A Loss-Augmented Approach to Training Syntactic Machine Translation Systems

Tong Xiao, Derek F. Wong, *Member, IEEE*, and Jingbo Zhu

***Abstract*—Current syntactic machine translation (MT) systems implicitly use beam-width unlimited search in learning model parameters (e.g., feature values for each translation rule). However, a limited beam-width has to be adopted in decoding new sentences, and the MT output is in general evaluated by various metrics, such as BLEU and TER. In this paper, we address: 1) the mismatch of adopted beam-widths between training and decoding; and 2) the mismatch of training criteria and MT evaluation metrics. Unlike previous work, we model the two problems in a single training paradigm simultaneously. We design a loss-augmented approach that explicitly considers the limited beam-width and evaluation metric in training, and present a simple but effective method to learn the model. By using beam search and BLEU-related losses, our approach improves a state-of-the-art syntactic MT system by +1.0 BLEU on Chinese-to-English and English-to-Chinese translation tasks. It even outperforms seven previous training approaches over 0.8 BLEU points. More interestingly, promising improvements are observed when our approach works with TER.**

***Index Terms*—Loss-augmented training, machine translation, syntax-based model.**

I. Introduction

In Machine Translation (MT), syntactic approaches have had witnessed progress over the past decade. Many good models have been developed. The simplest of these is learning formally syntactic translation from word-aligned string pairs where translation is not required to respect syntactic annotation [1]. More sophisticated models can benefit from parse trees on either language side or both [2]–[6]. These systems have good abilities in handling long-distance dependency and syntactic movement. For example, systems based on source or target syntax have shown very promising results in several MT evaluation tasks [4], [7], and modern systems learned from syntax on both language sides can achieve state-of-the-art performance on Chinese-to-English translation which is a well-known difficult task [6].

Despite differences in detailed modeling, the learning procedure of syntactic MT systems resembles the usual pipeline in Statistical Machine Translation (SMT): 1) translation is modeled via a weighted log-linear model where each translation rule/derivation is associated with a number of translation features; 2) then translation rules and features are learned on large-scale bitext (call it *training*)[1]; 3) then feature weights are tuned on a handful of sentences with reference translations (call it *tuning*). After that, new sentences can be decoded with the learned model (call it *decoding*) and the translation is evaluated by various metrics, e.g., BLEU [8] and TER [9]. Here beam pruning and/or other aggressive pruning methods have to be adopted due to the large complexity of underlying structure of sentences. However, too few derivations are harmful to syntactic translation [6]. For good performance, syntactic MT systems have to access a large number of rule derivations and run slower than the phrase-based counterparts.[2]

In most syntactic MT systems, the training process is straightforward: we define a training criterion and learn features by the criterion. For example, a widely used method is that translation rules are extracted from a parsed bilingual corpus, and then probability-like features are learned by relative frequency estimation. Such a method is easy to implement but there appear mismatches between training and decoding. In particular,

1) The search strategies are mismatched. In decoding, exact search is impractical and beam pruning is employed to throw away all hypotheses beyond the beam width. Yet current training methods directly learn features by accessing all possible derivations over the given string/tree pair, as if exact search is always applicable. In other words, beam-width unlimited search is implicitly assumed in training, whereas beam-width limited search is adopted in decoding.
2) The training criterion and MT evaluation metric are mismatched. Learning MT features are generally designed for a maximum likelihood story. However, the evaluation metric is based on some different factors, e.g., the n-gram precision as in BLEU.

Manuscript received October 10, 2015; revised February 10, 2016, May 6, 2016, and June 14, 2016; accepted July 13, 2016. Date of publication July 27, 2016; date of current version September 2, 2016. The work of T. Xiao and J. Zhu was supported in part by the National Science Foundation of China under Grants 61272376, 61300097, and 61432013. The work of D. F. Wong was supported in part by the Science and Technology Development Fund, Macao S.A.R. (FDCT) and the University of Macau under References 057/2014/A and MYRG2015-00175-FST. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Imed Zitouni. *(Corresponding author: Tong Xiao.)*

T. Xiao and J. Zhu are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110004, China (e-mail: xiaotong@mail.neu.edu.cn; zhujingbo@mail.neu.edu.cn).

D. F. Wong is with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: derekfw@umac.mo).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2594383

[1]We assume that the language model is ready in advance. It is not involved in the training process of MT systems in this paper.

[2]For phrase-based models with an integrated n-gram language model, a left-to-right decoder has a time complexity of $2^m V^{n-1} m^2$, where m is the length of source sentence, V is the size of target-language vocabulary, and n is the order of language model. For syntactic models, even a binary grammar with L distinct non-terminal labels results in a time complexity of $m^3 V^{4(n-1)} L$ if we use the CKY (Cocke–Kasami–Younger) algorithm. In most large-scale systems, we decode input source sentences with less than 40 words but generate the target string over a vocabulary of over 500 K entries. As a result, syntax-based models have a much higher complexity than phrase-based models when we develop practical systems.

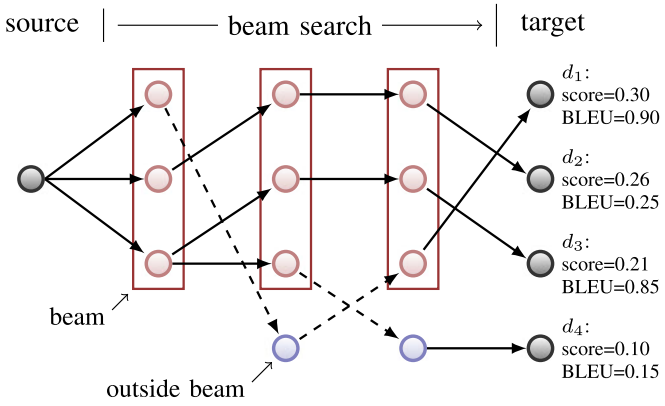


Fig. 1. Illustration of the mismatch problems—the decoder has to choose a derivation that is not the "best" one learned by the trainer.

The mismatch problems may result in an unpleasant situation: desired derivations cannot be kept within the beam even when we decode a training sentence with the parameters learned from it. See Fig. 1 for an example of derivations (or decoding paths) in multi-stage beam search. In Fig. 1 d_1 is the best "path" predicted by the model learned via maximum likelihood estimation; whereas the decoder prunes out d_1 at the second stage when we decode the source sentence s with beam search. Instead it chooses an "always-in-beam" path d_2 with a bad translation (as indicated by the BLEU score). To make matters worse, a good-quality derivation d_3 has a low rank although it survives in beam pruning.

A natural solution to these problems is to take the mismatch problem into account and guide the training process towards models that make use of the real factors in decoding, that is, we prefer to learn the MT model from deviations that are less likely to be pruned and are of a high BLEU score. Several research groups have been aware of this and made good attempts. E.g., Liu and Huang [10] and Yu *et al.* [11] proposed to model search errors for MT training and learned feature weights on bitext using violation-fixing perceptron [12]. Another line of research is large-scale BLEU-oriented training for MT. E.g., He and Deng [13] developed an expected BLEU training method to learn phrase and lexicon translation features for phrase-based MT. But all these methods are task specific, rather than unifying different kinds of problems. To date, it is rare to see studies on addressing the two problems in a single framework of MT training.

In this paper we instead present a loss-augmented approach to learning features for syntactic MT. We introduce a loss function into the training criterion and model additional factors (e.g., search error and evaluation metric) by designing an appropriate loss. In this way we can guide MT training towards derivations that always stay within the beam and are of a good evaluation score. In particular, our model has several benefits:

1) Unlike previous work, we address the mismatch of search strategies between training and decoding as well as the mismatch of optimization criteria between evaluation metric and likelihood simultaneously, rather than optimizing for individual problems.
2) Our approach does not require forced decoding-like methods for oracle generation which suffers greatly from a high failure rate and insufficient use of training data [10], [11]. Instead, all bilingual data can be used to learn MT features.
3) Our approach is compatible with the general framework of MT training and is very easy to implement. It does not rely on sophisticated optimization like stochastic gradient descent which is not involved in popular MT toolkits. All we need is a slight modification to the existing modules if one already has a syntactic MT system.

We experiment with our approach in a state-of-the-art linguistically-motivated system. On large-scale Chinese-to-English and English-to-Chinese translation tasks, we obtain an average improvement of +0.8 BLEU. More interestingly, promising improvements are observed when our approach works with TER. In addition, we study various factors affecting our approach, including model convergency, training speed and sensitivity analysis of system parameters.

The rest of the paper is structured as follows. Section II briefly introduces the MT system used in this work. Section III describes our loss-augmented approach. Then, Section IV presents experimental evaluation of our approach. After reviewing the related work in Section V, the paper is concluded with a summary.

II. The Used MT System

The system used in this work is based on synchronous context-free grammars (SCFGs) which are popular in SMT. In the following we describe the translation model, rule extraction, training and decoding methods used in the system.

A. Translation Rules and Derivations

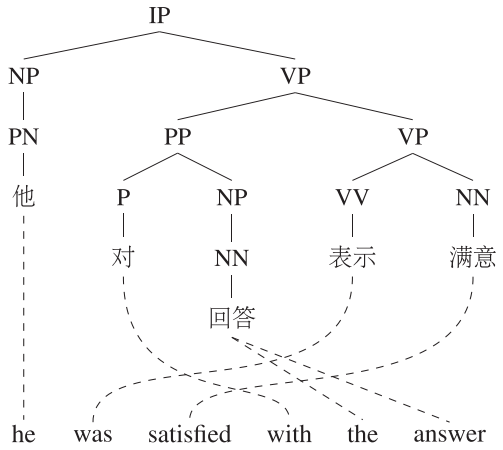
An SCFG is a system $\langle N, W_s, W_t, S, \Psi \rangle$, where N is a set of non-terminals, W_s and W_t are sets of terminals (or words) in the source and target languages, $S \subseteq N$ is a set of start symbols, and Ψ is a finite set of productions. Each production is an SCFG rewrite rule of the form

$$\text{LHS} \rightarrow \langle \alpha, \beta \rangle$$

where LHS is a non-terminal; α and β are sequences of terminals and non-terminals in the source and target languages. The only constraint here is that non-terminals in α and β are aligned in a 1-to-1 fashion.

When applying SCFGs, we can rewrite an aligned pair of non-terminals with the source and target-language sides of an SCFG rule. This operation requires that the labels of the substituted non-terminals must match the LHS label of the rewriting rule. Given a string pair, the generation process can be described as a sequence of SCFG rule applications, say, a *derivation* of SCFG rules.

In MT, different versions of SCFGs can be applied. A simple but effective way is hierarchical phrase-based translation [1]. It induces SCFGs with no syntactic annotations. Instead, SCFGs are extracted from word-aligned parallel data using Hiero-style heuristics [14]. E.g., we can first extract initial phrase pairs from the bitext and then obtain hierarchical phrase rules (i.e.,



Hiero-style SCFG Rules

h_1	X → ⟨他, he⟩
h_2	X → ⟨对, with⟩
h_3	X → ⟨回答, the answer⟩
h_4	X → ⟨表示 满意, was satisfied⟩
h_5	X → ⟨X_1 对 X_2 表示 满意, X_1 was satisfied with X_2⟩
...	

