Latent Part-of-Speech Sequences for Neural Machine Translation

Xuewen Yang[1], Yingru Liu[1], Dongliang Xie[2], Xin Wang[1], and Niranjan Balasubramanian[1,3]

[1]Stony Brook University
[2]Beijing University of Posts and Telecommunications
[1]{xuewen.yang,yingru.liu,x.wang}@stonybrook.edu
[2]xiedl@bupt.edu.cn
[3]niranjan@cs.stonybrook.edu

Abstract

Learning target side syntactic structure has been shown to improve Neural Machine Translation (NMT). However, incorporating syntax through latent variables introduces additional complexity in inference, as the models need to marginalize over the latent syntactic structures. To avoid this, models often resort to greedy search which only allows them to explore a limited portion of the latent space.

In this work, we introduce a new latent variable model, LaSyn, that captures the co-dependence between syntax and semantics, while allowing for effective and efficient inference over the latent space. LaSyn decouples direct dependence between successive latent variables, which allows its decoder to exhaustively search through the latent syntactic choices, while keeping decoding speed proportional to the size of the latent variable vocabulary. We implement LaSyn by modifying a transformer-based NMT system and design a neural expectation maximization algorithm that we regularize with part-of-speech information as the latent sequences. Evaluations on four different MT tasks show that incorporating target side syntax with LaSyn improves both translation quality, and also provides an opportunity to improve diversity.

1 Introduction

Syntactic information has been shown to improve the translation quality in NMT models. On the source side, syntax can be incorporated in multiple ways — either directly during encoding (Chen et al., 2018; Sennrich and Haddow, 2016; Eriguchi et al., 2016), or indirectly via multi-task learning to produce syntax informed representations (Eriguchi et al., 2017; Baniata et al., 2018; Niehues and Cho, 2017; Zaremoodi et al., 2018). On the target side, however, incorporating the syntax is more challenging due to the additional complexity in inference when decoding over latent states. To avoid this, existing methods resort to approximate inference over the latent states using a two-step decoding process (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017). Typically, the first stage decoder produces a beam of latent states, which serve as conditions to feed into the second stage decoder to obtain the target words. Thus, training and inference in these models can only explore a limited sub-space of the latent states.

In this work, we introduce LaSyn, a new target side syntax model that allows exhaustive exploration of the latent states to ensure a better translation quality. Similar to prior work, LaSyn approximates the co-dependence between syntax and semantics of the target sentences by modeling the joint conditional probability of the target words and the syntactic information at each position. However, unlike prior work, LaSyn eliminates the sequential dependence between the latent variables and simply infers the syntactic information at a given position based on the source text and the partial translation context. This allows LaSyn to search over a much larger set of latent state sequences. In terms of time complexity, unlike typical latent sequential models, LaSyn only introduces an additional term that is linear in the size of latent variable vocabulary and the length of the sentence.

We implement LaSyn by modifying a transformer-based encoder-decoder model. The implementation uses a hybrid decoder that predicts two posterior distributions: the probability of syntactic choices at each position $P(\mathbf{z}_n|\mathbf{x}, \mathbf{y}_{<n})$, and the probability of the word choices at each position conditioned on each of the possible values for the latent states $P(\mathbf{y}_n|\mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n})$. The model cannot be trained by directly optimizing the data log-likelihood

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 780–790, Hong Kong, China, November 3–7, 2019.

because of its non-convex property. We devise a neural expectation maximization (NEM) algorithm, whose E-step computes the posterior distribution of latent states under current model parameters, and M-step updates model parameters using gradients from back-propagation. Given some supervision signal for the latent variables, we can modify this EM algorithm to obtain a regularized training procedure. We use parts-of-speech (POS) tag sequences, automatically generated by an existing tagger, as the source of supervision.

Because the decoder is exposed to more latent states during training, it is more likely to generate diverse translation candidates. To obtain diverse sequences, we can decode the most likely translations for different POS tag sequences. This is a more explicit and effective way of performing diverse translation than other methods based on diverse or re-ranking beam search (Vijayakumar et al., 2018; Li and Jurafsky, 2016), or coarse codes planning (Shu and Nakayama, 2018).

We evaluate LaSyn on four translation tasks. Evaluations show that LaSyn outperforms models that only use partial exploration of the latent states for incorporating target side syntax. A diversity based evaluation also shows that when using different POS tag sequences during inference, LaSyn produces more diverse and meaningful translations compared to existing models.

