Max-Violation Perceptron and Forced Decoding for Scalable MT Training

Heng Yu[1]* **Liang Huang**[2]† **Haitao Mi**[3] **Kai Zhao**[2]

[1]Institute of Computing Tech. Chinese Academy of Sciences
yuheng@ict.ac.cn

[2]Queens College & Grad. Center City University of New York
{huang@cs.qc,kzhao@gc}.cuny.edu

[3]T.J. Watson Research Center IBM
hmi@us.ibm.com

Abstract

While large-scale discriminative training has triumphed in many NLP problems, its definite success on machine translation has been largely elusive. Most recent efforts along this line are not scalable (training on the small dev set with features from top ∼100 most frequent words) and overly complicated. We instead present a very simple yet theoretically motivated approach by extending the recent framework of "violation-fixing perceptron", using forced decoding to compute the target derivations. Extensive phrase-based translation experiments on both Chinese-to-English and Spanish-to-English tasks show substantial gains in BLEU by up to +2.3/+2.0 on dev/test over MERT, thanks to 20M+ sparse features. This is the first successful effort of large-scale online discriminative training for MT.

1 Introduction

Large-scale discriminative training has witnessed great success in many NLP problems such as parsing (McDonald et al., 2005) and tagging (Collins, 2002), but not yet for machine translation (MT) despite numerous recent efforts. Due to scalability issues, most of these recent methods can only train on a small dev set of about a thousand sentences rather than on the full training set, and only with 2,000–10,000 rather "dense-like" features (either unlexicalized or only considering highest-frequency words), as in MIRA (Watanabe et al., 2007; Chiang et al., 2008; Chiang, 2012), PRO (Hopkins and May, 2011), and RAMP (Gimpel and Smith, 2012). However, it is well-known that the most important features for NLP are lexicalized, most of which can not be seen on a small dataset. Furthermore, these methods often involve complicated loss functions and intricate choices of the "target" derivations to update towards or against (e.g. k-best/forest oracles, or hope/fear derivations), and are thus hard to replicate. As a result, the classical method of MERT (Och, 2003) remains the default training algorithm for MT even though it can only tune a handful of dense features. See also Section 6 for other related work.

As a notable exception, Liang et al. (2006) do train a structured perceptron model on the training data with sparse features, but fail to outperform MERT. We argue this is because structured perceptron, like many structured learning algorithms such as CRF and MIRA, assumes exact search, and search errors inevitably break theoretical properties such as convergence (Huang et al., 2012). Empirically, it is now well accepted that standard perceptron performs poorly when search error is severe (Collins and Roark, 2004; Zhang et al., 2013).

To address the search error problem we propose a very simple approach based on the recent framework of "violation-fixing perceptron" (Huang et al., 2012) which is designed specifically for inexact search, with a theoretical convergence guarantee and excellent empirical performance on beam search parsing and tagging. The basic idea is to update when search error happens, rather than at the end of the search. To adapt it to MT, we extend this framework to handle latent variables corresponding to the hidden derivations. We update towards "gold-standard" derivations computed by forced decoding so that each derivation leads to the exact reference translation. Forced decoding is also used as a way of data selection, since those reachable sentence pairs are generally more literal and of higher quality, which the training should focus on. When the reachable subset is small for some language pairs, we augment

*Work done while visiting City University of New York.
†Corresponding author.

it by including reachable prefix-pairs when the full sentence pair is not.

We make the following contributions:

1. Our work is the first successful effort to scale online structured learning to a large portion of the training data (as opposed to the dev set).

2. Our work is the first to use a principled learning method customized for inexact search which updates on partial derivations rather than full ones in order to fix search errors. We adapt it to MT using latent variables for derivations.

3. Contrary to the common wisdom, we show that simply updating towards the exact reference translation is helpful, which is much simpler than k-best/forest oracles or loss-augmented (e.g. hope/fear) derivations, avoiding sentence-level BLEU scores or other loss functions.

4. We present a convincing analysis that it is the search errors and standard perceptron's inability to deal with them that prevent previous work, esp. Liang et al. (2006), from succeeding.

5. Scaling to the training data enables us to engineer a very rich feature set of sparse, lexicalized, and non-local features, and we propose various ways to alleviate overfitting.

For simplicity and efficiency reasons, in this paper we use phrase-based translation, but our method has the potential to be applicable to other translation paradigms. Extensive experiments on both Chinese-to-English and Spanish-to-English tasks show statistically significant gains in BLEU by up to +2.3/+2.0 on dev/test over MERT, and up to +1.5/+1.5 over PRO, thanks to 20M+ sparse features.

2 Phrase-Based MT and Forced Decoding

We first review the basic phrase-based decoding algorithm (Koehn, 2004), which will be adapted for forced decoding.

2.1 Background: Phrase-based Decoding

We will use the following running example from Chinese to English from Mi et al. (2008):

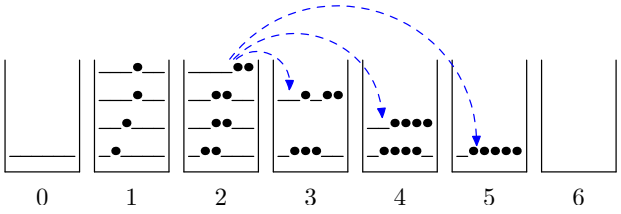


Figure 1: Standard beam-search phrase-based decoding.