Rules Transformed from GHKM Extraction

r_1	NP → ⟨ 他, he ⟩
r_2	NP → ⟨ 回答, the answer ⟩
r_3	VP → ⟨ 表示 满意, was satisfied ⟩
r_4	IP → ⟨ NP_1 VP_2, NP_1 VP_2 ⟩
r_5	VP → ⟨ 对 NP_1 VP_2, VP_2 with NP_1 ⟩
...	

Fig. 2. Hiero-syle and tree-to-string rules extracted from a pair of word-aligned Chinese-English sentences with a source-language (Chinese) parse tree.

rules with non-terminals on the right hand side). See Fig. 2 for example Hiero-style rules extracted from a sentence pair with word alignments, where the non-terminals are labeled with X only.

Hiero-style grammars are formally syntactic, but rules are not constrained by source (or target) language syntax. An alternative is to perform GHKM rule extraction to introduce linguistically-motivated non-terminals from source (or target) phrase structure trees [15]. In the GHKM approach, translation equivalency relations are modeled from source-language syntactic trees to target language strings using derivations of GHKM rules [4]. A GHKM rule is a tuple of a source tree-fragment s_r, a target string t_r, and the alignments between the frontier non-terminals of s_r and t_r, for example,

$$\text{VP(VV(提高) } x_1\text{:NN)} \rightarrow \text{increases } x_1$$

is a GHKM rule. We can transform it into the SCFG form by keeping the frontier non-terminal annotations and discarding the internal tree structure (or flattening the tree) [16], like this

$$\text{VP} \rightarrow \langle\ \text{提高 NN}_1\text{, increases NN}_1\ \rangle$$

We refer to SCFG rules transformed from the GHKM rules as tree-to-string rules. As the non-terminals are annotated with source syntactic labels, applying tree-to-string rules is constrained to the "well-formed" constituents. See Fig. 2 for tree-to-string rules extracted from a tree-string pair.

For diverse rule types, we jointly use Hiero-style and tree-to-string rules in decoding [17], [18]. We follow the idea presented in [16]. Under such a model, a derivation can be made up of a mixture of Hiero-style and tree-to-string rules. It is doing something similar to "fuzzy" MT [6]: rule substitution does not require exact match of non-terminal labels, instead syntactic categories are used to model syntactic compatibility. In this way, more syntax-sensitive derivations are available to translation.

B. Mathematical Model and Features

Following the general framework of weighted synchronous grammar [19], we assign a score (or weight) to each translation rule r via a log-linear model [20]:

$$w(r) = \prod_i \theta_i(r)^{\lambda_i} \tag{1}$$

where $\{\theta_i(\cdot)\}$ are the features defined on rules, and $\{\lambda_i\}$ are the corresponding weights. We have the following feature functions for rule $r = \text{LHS} \rightarrow \langle \alpha, \beta \rangle$:

1) Phrase-based-like translation probabilities $\text{P}(\alpha|\beta)$ and $\text{P}(\beta|\alpha)$.
2) Rule generation probabilities $\text{P}(r|\text{LHS})$, $\text{P}(r|\alpha)$ and $\text{P}(r|\beta)$.
3) Lexical weights $\text{P}_{lex}(\alpha|\beta)$ and $\text{P}_{lex}(\beta|\alpha)$ estimated by Koehn *et al.*'s method [21].
4) A constant (exp(1)) to learn a preference for less or more rules in a derivation.
5) Indicators (exp(1)) for glue rules, Hiero-style and tree-to-string rules individually.
6) Number of substitutions (exp(#)) where the label of the substitution site matches LHS of the rule, and number for the unmatched case. They are analogies to the fuzzy matching features of syntactic rules [6].

Let t_d be the target string encoded in a derivation d. Then the score of d is defined as the product of rule weights, with a multiplication of the n-gram language model $lm(t_d)$ and the length of target string $\exp(|t_d|)$.

$$w(d) = \left(\prod_{r \in d} w(r) \right) \times lm(t_d)^{\lambda_{lm}} \times \exp(\lambda_{wb} \cdot |t_d|) \tag{2}$$

where λ_{lm} and λ_{wb} are the feature weights of the language model and target sentence length.

Then, the probability of d is defined as

$$\text{P}(d) = \frac{w(d)}{\sum_{d'} w(d')} \tag{3}$$

C. Training and Decoding

Given a set of translation rules, training can be regarded as a process of estimating the values of the associated parameters $\{\theta_i\}$ (or θ for short) from the bilingual data. In this work we re-

strict MT training to phrase-based-like translation probabilities (i.e., $\mathrm{P}(\alpha|\beta)$ and $\mathrm{P}(\beta|\alpha)$) and rule generation probabilities (i.e., $\mathrm{P}(r|\mathrm{LHS})$, $\mathrm{P}(r|\alpha)$ and $\mathrm{P}(r|\beta)$), and do not discuss the learning process of the remaining "off-the-shelf" features.[3]

A popular method for training these parameters is to maximize the likelihood value of the bilingual data, which is implemented by either summing the likelihood over all derivations or using the best derivation (Viterbi) as an approximation. For example, given a pair of sentences (s, t), we can formulate training as to obtain the optimal parameter set by maximizing the probability of Viterbi derivations in log-scale:

$$\hat{\theta} = \arg\max_{\theta} \log(\mathrm{P}_\theta(\hat{d})) \tag{4}$$

$$\text{where} \quad \hat{d} = \arg\max_{d \in D(s,t)} \log(\mathrm{P}_\theta(d)). \tag{5}$$

Here $\hat{d}$ is the Viterbi derivation for (s, t), and $D(s, t)$ is the set of all derivations from the source string s to the target-string t. The subscript θ in $\mathrm{P}_\theta(\cdot)$ is to emphasize that the probability is determined by the given parameter set. $\hat{\theta}$ is the maximum-likelihood estimates of the features, and can be obtained using either the Viterbi training algorithm [22] or the relative frequency estimation. In this work we take both methods as baselines for comparison.

Given a set of translation rules and features (as well as feature weights), decoding can be cast as finding the translation encoded in the most likely derivation of rules

$$\hat{d} = \arg\max_{d \in D(s,\cdot)} \log(\mathrm{P}_\theta(d)) \tag{6}$$

where $D(s, \cdot)$ represents the set of derivations from the source-string to any target-string. The decoding process is in general implemented with the algorithms used in syntactic parsing, such as the CKY parsing algorithm [23]. However, exact decoding is impractical for most SMT systems, especially when we decode with a high-order language model. Beam search has to be employed to prune the search space and explore the most promising candidates. Take CKY-style decoding as an example. On each source span (corresponding to a search stage), only the top-k partial derivations according to model score are kept and the rest are discarded.

III. Loss-Augmented Training

Here we present a way to incorporate prior knowledge into the training process of syntactic MT systems. In this method, all we need is to add a loss function to the training objective. Although the loss function can be designed in many ways, we resort to two simple forms—1) a search-related loss that penalizes derivations outside beam; 2) an evaluation-metric-like loss that penalizes derivations with a lower evaluation score. Such a method has several advantages:

1) Because the loss function is only incorporated in training, there is no modification of the tuning/decoding step. The decoding of test sentences can proceed as usual.
2) Our loss function is designed to address the mismatch problems. By training with this loss function, the learned model can be easily used to alleviate the problems.
3) Loss-augmented training unifies different problems. It is convenient to exploit various factors in a single framework of MT training.

A. The Model

We slightly modify the objective function of training and make the training process sensitive to the mismatch problems described in Section I. More specifically, we use a loss function to model the additional factors that are taken into account in training, e.g., this function can penalize derivations outside beams, and/or penalize derivations with a lower BLEU score.

We incorporate the loss function into the training process. For the method shown in Eqs. (4) and (5), we re-define the Viterbi derivation as

$$\hat{d} = \arg\max_{d \in D(s,\cdot)} \log(\mathrm{P}_\theta(d)) - \mathrm{loss}\,(t, d, D(s, \cdot)) \tag{7}$$

where t is the reference target-string, $\log(\mathrm{P}_\theta(d))$ is the score of the MT model, and $\mathrm{loss}(t, d, D(s, \cdot))$ is the loss function that imposes a penalty score. Then, we reuse Eq. (4) to obtain the optimized parameters.

The idea of this approach is pretty simple. We guide the system for a preference of derivations with a "best" loss-augmented score (as in Eq. (7)), rather than the "best" derivations in the original MT model. The preferred derivations are then employed in parameter optimization via a standard training method (as in Eq. (4)).

Note that Eq. (7) differs from Eq. (5) also in derivation space. Eq. (7) learns a translation model with derivations that are more likely to be explored during real decoding (i.e., $D(s, \cdot)$), instead of the "best" derivations indicated by two-way training (i.e., $D(s, t)$) as in Eq. (5).[4] In this way, Eq. (7) can explore derivations that do not exactly match t. The matching degrees can be controlled by the loss function. For example, when the loss function is designed as minus BLEU score scaled by a large positive number, Eq. (7) does exactly the same thing as Eq. (5). In addition, in this work we enforce a constraint that all derivations used training differ from t in 25% of the words at most, that is, in Eq. (7) $D(s, \cdot)$ excludes derivations whose target string is far from t. Thus we can save computation on unlikely derivations and speed up the system.

B. Loss Functions

Our loss function possesses two properties. First, it penalizes derivations which are likely to be outside beam at some search stages. Second, it penalizes derivations which possess a

[3]Lexical weights can be learned with word alignments and word-based translation probabilities, and n-gram language models can be learned over the target-side data using back-off methods. For indicator and count-based features, we can compute them on the fly during decoding.

[4]In two-way training, analysis is done in parallel on bilingual sentences, where the source-strings and target-strings are given in advance. On the other hand, the decoder runs in "one-way" mode (i.e., only the source sentence is provided) and seeks the best derivation in the absent of target-string.

lower evaluation score (e.g., a lower BLEU score). In this work these factors are modeled with two loss functions $\text{loss}_{\text{beam}}(\cdot)$ and $\text{loss}_{\text{goodness}}(\cdot)$. They are then interpolated in a linear fashion to form the loss function used in Eq. (7)

$$\begin{aligned}\text{loss}\ (t, d, D(s,\cdot)) = \alpha_1 \cdot \text{loss}_{\text{beam}}(t, d, D(s,\cdot)) \\ + \alpha_2 \cdot \text{loss}_{\text{goodness}}(t, d, D(s,\cdot))\end{aligned} \quad (8)$$

where α_1 and α_2 are the coefficients.