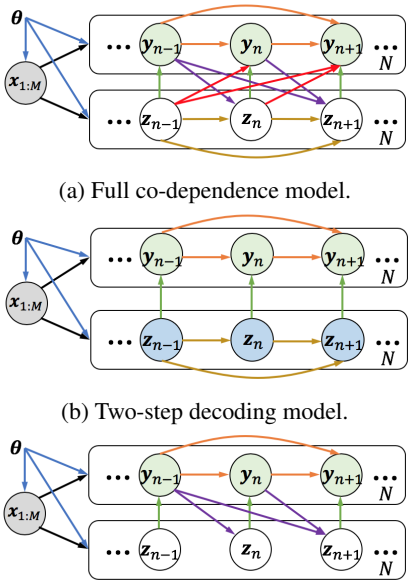
2 A Latent Syntax Model for Decoding

In a standard sequence-to-sequence model, the decoder directly predicts the target sequence $\mathbf{y}$ conditioned on the source input $\mathbf{x}$. The translation probability $P(\mathbf{y}|\mathbf{x})$ is modeled directly using the probability of each target word $\mathbf{y}_n$ at time step n conditioned on the source sequence $\mathbf{x}$ and the current partial target sequence $\mathbf{y}_{<n}$ as follows:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{n=1}^{N} P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \qquad (1)$$

where, $\boldsymbol{\theta}$ denotes the parameters of both the encoder and the decoder.

In this work, we model syntactic information of target tokens using an additional sequence of variables, which captures the syntactic choices[1] at each time step. There are multiple ways of incorporating this additional information in a sequence-to-sequence model.



(a) Full co-dependence model.

(b) Two-step decoding model.

(c) LaSyn: Our Latent syntax model

Figure 1: Target-side Syntax Models: (a) An ideal solution that captures full co-dependence between syntax and semantics. (b) A widely-used two-step decoding model (Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017). (c) LaSyn, our latent syntax model that uses non-sequential latent variables for exhaustive search of latent states.

An ideal solution should capture the co-dependence between syntax and semantics. In a sequential translation process, the word choices at each time step depend on both the semantics and the syntax of the words generated at the previous time steps. The same dependence also holds for the syntactic choices at each time step. Figure 1a shows a graphical model that captures this *full* co-dependence between the syntactic variable sequence $\mathbf{z}_1, \ldots, \mathbf{z}_N$ and the output word sequence $\mathbf{y}_1, \ldots, \mathbf{y}_N$. Such a model can be implemented using two decoders, one to decode syntax and the other to decode output words. The main difficulty, however, is that inference in this model is intractable since it involves marginalizing over the latent $\mathbf{z}$ sequences.

To keep inference tractable, existing approaches treat syntactic choices $\mathbf{z}$ as observed sequential variables (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017), as shown in Figure 1b. These models use a two-stage decoding process, where for each time step they

[1]The variables can be used to model any linguistic information that can be expressed as choices for each word position (e.g., morphological choices).

first produce most likely latent state $\mathbf{z}_n$ and then use this as input to a second stage that decodes words. However, this process is unsatisfactory in two respects. First, the inference of syntactic choices is still approximate as it does not explore the full space of $\mathbf{z}$. Second, these models are not well-suited for controllable or diverse translations. Using such a model to decode from an arbitrary $\mathbf{z}$ sequence is a divergence from its training, where it only learns to decode from a limited space of $\mathbf{z}$ sequences.

2.1 Model Description

Our goal is to design a model that allows for exhaustive search over syntactic choices. We introduce LaSyn, a new latent model shown in Figure 1c. The syntactic choices are modeled as *true latent variables* i.e., unobserved variables. Compared to the ideal model in Figure 1a, LaSyn includes two simplifications for tractability: (i) The dependence between successive syntactic choices is modeled indirectly, via the word choices made in the previous time steps. (ii) The word choice at each position depends only on the syntactic choice at the current position and the previous predicted words. Dependence on previous syntactic choices is modeled indirectly.

Under this model, the joint conditional probability of the target word $\mathbf{y}_n$ together with its corresponding latent syntactic choice $\mathbf{z}_n$[2] is given by:

$$P(\mathbf{y}_n, \mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}) = P(\mathbf{y}_n | \mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n}) \times P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}) \quad (2)$$

We implement LaSyn by modifying the Transformer-based encoder-decoder architecture (Vaswani et al., 2017). As shown in Figure 2, LaSyn consists of a shared encoder for encoding source sentence $\mathbf{x}$ and a hybrid decoder that manages the decoding of the latent sequence $\mathbf{z}$ (left branch) and the target sentence $\mathbf{y}$ (right branch) separately.

