Hierarchical MT Training using Max-Violation Perceptron

Kai Zhao† Liang Huang†
†Graduate Center & Queens College
City University of New York
{kzhao@gc,huang@cs.qc}.cuny.edu

Haitao Mi‡ Abe Ittycheriah‡
‡T. J. Watson Research Center
IBM
{hmi,abei}@us.ibm.com

Abstract

Large-scale discriminative training has become promising for statistical machine translation by leveraging the huge training corpus; for example the recent effort in phrase-based MT (Yu et al., 2013) significantly outperforms mainstream methods that only train on small tuning sets. However, phrase-based MT suffers from limited reorderings, and thus its training can only utilize a small portion of the bitext due to the distortion limit. To address this problem, we extend Yu et al. (2013) to syntax-based MT by generalizing their latent variable “violation-fixing” perceptron from graphs to hypergraphs. Experiments confirm that our method leads to up to +1.2 BLEU improvement over mainstream methods such as MERT and PRO.

1 Introduction

Many natural language processing problems including part-of-speech tagging (Collins, 2002), parsing (McDonald et al., 2005), and event extraction (Li et al., 2013) have enjoyed great success using large-scale discriminative training algorithms. However, a similar success on machine translation has been elusive, where the mainstream methods still tune on small datasets.

What makes large-scale MT training so hard then? After numerous attempts by various researchers (Liang et al., 2006; Watanabe et al., 2007; Arun and Koehn, 2007; Blunsom et al., 2008; Chiang et al., 2008; Flanigan et al., 2013; Green et al., 2013), the recent work of Yu et al. (2013) finally reveals a major reason: it is the vast amount of (inevitable) search errors in MT decoding that astray learning. To alleviate this problem, their work adopts the theoretically-motivated framework of violation-fixing perceptron (Huang et al., 2012) tailored for inexact search, yielding great results on phrase-based MT (outperforming small-scale MERT/PRO by a large margin for the first time). However, the underlying phrase-based model suffers from limited distortion and thus can only employ a small portion (about 1/3 in their Ch-En experiments) of the bitext in training.

To better utilize the large training set, we propose to generalize from phrase-based MT to syntax-based MT, in particular the hierarchical phrase-based translation model (HIERO) (Chiang, 2005), in order to exploit sentence pairs beyond the expressive capacity of phrase-based MT.

The key challenge here is to extend the latent variable violation-fixing perceptron of Yu et al. (2013) to handle tree-structured derivations and translation hypergraphs. Luckily, Zhang et al. (2013) have recently generalized the underlying violation-fixing perceptron of Huang et al. (2012) from graphs to hypergraphs for bottom-up parsing, which resembles syntax-based decoding. We just need to further extend it to handle latent variables. We make the following contributions:

1. We generalize the latent variable violation-fixing perceptron framework to inexact search over hypergraphs, which subsumes previous algorithms for PBMT and bottom-up parsing as special cases (see Fig. 1).

2. We show that syntax-based MT, with its better handling of long-distance reordering, can exploit a larger portion of the training set, which facilitates sparse lexicalized features.

3. Experiments show that our training algorithm outperforms mainstream tuning methods (which optimize on small devsets) by +1.2 BLEU over MERT and PRO on FBIS.
2 Review: Syntax-based MT Decoding

For clarity reasons we will describe HIERO decoding as a two-pass process, first without a language model, and then integrating the LM. This section mostly follows Huang and Chiang (2007).

In the first, −LM phase, the decoder parses the source sentence using the source projection of the synchronous grammar (see Fig. 2 (a) for an example), producing a −LM hypergraph where each node has a signature $N_{[i:j]}$, where $N$ is the nonterminal type (either X or S in HIERO) and $[i:j]$ is the span, and each hyperedge $e$ is an application of the translation rule $r(e)$ (see Figure 3).

To incorporate the language model, each node also needs to remember its target side boundary words. Thus a $−LM$ node $N_{[i:j]}$ is split into multiple $+LM$ nodes of signature $N_{[i:a]}^{[a:b]}$, where $a$ and $b$ are the boundary words. For example, with a bigram LM, $X_{[1:5]}^{\text{held Sharon}}$ is a node whose translation starts with “held” and ends with “Sharon”.

More formally, the whole decoding process can be cast as a deductive system. Take the partial translation of “held talks with Sharon” in Figure 2 (b) for example, the deduction is

$$\begin{align*}
X_{[2:3]}^{\text{Sharon Sharon}} \colon s_1 \\
X_{[3:5]}^{\text{talks talks}} \colon s_2 \\
X_{[1:5]}^{\text{held Sharon}} \colon s_1 + s_2 + s(r_5) + \lambda
\end{align*}$$

where $s(r_5)$ is the score of rule $r_5$, and the LM combo score $\lambda$ is $\log P_{\text{lm}}(\text{talks} | \text{held})P_{\text{lm}}(\text{Sharon} | \text{talks})P_{\text{lm}}(\text{Sharon} | \text{with})$.