For the design of $\text{loss}_{\text{beam}}(\cdot)$, we first introduce the concept of $rank$ for a derivation. Given a source span u and a derivation d, we define $\eta(d, u, D(s,\cdot))$ (or $\eta(d)$ for short) to be the rank of d in competing with other derivations within $D(s,\cdot)$ on u, i.e., the rank in the beam. Then, we define $\text{loss}_{\text{beam}}(\cdot)$ to be

$$\text{loss}_{\text{beam}}(t, d, D(s,\cdot)) = \sum_{u \in \text{span}(d)} l_{\text{beam}}(t, d, u, D(s,\cdot)) \quad (9)$$

where $\text{span}(d)$ is the set of source spans associated with each rule application in d.span(d) models all the span units and corresponding search stages for a derivation. For example, in Fig. 2, we have a derivation d that is formed by applying rules $\{r_4, r_1, r_5, r_2, r_3\}$. Hence, span $(d) = \{(1,5),(1,1),(2,5),(3,3),(4,5)\}$[5] where each span corresponds to a rule application (see below).

(1,5) ⇒ applying r4 on 他$_1$ 对$_2$ 回答$_3$ 表示$_4$ 满意$_5$
(1,1) ⇒ applying r1 on 他$_1$
(2,5) ⇒ applying r5 on 对$_2$ 回答$_3$ 表示$_4$ 满意$_5$
(3,3) ⇒ applying r2 on 回答$_3$
(4,5) ⇒ applying r3 on 表示$_4$ 满意$_5$

$l_{\text{beam}}(\cdot)$ is designed to model the behavior of d in beam pruning. In beam pruning, d (or its partial derivation) has to compete with other derivations to survive on each span $u \in \text{span}(d)$. If the rank of d is beyond the beam width, it is pruned out. Therefore we define $l_{\text{beam}}(\cdot)$ to be

$$l_{\text{beam}}(t, d, u, D(s,\cdot)) = \begin{cases} C, & \eta(d) > k + k_m \\ \frac{\left(\eta(d) - (k - k_m)\right) \cdot C}{2k_m}, & k - k_m < \eta(d) \le k + k_m \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

Here k is the beam width, k_m is a parameter indicating the rank margin from the beam-width boundary that we start to penalize the given candidate, and C is the clipping level of the penalty score we tend to give. See Fig. 3 for an example curve of $l_{\text{beam}}(\cdot)$. It can be seen as a shifted and rescaled version of the hard tanh function.[6] The idea behind this model is that we would like to penalize derivations with a constant C when they are outside the beam over a margin k_m; on the other hand, when derivations are within the beam and their associated ranks are below $k - k_m$, we do not disturb them and assign them a zero-loss; otherwise, d is penalized with a linear function whose slope is $C/2k_m$. Thus, $l_{\text{beam}}(\cdot)$ is a continuous and bounded function, which possesses the desired characteristics preferred by the influence curve of robust statistics [24].

[5]In this work a span (beg, end) means a range from position beg to position end.

[6]Here we do not choose the standard version of the tanh function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ because it is sometimes numerical unstable and is not so easy to optimize. Instead the hard tanh function has almost the same curve as tanh but is much easier to compute.

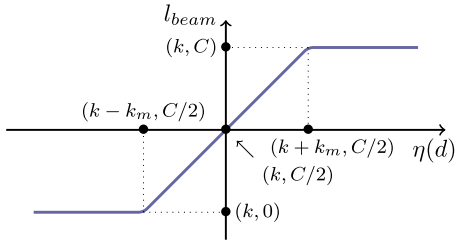


Fig. 3. Curve of $l_{\text{beam}}(\cdot)$.

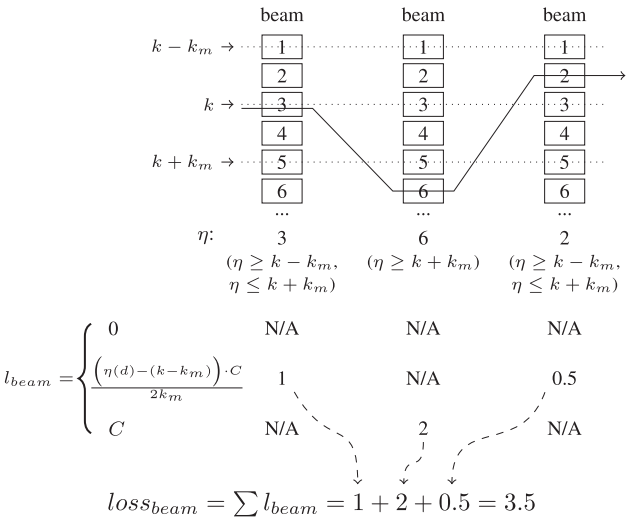


Fig. 4. $l_{\text{beam}}(\cdot)$ and $\text{loss}_{\text{beam}}(\cdot)$ for an example derivation or decoding path ($k = 3$, $k_m = 2$ and $C = 2$).

Fig. 4 shows the values of $l_{\text{beam}}(\cdot)$ and $\text{loss}_{\text{beam}}(\cdot)$ for an example derivation. Note that, to use $\text{loss}_{\text{beam}}$, we should adopt a larger beam width which is considerably greater than $k + k_m$ during training (say, 100, for $k = 30$ and $k_m = 10$). In this way, more derivations can be involved when we train the model with this loss function.

Next, we switch to the design of $\text{loss}_{\text{goodness}}(\cdot)$. As there are plenty of evaluation methods, we have many choices to define $\text{loss}_{\text{goodness}}(\cdot)$. In this work we use BLEU for technical description and experiments. We choose BLEU because it is the most widely used metric in MT evaluation. Note that although we restrict ourselves to BLEU in this paper, our method is applicable to other metrics. In the experiment section, we show that the loss-augmented training method also works well with TER and obtains promising improvements by several MT evaluation metrics.

We choose a similar formulation as that used in [25]. We define $\text{loss}_{\text{goodness}}(\cdot)$ as an approximation of the (minus) BLEU in log-scale:

$$\begin{aligned}&\text{loss}_{\text{goodness}}(t, d, D(s,\cdot)) \\ &= -\log|t_d| - \frac{1}{4} \cdot \sum_{n=1}^{4} \log \frac{\sum_{g_n \in \varphi(d)} c(g_n, t)}{|t_d| - n + 1}\end{aligned} \quad (11)$$

1: **Function** LEARNTHETA(θ, w, $loss(\cdot)$)
2: Initialize the model for $\hat{\theta}^{(0)}$
3: **For** i = 0 to (I - 1) (or until it converges) **do**
4: $\hat{d}^{(i)} = \arg\max_{d \in D(s,t)} \log(\mathrm{P}_{\theta^{(i)}}(d)) - loss_{\theta^{(i)}}(t, d, D(s, \cdot))$
5: $\hat{\theta}^{(i+1)} = \arg\max_{\theta} \log(\mathrm{P}_{\theta}(\hat{d}^{(i)}))$
6: **return** $\hat{\theta}^{(I)}$

Fig. 5. Algorithm of learning feature values under our model.

1: **Function** LEARNMODEL(θ, w, $loss(\cdot)$)
2: Initialize the model for $\hat{\theta}^{(0)}$ and $\hat{w}^{(0)}$
3: **For** i = 0 to (T - 1) **do**
4: $\hat{\theta}^{(i+1)} = \text{LEARNTHETA}(\hat{\theta}^{(i)}, \hat{w}^{(i)}, loss(\cdot))$
5: $\hat{w}^{(i+1)} = \text{LEARNWEIGHTS}(\hat{\theta}^{(i+1)}, \hat{w}^{(i)})$
6: **return** ($\hat{\theta}^{(T)}$, $\hat{w}^{(T)}$)

Fig. 6. Algorithm of learning feature values and weights.

where $|t_d|$ is the length of the target string yielded by d, $\varphi(d)$ is the set of n-grams encoded in d, g_n is an n-gram, and $c(g_n, t)$ counts the occurrence of g_n in the reference target-string t.

In Eq. (11), $\log|t_d|$ indicates a brevity penalty. We use the length of the translation to penalize short translations. It has been shown to have a similar effect as the standard version in original BLEU and can be applied to any partial translations [26].

$\frac{1}{4} \cdot \sum_{n=1}^{4} \log \frac{\sum_{g_n \in \varphi(d)} c(g_n, t)}{|t_d| - n + 1}$ is the log-scale geometric average of n-gram precisions. The clipping count of n-gram is ignored to speed up n-gram counting, as is in [25]. Obviously, such a loss function tends to give larger penalties to the derivations whose associated target strings deviate more from the given target-language training sentence t, that is, it prefers the derivations that match t better.

C. Training

To learn model parameters, we employ a Viterbi training-like method. See Fig. 5 for the pseudo-code of the algorithm.

The input of the algorithm is the feature functions (i.e., θ), fixed feature weights (i.e., w) and loss function $\text{loss}(\cdot)$. It first initializes the model with the parameters obtained from an existing training method. Here we use the method presented in [14] and [15] where the model is learned from the extracted rules using relative frequency estimation. We choose this method because it has been successfully used in several successful syntactic MT systems.

Afterwards, the parameters are updated in an iterative fashion via the following two steps:

1) We learn the Viterbi derivation given the parameters according to Eq. (7) (as in line 4 of Fig. 5). To do this, we just need to incorporate the loss function into our model for decoding (loss-augmented decoding), and output the 1-best derivation for each source sentence in the training set.
2) We learn the parameters given the derivation (as in line 5 of Fig. 5). Like most state-of-the-art SMT systems [1], [3], [21], [27], maximum likelihood estimation can be used to train the model parameters. In this work we choose a simple way for implementation and perform relative frequency estimation on the Viterbi derivations. Also, the add-alpha smoothing technique is employed to train the model for better generalization ability on test data.

Like the hard expectation-maximization (EM) algorithm, the above method focuses on maximizing the likelihood of the 1-best derivation. A natural extension is to use n-best derivations or derivation forest instead. In n-best or forest-based training, derivations (as well as translation rules) are associated with their posteriors or weights [28]. Once the derivations and rules are weighted, the parameter estimation procedure can proceed as usual, but with weight counts [29]. In this work we experiment with the 1-best, n-best and forest-based methods.

Another note on the algorithm. The main body of the learning process in Fig. 5 has similarities to the hard version of the EM algorithm [30], e.g., both of them update the Viterbi derivation ($\hat{d}^{(i)}$) and the optimal parameters ($\hat{\theta}^{(i+1)}$) in an iterative manner. On the other hand, there appear differences between them. In EM, the goal is to find maximum likelihood (or maximum a posteriori) estimates of model parameters, i.e., the objective is to maximize a likelihood function. However, the objective of our model is to find the derivation via a likelihood function plus additional losses. As arbitrary loss functions can be used, the resulting training algorithm is not guaranteed to converge to a local optimum. Fortunately, as is shown in the evaluation section (see Section IV), our method works well in real world tasks and shows a good ability of convergency in most cases.

Fig. 6 shows the complete version of our training paradigm. The LEARNTHETA procedure learns feature values. We can use either the algorithm in Fig. 5 or the baseline methods presented in Section II. The LEARNWEIGHTS procedure learns feature weights on the tuning set. In this work we use the Minimum Error Rate Training (MERT) method [31]. One complete pass of the two procedures is a training epoch, and the feature values and weights can be updated for T epoches.