The encoder consists of the standard self-attention layer, which generates representations of each token in the source sentence $\mathbf{x}$. The hybrid decoder consists of a self-attention layer that encodes the output generated thus far (i.e., the partial translation), followed by a inter-attention layer

[2]Note that $\mathbf{z}_n \in V_\mathbf{z}$, where $V_\mathbf{z}$ is the vocabulary of latent syntax for the target, which differs from language to language.

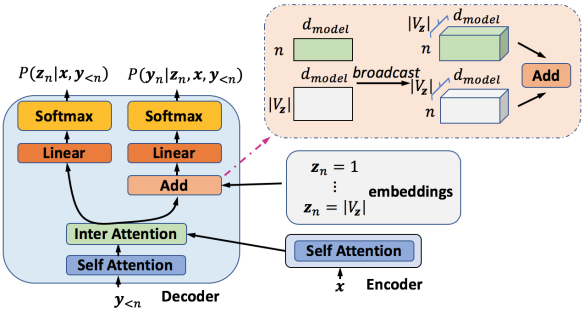


Figure 2: The architecture of LaSyn.

which computes the attention across the encoder and decoder representations.

The decoder's left branch predicts the latent variable distribution $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ by applying a simple linear transformation and softmax on the inter-attention output, which contains information about the encoded input $\mathbf{x}$ and the partial translation $\mathbf{y}_{<n}$.

The right branch predicts the target word distribution $P(\mathbf{y}_n | \mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n})$ using the inter-attention output and the embeddings of all the available choices for $\mathbf{z}_n$. The choices for $\mathbf{z}_n$ are represented as embeddings that can be learned during training. We then combine the inter-attention output and the latent choice embeddings through an `Add` operation, which is a simple composition function that captures all combinations of additive interactions between the two. The dimension of the inter-attention is $n \times d_{model}$ and that of the latent embeddings is $|V_\mathbf{z}| \times d_{model}$, where $|V_\mathbf{z}|$ is the total number of choices for $\mathbf{z}_n$ or the size of the latent variable vocabulary. We broadcast them to the same dimension $n \times |V_\mathbf{z}| \times d_{model}$ and then simply add them together point-wise as shown in Figure 2. This is then fed to a linear transform and softmax over the target word vocabulary.

2.2 Inference with Exhaustive Search for Latent States

When using additional variables to model target side syntax, exact inference requires marginalizing over these additional variables.

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z} \in F(\mathbf{z})} P(\mathbf{y}|\mathbf{z}, \mathbf{x}) P(\mathbf{z}|\mathbf{x}) \quad (3)$$

To avoid this exponential complexity, prior works use a two-step decoding process with models similar to the one shown in Figure 1b. They use greedy or beam search to explore a subset $B(\mathbf{z})$ of

the latent space to compute the posterior distribution as follows:

$$P(\mathbf{y}|\mathbf{x}) \simeq \sum_{\mathbf{z}\in B(\mathbf{z})} P(\mathbf{y}|\mathbf{z},\mathbf{x})P(\mathbf{z}|\mathbf{x}) \quad (4)$$

Finding the most likely translation using LaSyn also requires marginalizing over the latent states. However, because the latent states in LaSyn don't directly depend on each other, we can exhaustively search over the latent states. In particular, we can show that when $\mathbf{y}$ is fixed (observed), the $\{\mathbf{z}_n\}_{n=1}^{N}$ variables are d-separated (Bishop, 2006) i.e., are mutually independent. As a result, the time complexity for searching latent sequence $\mathbf{z}$ drops from $|V_\mathbf{z}|^N$ to $N|V_\mathbf{z}|$.

The posterior distribution for the translation probability at a time step n can be computed as follows:

$$\begin{aligned} &P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n}) \\ &= \sum_{\mathbf{z}_n\in F(\mathbf{z}_n)} P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \\ &= \sum_{\mathbf{z}_n\in F(\mathbf{z}_n)} P(\mathbf{y}_n|\mathbf{z}_n,\mathbf{x},\mathbf{y}_{<n}) \times P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \end{aligned} \quad (5)$$

where, $F(\mathbf{z}_n)$ is the full search space of latent states $\mathbf{z}_n$ and the joint probability is factorized as specified in Equation 2.

For decoding words, we use standard beam search[3] with $P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})$ as the beam cost. With this inference scheme, we can easily control decoding for diversity, by feeding different $\mathbf{z}$ sequences to the right branch of the decoder and decode diverse $\mathbf{y}_n$ by directly using $P(\mathbf{y}_n|\mathbf{z}_n,\mathbf{x},\mathbf{y}_{<n})$ as the beam cost. Unlike the two-step decoding models which only evaluate a small number of $\mathbf{z}_n$ values at each time step (constrained by beam size), LaSyn evaluates all possible values for $\mathbf{z}_n$ at each time step, while avoiding the evaluation of all possible sequences.