Bùshí yǔ Shālóng jǔxíng le huìtán
Bush with Sharon hold -ed meeting

'Bush held a meeting with Sharon'

Phrase-based decoders generate partial target-language outputs in left-to-right order in the form of *hypotheses* (or states) (Koehn, 2004). Each hypothesis has a *coverage vector* capturing the source-language words translated so far, and can be extended into a longer hypothesis by a phrase-pair translating an uncovered segment. For example, the following is one possible derivation:

$$\dfrac{\dfrac{\dfrac{(0\_\_\_\_\_) : (0, \text{“”})}{(\bullet_1\_\_\_\_\_) : (s_1, \text{“Bush”})}\ r_1}{(\bullet\_\_\bullet\bullet\bullet_6) : (s_2, \text{“Bush held talks”})}\ r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet) : (s_3, \text{“Bush held talks with Sharon”})}\ r_3$$

where a $\bullet$ in the coverage vector indicates the source word at this position is "covered" and where each s_i is the score of each state, each adding the rule score and the distortion cost (dc) to the score of the previous state. To compute the distortion cost we also need to maintain the ending position of the last phrase (e.g., the $_3$ and $_6$ in the coverage vectors). In phrase-based translation there is also a *distortion-limit* which prohibits long-distance reorderings.

The above states are called $-$LM states since they do not involve language model costs. To add a bigram model, we split each $-$LM state into a series of $+$LM states; each $+$LM state has the form $(v,^a)$ where a is the last word of the hypothesis. Thus a $+$LM version of the above derivation might be:

$$\dfrac{\dfrac{\dfrac{(0\_\_\_\_\_,^{\text{<s>}}) : (0, \text{“<s>”})}{(\bullet_1\_\_\_\_\_,^{\text{Bush}}) : (s'_1, \text{“<s> Bush”})}\ r_1}{(\bullet\_\_\bullet\bullet\bullet_6,^{\text{talks}}) : (s'_2, \text{“<s> Bush held talks”})}\ r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet,^{\text{Sharon}}) : (s'_3, \text{“<s> Bush held ... with Sharon”})}\ r_3$$

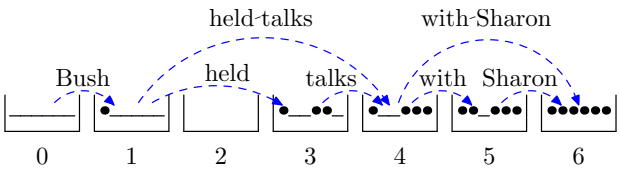


Figure 2: Forced decoding and y-good derivation lattice.

where the score of applying each rule now also includes a *combination cost* due to the bigrams formed when applying the phrase-pair, e.g.

$$s'_3 = s'_2 + s(r_3) + dc(|6-3|) - \log P_{lm}(\text{with} \mid \text{talk})$$

To make this exponential-time algorithm practical, beam search is the standard approximate search method (Koehn, 2004). Here we group +LM states into n bins, with each bin B_i hosting at most b states that cover exactly i Chinese words (see Figure 1).

2.2 Forced Decoding

The idea of forced decoding is to consider only those (partial) derivations that can produce (a prefix of) the exact reference translation (assuming single reference). We call these partial derivations "y-good" derivations (Daumé, III and Marcu, 2005), and those that deviate from the reference translation "y-bad" derivations. The forced decoding algorithm is very similar to +LM decoding introduced above, with the new "forced decoding LM" to be defined as only accepting two consecutive words on the reference translation, ruling out any y-bad hypothesis:

$$P_{forced}(b \mid a) = \begin{cases} 1 & \text{if } \exists j, \text{ s.t. } a = y_j \text{ and } b = y_{j+1} \\ 0 & \text{otherwise} \end{cases}$$

In the +LM state, we can simply replace the boundary word by the index on the reference translation:

$$\frac{\dfrac{\dfrac{(0\_\_\_\_\_,{}^{0}) : (0, \text{“<s>”})}{(\bullet_1\_\_\_\_,{}^{1}) : (w'_1, \text{“<s> Bush”})}\, r_1}{(\bullet\_\_\bullet\bullet\bullet_6,{}^{3}) : (w'_2, \text{“<s> Bush held talks”})}\, r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet,{}^{5}) : (w'_3, \text{“<s> Bush held talks with Sharon”})}\, r_3$$

The complexity of this forced decoding algorithm is reduced to $O(2^n n^3)$ where n is the source sentence length, without the expensive bookkeeping for English boundary words.

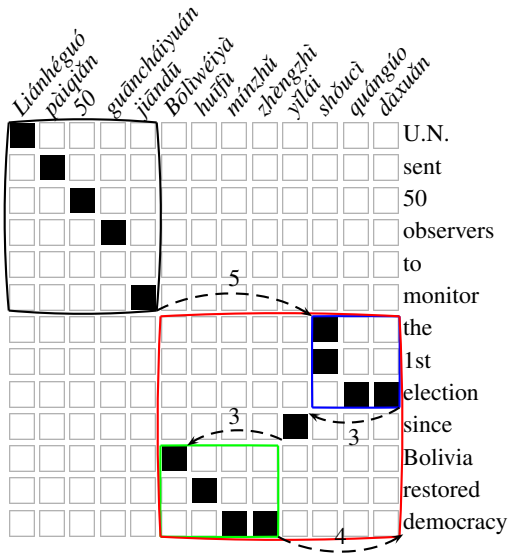


Figure 3: Example of unreachable sentence pair and reachable prefix-pair. The first big jump is disallowed for a distortion limit of 4, but we can still extract the top-left box as a reachable prefix-pair. Note that this example is perfectly reachable in syntax-based MT.