Note that this method provides a very flexible way for learning syntactic MT models. For example, it is a standard way to train syntactic MT models if we use the relative-frequency estimation for LEARNTHETA and MERT for LEARNWEIGHTS [14]; On the other hand, it is the loss-augmented training if we use the algorithm in Fig. 5 for LEARNTHETA and iterate the training process for epochs.

D. Interpolation

The loss-augmented training and maximum likelihood/relative frequency-based training are two ways to estimate parameters of the MT model. The loss-augmented method focuses more on introducing additional factors into the training process, while the usual method (see Section II) is likelihood-oriented and fits better in a statistic story. It is therefore desirable to interpolate the two models when having the final parameter estimation. To do this, we can add an

additional round of training. Let θ_{baseline} be the model obtained via the baseline method described in Section II, θ_{new} be the model obtained via loss-augmented training. The final model θ_{final} is a linear interpolation of θ_{baseline} and θ_{new}

$$\theta_{\text{final}} = b \cdot \theta_{\text{baseline}} + (1 - b) \cdot \theta_{\text{new}} \tag{12}$$

where b controls the preference to the baseline model. Also, we re-tune the feature weights for θ_{final}.

IV. Evaluation

We evaluated our approach on Chinese-to-English and English-to-Chinese translation.

A. Experimental Setup

Our training corpus consists of 2.7 million sentence Chinese-English bitext from NIST12 OpenMT. We ran GIZA++ on the bitext to produce bidirectional alignments and then the grow-diag-final-and method to obtain symmetric alignments. For Chinese word segmentation, we ran NiuParser on the Chinese side[32]. For syntactic parsing, we ran the Berkeley Parser[33] on both Chinese and English sides. The parse trees were then binarized in a left-heavy fashion.[7] For the syntactic model, syntax-based (tree-to-string) rules with up to five non-terminals were extracted. For the hierarchical phrase-based model, phrasal rules and hierarchical phrase-based rules with up to two non-terminals were extracted. All these rules were learned using the NiuTrans open-source toolkit [34].

We trained two 5-gram language models: one on the Xinhua portion of the English Gigaword in addition to the English-side of the bitext, used by Chinese-to-English systems; one on the Xinhua portion of the Chinese Gigaword in addition to the Chinese-side of the bitext, used by English-to-Chinese systems. All language models were smoothed using the modified Kneser-Ney smoothing method [35].

For Chinese-to-English translation, our tuning set (919 sentences) was the NIST MT 03 evaluation data. The test sets (3486 sentences) contained all newswire evaluation data of NIST MT 04, 05 and 06. For English-to-Chinese translation, our tuning set (995 sentences) and test set (1859 sentences) were the evaluation data sets of SSMT 07 and NIST MT 08 Chinese-English track, respectively. All source-side parse trees were produced in the same way as that used on the training data.

We used a CKY-style decoder with cube pruning [14] and beam search to decode new sentences. Both tree-to-string rules and Hiero-style rules were used in the same decoding paradigm as described in [16]. All feature weights were optimized using minimum error rate training [31].

We used a beam width of 30 for decoding. To train our model, we adopted a beam width of 100. k, k_m and C (Eq. (10)) were set to 30, 10 and 10, respectively. The weighting coefficients α_1 and α_2 (Eq. (8)) were set to 0.1 and 5, which were tuned on the development set. The iteration number of our approach (i.e., I in Fig. 5) was set to 5 which was sufficient for a stable result. By default, the joint training was performed for one epoch (i.e., $T = 1$ in Fig. 6). In additon, add-alpha smoothing (alpha $= 0.5$) was used for parameter estimation in this work.

B. Baselines

Several baseline methods were chosen for comparison in this study.

1) The first of these is relative frequency estimation on all extracted rules. Due to its simplicity and effectiveness, this method has been widely adopted in current state-of-the-art syntactic MT systems [6], [14], [15]. Therefore, we chose it as the primary baseline in our experiments.
2) The second baseline is the system trained on the best forced-decoding derivations [36]. This method first generates the derivations that reach the target-string exactly, and then learns the parameters from these derivations. In this way, it can lead to a model that focuses more on the desired training paths instead of considering all paths equally.
3) In addition, SEARN [37] was selected for our empirical comparison. This algorithm maintains two sets of parameters. The first set is the parameters learned from the "optimal" derivations which are obtained via forced-decoding. The second parameter set is learned from the derivations that the MT system actually sees in practice. Then a new model is obtained by interpolating these two sets of the parameters. Since beam search is employed to generate the second set of training paths, SEARN implicitly considers the search problem during training.
4) We also implemented the discriminative method that learned the parameters on the best BLEU derivations in the n-best list [38]. Basically this method is on the same theme as SEARN. It implicitly takes beam-width limitation into account by using the best candidate within the n-best list.
5) Moreover, we employed the method that maximizes the likelihood (see Section II-C), including learning from Viterbi derivations (i.e., Viterbi training), learning from n-best derivations (i.e., n-best training), and learning from derivation forests (i.e., forest-based training). Note that the forest-based training is doing something similar to the primary baseline where the probabilities are obtained via relative frequency estimation on all the rules involved in the derivation forest. The difference lies in that the forest-based training method uses fractional counts for maximum likelihood estimation, while the primary baseline counts each rule occurrence as unit one.

C. Results

1) Impact of Loss Functions: We start with a study on how the proposed loss functions affect the translation quality in terms of case insensitive BLEU.[8] To isolate the contributions of each loss function from the other, we investigate the individual impact of $\text{loss}_{\text{beam}}$ and $\text{loss}_{\text{goodness}}$, respectively, i.e., varying α_1 when

[7] We used left-heavy binarization because it showed good results for Chinese syntactic trees [16].

[8] In the evaluation, we report uncased BLEU4 on zh-en and uncased BLEU5 on en-zh, respectively.

TABLE I
BLEU[%] RESULTS BY VARYING α_1 WHEN $\alpha_2 = 0$

α_1	zh-en		en-zh	
	Tune	Test	Tune	Test
0	38.05	36.07	34.09	31.40
0.01	38.15	36.25	34.21	31.63
0.1	**38.44**	**36.64**	**34.52**	**31.88**
1	37.73	36.02	33.59	30.94

TABLE II
BLEU[%] RESULTS BY VARYING α_2 WHEN $\alpha_1 = 0$

α_2	zh-en		en-zh	
	Tune	Test	Tune	Test
0	38.05	36.07	34.09	31.40
0.1	38.48	36.56	34.17	31.67
1	38.57	36.68	**34.50**	**31.82**
5	**38.66**	**36.70**	34.35	31.77
10	38.12	36.32	34.07	31.55

TABLE III
BLEU[%] RESULTS (TUNE/TEST) ON ZH-EN UNDER DIFFERENT SETTINGS OF α_1 AND α_2

	$\alpha_1 = 0$	$\alpha_1 = 0.01$	$\alpha_1 = 0.1$	$\alpha_1 = 1$
$\alpha_2 = 0$	38.05/36.07	38.15/36.25	38.44/36.64	37.73/36.02
$\alpha_2 = 0.1$	38.48/36.56	38.48/36.56	38.46/36.60	38.01/36.10
$\alpha_2 = 1$	38.57/36.68	38.60/36.79	38.75/36.80	38.00 36.20
$\alpha_2 = 5$	38.66/36.70	38.80/36.80	**38.88/36.86**	38.12/36.23
$\alpha_2 = 10$	38.12/36.32	38.10/36.25	38.17/36.46	37.86/36.06

TABLE IV
BLEU[%] RESULTS (TUNE/TEST) ON EN-ZH UNDER DIFFERENT SETTINGS OF α_1 AND α_2

	$\alpha_1 = 0$	$\alpha_1 = 0.01$	$\alpha_1 = 0.1$	$\alpha_1 = 1$
$\alpha_2 = 0$	34.09/31.40	34.21/31.63	34.52/31.88	33.59/30.94
$\alpha_2 = 0.1$	34.17/31.67	34.45/31.86	34.76/32.02	33.80/31.41
$\alpha_2 = 1$	34.50/31.82	34.74/32.03	**35.01/32.43**	34.08/32.00
$\alpha_2 = 5$	34.35/31.77	34.70/31.98	34.95/32.26	34.05/31.75
$\alpha_2 = 10$	34.07/31.55	34.23/31.43	34.39/31.76	33.29/31.15

$\alpha_2 = 0$, and varying α_2 when $\alpha_1 = 0$. Tables I and II show that $\text{loss}_{\text{beam}}$ and $\text{loss}_{\text{goodness}}$ have similar influence. Either of them can lead to a BLEU improvement in [0.4, 0.7] on the zh-en and en-zh tasks.

We then study the effectiveness of using $\text{loss}_{\text{beam}}$ and $\text{loss}_{\text{goodness}}$ simultaneously. To do this, we adjust the system along the dimensions of α_1 and α_2. Seen from Tables III and IV, we can have a larger BLEU improvement by seeking a better corelation between $\text{loss}_{\text{beam}}$ and $\text{loss}_{\text{goodness}}$. Our system outperforms the baseline over 0.8 BLEU points when $\alpha_1 = 0.1$ and $\alpha_2 \in [1, 5]$.

2) Comparison With Other Approaches: Table V shows the BLEU result of various systems. We see, first of all, that the proposed approach outperforms the baselines on both translation tasks. Learning rules and parameters from derivation forests performs best, followed by the n-best and 1-best (Viterbi-style) methods. This result indicates that our approach can learn a better model by accessing more derivations in training.

Also, we interpolate different models with the primary baseline. Table V shows that the model interpolation helps. The best system is obtained when we have an interpolation factor of $b = 0.5$. It achieves +0.87 (zh-en) and +0.88 (en-zh) BLEU improvements over the baselines with the same interpolation coefficient. It even achieves an average improvement of +1.09 BLEU over the baselines with no model interpolation.

In Table V, we also report the results of a standard hierarchical phrase-based system [14] and a standard tree-to-string system [4]. For tree-to-string translation, we use the binarized parse trees in training and decoding as described in Section IV-A. Both our primary baseline and improved systems outperform the hierarchical phrase-based and tree-to-string counterparts. As described in Section III, loss-augmented training is a general approach. So we also apply it to the hierarchical phrase-based and tree-to-string systems. It is observed that it works well with these systems (see Table V). On both translation tasks, the improvement is over 0.8 BLEU points, which indicates that our approach is applicable to different types of systems.

3) Impact of Beam Width: In addition to BLEU, we also study how fast our proposed approach and the baseline can run on the training set. Fig. 7 shows the speed comparison for different settings of beam width.[9] Note that the curves of Viterbi-based, n-best-based and forest-based loss-augmented training are not distinguishable. Because loss-augmented training requires running the decoder on the entire training set, most of the training time is consumed by the generation of desired derivations rather than the parameter estimation program. As a result, the training time does not vary too much for these methods. So we remove the curves of n-best and forest-based training for clear figures.