3 Violation-Fixing Perceptron for HIERO

As mentioned in Section 1, the key to the success of Yu et al. (2013) is the adoption of violation-fixing perceptron of Huang et al. (2012) which is tailored for vastly inexact search. The general idea is to update somewhere in the middle of the search (where search error happens) rather than at the very end (standard update is often invalid). To adapt it to MT where many derivations can output the same translation (i.e., spurious ambiguity), Yu et al. (2013) extends it to handle latent variables which correspond to phrase-based derivations. On the other hand, Zhang et al. (2013) has generalized Huang et al. (2012) from graphs to hypergraphs for bottom-up parsing, which resembles HIERO decoding. So we just need to combine the two generalizing directions (latent variable and hypergraph, see Fig. 1).

3.1 Latent Variable Hypergraph Search

The key difference between bottom-up parsing and MT decoding is that in parsing the gold tree for each input sentence is unique, while in MT many derivations can generate the same reference translation. In other words, the gold derivation to update towards is a latent variable.
Here we formally define the latent variable “max-violation” perceptron over a hypergraph for MT training. For a given sentence pair \( \langle x, y \rangle \), we denote \( H(x) \) as the decoding hypergraph of Hi-ERO without any pruning. We say \( D \in H(x) \) if \( D \) is a full derivation of \( x \), and \( D \) can be derived from the hypergraph. Let \( \text{good}(x, y) \) be the set of \( y \)-good derivations for \( \langle x, y \rangle \):

\[
\text{good}(x, y) \triangleq \{ D \in H(x) \mid c(D) = y \},
\]

where \( c(D) \) is the translation from derivation \( D \).

We then define the set of \( y \)-good partial derivations that cover \( x_{[i:j]} \) with root \( N_{[i:j]} \) as

\[
\text{good}_{N_{[i:j]}^y}(x, y) \triangleq \{ d \in D \mid D \in \text{good}(x, y), \quad \text{root}(d) = N_{[i:j]} \}
\]

We further denote the real decoding hypergraph with beam-pruning and cube-pruning as \( H'(x) \). The set of \( y \)-bad derivations is defined as

\[
\text{bad}_{N_{[i:j]}^y}(x, y) \triangleq \{ d \in D \mid D \in H'(x, y), \quad \text{root}(d) = N_{[i:j]} \}, \quad d \notin \text{good}_{N_{[i:j]}^y}(x, y) \}
\]

Note that the \( y \)-good derivations are defined over the unpruned whole decoding hypergraph, while the \( y \)-bad derivations are defined over the real decoding hypergraph with pruning.

The max-violation method performs the update where the model score difference between the incorrect Viterbi partial derivation and the best \( y \)-good partial derivation is maximal, by penalizing the incorrect Viterbi partial derivation and rewarding the \( y \)-good partial derivation.

More formally, we first find the Viterbi partial derivation \( d^- \) and the best \( y \)-good partial derivation \( d^+ \) for each \( N_{[i:j]} \) group in the pruned +LM hypergraph:

\[
d^+_N(x, y) \triangleq \arg\max_{d \in \text{good}_N(x, y)} w \cdot \Phi(x, d),
\]

\[
d^-_{N_{[i:j]}^y}(x, y) \triangleq \arg\max_{d \in \text{bad}_N(x, y)} w \cdot \Phi(x, d),
\]

where \( \Phi(x, d) \) is the feature vector for derivation \( d \). Then it finds the group \( N_{[i:j]}^y \) with the maximal score difference between the Viterbi derivation and the best \( y \)-good derivation:

\[
N_{[i:j]}^y \triangleq \arg\max_{N_{[i:j]}} \quad w \cdot \Delta \Phi(x, d^+_N(x, y), d^-_{N_{[i:j]}^y}(x, y)),
\]

and update as follows:

\[
w \leftarrow w + \Delta \Phi(x, d^+_N(x, y), d^-_{N_{[i:j]}^y}(x, y)),
\]

where \( \Delta \Phi(x, d, d') \triangleq \Phi(x, d) - \Phi(x, d') \).

### 3.2 Forced Decoding for HiERO

We now describe how to find the gold derivations. Such derivations can be generated in way similar to Yu et al. (2013) by using a language model tailored for forced decoding:

\[
P_{\text{forced}}(q \mid p) = \begin{cases} 1 & \text{if } q = p + 1 \\ 0 & \text{otherwise} \end{cases},
\]

where \( p \) and \( q \) are the indices of the boundary words in the reference translation. The +LM node now has signature \( N_{[i:j]}^y \), where \( p \) and \( q \) are the indexes of the boundary words. If a boundary word does not occur in the reference, its index is set to \( \infty \) so that its language model score will always be \( -\infty \); if a boundary word occurs more than once in the reference, its \(-\)LM node is split into multiple +LM nodes, one for each such index.\(^2\)