2.3 Reachable Prefix-Pairs

In practice, many sentence pairs in the parallel text fail in forced decoding due to two reasons:

1. **distortion limit**: long-distance reorderings are disallowed but are very common between languages with very different word orders such as English and Chinese.

2. **noisy alignment and phrase limit**: the word-alignment quality (typically from GIZA++) are usually very noisy, which leads to unnecessarily big chunks of rules beyond the phrase limit.

If we only rely on the reachable whole sentence pairs, we will not be able to use much of the training set for Chinese-English. So we propose to augment the set of reachable examples by considering reachable prefix-pairs (see Figure 3 for an example).

3 Violation-Fixing Perceptron for MT

Huang et al. (2012) establish a theoretical framework called "violation-fixing perceptron" which is tailored for structured learning with inexact search and has provable convergence properties. The high-level idea is that standard full update does not fix search errors; to do that we should instead update when search error occurs, e.g., when the gold-

standard derivation falls below the beam. Huang et al. (2012) show dramatic improvements in the quality of the learned model using violation-fixing perceptron (compared to standard perceptron) on incremental parsing and part-of-speech tagging.

Since phrase-based decoding is also an incremental search problem which closely resembles beam-search incremental parsing, it is very natural to employ violation-fixing perceptron here for MT training. Our goal is to produce the exact reference translation, or in other words, we want at least one y-good derivation to survive in the beam search.

To adapt the violation-fixing perceptron framework to MT we need to extend the framework to handle latent variables since the gold-standard derivation is not observed. This is done in a way similar to the latent variable structured perceptron (Zettlemoyer and Collins, 2005; Liang et al., 2006; Sun et al., 2009) where each update is from the best (y-bad) derivation towards the best y-good derivation in the current model; the latter is a constrained search which is exactly forced decoding in MT.

3.1 Notations

We first establish some necessary notations. Let $\langle x, y \rangle$ be a sentence pair in the training data, and

$$d = r_1 \circ r_2 \circ \ldots \circ r_{|d|}$$

be a (partial) derivation, where each $r_i = \langle c(r_i), e(r_i) \rangle$ is a rule, i.e., a Chinese-English phrase-pair. Let $|c(d)| \triangleq \sum_i |c(r_i)|$ be the number of Chinese words covered by this derivation, and $e(d) \triangleq e(r_1) \circ e(r_2) \ldots \circ e(r_{|d|})$ be the English prefix generated so far. Let $D(x)$ be the set of all possible partial derivations translating part of the input sentence x. Let $pre(y) \triangleq \{y_{[0:j]} \mid 0 \leq j \leq |y|\}$ be the set of prefixes of the reference translation y, and $good_i(x, y)$ be the set of partial y-good derivations whose English side is a prefix of the reference translation y, and whose Chinese projection covers exactly i words on the input sentence x, i.e.,

$$good_i(x, y) \triangleq \{d \in D(x) \mid e(d) \in pre(y), |c(d)| = i\}.$$

Conversely, we define the set of y-bad partial derivations covering i Chinese words to be:

$$bad_i(x, y) \triangleq \{d \in D(x) \mid e(d) \notin pre(y), |c(d)| = i\}.$$

Basically, at each bin B_i, y-good derivations $good_i(x, y)$ and y-bad ones $bad_i(x, y)$ compete for the b slots in the bin:

$$B_0 = \{\epsilon\} \quad (1)$$

$$B_i = \mathbf{top}^b \bigcup_{j=1..l} \{d \circ r \mid d \in B_{i-j}, |c(r)| = j\} \quad (2)$$

where r is a rule covering j Chinese words, l is the phrase-limit, and $\mathbf{top}^b\ S$ is a shorthand for $\mathbf{argtop}^b_{d \in S}\ \mathbf{w} \cdot \mathbf{\Phi}(x, d)$ which selects the top b derivations according to the current model $\mathbf{w}$.

3.2 Algorithm 1: Early Update

As a special case of violation-fixing perceptron, early update (Collins and Roark, 2004) stops decoding whenever the gold derivation falls off the beam, makes an update on the prefix so far and move on to the next example. We adapt it to MT as follows: if at a certain bin B_i, all y-good derivations in $good_i(x, y)$ have fallen off the bin, then we stop and update, rewarding the best y-good derivation in $good_i(x, y)$ (with respect to current model $\mathbf{w}$), and penalizing the best y-bad derivation in the same step:

$$d_i^+(x, y) \triangleq \mathop{\mathbf{argmax}}_{d \in good_i(x,y)} \mathbf{w} \cdot \mathbf{\Phi}(x, d) \quad (3)$$

$$d_i^-(x, y) \triangleq \mathop{\mathbf{argmax}}_{d \in bad_i(x,y) \cap B_i} \mathbf{w} \cdot \mathbf{\Phi}(x, d) \quad (4)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x,\ d_i^+(x, y),\ d_i^-(x, y)) \quad (5)$$

where $\Delta\mathbf{\Phi}(x, d, d') \triangleq \mathbf{\Phi}(x, d) - \mathbf{\Phi}(x, d')$ is a shorthand notation for the difference of feature vectors. Note that the set $good_i(x, y)$ is independent of the beam search and current model and is instead precomputed in the forced decoding phase, whereas the negative signal $d_i^-(x, y)$ depends on the beam.