2.3 Training with Neural Expectation Maximization

The log-likelihood of LaSyn's parameters $\boldsymbol{\theta}$[4] computed on one training pair $\langle\mathbf{x},\mathbf{y}\rangle \in D$ is given by:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log \sum_{\mathbf{z}_n\in V_\mathbf{z}} P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \end{aligned} \quad (6)$$

Directly optimizing the log-likelihood function (equation 6) with respect to model parameter $\boldsymbol{\theta}$ is challenging because of the highly non-convex function $P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})$ and the marginalization over $\mathbf{z}_n$.[5] Alternatively, we optimize the system parameters by Expectation Maximization (EM).

Using Jensen's inequality, equation (6) can be re-written as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} \log \sum_{\mathbf{z}_n\in V_\mathbf{z}} Q(\mathbf{z}_n) \frac{P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})}{Q(\mathbf{z}_n)} \\ &\geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n\in V_\mathbf{z}} Q(\mathbf{z}_n) \log \frac{P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})}{Q(\mathbf{z}_n)} \\ &= \mathcal{L}_{lower}(Q,\boldsymbol{\theta}) \end{aligned} \quad (7)$$

where $\mathcal{L}_{lower}(Q,\boldsymbol{\theta})$ is the lower bound of the log-likelihood and Q is any auxiliary probability distribution defined on $\mathbf{z}_n$. $\boldsymbol{\theta}$ is omitted from the expression for simplicity.

We set $Q(\mathbf{z}_n) = P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old})$, the probability of the latent state computed by the decoder (shown in the left branch in Figure 2). Substituting this in equation (7), we obtain the lower bound as

$$\begin{aligned} &\mathcal{L}_{lower}(Q,\boldsymbol{\theta}) = \\ &\sum_{n=1}^{N} \sum_{\mathbf{z}_n\in V_\mathbf{z}} P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \times \log P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \\ &- P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \times \log P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \\ &= \mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) + C \end{aligned} \quad (8)$$

[3]Note our primary goal is to perform exhaustive search in the latent space. Search in the target vocabulary space remains exponential in our model.

[4]This includes the trainable parameters of the encoder, decoder, and the latent state embeddings.

[5]Note that marginalization is an issue during training, unlike in inference. As $P(y_n,z_n)$ is already an non-convex function with respect to θ, summing $P(y_n,z_n)$ over different values of z_n makes the function more complicated. Besides, we also need to compute gradients to update the parameters and computing the gradient of a log-of-sum function is costly and unstable. During the translation, we only need to compute the value of $\mathcal{L}(\theta)$ as score for beam searching. Therefore, the marginalization is not an issue.

where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{n=1}^{N} \sum_{\mathbf{z}_n \in V_{\mathbf{z}}} P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{\leq n}; \boldsymbol{\theta}^{old}) \times \log P(\mathbf{y}_n, \mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}; \boldsymbol{\theta}) \quad (9)$$

EM algorithm for optimizing $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ consists of two major steps. In the E-step, we compute the posterior distribution of $\mathbf{z}_n$ with respect to $\boldsymbol{\theta}^{old}$ by

$$\gamma(\mathbf{z}_n = i) = P(\mathbf{z}_n = i | \mathbf{x}, \mathbf{y}_{\leq n}) = \frac{P(\mathbf{y}_n, \mathbf{z}_n = i | \mathbf{x}, \mathbf{y}_{<n})}{\sum_{\mathbf{z}_n = j} P(\mathbf{y}_n, \mathbf{z}_n = j | \mathbf{x}, \mathbf{y}_{<n})} \quad (10)$$

where $\gamma(\mathbf{z}_n = i)$ is the responsibility of $\mathbf{z}_n = i$ given $\mathbf{y}_n$, and can be calculated by equation (2).

In the M-step, we aim to find the configuration of $\boldsymbol{\theta}$ that would maximize the expected log-likelihood using the posteriors computed in the E-step. In conventional EM algorithm for shallow probabilistic graphical model, the M-step is generally supposed to have closed-form solution. However, we model the probabilistic dependencies by deep neural networks, where $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ is highly non-convex and non-linear with respect to network parameters $\boldsymbol{\theta}$. Therefore, there exists no analytical solution to maximize it. However, since deep neural network is differentiable, we can update $\boldsymbol{\theta}$ by taking a gradient ascent step:

$$\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} + \eta \frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})}{\partial \boldsymbol{\theta}}, \quad (11)$$

The resulting algorithm belongs to the class of *generalized EM algorithms* and is guaranteed (for a sufficiently small learning rate η) to converge to a (local) optimum of the data log likelihood (Wu, 1983).