We see that loss-augmented training is slower than the primary baseline based on relative frequency estimation, e.g., in the default setting (beam width = 100), it is 0.90 sentence/second vs. 28.0 sentence/second on the zh-en task. It is even slower when we enlarge the beam width. To speed up the system, a good way might be to parallelize the training process. E.g., in our implementation, we distribute both the derivation generation and parameter learning (lines 4-5 in Fig. 5) to a cluster of computers in a standard map-reduce fashion. In this way, the loss-augmented training can be finished in no more than 12 hours, and is not a bottleneck in system development and tuning.[10]

Then we plot BLEU as a function of beam width. Fig. 8 shows that both the baseline and loss-augmented training can benefit from larger beam width. But a too large beam does not help. E.g., when we use a beam width of 200, there is almost no BLEU improvement on the test data, but the systems are very slow.

[9]The speed is measured in the single thread mode on an Intel Xeon 3.5GHz CPU.

[10]The decoding speed on the test data is not for comparison because all these systems use the same decoder with a same-sized rule table.

TABLE V
BLEU[%] SCORES OF VARIOUS SYSTEMS[a]

Entry		zh-en		en-zh		Average
		Tune (919)	Test (3486)	Tune (995)	Test (1859)	Improvement on Test
No Interpolation ($b = 0$)	Primary Baseline	37.98	36.01	34.03	31.23	0
	Forced Decoding-based Training	37.32	35.72	33.56	31.27	−0.13
	SEARN	37.06	35.21	33.22	30.50	−0.76
	Learning from Best-BLEU Derivation (in n-best List)	37.23	35.14	33.10	30.72	−0.69
	Viterbi Training	37.63	35.67	33.71	30.76	−0.40
	n-best Training	37.75	35.89	33.98	31.00	−0.18
	Forest-based Training	38.05	36.07	34.09	31.40	+0.12
	Loss-Augmented Training (Viterbi)	38.62	36.57	34.77*	31.86	+0.54
	Loss-Augmented Training (n-best)	38.68*	36.59*	34.89*	31.97*	+0.58
	Loss-Augmented Training (Forest)	38.88*	36.86*	34.95*	32.26*	+0.95
Interpolation with Primary Baseline ($b = 0.1$)	Primary Baseline	37.98	36.01	34.03	31.23	0
	Forced Decoding-based Training	37.38	35.85	33.65	31.40	0
	SEARN	37.52	35.67	33.60	31.07	−0.25
	Learning from Best-BLEU Derivation (in n-best List)	37.68	35.68	33.65	31.17	−0.20
	Viterbi Training	37.65	35.97	33.80	30.94	−0.15
	n-best Training	37.73	36.05	34.04	31.08	−0.06
	Forest-based Training	38.13	36.19	34.20	31.55	+0.25
	Loss-Augmented Training (Viterbi)	38.65	36.59	34.86*	31.86	+0.60
	Loss-Augmented Training (n-best)	38.75*	36.68*	34.94*	31.90	+0.67
	Loss-Augmented Training (Forest)	38.88*	36.92*	34.99*	32.34*	+1.01
Interpolation with Primary Baseline ($b = 0.5$)	Primary Baseline	37.98	36.01	34.03	31.23	0
	Forced Decoding-based Training	37.70	35.94	33.73	31.53	+0.11
	SEARN	38.05	36.10	34.15	31.45	+0.16
	Learning from Best-BLEU Derivation (in n-best List)	38.10	36.14	34.19	31.55	+0.23
	Viterbi Training	37.99	36.07	33.97	31.08	−0.05
	n-best Training	38.04	36.15	34.12	31.20	+0.06
	Forest-based Training	38.26	36.21	34.27	31.55	+0.26
	Loss-Augmented Training (Viterbi)	38.75*	36.64*	34.98*	31.90	+0.65
	Loss-Augmented Training (n-best)	38.72	36.68*	35.02*	31.96*	+0.70
	Loss-Augmented Training (Forest)	38.99*	36.98*	35.10*	32.43*	+1.09
Interpolation with Primary Baseline ($b = 0.9$)	Primary Baseline	37.98	36.01	34.03	31.23	0
	Forced Decoding-based Training	37.99	36.00	33.87	31.36	+0.06
	SEARN	38.06	35.90	33.99	31.30	-0.02
	Learning from Best-BLEU Derivation (in n-best List)	37.98	35.95	34.05	31.26	-0.02
	Viterbi Training	37.95	36.00	34.03	31.10	-0.07
	n-best Training	38.08	36.10	34.11	31.22	+0.04
	Forest-based Training	38.12	36.10	34.16	31.46	+0.16
	Loss-Augmented Training (Viterbi)	38.06	36.23	34.27	31.55	+0.27
	Loss-Augmented Training (n-best)	38.10	36.22	34.25	31.60	+0.29
	Loss-Augmented Training (Forest)	38.20	36.30	34.42	31.66*	+0.36
Hierarchial Phrase-based		37.35	35.29	33.10	30.47	-0.74
Hierarchial Phrase-based + Loss-Augmented Training		38.01	36.07	33.99	31.30	0.07
Tree-to-String		36.28	34.63	32.90	30.53	-1.04
Tree-to-String + Loss-Augmented Training		37.15	35.64	33.94	31.50	-0.05

[a] * = significantly better than all seven of the baselines at $p < 0.01$.

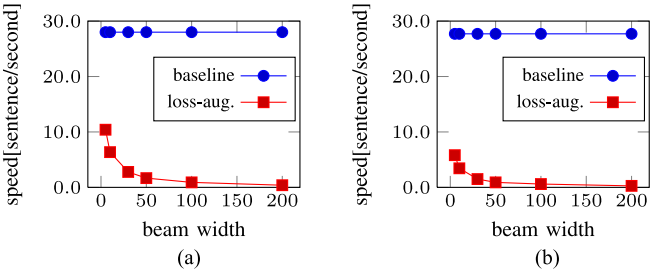


Fig. 7. Training speed (sentence/second) against beam width.

4) Convergence of Different Measures: The proposed approach requires an iterative process of learning model parameters (see Fig. 5). It is therefore worth a study on the convergency issue. Fig. 9 shows the BLEU score and 1-best model score of our approach against iteration number (i.e., I). As model interpolation is useful, we interpolate our model and primary baseline with $b = 0.5$. Here iteration 0 represents the system where loss-augmented training is not employed. The curves in Fig. 9 show that the system converges by both measures. In our experiments three iterations are enough to obtain a stable improvement.

5) Joint Training With Weight Tuning: As described in Fig. 6, we can learn feature values (i.e., θ) and feature weights (i.e., w) jointly. The process can be iterated for epoches. We investigate the system behavior in this joint learning framework. Fig. 10 shows that our system can benefit from joint training with weight tuning. More interestingly, forest-based training obtains a larger improvement than the 1-best and n-best counterparts. As

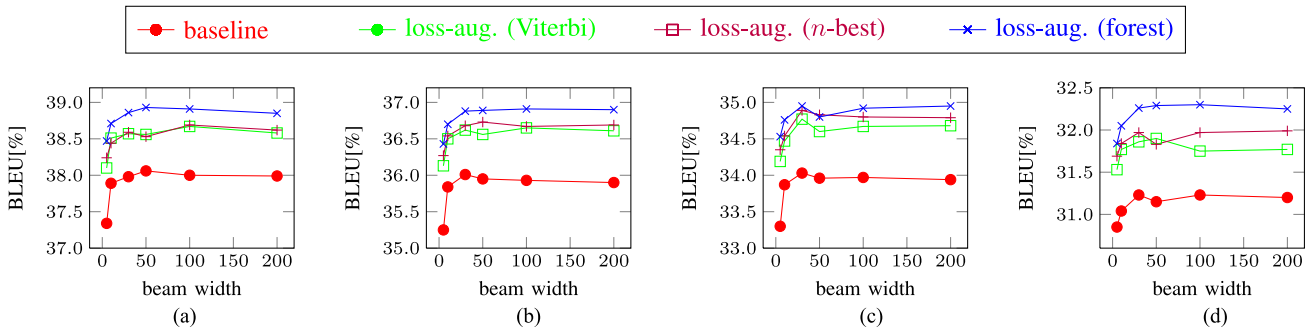


Fig. 8. BLEU[%] against beam width. (a) BLEU[%] on tuning (zh-en). (b) BLEU[%] on test (zh-en). (c) BLEU[%] on tuning (en-zh). (d) BLEU[%] on test (en-zh).

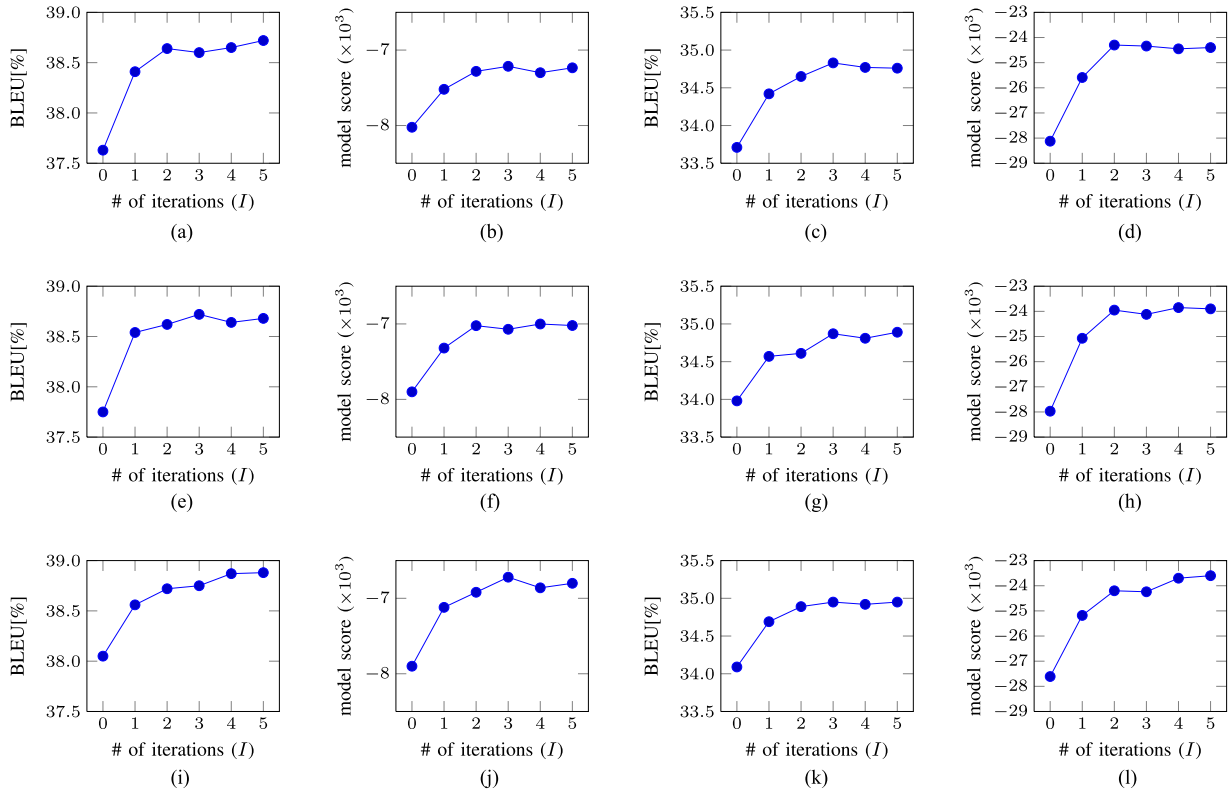


Fig. 9. BLEU[%] and (1-best) model score against iteration number (I). (a)–(d) show the case of Viterbi training. (e)–(h) show the case of 100-best training. (i)–(l) show the case of forest-based training.

more rules and parameters are accessed in forest-based training, it requires more training epoches for convergence.