In practice, however, there are exponentially many y-good derivations for each reachable sentence pair, and our goal is just to make sure (at least) one y-good derivation triumphs at the end. So it is possible that at a certain bin, all y-good partial derivations fall off the bin, but the search can still continue and produce the exact reference translation through some other y-good path that avoids that bin. For example, in Figure 1, the y-good states in steps 3 and 5 are not critical; it is totally fine to miss them in the search as long as we save the y-good states

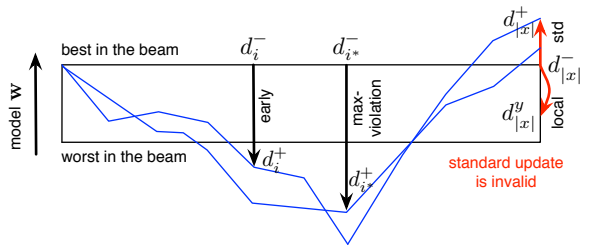


Figure 4: Illustration of four update methods. The blue paths denote (possibly lots of) gold-standard derivations from forced decoding. Standard update in this case is invalid as it reinforces the error of $\mathbf{w}$ (Huang et al., 2012).

in bins 1, 4 and 6. So we actually use a "softer" version of the early update algorithm: only stop and update when there is no hope to continue. To be more concrete, let l denote the phrase-limit then we stop where there are l consecutive bins without any y-good states, and update on the first among them.

3.3 Algorithm 2: Max-Violation Update

While early update learns substantially better models than standard perceptron in the midst of inexact search, it is also well-known to be converging much slower than the latter, since each update is on a (short) prefix. Huang et al. (2012) propose an improved method "*max-violation*" which updates at the worst mistake instead of the first, and converges much faster than early update with similar or better accuracy. We adopt this idea here as follows: decode the whole sentence, and find the step i^* where the difference between the best y-good derivation and the best y-bad one is the biggest. This amount of difference is called the amount of "violation" in Huang et al. (2012), and the place of maximum violation is intuitively the site of the biggest mistake during the search. More formally, the update rule is:

$$i^* \triangleq \operatorname*{argmin}_i \mathbf{w} \cdot \Delta\boldsymbol{\Phi}(x,\, d_i^+(x,y),\, d_i^-(x,y)) \quad (6)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x,\, d_{i^*}^+(x,y),\, d_{i^*}^-(x,y)) \quad (7)$$

3.4 Previous Work: Standard and Local Updates

We compare the above new update methods with the two existing ones from Liang et al. (2006).

Standard update (also known as "bold update" in Liang et al. (2006)) simply updates at the very end, from the best derivation in the beam towards the best gold-standard derivation (regardless of whether it survives the beam search):

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x, d_{|x|}^+(x,y), d_{|x|}^-(x,y)) \quad (8)$$

Local update, however, updates towards the derivation in the final bin that is most similar to the reference y, denoted $d_{|x|}^y(x,y)$:

$$d_{|x|}^y(x,y) = \operatorname*{argmax}_{d \in B_{|x|}} \mathrm{Bleu}^{+1}(y, e(d)) \quad (9)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x, d_{|x|}^y(x,y), d_{|x|}^-(x,y)) \quad (10)$$

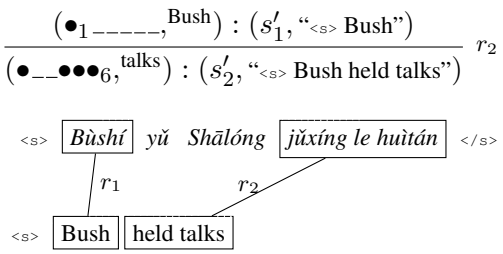
where $\mathrm{Bleu}^{+1}(\cdot,\cdot)$ returns the sentence-level BLEU.

Liang et al. (2006) observe that standard update performs worse than local update, which they attribute to the fact that the former often update towards a gold derivation made up of "unreasonable" rules. Here we give a very different but theoretically more reasonable explanation based on the theory of Huang et al. (2012), who define an update $\Delta\boldsymbol{\Phi}(x, d^+, d^-)$ to be **invalid** if d^+ scores higher than d^- (i.e., $\mathbf{w} \cdot \Delta\boldsymbol{\Phi}(x, d^+, d^-) > 0$, or update $\Delta\mathbf{w}$ points to the same direction as $\mathbf{w}$ in Fig. 4), in which case there is no "violation" or mistake to fix. Perceptron is guaranteed to converge if all updates are valid. Clearly, early and max-violation updates are valid. But standard update is not: it is possible that at the end of search, the best y-good derivation $d_{|x|}^+(x,y)$, though pruned earlier in the search, ranks even higher in the current model than anything in the final bin (see Figure 4). In other words, there is no mistake at the final step, while there must be some search error in earlier steps which expels the y-good subderivation. We will see in Section 5.3 that invalid updates due to search errors are indeed the main reason why standard update fails. Local update, however, is always valid in that definition.

Finally, it is worth noting that in terms of implementation, standard and max-violation are the easiest, while early update is more involved.

4 Feature Design

Our feature set includes the following 11 dense features: LM, four conditional and lexical translation probabilities ($p_c(e|f)$, $p_c(f|e)$, $p_l(e|f)$, $p_l(f|e)$), length and phrase penalties, distortion cost, and three lexicalized reordering features. All these features are inherited from Moses (Koehn et al., 2007).



features for applying r_2 on span $x_{[3:6]}$

WordEdges	$c(r_2)_{[0]} =$ *jǔxíng*, $c(r_2)_{[-1]} =$ *huìtán* $e(r_2)_{[0]} =$ held, $e(r_2)_{[-1]} =$ talks $x_{[2:3]} =$ *Shālóng*, $x_{[6:7]} =$ </s>, $\lvert c(r_2)\rvert = 3$ ... (**combos** of the above **atomic** features) ...
non-local	$e(r_0 \circ r_1)_{[-2:]} \circ id(r_2)$ $id(r_1) \circ id(r_2)$

Figure 5: Examples of WordEdges and non-local features. The notation uses the Python style subscript syntax.