2.4 Regularized EM training

The EM training we derived does not assume any supervision for the latent variables $\mathbf{z}$. This can be seen as inferring the latent syntax of the target sentences by clustering the target side tokens into $|V_{\mathbf{z}}|$ different categories. Given some token-level syntactic information, we can modify the training procedure to regularize the generation of latent sequence $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ such that true latent sequences have higher probabilities under the model. In this work, we consider parts-of-speech sequences of the target sentences for regularization.

The regularized EM training objective is thus redefined as

$$\mathcal{L}_{total}(\boldsymbol{\theta}) = \mathcal{L}_{lower}(\boldsymbol{\theta}) + \lambda \mathcal{L}_{\mathbf{z}}(\boldsymbol{\theta}), \quad (12)$$

where $\mathcal{L}_{lower}(\boldsymbol{\theta})$ is the EM lower bound in equation (7) and $\mathcal{L}_{\mathbf{z}}(\boldsymbol{\theta})$ denotes cross entropy loss between $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ and the true POS tag sequences and λ is a hyper-parameter that controls the impact of the regularization.

This regularized training algorithm is shown in Algorithm 1.

Algorithm 1 Training NMT with latent POS tag sequences through Regularized Neural EM

Objective: Maximize the log likelihood function $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ with respect to $\boldsymbol{\theta}$ over observed variable $\mathbf{y}$ and latent variable $\mathbf{z}$, governed by parameters $\boldsymbol{\theta}$.
Input: Parallel corpus training data $\langle \mathbf{x}, \mathbf{y} \rangle$; POS tag sequence $\mathbf{z}$ of target sentence; the number of EM update steps per batch K.
Initialize: Initialize random values for the parameters $\boldsymbol{\theta}^{old}$.
while Training loss has not converged **do**
  Select $\langle \mathbf{x}, \mathbf{y} \rangle \in D$, parse $\mathbf{z}$ of $\mathbf{y}$.
  for $k \leftarrow 1$ to K **do**
    1. E-step: Evaluate $\gamma(\mathbf{z}_n = i)$ for $i \leq |V_{\mathbf{z}}|$.
    2. M-step: Evaluate $\boldsymbol{\theta}^{new}$ given by equation (11).
    3. Let $\boldsymbol{\theta}^{old} \leftarrow \boldsymbol{\theta}^{new}$.
  end for
end while

3 Evaluation

We evaluate LaSyn on four translation tasks, including three with moderate sized datasets IWSLT 2014 (Cettolo et al., 2015) German-English (De-En), English-German (En-De), English-French (En-Fr), and one with a relatively larger dataset, the WMT 2014 English-German (En-De). We describe the datasets in more details in the appendix.

We compare against three types of baselines: (i) general Seq2Seq models that use no syntactic information, (ii) models that incorporate source side syntax directly, and multitask learning models which include syntax indirectly, and (iii) models that use syntax on the target side. We also define a LaSyn **Empirical Upper Bound (EUB)**, which is our proposed model using true POS tag sequences for inference.

We use BLEU as the evaluation metric (Papineni et al., 2002) for translation quality. For diverse translation evaluation, we utilize *distinct-1* score (Li et al., 2016) as the evaluation metric,

which is the number of distinct unigrams divided by total number of generated words.

For all translation tasks, we choose the *base* configuration of Transformer with $d_{model} = 512$. During training, we choose Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ with initial learning rate is 0.0002 with 4000 warm-up steps. We describe additional implementation and training details in the Appendix.

3.1 Results on IWSLT'14 Tasks

Table 1 compares LaSyn versions against some of the state-of-the-art models on the IWSLT'14 dataset. LaSyn-K rows show results when varying the number of EM update steps per batch (K).

On the De-En task, LaSyn provides a 1.7 points improvements over the Transformer baseline, demonstrating that the LaSyn's improvements come from incorporating target side syntax effectively. This result is also better than a transformer-based source side syntax model by 1.5 points. LaSyn results are also better than the published numbers for LSTM-based models that use multi-task learning for source side and models that uses target side syntax. Note that since the underlying architectures are different, we only present these results to show that the results with LaSyn are comparable with other models that have incorporated syntax.

On the En-De task, our model achieves 29.2 in terms of BLEU score, with 2.6 points improvement over the Transformer baseline and 2.4 points improvement over Transformer-based Source Side Syntax model. Compared with NPMT (Huang et al., 2018), which is a BiLSTM based model, we achieve 3.84 point improvement.