6) Sensitivity Analysis: There are four parameters in our model, including the margin k_m, the clipping level C, the loss weights α_1 and α_2. Note that k_m and C are only related to $\text{loss}_{\text{beam}}$. So we first study the system behavior under different settings of k_m and C when $\alpha_1 = 0.1$ and $\alpha_2 = 0$. In this way it is easier to see the influence of k_m and C to $\text{loss}_{\text{beam}}$ only. To do this, we fix $k_m = 10$ and vary the value of C. Table VI shows that a too high/low clipping level is harmful. In our system, choosing C in [10,20] achieves the best result on both translation tasks.

We then study the impact of margin k_m given the optimal choice of C (=15). Table VII shows that introducing a margin into training can enhance the generalization ability of the model on the test set. However, a large value of k_m is not beneficial. This is because a large margin disturbs derivations within the beam and over-twists their associated ranks. This phenomenon may accompany with the overfitting problem, and thus has a side-effect on the test set.

Note that the best choice of k_m and C may change when $\alpha_2 \neq 0$. A better way to fine tune these parameters is to consider both α_1 and α_2 while adjusting k_m and C. However it is impractical to find the optimized values of k_m, C, α_1 and α_2 simultaneously. We instead use a simple method. For each choice of α_1, we find the best value of α_2 in Tables III and IV. Then, we adjust k_m and C in a similar way as in Tables VII and VI. Table VIII shows the best setting of k_m and C for different values of α_1

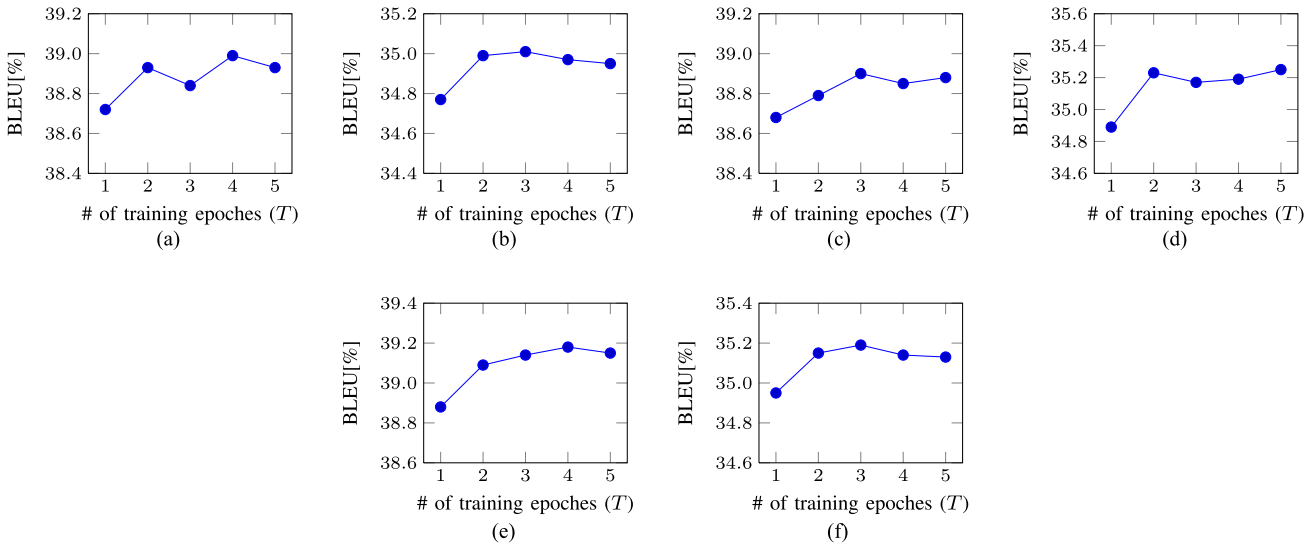


Fig. 10. BLEU[%] against training epoch number (T). (a) and (b) show the case of Viterbi training. (c) and (d) show the case of n-best training. (e) and (f) show the case of forest-based training.

TABLE VI
BLEU[%] RESULTS UNDER DIFFERENT SETTINGS OF C

C	zh-en		en-zh	
	Tune	Test	Tune	Test
0	37.53	35.78	34.12	31.47
1	38.12	36.31	34.56	31.89
5	38.33	36.54	34.76	32.18
10	38.75	36.80	35.01	**32.43**
15	**38.78**	36.83	**35.05**	32.40
20	38.73	**36.95**	34.88	32.27
50	38.15	36.56	34.63	31.91
100	38.06	36.30	34.39	31.64

TABLE VII
BLEU[%] RESULTS UNDER DIFFERENT SETTINGS OF k_m

k_m	zh-en		en-zh	
	Tune	Test	Tune	Test
0	38.10	36.40	34.69	31.80
5	38.28	36.61	34.75	32.10
10	**38.78**	36.83	**35.05**	**32.40**
15	38.75	36.76	35.01	32.29
20	38.68	**36.84**	34.95	32.30
30	38.50	36.73	34.84	32.26

TABLE VIII
BLEU[%] RESULTS FOR TUNED PARAMETERS α_1, α_2, k_m AND C

				zh-en		en-zh	
α_1	α_2	C	k_m	Tune	Test	Tune	Test
0	5	0	0	38.66	36.70	34.35	31.77
0.01	5	15	15	38.89	**36.97**	34.78	32.09
0.1	5	10	10	**38.98**	36.92	**34.99**	**32.30**
1	1	10	10	38.12	36.27	34.15	32.17

and α_2. We see a better BLEU score under the fine tuned parameters. But the improvement is modest. Also, the optimal value of k_m and C does not vary too much for different choices of α_1 and α_2.

Another note on α_1 and α_2. Because the model score of MT systems and the scores of loss functions are not in the same scale, their influences are not directly comparable. After normalizing these scores with their associated variances, it is found that 1) the MT model score is more important than the loss functions used in this work (the relative weight is 2.79); and 2) $\text{loss}_{\text{beam}}(\cdot)$ and $\text{loss}_{\text{goodness}}(\cdot)$ possess similar influence (relative weight is 1.24 which slightly favors $\text{loss}_{\text{beam}}(\cdot)$). This result agrees with the observation in Tables I and II that $\text{loss}_{\text{beam}}(\cdot)$ and $\text{loss}_{\text{goodness}}(\cdot)$ have similar contributions to the BLEU improvement.

7) Experiments on Other Metrics: Although we restrict ourselves to BLEU-based loss in this work, our approach is applicable to other MT evaluation metrics. There are many choices, e.g., TER and WER. To integrate these evaluation metrics into loss-augmented training, all we need is to replace $\text{loss}_{\text{goodness}}$ with the negative score of the corresponding metric.[11] Here we test loss-augmented training for TER which is another popular method of MT evaluation.

Another interesting question is how the loss-augmented training performs if we use more metrics for evaluation rather than BLEU and TER. Therefore we report the result of NIST [39], GTM [40], WER [41], and PER [42]. In this way, we have a better assessment of the translation quality.

We reuse the default setting in the previous experiment and tune feature weights by minimizing TER on the tuning set. Tables IX and X show that the loss-augmented training approach works well with TER. It obtains a +0.60 -TER improvement on the test sets. More interestingly, our approach

[11]We assume that a higher score means better translation. For error oriented methods (e.g., TER), we take its negative score as the measure of translation quality.

TABLE IX
EVALUATION RESULT OF BLEU-BASED AND TER-BASED SYSTEMS ON THE ZH-EN TASK[a]

Entry		Tune (zh-en)						Test (zh-en)					
		BLEU	TER	NIST	WER	PER	GTM	BLEU	TER	NIST	WER	PER	GTM
BLEU-based training&tuning	Primary Baseline	37.98	60.52	9.676	65.79	42.35	80.83	36.01	60.18	9.412	68.25	43.60	77.39
	Loss-aug. Training	38.88	59.87	9.780	65.24	41.69	81.00	36.86	59.55	9.507	67.85	43.02	77.57
	diff.	0.90	−0.65	0.104	−0.55	−0.67	0.17	0.85	−0.63	0.095	−0.40	−0.58	0.18
TER-based training&tuning	Primary Baseline	36.87	59.73	9.598	65.10	41.58	81.15	35.06	59.54	9.324	67.75	42.98	77.55
	Loss-aug. Training	37.32	59.01	9.665	64.50	41.00	81.16	35.43	58.82	9.368	67.01	42.33	77.78
	diff.	0.45	−0.72	0.067	−0.60	−0.58	0.01	0.37	−0.72	0.044	−0.74	−0.65	0.23

[a] BLEU, TER, WER, PER, and GTM scores are reported in percentage

TABLE X
EVALUATION RESULT OF BLEU-BASED AND TER-BASED SYSTEMS ON THE EN-ZH TASK[a]

Entry		Tune (en-zh)						Test (en-zh)					
		BLEU	TER	NIST	WER	PER	GTM	BLEU	TER	NIST	WER	PER	GTM
BLEU-based training&tuning	Primary Baseline	34.03	52.32	9.482	65.44	39.67	77.02	31.23	52.94	9.302	65.45	39.19	75.67
	Loss-aug. Training	34.95	51.62	9.593	64.70	38.92	77.31	32.26	52.19	9.411	64.79	38.48	75.94
	diff.	0.92	−0.70	0.111	−0.74	−0.75	0.29	1.03	−0.75	0.109	−0.66	−0.71	0.27
TER-based training&tuning	Primary Baseline	32.97	51.47	9.402	64.63	38.92	77.22	30.17	52.12	9.243	63.93	38.31	75.65
	Loss-aug. Training	33.32	50.56	9.465	63.90	38.20	77.60	30.59	51.33	9.281	63.21	37.73	75.82
	diff.	0.35	−0.91	0.065	−0.73	−0.72	0.38	0.42	−0.79	0.038	−0.72	−0.58	0.17

[a] BLEU, TER, WER, PER, and GTM scores are reported in percentage

outperforms the baseline over 0.3 BLEU points even when we train and tune the system using TER. Also, loss-augmented training shows promising results for other metrics. E.g., there are also improvements of NIST, WER and PER on both tasks. This result indicates that the improvement of translation quality here is general and can be demonstrated by several metrics. In addition, it is observed that the BLEU score drops when we switch from BLEU-based tuning to TER-based tuning. It reflects some inconsistency between different error measures. Another explanation is that TER prefers short translations which result in a large brevity penalty in BLEU.