4.1 Local Sparse Features: Ruleid & WordEdges

We first add the rule identification feature for each rule: $id(r_i)$. We also introduce lexicalized **WordEdges** features, which are shown to be very effective in parsing (Charniak and Johnson, 2005) and MT (Liu et al., 2008; He et al., 2008) literatures. We use the following atomic features when applying a rule $r_i = \langle c(r_i), e(r_i) \rangle$: the source-side length $|c(r_i)|$, the boundary words of both $c(r_i)$ and $e(r_i)$, and the surrounding words of $c(r_i)$ on the input sentence x. See Figure 5 for examples. These atomic features are concatenated to generate all kinds of **combo features**.

Chinese	English	class size		budget
word		52.9k	64.2k	5
characters	-	3.7k	-	3
Brown cluster, full string		200		3
Brown cluster, prefix 6		6	8	2
Brown cluster, prefix 4		4	4	2
POS tag		52	36	2
word type	-	4	-	1

Table 1: Various levels of backoff for WordEdges features. Class size is estimated on the small Chinese-English dataset (Sec. 5.3). The POS tagsets are ICTCLAS for Chinese (Zhang et al., 2003) and Penn Treebank for English (Marcus et al., 1993).

4.2 Addressing Overfitting

With large numbers of lexicalized combo features we will face the overfitting problem, where some combo features found in the training data are too rare to be seen in the test data. Thus we propose three ways to alleviate this problem.

First, we introduce various levels of backoffs for each word w (see Table 1). We include w's Brown cluster and its prefixes of lengths 4 and 6 (Brown et al., 1992), and w's part-of-speech tag. If w is Chinese we also include its word type (punctuations, digits, alpha, or otherwise) and (leftmost or rightmost) character. In such a way, we significantly increase the feature coverage on unseen data.

However, if we allow arbitrary combinations, we can extract a hexalexical feature (4 Chinese + 2 English words) for a local window in Figure 5, which is unlikely to be seen at test time. To control model complexity we introduce a *feature budget* for each level of backoffs, shown in the last column in Table 1. The total budget for a combo feature is the sum of the budgets of all atomic features. In our experiments, we only use the combo features with a total budget of 10 or less, i.e., we can only include bilexical but not trilexical features, and we can include for example combo features with one Chinese word plus two English tags (total budget: 9).

Finally, we use two methods to alleviate overfitting due to one-count rules: for large datasets, we simply remove all one-count rules, but for small datasets where out-of-vocabulary words (OOVs) abound, we use a simple leave-one-out method: when training on a sentence pair (x, y), do not use the one-count rules extracted from (x, y) itself.

4.3 Non-Local Features

Following the success of non-local features in parsing (Huang, 2008) and MT (Vaswani et al., 2011), we also introduce them to capture the contextual information in MT. Our non-local features, shown in Figure 5, include bigram rule-ids and the concatenation of a rule id with the translation history, i.e. the last two English words. Note that we also use backoffs (Table 1) for the words included. Experiments (Section 5.3) show that although the set of non-local features is just a tiny fraction of all features, it contributes substantially to the improvement in BLEU.

Scale	Language Pair	Training Data # sent.	Training Data # words	Reachability sent.	Reachability words	# feats	ΔBLEU # refs	ΔBLEU dev/test	Sections
small	CH-EN	30K	0.8M/1.0M	21.4%	8.8%	7M	4	+2.2/2.0	5.2, 5.3
large		230K	6.9M/8.9M	32.1%	12.7%	23M		+2.3/2.0	5.2, 5.4
large	SP-EN	174K	4.9M/4.3M	55.0%	43.9%	21M	1	+1.3/1.1	5.5

Table 2: Overview of all experiments. The ΔBLEU column shows the absolute improvements of our method MAX-FORCE on dev/test sets over MERT. The Chinese datasets also use prefix-pairs in training (see Table 3).

5 Experiments

In order to test our approach in different language pairs, we conduct three experiments, shown in Table 2, on two significantly different language pairs (long vs. short distance reorderings), Chinese-to-English (CH-EN) and Spanish-to-English (SP-EN).

5.1 System Preparation and Data

We base our experiments on **Cubit**, a state-of-art phrase-based system in Python (Huang and Chiang, 2007).[1] We set phrase-limit to 7 in rule extraction, and beam size to 30 and distortion limit 6 in decoding. We compare our violation-fixing perceptron with two popular tuning methods: MERT (Och, 2003) and PRO (Hopkins and May, 2011).

For word alignments we use GIZA++-ℓ_0 (Vaswani et al., 2012) which produces sparser alignments, alleviating the garbage collection problem. We use the SRILM toolkit (Stolcke, 2002) to train a trigram language model with modified Kneser-Ney smoothing on 1.5M English sentences.

Our dev and test sets for CH-EN task are from the newswire portion of 2006 and 2008 NIST MT Evaluations (616/691 sentences, 18575/18875 words), with four references.[2] The dev and test sets for SP-EN task are from newstest2012 and newstest2013, with only one reference. Below both MERT and PRO tune weights on the dev set, while our method on the training set. Specifically, our method only uses the dev set to know when to stop training.