On the En-Fr task, our model set a new state-of-the-art with a BLEU score of 40.6, which is 1.7 points improvement over the second best model which uses Transformer to incorporate source side syntax knowledge. Our model also surpasses the basic Transformer model by about 2.1 points.

We notice that across all tasks, the performance of our model improves with number of EM update steps per batch (K). With larger K values, we get better lower bounds $\mathcal{L}_{lower}(\boldsymbol{\theta})$ on each training batch, thus leading to better optimization. For update steps beyond $K > 5$, the performance does not improve any more.

Last, the EUB row indicates the performance that can be obtained when feeding in the true POS tags. The large improvement here shows the potential for improvement when modeling target side syntax.

Table 2 shows one example where LaSyn produces correct translations for a long input sentence. The output of LaSyn is close to the reference and the output of LaSyn when given the gold POS tag sequence is even better, demonstrating the benefits of modeling syntax. The transformer model however fails to decode the later portions of the long input accurately.

3.2 Speed

We compare the speeds of our (un-optimized) implementation of LaSyn with a vanilla transformer with no latents in its decoder. Table 3 shows the training time per epoch, and the inference time for the whole test set. computed on the IWSLT'14 De-En task. When $K = 1$, LaSyn takes almost twice as much time as the vanilla transformer for training. Increasing K increases training time further. For inference, LaSyn takes close to four times as much time compared to the vanilla Transformer. In terms of complexity, LaSyn only adds a linear term (in POS tag size to the decoding complexity. Specifically, its decoding complexity is $B \times O(m) \times O(|V_{\mathbf{z}}| \times N)$ where B is beam size, m is a constant proportional to the tag set size and N is output size. As the table shows, empirically, our current implementation incurs $m \simeq 4$. We leave further optimizations for future work.

3.3 Diversity

We compare the diversity of translations using *distinct-1* score (Li et al., 2016), which is simply the number of distinct unigrams divided by total number of generated words. We use our model to generate 10 translations for each source sentence of the test dataset. We then compare our results with baseline Transformer. The result is shown in Table 4. Much like translation quality, LaSyn's diversity increases with number of EM updates and is better than diversity of the transformer and a source side encoder model.

3.3.1 Controlling Diversity with POS Sequences

One of the main strengths of LaSyn is that it can generate translations conditioned on a given POS sequence. First, we present some examples that we generate by decoding over different POS tag sequences. Given a source sentence, we use

Method Type	Model	BLEU		
		De-En	En-De	En-Fr
BiLSTM	BiLSTM (Denkowski and Neubig, 2017)	–	–	34.8
	Dual Learning (Wang et al., 2018b)	32.35	–	–
	AST (Cheng et al., 2018)	–	–	38.03
	NPMT (Huang et al., 2018)	–	25.36	–
Multi-Task (BiLSTM)	MTL-NMT (Niehues and Cho, 2017)	27.78	–	–
Source Side Syn. (Transformer)	Source-NMT (Sennrich and Haddow, 2016)	33.5	26.8	38.9
Target Side Syn. (BiLSTM)	DSP-NMT (Shu and Nakayama, 2018)	29.78	–	–
	Tree-decoder (Wang et al., 2018a)	32.65	–	–
Transformer	Transformer	33.3	26.6	38.5
	LaSyn (Unsupervised)	30.8	25.2	34.5
	LaSyn (K=1)	34.63	28.1	39.7
	LaSyn (K=3)	34.91	28.9	40.4
	LaSyn (K=5)	**35.0**	**29.2**	**40.6**
	LaSyn **EUB**	51.4	47.3	54.2

Table 1: **IWSLT'14 English-German and English-French results** - shown are the BLEU scores of various models on TED talks translation tasks. We highlight the **best** model in bold.

SRC	letztes jahr habe ich diese beiden folien gezeigt , um zu veranschaulichen , dass die arktische eiskappe , die fr annhernd drei millionen jahre die grsse der unteren 48 staaten hatte , um 40 prozent geschrumpft ist .
REF	last year i showed these two slides so that demonstrate that the arctic ice cap , which for most of the last three million years has been the size of the lower 48 states , has shrunk by 40 percent .
Transformer	last year , i showed these two slides to illustrate that the arctic ice caps that had the size of the lower 48 million states to 40 percent .
LaSyn	last year , i showed these two slides to illustrate that the arctic ice cap , which for nearly three million years had the size of the lower 48 states , was shrunk by 40 percent .
LaSyn (groundtrue POS)	last year i showed these two slides just to illustrate that the arctic ice cap , which for nearly about the last three million years has been the size of the lower 48 states , has shrunk by 40 percent .