8) Real Examples and Analysis: We also examine whether the improvements persist when the translation quality is judged by humans. To do this, we randomly select 200 sentences from the data set used in the zh-en task. Two judges participate in this experiment. Both of them are qualified translators who are skilled in reading and writing English. In the evaluation, they are required to carefully read the translation, and do 5-point scoring on each MT output according to its translation quality. We observe that the improvement in human evaluation is consistent with the improvement of automatic evaluation score. The loss-augmented system outperforms the baseline by more than 0.15 points (3.60 vs. 3.43), which is statistically significant at $p < 0.01$.

We then analyze the data to see where the improvement is from. See Fig. 11 for real examples generated by different systems. In Example 1, a good rule NP 出口商品 is pruned by the baseline system due to beam search. This results in a bad translation "export mix" (the reference is "export commodity structure"). The situation is much better when loss-augmented training is used, where the desired rule survives in beam search and leads to the correct translation. Similar phenomena are observed in Examples 2 and 3. In Example 2, a desired rule NP $\rightarrow \langle$ NR_1 众多 的 NP_2, many NR_1 NP_2 $\rangle$ is selected by the loss-augmented system, while it is pruned out in an intermediate search stage in the baseline system. Similarly, in Example 3, NP $\rightarrow \langle$ NP_1 NP_2 综合 利用, integrated utilization of NP_1 $NP_2\rangle$ is kept in beam search and generates the correct translation "integrated utilization of national mineral resources" for "全国 矿产 资源 综合 利 用".

V. RELATED WORK

Over the past decades, researchers have developed powerful methods of learning feature values and weights for modern SMT systems. Among these are maximum likelihood training [21], [27], discriminative training [43], Bayesian training [44], [45], MERT [31], minimum risk training [46], [47], maximum margin training [48], [49], and pairwise ranking optimization (PRO) [50], [51]. Although widely used, these approaches do not address the mismatch problems between training and decoding. Most systems are still resorting to a paradigm of relative frequency estimation plus MERT/PRO.

To our knowledge, only a few studies consider the limited beam-width problem in learning NLP models. Perhaps the earliest is Daumé's work [37], where the beam-width is implicitly considered in training. He proposed a general approach (SEARN) that learned structured prediction systems from both ideal decoding paths (associated with the benchmark) and actual decoding paths (obtained using beam search), instead of from ideal paths only. Similar ideas can be found in incremental parsing [12], [52], where the assumption of exact inference is removed

Example 1

Source	外商 投资 企业 在 改善 中国 出口 商品 结构 中 发挥 显著 作用 . foreign funded enterprise in improve Chinese export commodity structure play prominent role .
Baseline	Foreign-funded enterprises played a remarkable role in improving China 's export mix .
Loss-Aug.	Foreign-funded enterprises played a prominent role in improving China 's export commodity structure .
Reference	Foreign-invested enterprises have played a prominent role in improving China 's export commodity structure .

Example 2

Source	... , 天津市 众多 的 " 三资 企业 " 正在 积极 寻求 进入 俄方 市场 . ... , Tianjin many " triple-funded enterprise " are actively seek enter Russian market .
Baseline	... , the Tianjin enterprises are actively seeking to enter the Russian market .
Loss-Aug.	... , many Tianjin "triple-funded enterprises " are actively seeking to enter the Russian market .
Reference	... , many Tianjin "triple-funded enterprises " are actively seeking to enter the Russian market .

Example 3

Source	... , 金川 公司 被 中国 政府 列为 全国 矿产 资源 综合 利用 三 大 基地 之一 ... , ... , Jinchuan company was Chinese government listed national mineral resource integrated utilization three top base one of ... ,
Baseline	... , the Jinchuan company listed as one of the top three bases of the use of mineral resources across the country , ...
Loss-Aug.	... , the Jinchuan company was listed as one of the top three bases of the integrated utilization of mineral resources across the country ,...
Reference	... , the Jinchuan company was listed by the Chinese government as one of the top three bases of integrated utilization of national mineral resources , ...

Fig. 11. Comparison of translations (with word segmentation and tokenization).

in training and the decoder with approximate inference is used in the training stage. But these approaches were not actually experimented in SMT systems.

In MT, Liang *et al.* [38] proposed a discriminative method that was similar to SEARN and applied it to their MT system. In their work the beam-width limitation problem was implicitly addressed by selecting the best candidate from the n-best list. But they did not explicitly modeled the beam-width limitation in training. More recently, Yu *et al.* [11] and Zhao *et al.* [53] adapted the max-violation perceptron algorithm [12] to phrasal and hierarchical phrase-based models. In this way, MT systems can learn better weights by avoiding invalid updates in perceptron (i.e., towards derivations with good scores but outside beams). These systems require forced-decoding-like methods to obtain reference derivation for weight updates. It may result in decoding failure and insufficient use of training data.

Another line of research is to develop BLEU-oriented objectives for learning feature values of MT models. For example, He and Li [13] proposed an expected BLEU-based method. They designed an objective with an expected BLEU score and KL regularization, and trained MT models via the extended Baum-Welch algorithm. By learning phrase and lexicon translation probabilities with this approach, their phrase-based system achieved very promising result on German-to-English and Chinese-to-English translation tasks. Also, Rosti *et al.* [54] used expected BLEU training to tune weights of component systems in system combination.

To date, few studies have addressed the two mismatch problems in a single MT training paradigm. This motivates us to develop a loss-augmented approach to addressing both problems simultaneously. Apart from this, there appear other differences from previous work. For example, our approach does not require forced decoding, and thus can make use of full data. Also, it does not rely on complicated optimization algorithms and is very easy to implement. More importantly, we are the first to report results on addressing the mismatch problems in linguistically-motivated systems. In our experiments, we demonstrate the effectiveness of our approach in a good syntactic MT system by a quantitative comparison with several baselines.

Another note on loss-augmented training. Loss-augmented approaches are not new in NLP. They have been used in several NLP tasks [55]–[58]. But it is rare to see work on applying them to learn probability-like features of MT models. In this work we employ loss-augmented approaches to introduce additional factors into MT training. Although we do not focus on the theoretical side of loss-augmented training in MT, it is interesting to see that such a method works well in improving a syntactic MT system.

Actually the forced decoding method discussed in this paper is also related to forced alignment [59]–[62]. Forced alignment is usually used to align transcription (in text) to audio signals of spoken word. Like forced decoding, forced alignment is required to align the input sequence to a given output sequence. On the other hand, there is an obvious difference between them. Forced decoding in MT needs to take the reordering problem into account, while forced alignment in speech recognition does not need to do so due to its nature of monotonic alignment.

VI. CONCLUSION AND FUTURE WORK

We have proposed an approach to addressing the two problems simultaneously: 1) the mismatch of adopted beam-widths between training and decoding; and 2) the mismatch of optimization criterion between likelihood value and evaluation score. We use a loss-augmented approach that models beam-width limitation and goodness of translation as two factors in training. In this way, we can introduce additional factors into training (e.g., search error and evaluation metric) by designing an appropriate loss. On large-scale Chinese-to-English and English-to-Chinese MT tasks, it obtains significant improvements over the state-of-the-art baselines.

Although the framework presented here is simple, it requires more parameters to tune. An interesting issue we would investigate is how to fine tune these parameters in an easier manner. E.g., we plan to use model selection techniques to learn better models. Another issue that needs further investigation is that how our approach behaves when it works with expected BLEU training. A straightforward way to do this is to design

an expected BLEU-based loss and introduce it into our training objective. Actually the $\text{loss}_{\text{goodness}}$ function used in this work is doing something similar to the objective function used in expected BLEU training. It is thus worth an empirical comparison of these approaches as well as an exploration on the combination of them.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their pertinent and insightful comments, K.-Y. Su for his great help in improving the early version of this paper, and C. Zhang for language refinement.

REFERENCES

[1] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, Ann Arbor, MI, USA, Jun. 2005, pp. 263–270.

[2] J. Eisner, "Learning non-isomorphic tree mappings for machine translation," in *Proc. 41st Annu. Meeting Assoc. Comput. Linguistics*, Sapporo, Japan, Jul. 2003, pp. 205–208.

[3] M. Galley, M. Hopkins, K. Knight, and D. Marcu, "What's in a translation rule?" in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, Boston, MA, USA, 2004, pp. 273–280.

[4] Y. Liu, Q. Liu, and S. Lin, "Tree-to-string alignment template for statistical machine translation," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, Sydney, Australia, Jul. 2006, pp. 609–616.

[5] L. Huang, K. Kevin, and A. Joshi, "Statistical syntax-directed translation with extended domain of locality," in *Proc. Conf. Assoc. Mach. Transl. Amer.*, Cambridge, MA, USA, Aug. 2006, pp. 66–73.

[6] D. Chiang, "Learning to translate with source and target syntax," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, Uppsala, Sweden, Jul. 2010, pp. 1443–1452.

[7] D. Marcu, W. Wang, A. Echihabi, and K. Knight, "SPMT: Statistical machine translation with syntactified target language phrases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sydney, Australia, Jul. 2006, pp. 44–52.

[8] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, Philadelphia, PA, USA, Jul. 2002, pp. 311–318.

[9] M. Snover, B. Dorr, R. Schwartz, J. Makhoul, L. Micciula, and R. Weischedel, "A study of translation error rate with targeted human annotation," Univ. Maryland, College Park, MD, USA and , BBN Technol., Cambridge, MA, USA, Tech. Rep. LAMP-TR-126, CS-TR-4755, UMIACS-TR-2005-58, Jul. 2005.

[10] L. Liu and L. Huang, "Search-aware tuning for machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Doha, Qatar, Oct. 2014, pp. 1942–1952.

[11] H. Yu, L. Huang, H. Mi, and K. Zhao, "Max-violation perceptron and forced decoding for scalable MT training," in *Proc. Conf. Empirical Methods Natural Lang. Process*, Seattle, WA, USA, Oct. 2013, pp. 1112–1123.

[12] L. Huang, S. Fayong, and Y. Guo, "Structured perceptron with inexact search," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, Montréal, QC, Canada, Jun. 2012, pp. 142–151.

[13] X. He and L. Deng, "Maximum expected Bleu training of phrase and lexicon translation models," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics (Vol. 1, Long Papers)*, Jeju Island, Korea, Jul. 2012, pp. 292–301.

[14] D. Chiang, "Hierarchical phrase-based translation," *Comput. Linguistics*, vol. 33, pp. 45–60, 2007.

[15] M. Galley *et al.*,"Scalable inference and training of context-rich syntactic translation models," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, Sydney, Australia, Jul. 2006, pp. 961–968.