5.2 Forced Decoding Reachability on Chinese

As mentioned in Section 2.2, we perform forced decoding to select reachable sentences from the training data; this part is done with exact search without any beam pruning. Figure 6 shows the reachability ratio vs. sentence length on the small CH-EN training data, where the ratio decreases sharply with sentence length, and increases with distortion limit. We can see that there are a lot of long distance reorderings beyond small distortion limits. In the extreme case of unlimited distortion, a large amount of sentences will be reachable, but at the cost of much slower decoding ($O(n^2V^2)$ in beam search decoding, and $O(2^n n^3)$ in forced decoding). In fact forced decoding is too slow in the unlimited mode that we only plot reachability for sentences up to 30 words.

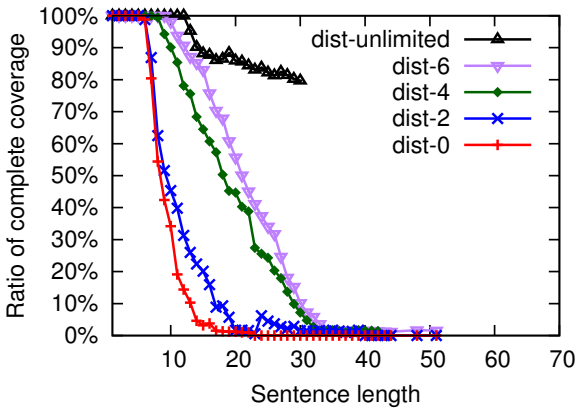


Figure 6: Reachability ratio vs. sentence length on the small CH-EN training set.

	small sent.	small words	large sent.	large words
full	21.4%	8.8%	32.1%	12.7%
+prefix	61.3%	24.6%	67.3%	32.8%

Table 3: Ratio of sentence reachability and word coverage on the two CH-EN training data (distortion limit: 6).

Table 3 shows the statistics of forced decoding on both small and large CH-EN training sets. In the

[1] http://www.cis.upenn.edu/~lhuang3/cubit/. We will release the new version at http://acl.cs.qc.edu.

[2] We use the "average" reference length to compute the brevity penalty factor, which does not decrease with more references unlike the "shortest" heuristic.

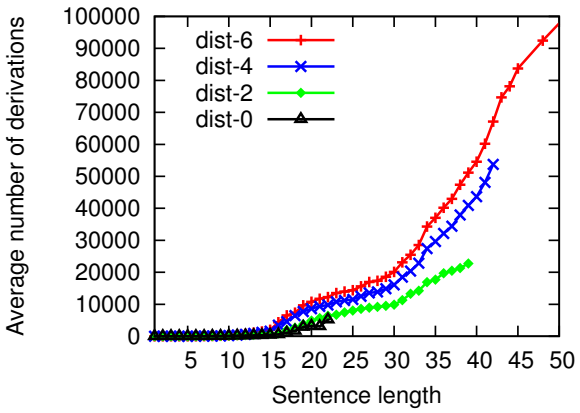


Figure 7: Average number of derivations in gold lattices.

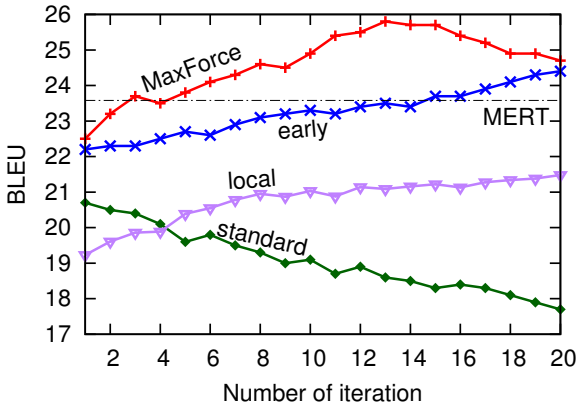


Figure 8: BLEU scores on the heldout dev set for different update methods (trained on small CH-EN data).

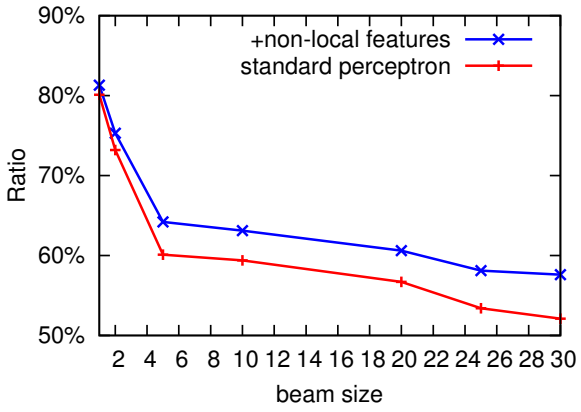


Figure 9: Ratio of invalid updates in standard update.

small data-set, 21.4% sentences are fully reachable which only contains 8.8% words (since shorter sentences are more likely to be reachable). Larger data improves reachable ratios significantly thanks to better alignment quality, but still only 12.7% words can be used. In order to add more examples for perceptron training, we pick all non-trivial reachable prefix-pairs (with 5 or more Chinese words) as additional training examples (see Section 2.2). As shown in Table 3, with prefix-pairs we can use about 1/4 of small data and 1/3 of large data for training, which is 10x and 120x bigger than the 616-sentence dev set.

After running forced decoding, we obtain gold translation lattice for each reachable sentence (or prefix) pair. Figure 7 shows, as expected, the average number of gold derivations in these lattices grows exponentially with sentence length.