Table 2: Translation examples on IWSLT'14 De-En dataset from our model and the Transformer baseline. We put correct translation segment in blue and highlight the wrong one in red.

Model	Training Time/Epoch↓	Inference Time↓
Transformer	**3.6** min	**12.8** s
LaSyn (K=1)	6.3 min	56.1 s
LaSyn (K=3)	18.1 min	55.6 s
LaSyn (K=5)	28.0 min	55.6 s

Table 3: **IWSLT'14 De-En training and inference speed evaluation**. ↓ means the smaller the better. We highlight the **best** model in bold.

Model	distinct-1		
	De-En	En-De	En-Fr
Transformer	0.231	0.242	0.258
Source-NMT	0.232	0.244	0.260
LaSyn (Unsupervised)	0.228	0.231	0.239
LaSyn (K=1)	0.237	0.251	0.265
LaSyn (K=3)	0.241	0.253	0.270
LaSyn (K=5)	**0.245**	**0.255**	**0.273**
LaSyn **EUB**	0.328	0.516	0.354

Table 4: **IWSLT'14 En-De/De-En/En-Fr diversity translation evaluation**. We highlight the **best** model in bold.

LaSyn to provide the most-likely target pos tag sequence. Then, we obtain a random set of valid POS tag sequences that differ from this maximum likely sequence by some edit distance. For each of these randomly sampled POS tag sequences, we let LaSyn generate a translation that fits the sequence. Table 5 shows some example sentences. LaSyn is able to generate diverse translations that reflect the sentence structure implied by the input POS tags. However, in trying to fit the translation into the specified sequence, it deviates somewhat from the ideal translation.

To understand how diversity plays against translation quality, we also conduct a small scale quantitative evaluation. We pick a subset of the test dataset, and for each source sentence in this subset, we sample 10 POS tag sequences whose edit distance to their corresponding Top-1 POS tag sequence equal to a specific value, we then use them

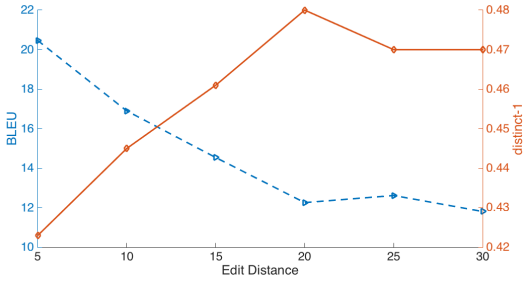


Figure 3: Diversity vs. Translation Quality: BLEU and *distinct-1* scores for targets decoded using POS sequences of increasing edit distance.

SRC	und natrlich auch , wie nimmt gestaltung einfluss auf die wahrnehmung .
REF	and of course how design affects perception .
0	cc in nn wrb nn vbz vbz nn .
	and of course how design is affecting perception .
20	ls rb vb in dt nn vbz jj jj jj .
	i also think that the design is affecting perception .
30	rb , dt nn nn dt nn in prp rb .
	also , the way design adds influence in perception too .
30	prp vbz in prp vb vbn rb in nn .
	it turns out it included design impact on perception .

Table 5: Examples of translations decoded from specified POS sequences with different edit distances (shown as values in first column). SRC: source sentence. REF: reference sentence.

to decode W translations. We calculate their final BLEU and *distinct-1* scores. The results are shown in Figure: 3. As the edit distance increases, diversity increases dramatically but at the cost of translation quality. Since the POS tag sequence acts as a template for generation, as we move further from the most likely template, the model struggles to fit the content accurately. Understanding this trade-off can be useful for re-ranking or other scoring functions.

3.4 Results on WMT'14 En-De

To assess the impact on a larger dataset, we show results on the WMT'14 English-German in table 6. Compared to the previously reported systems, we see that our transformer implementation is a strong baseline. LaSyn produces small gains, with the best gain at K=5 – a BLEU score improvement of 0.6. This demonstrates that syntactic information can contribute more to the increase of translation quality on a smaller dataset.

4 Related Work

Attention-based Neural Machine Translation (NMT) models have shown promising results in various large scale translation tasks (Bahdanau et al., 2015; Luong et al., 2015; Sennrich et al., 2016; Vaswani et al., 2017) using an `Seq2Seq` structure. Many Statistical Machine Translation (SMT) approaches have leveraged benefits from modeling syntactic information (Chiang et al., 2009; Huang and Knight, 2006; Shen et al., 2008). Recent efforts have demonstrated that incorporating syntax can also be useful in neural methods as well.