[16] T. Xiao, A. de Gispert, J. Zhu, and B. Byrne, "Effective incorporation of source syntax into hierarchical phrase-based translation," in *Proc. 25th Int. Conf. Comput. Linguistics*, Dublin, Ireland, Aug. 2014, pp. 2064–2074.

[17] Y. Liu, H. Mi, Y. Feng, and Q. Liu, "Joint decoding with multiple translation models," in *Proc. Joint Conf. 47th Ann. Meeting Assoc. Comput. Linguistics 4th Int. Joint Conf. Natural Lang. Process.*, Suntec, Singapore, Aug. 2009, pp. 576–584.

[18] M. Cmejrek, H. Mi, and B. Zhou, "Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Seattle, WA, USA, Oct. 2013, pp. 545–555.

[19] A. V. Aho and J. D. Ullman, "Syntax directed translations and the pushdown assembler," *J. Comput. Syst. Sci.*, vol. 3, pp. 37–57, 1969.

[20] F. Och and H. Ney, "Discriminative training and maximum entropy models for statistical machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, Philadelphia, PA, USA, Jul. 2002, pp. 295–302.

[21] P. Koehn, F. Och, and D. Marcu, "Statistical phrase-based translation," in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, Edmonton, AB, Canada, Jun. 2003, pp. 48–54.

[22] R. M. Neal and G. E. Hinton, *A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants*, M. I. Jordan, Ed. Norwell, MA, USA: Kluwer, 1998.

[23] D. H. Younger, "Recognition and parsing of context-free languages in time n^3," *Inf. Control*, vol. 10, pp. 189–208, 1967.

[24] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*. Hoboken, NJ, USA: Wiley, 1983.

[25] R. Tromble, S. Kumar, F. Och, and W. Macherey, "Lattice minimum Bayes-risk decoding for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Honolulu, HI, USA, Oct. 2008, pp. 620–629.

[26] S. Kumar, W. Macherey, C. Dyer, and F. Och, "Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices," in *Proc. Joint Conf. 47th Annu. Meeting. Assoc. Comput. Linguistics 4th Int. Joint Conf. Natural Lang. Process.*, Suntec, Singapore, Aug. 2009, pp. 163–171.

[27] P. E. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Comput. Linguistics*, vol. 19, pp. 263–311, 1993.

[28] H. Mi and L. Huang, "Forest-based translation rule extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Honolulu, HI, USA, Oct. 2008, pp. 206–214.

[29] S. Matsoukas, A.-V. I. Rosti, and B. Zhang, "Discriminative corpus weight estimation for machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Singapore, Aug. 2009, pp. 708–717.

[30] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. Ser. B*, vol. 39, pp. 1–38, 1977.

[31] F. Och, "Minimum error rate training in statistical machine translation," in *Proc. 41st Annu. Meeting Assoc. Comput. Linguistics*, Sapporo, Japan, Jul. 2003, pp. 160–167.

[32] J. Zhu, M. Zhu, Q. Wang, and T. Xiao, "Niuparser: A Chinese syntactic and semantic parsing toolkit," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Asian Fed. Natural Lang. Process. Sys. Demonstrations*, Beijing, China, Jul. 2015, pp. 145–150.

[33] S. Petrov and D. Klein, "Improved inference for unlexicalized parsing," in *Proc. Human Lang. Technol., Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, Rochester, NY, USA, Apr. 2007, pp. 404–411.

[34] T. Xiao, J. Zhu, H. Zhang, and Q. Li, "NiuTrans: An open source toolkit for phrase-based and syntax-based machine translation," in *Proc. 50th Annu. Meeting. Assoc. Comput. Linguistics Syst. Demonstrations*, Jeju Island, Korea, Jul. 2012, pp. 19–24.

[35] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, pp. 359–393, 1999.

[36] L. Schwartz, "Multi-source translation methods," in *Proc. 8th Conf. Assoc. Mach. Transl. Amer.*, Waikiki, HI, USA, 2008, pp. 279–288.

[37] H. Daumé III, "Practical structured learning techniques for natural language processing," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, USA, 2006.

[38] P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar, "An end-to-end discriminative approach to machine translation," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, Sydney, Australia, Jul. 2006, pp. 761–768.

[39] G. Doddington, "Automatic evaluation of machine translation quality using n-gram cooccurrence statistics," in *Proc. 2nd Int. Conf. Human Lang. Technol. Conf.*, San Diego, CA, USA, 2002, pp. 128–132.

[40] J. Turian, L. Shen, and I. D. Melamed, "Evaluation of machine translation and its evaluation," in *Proc. Mach. Transl. Summit IX*, New Orleans, LA, USA, 2003, pp. 386–393.
[41] S. Nieβen, F. J. Och, G. Leusch, and H. Ney, "An evaluation tool for machine translation: Fast evaluation for machine translation research," in *Proc. 2nd Int. Conf. Lang. Resources Eval.*, Athens, Greece, May 2000, pp. 39–45.
[42] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf, "Accelerated DP based search for statistical translation," in *Proc. Eur. Conf. Speech Commun. Technol.*, Rhodes, Greece, Sep. 1997, pp. 2667–2670.
[43] P. Blunsom, T. Cohn, and M. Osborne, "A discriminative latent variable model for statistical machine translation," in *Proc. Assoc. Comput. Linguistics Human Lang. Technol.*, Columbus, OH, USA, Jun. 2008, pp. 200–208.
[44] P. Blunsom, T. Cohn, C. Dyer, and M. Osborne, "A GIBBs sampler for phrasal synchronous grammarinduction," in *Proc. Joint Conf. 47th Annu. Meeting Assoc. Comput. Linguistics 4th Int. Joint Conf. Natural Lang. Process.*, Suntec, Singapore, Aug. 2009, pp. 782–790.
[45] T. Cohn and P. Blunsom, "A Bayesian model of syntax-directed tree to string grammarinduction," in *Proc. Conf. Empirical Methods. Natural Lang. Process.*, Singapore, Aug. 2009, pp. 352–361.
[46] D. A. Smith and J. Eisner, "Minimum risk annealing for training log-linear models," in *Proc. Int. Conf. Comput. Linguistics Assoc. Comput. Linguistics Main Conf. Poster Sessions*, Sydney, Australia, Jul. 2006, pp. 787–794.
[47] Z. Li and J. Eisner, "First- and second-order expectation semirings with applications to minimum-risk training on translation forests," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Singapore, Aug. 2009, pp. 40–51.
[48] T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki, "Online large-margin training for statistical machine translation," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, Prague, Czech Republic, Jun. 2007, pp. 764–773.
[49] D. Chiang, K. Knight, and W. Wang, "11,001 new features for statistical machine translation," in *Proc. Human Lang. Technol., Annu. Conf. North Amer. Chapter. Assoc. Comput. Linguistics*, Boulder, CO, USA, Jun. 2009, pp. 218–226.
[50] M. Hopkins and J. May, "Tuning as ranking," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Edinburgh, U.K., Jul. 2011, pp. 1352–1362.
[51] M. Dreyer and Y. Dong, "APRO: All-pairs ranking optimization for MT tuning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, Denver, CO, USA, May–Jun. 2015, pp. 1018–1023.
[52] Y. Zhang and S. Clark, "Syntactic processing using the generalized perceptron and beam search," *Comput. Linguistics*, vol. 37, pp. 105–151, 2011.
[53] K. Zhao, L. Huang, H. Mi, and A. Ittycheriah, "Hierarchical MT training using max-violation perceptron," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 2, Short Papers)*, Baltimore, MD, USA, Jun. 2014, pp. 785–790.
[54] A.-V. Rosti, B. Zhang, S. Matsoukas, and R. Schwartz, "Expected BLEU training for graphs: BBN system description for WMT11 system combination task," in *Proc. 6th Workshop. Statist. Mach. Transl.*, Edinburgh, U.K., Jul. 2011, pp. 159–165.
[55] M.-W. Chang, L. Ratinov, and D. Roth, "Guiding semi-supervision with constraint-driven learning," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*, Prague, Czech Republic, Jun. 2007, pp. 280–287.
[56] A. Haghighi, J. Blitzer, J. DeNero, and D. Klein, "Better word alignments with supervised ITG models," in *Proc. Joint Conf. 47th Annu. Meeting Assoc. Comput. Linguistics 4th Int. Joint Conf. Natural Lang. Process. Asian Fed. Natural Lang. Process.*, Suntec, Singapore, Aug. 2009, pp. 923–931.
[57] G. Druck, K. Ganchev, and J. Graca, "Rich prior knowledge in learning for natural language processing," in *Proc. Tut. Abstracts Assoc. Comput. Linguistics*, Portland, OR, USA, Jun. 2011, Art. no. 5.
[58] A. Saluja, I. Lane, and Y. Zhang, "Machine translation with binary feedback: A largemargin approach," in *Proc. 10th Biennial Conf. Assoc. Mach. Transl. Amer.*, San Diego, CA, USA, Oct. 2012.
[59] A. Ljolje and M. Riley, "Automatic segmentation and labeling of speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1991, pp. 473–476.
[60] P. J. Moreno, C. Joerg, J. manuel Van Thong, and O. Glickman, "A recursive algorithm for the forced alignment of very long audio segments," in *Proc. Int. Conf. Spoken Lang. Process.*, Sydney, Australia, 1998, pp. 2711–2714.
[61] P. J. Moreno and C. Alberti, "A factor automaton approach for the forced alignment of long speech recordings," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2009, pp. 4869–4872.
[62] D. Caseiro, H. Meinedo, A. Serralheiro, I. Trancoso, and J. Neto, "Spoken book alignment using WFSTs," in *Proc. 2nd Int. Conf. Human Lang. Technol. Res.*, 2002, pp. 194–196.

Tong Xiao received the Bachelor's, Master's, and Ph.D. degrees in computer science all from Northeastern University, Shenyang, China, in 2005, 2008, and 2012, respectively. In 2014, he was elected as a candidate of the Excellent Ph.D. Dissertation Award established by Chinese Information Processing Society of China. He is currently an Associate Professor in the College of Computer Science and Engineering, Northeastern University. He is a Co-PI of the NiuTrans MT project. His current research interests include machine translation and language modeling.

Derek F. Wong received the Ph.D. degree in automation from Tsinghua University, Beijing, China, in 2005. He is currently an Assistant Professor in the Department of Computer and Information Science, University of Macau, Macau, China, with a secondary appointment as a Project Manager in the Instituto de Engenharia de Sistemas e Computadores de Macau during 2003–2013. His active and diverse research interests include the areas of natural language processing and machine translation. He is the Leader of the Natural Language Processing & Portuguese—Chinese Machine Translation (NLP^2CT) research group and the Founder of the NLP^2CT laboratory.

Jingbo Zhu received the Ph.D. degree in computer science from Northeastern University, Shenyang, China, in 1999. He has been with Northeastern University, since 1999. He is currently a Full Professor with the College of Computer Science and Engineering and is in charge of research activities within the Natural Language Processing Laboratory. He has published more than 180 papers and holds four US patents. His current research interests include syntactic parsing, machine translation, and machine learning for natural language processing.