5.3 Analysis on Small Chinese-English Data

Figure 8 shows the BLEU scores of different learning algorithms on the dev set. MAXFORCE[3] performs the best, peaking at iteration 13 while early update learns much slower (the first few iterations are faster than other methods due to early stopping but this difference is immaterial later). The local and standard updates, however, underperform MERT; in particular, the latter gets worse as training goes on.

As analysized in Section 3.4, the reason why standard update (or "bold update" in Liang et al. (2006)) fails is that inexact search leads to many invalid updates. This is confirmed by Figure 9, where more than half of the updates remain invalid even at a beam of 30. These analyses provide an alternative but theoretically more reasonable explanation to the findings of Liang et al. (2006): while they blame "unreasonable" gold derivations for the failure of standard update, we observe that it is the search errors that make the real difference, and that an update that respects search errors towards a gold subderivation is indeed helpful, even if that subderivation might be "unreasonable".

In order to speedup training, we use mini-batch parallelization of Zhao and Huang (2013) which has been shown to be much faster than previous parallelization methods. We set the mini-batch size to 24 and train MAXFORCE with 1, 6, and 24 cores on a small subset of the our original reachable sen-

[3] Stands for **Max**-Violation Perceptron w/ **Force**d Decoding

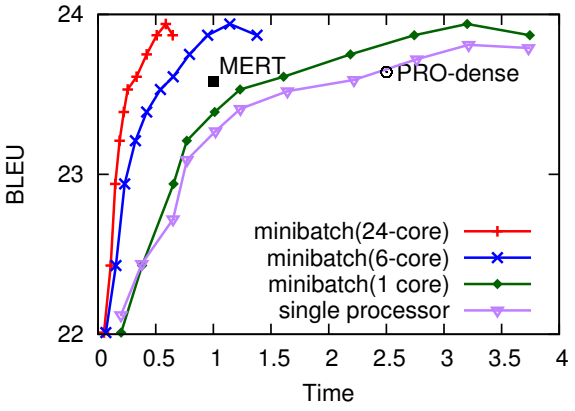


Figure 10: Minibatch parallelization speeds up learning.

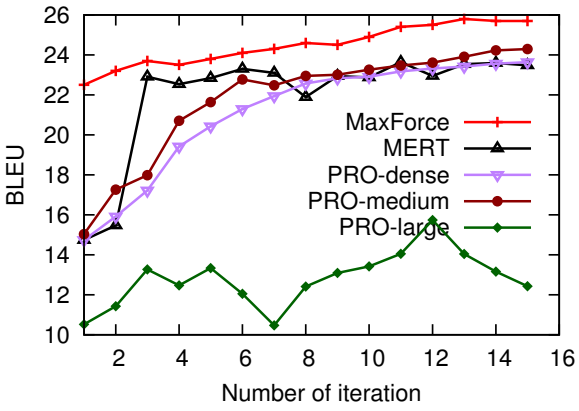


Figure 11: Comparison between different training methods. Ours trains the training set while others on dev set.

tences. The number of sentence pairs in this subset is 1,032, which contains similar number of words to our 616-sentence dev set (since reachable sentences are much shorter). Thus, it is reasonable to compare different learning algorithms in terms of speed and performance. Figure 10 shows that first of all, minibatch improves BLEU even in the serial setting, and when run on 24 cores, it leads to a speedup of about 7x. It is also interesting to know that on 1 CPU, minibatch perceptron takes similar amount of time to reach the same performance as MERT and PRO.

Figure 11 compares the learning curves of MAXFORCE, MERT, and PRO. We test PRO in three different ways: PRO-dense (dense features only), PRO-medium (dense features plus top 3K most frequent sparse features[4]), and PRO-large (dense features plus all sparse features). The results show that PRO-dense performs almost the same as MERT but with a stabler learning curve while PRO-medium improves by +0.6. However, PRO-large decreases the performance significantly, which indicates PRO is not scalable to truly sparse features. By contrast, our method handles large-scale sparse features well and outperforms all other methods by a large margin and with a stable learning curve.

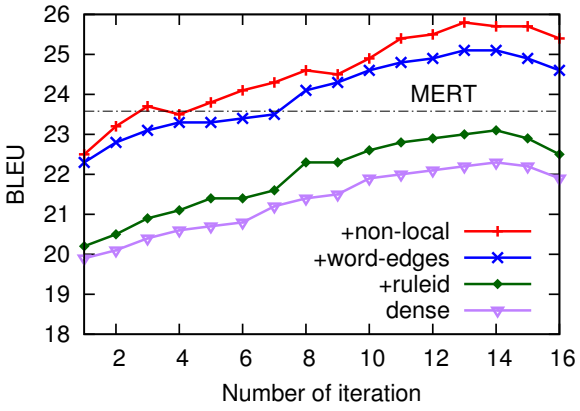


Figure 12: Incremental contributions of different feature sets (dense features, ruleid, WordEdges, and non-local).

type	count	%	BLEU
dense	11	-	22.3
+ruleid	+9,264	+0.1%	+0.8
+WordEdges	+7,046,238	+99.5%	+2.0
+non-local	+22,536	+0.3%	+0.7
all	7,074,049	100%	25.8

Table 4: Feature counts and incremental BLEU improvements. MAXFORCE with all features is +2.2 over MERT.