Model	BLEU
BiRNN+GCN (Bastings et al., 2017)	23.9
ConvS2S (Gehring et al., 2017)	25.16
MoE (Shazeer et al., 2017)	26.03
Transformer (base)	27.3
LaSyn (K=1) (base)	27.6
LaSyn (K=3) (base)	27.8
LaSyn (K=5) (base)	**27.9**

Table 6: **WMT'14 English-German results** - shown are the BLEU scores of various models on TED talks translation tasks. We highlight the **best** model in bold.

One branch uses features on the source side to help improve the translation performance (Sennrich and Haddow, 2016; Morishita et al., 2018; Eriguchi et al., 2016). Sennrich *et al.* (2016) explored linguistic features like lemmas, morphological features, POS tags and dependency labels and concatenate their embeddings with sentence features to improve the translation quality. In a similar vein, Morishita *et al.* (2018) and Eriguchi *et al.* (2016), incorporated hierarchical subword features and phrase structure into the source side representations. Despite the promising improvements, these approaches are limited in that the trained translation model requires the availability of external tools during inference – the source text needs to be processed first to extract syntactic structure (Eriguchi et al., 2017).

Another branch uses multitask learning, where the encoder of the NMT model is trained to produce multiple tasks such as POS tagging, named-entity recognition, syntactic parsing or semantic parsing (Eriguchi et al., 2017; Baniata et al., 2018; Niehues and Cho, 2017; Zaremoodi et al., 2018). These can be seen as models that implicitly generate syntax informed representations during encoding. With careful model selection, these methods have demonstrate some benefits in NMT.

The third branch directly models the syntax of

the target sentence during decoding (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017; Bastings et al., 2017; Li et al., 2018). Aharoni *et al.* (2017) treated constituency trees as sequential strings and trained a `Seq2Seq` model to translate source sentences into these tree sequences. Wang *et al.* (2018a) and Wu *et al.* (2017) proposed to use two RNNs, a Rule RNN and a Word RNN, to generate a target sentence and its corresponding tree structure. Gu *et al.* (2018) proposed a model to translate and parse at the same time. However, apart from the complex tree structure to model, they all have a similar architecture as shown in Figure 1b, which limits them to only exploring a small portion of the syntactic space during inference.

LaSyn uses simpler parts-of-speech information in a latent syntax model, avoiding the typical exponential search complexity in the latent space with a linear search complexity and is optimized by EM. This allows for better translation quality as well as diversity. Similar to our work, (Shankar et al., 2018) and (Shankar and Sarawagi, 2019) proposed a latent attention mechanism to further reduce the complexity of model implementation by taking a top-K approximation instead of the EM algorithm as in LaSyn.

5 Conclusion

Modeling target-side syntax through true latent variables is difficult because of the additional inference complexity. In this work, we presented LaSyn, a latent syntax model that allows for efficient exploration of a large space of latent sequences. This yields significant gains on four translation tasks, IWSLT'14 English-German, German-English, English-French and WMT'14 English-German. The model also allows for better decoding of diverse translation candidates. This work only explored parts-of-speech sequences for syntax. Further extensions are needed to tackle tree-structured syntax information.

6 Acknowledgements

This work is supported in part by the National Science Foundation under Grants NSF CNS 1526843 and IIS 1815358. We thank Jiangbo Yuan, Wanying Ding, Yang Liu, Guillaume Rabusseau and Jeffrey Heinz for valuable discussions.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 132–140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Laith H. Baniata, Seyoung Park, and Seong-Bae Park. 2018. A multitask-based neural machine translation model with part-of-speech tags integration for arabic dialects. *Applied Sciences*, 8.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2015. Report on the 11 th iwslt evaluation campaign , iwslt 2014. In *Proceedings of IWSLT 2014*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4792–4799.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1766.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the*

55th Annual Meeting of the Association for Computational Linguistics, pages 72–78.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252.

Jetic Gū, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 401–413.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 240–247.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372.

Xintong Li, Lemao Liu, Zhaopeng Tu, Shuming Shi, and Max Meng. 2018. Target foresight based attention for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1380–1390.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2018. Improving neural machine translation by incorporating hierarchical subword features. In *COLING*, pages 618–629.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL 2016*.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *EMNLP*.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, pages 577–585.

Raphael Shu and Hideki Nakayama. 2018. Discrete structural planning for neural machine translation.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *AAAI*, pages 7371–7379.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018a. A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, GuiQuan liu, and Tie-Yan Liu. 2018b. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

CF Jeff Wu. 1983. On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 698–707.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.