in the test set, (b) a target language synonym of the original constraint term, (c) a target language antonym of the original constraint term, or (d) a totally random term in the target language.

While antonym and random term constraints might not seem to correspond to realistic use cases, they let us to examine the effects of specifying a constraint where the source and target language terms are extremely semantically divergent. Additionally, they simulate scenarios where translations for names vary dramatically by region. For example, the cleaning product called "Mr. Clean" in the U.S. is called "Meister Proper" in Germany.

As can be seen in Figure 3, our qualitative exploration reveals that synonym, antonym, and random terminology constraints are frequently not included in the output. For example the MST model fails to generate the antonym *Verkleinerung*. This suggests that target side constraints that are unseen during training may be ignored by the model, and that the models are not learning to systematically copy arbitrary constraints.

## 8.1 Data Augmentation Experiment

The results of our qualitative exploration suggests that the model would benefit from seeing more semantically divergent terminology constraints during training. To that end, we propose a data augmentation experiment to increase the robustness of the APE models. We create novel training instances by replacing the target language term constraint with either a synonym or antonym (using Wiktionary), as well as replacing it's occurrence in the post-edited target translation. This results in 837,127 additional samples for eSCAPE corpus, and 2587 additional samples for official NMT data. See Figure 4 for examples.

We then train MST append and MS LevT with the augmented pretraining corpus. We experiment with using augmented data only for pretraining (*pretrain*) and for the fine-tuning process (*ft*). The result can be seen in Table 7.

Interestingly, on the WMT19 test set, the data augmentation helps with TER and BLEU while having only a slight effect on Term%. For the LevT, data augmentation helps all metrics.

Since the WMT19 APE test data contains few unusual constraints, the effect of the augmented data is relatively small. When we create antonym and synonyms examples from the WMT19 APE test data, we see fairly positive trends, with pretraining and fine-tuning yielding additive reductions in TER and gains in BLEU. This suggests that the augmentation method has a positive effect on the systematic copying behavior of the model.

### 8.2 Post-Edit Stability

To quantify the stability of the APE models, we compare the constrained APE output when given a target side synonym or antonym to the output of that same model under the original test set constraint using TER and BLEU. Under this setting, higher BLEU and lower TER indicate that the model makes minimal changes when inserting a semantically divergent constraint. We also report Term% to show how often the terminology was correctly translated given the input. We refer to a model with high Term% and BLEU but low TER as a stable model. Results of this experiment are shown in Table 8.

There are several takeaways from this experiment. First, the LevT TER scores are higher on average than the MST model suggesting that the LevT model is less stable, producing different translations for each target side constraint change.

Second, as sensitivity to constraints increases (i.e. Term% goes up), TER generally goes up, implying that models make more structural changes to the overall output in order to accommodate constraints. Future work on refinement tasks like APE may benefit from including an explicit objective function to encourage output stability.

Finally, synonyms are harder to translate than antonyms (i.e. Synonym Term% is lower than Antonym Term% for all models/training configurations). This may be because the original target side constraints are better represented in the decoder language model and are likely have higher probability than a synonym when either could be plausibly used in the same context. Antonyms may be less likely and therefore easier to override the preferences of decoder.

## 9 Conclusion and Future Work

This work introduces the terminology constrained APE task and several MST and LevT model variants for incorporating lexical constraints during post-editing. Furthermore, we show that constrained APE is necessary for preserving lexical constraints in a MT to APE pipeline. Evaluations on standard APE benchmarks show that terminology constraints are satisfied while improving the original MT quality. Finally, we show that the constrained APE models do not learn a robust systematic copying behavior, and propose a data augmentation method to help mitigate this issue. In future work, we hope to explore ways of modifying model architecture or training algorithms to further improve the systematic copying behavior.

## Acknowledgments

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

## References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Rajen Chatterjee, Christian Federmann, Matteo Negri, and Marco Turchi. 2019. Findings of the WMT 2019 Shared Task on Automatic Post-Editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 13–30, Florence, Italy. Association for Computational Linguistics.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the WMT 2018 Shared Task on Automatic Post-Editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 710–725, Belgium, Brussels. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training Neural Machine Translation to Apply Terminology Constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein Transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. MS-UEdin Submission to the WMT2018 APE Shared Task: Dual-Source Transformer for Automatic Post-Editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 835–839, Belgium, Brussels. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

WonKee Lee, Jaehun Shin, and Jong-Hyeok Lee. 2019. Transformer-based Automatic Post-Editing Model with Joint Encoder and Multi-source Attention of Decoder. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 112–117, Florence, Italy. Association for Computational Linguistics.

António V. Lopes, M. Amin Farajian, Gonçalo M. Correia, Jonay Trénous, and André F. T. Martins. 2019. Unbabel's Submission to the WMT2019 APE Shared Task: BERT-Based Encoder-Decoder for Automatic Post-Editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 118–123, Florence, Italy. Association for Computational Linguistics.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. ESCAPE: a Large-scale Synthetic Corpus for Automatic Post-Editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical Phrase-Based Post-Editing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 508–515, Rochester, New York. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically Constrained Neural Machine Translation with Levenshtein Transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543, Online. Association for Computational Linguistics.

Amirhossein Tebbifakhr, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source Transformer with Combined Losses for Automatic Post Editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 846–852, Belgium, Brussels. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.