We also investigate the individual contribution from each group of features (ruleid, WordEdges, and non-local features). So we perform experiments by adding each group incrementally. Figure 12 shows the learning curves and Table 4 lists the counts and incremental contributions of different feature sets. With dense features alone MAXFORCE does not do

[4]To prevent overfitting we remove all lexicalized features and only use Brown clusters. It is difficult to engineer the right feature set for PRO, whereas MAXFORCE is much more robust.

system	algorithm	# feat.	dev	test
Moses	MERT	11	25.5	22.5
Cubit	MERT	11	25.4	22.5
	PRO	11	25.6	22.6
		3K	26.3	23.0
		36K	17.7	14.3
	MAXFORCE	23M	**27.8**	**24.5**

Table 5: BLEU scores (with four references) using the large CH-EN data. Our approach is +2.3/2.0 over MERT.

well because perceptron is known to suffer from features of vastly different scales. Adding ruleid helps, but still not enough. WordEdges (which is the vast majority of features) improves BLEU by +2.0 points and outperforms MERT, when sparse features totally dominate dense features. Finally, the 0.3% non-local features contribute a final +0.7 in BLEU.

5.4 Results on Large Chinese-English Data

Table 5 shows all BLEU scores for different learning algorithms on the large CH-EN data. The MERT baseline on Cubit is essentially the same as Moses. Our MAXFORCE activates 23M features on reachable sentences and prefixes in the training data, and takes 35 hours to finish 15 iterations on 24 cores, peaking at iteration 13. It achieves significant improvements over other approaches: +2.3/+2.0 points over MERT and +1.5/+1.5 over PRO-medium on dev/test sets, respectively.

5.5 Results on Large Spanish-English Data

In SP-EN translation, we first run forced decoding on the training set, and achieve a very high reachability of 55% (with the same distortion limit of 6), which is expected since the word order between Spanish and English are more similar than than between Chinese and English, and most SP-EN reorderings are local. Table 6 shows that MAXFORCE improves the translation quality over MERT by +1.3/+1.1 BLEU on dev/test. These gains are comparable to the improvements on the CH-EN task, since it is well accepted in MT literature that a change of δ in 1-reference BLEU is roughly equivalent to a change of 2δ with 4 references.

system	algorithm	# feat.	dev	test
Moses	MERT	11	27.4	24.4
Cubit	MAXFORCE	21M	**28.7**	**25.5**

Table 6: BLEU scores **(with one reference)** on SP-EN.

6 Related Work

Besides those discussed in Section 1, there are also some research on tuning sparse features on the training data, but they integrate those sparse features into the MT log-linear model as a single feature weight, and tune its weight on the dev set (e.g. (Liu et al., 2008; He et al., 2008; Wuebker et al., 2010; Simianer et al., 2012; Flanigan et al., 2013; Setiawan and Zhou, 2013; He and Deng, 2012; Gao and He, 2013)). By contrast, our approach learns sparse features only on the training set, and use dev set as held-out to know when to stop.

Forced decoding has been used in the MT literature. For example, open source MT systems Moses and cdec have implemented it. Liang et al. (2012) also use the it to boost the MERT tuning by adding more y-good derivations to the standard k-best list.

7 Conclusions and Future Work

We have presented a simple yet effective approach of structured learning for machine translation which scales, for the first time, to a large portion of the whole training data, and enables us to tune a rich set of sparse, lexical, and non-local features. Our approach results in very significant BLEU gains over MERT and PRO baselines. For future work, we will consider other translation paradigms such as hierarchical phrase-based or syntax-based MT.

Acknowledgement

We thank the three anonymous reviewers for helpful suggestions. We are also grateful to David Chiang, Dan Gildea, Yoav Goldberg, Yifan He, Abe Ittycheriah, and Hao Zhang for discussions, and Chris Callison-Burch, Philipp Koehn, Lemao Liu, and Taro Watanabe for help with datasets. Huang, Yu, and Zhao are supported by DARPA FA8750-13-2-0041 (DEFT), a Google Faculty Research Award, and a PSC-CUNY Award, and Mi by DARPA HR0011-12-C-0015. Yu is also supported by the China 863 State Key Project (No. 2011AA01A207). The views and findings in this paper are those of the authors and are not endorsed by the US or Chinese governments.

References

Peter Brown, Peter Desouza, Robert Mercer, Vincent Pietra, and Jenifer Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, Michigan, June.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP 2008*.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *J. Machine Learning Research (JMLR)*, 13:1159–1187.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Hal Daumé, III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of ICML*.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of NAACL 2013*.

Jianfeng Gao and Xiaodong He. 2013. Training mrf-based phrase translation models using gradient ascent. In *Proceedings of NAACL:HLT*, pages 450–459, Atlanta, Georgia, June.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL 2012*.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of ACL*.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of COLING*, pages 321–328, Manchester, UK, August.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., June.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia, July.

Huashen Liang, Min Zhang, and Tiejun Zhao. 2012. Forced decoding for minimum error rate training in statistical machine translation. *Journal of Computational Information Systems*, (8):861868.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proceedings of EMNLP*, pages 89–97, Honolulu, Hawaii, October.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL*.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Hendra Setiawan and Bowen Zhou. 2013. Discriminative training of 150 million translation parameters and its application to pruning. In *Proceedings of NAACL:HLT*, pages 335–341, Atlanta, Georgia, June. ACL.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of ACL*, Jeju Island, Korea.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Xu Sun, Takuya Matsuzaki, and Daisuke Okanohara. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of IJCAI*.

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule markov models for fast tree-to-string translation. In *Proceedings of ACL 2011*, Portland, OR.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the L0-norm. In *Proceedings of ACL*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of ACL*, pages 475–484, Uppsala, Sweden, July.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, pages 184–187.

Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning with inexact hypergraph search. In *Proceedings of EMNLP 2013*.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of NAACL 2013*.