We have a similar deductive system for forced decoding. For the previous example, rule \( r_5 \) in Figure 2 (a) is rewritten as

\[
X \rightarrow \langle y\bar{u} X \bar{M} j\ddot{u}x X \ddot{M}, 1 X \ddot{M}, 4 X \ddot{M} \rangle,
\]

where 1 and 4 are the indexes of reference words “held” and “with” respectively. The deduction for \( X_{[1:5]} \) in Figure 2 (b) is

\[
X_{[1:5]}^{5+5} : s_1 \quad X_{[4:5]}^{2+3} : s_2
\]

\[
X_{[1:3]}^{1+5} : s(r_5) + \lambda + s_1 + s_2
\]

where \( \lambda = \log \prod_{i \in \{1,3,4\}} P_{\text{forced}}(i + 1 \mid i) = 0.\)

### 4 Experiments

Following Yu et al. (2013), we call our max-violation method \textsc{MaxForce}. Our implementation is mostly in Python on top of the \textsc{cdec} system (Dyer et al., 2010) via the \textsc{pydec} interface (Chahuneau et al., 2012). In addition, we use minibatch parallelization of (Zhao and Huang,\(^3\))

\(^3\)We only consider single reference in this paper.

\(^2\)Our formulation of index-based language model fixes a bug in the word-based LM of Yu et al. (2013) when a substring appears more than once in the reference (e.g. "the man...the man..."); thanks to Dan Gildea for pointing it out.
We evaluate MAXFORCE for HIERO over two CH-EN corpora, IWSLT09 and FBIS, and compare the performance with vanilla n-best MERT (Och, 2003) from Moses (Koehn et al., 2007), Hypergraph MERT (Kumar et al., 2009), and PRO (Hopkins and May, 2011) from cdec.

4.1 Features Design

We use all the 18 dense features from cdec, including language model, direct translation probability \( p(e|f) \), lexical translation probabilities \( p_l(e|f) \) and \( p_l(f|e) \), length penalty, counts for the source and target sides in the training corpus, and flags for the glue rules and pass-through rules.

For sparse features we use Word-Edges features (Charniak and Johnson, 2005; Huang, 2008) which are shown to be extremely effective in both parsing and phrase-based MT (Yu et al., 2013). We find that even simple Word-Edges features boost the performance significantly, and adding complex Word-Edges features from Yu et al. (2013) brings limited improvement and slows down the decoding. So in the following experiments we only use Word-Edges features consisting of combinations of English and Chinese words, and Chinese characters, and do not use word clusters nor word types. For simplicity and efficiency reasons, we also exclude all non-local features.

4.2 Datasets and Preprocessing

Our first corpus, IWSLT09, contains \( \sim 30k \) short sentences collected from spoken language. IWSLT04 is used as development set in MAXFORCE training, and as tuning set for n-best MERT, Hypergraph MERT, and PRO. IWSLT05 is used as test set. Both IWSLT04 and IWSLT05 contain 16 references. We mainly use this corpus to investigate the properties of MAXFORCE.

The second corpus, FBIS, contains \( \sim 240k \) sentences. NIST06 newswire is used as development set for MAXFORCE training, and as tuning set for all other tuning methods. NIST08 newswire is used as test set. Both NIST06 newswire and NIST08 newswire contain 4 references. We mainly use this corpus to demonstrate the performance of MAXFORCE in large-scale training.

For both corpora, we do standard tokenization, alignment and rule extraction using the cdec tools. In rule extraction, we remove all 1-count rules but keep the rules mapping from one Chinese word to one English word to help balancing between overfitting and coverage. We use a trigram language model trained from the target sides of the two corpora respectively.

4.3 Forced Decoding Reachability

We first report the forced decoding reachability for HIERO on FBIS in Table 1. With the full rule set, 65% sentences and 55% words of the whole corpus are forced decodable in HIERO. After pruning 1-count rules, our forced decoding covers significantly more words than phrase-based MT in Yu et al. (2013). Furthermore, in phrase-based MT, most decodable sentences are very short, while in HIERO the lengths of decodable sentences are more evenly distributed.

However, in the following experiments, due to efficiency considerations, we use the “tight” rule extraction in cdec that is more strict than the standard “loose” rule extraction, which generates a reduced rule set and, thus, a reduced reachability. We show the reachability distributions of both tight and loose rule extraction in Figure 4.

4.4 Evaluation on IWSLT

For IWSLT, we first compare the performance from various update methods in Figure 5. The max-violation method is more than 15 BLEU
points better than the standard perceptron (also known as “bold-update” in Liang et al. (2006)) which updates at the root of the derivation tree. This can be explained by the fact that in training ~58% of the standard updates are invalid (i.e., they do not fix any violation). We also use the “skip” strategy of Zhang et al. (2013) which updates at the root of the derivation only when it fixes a search error, avoiding all invalid updates. This achieves ~10 BLEU better than the standard update, but is still more than ~5 BLEU worse than Max-Violation update. Finally we also try the “local-update” method from Liang et al. (2006) which updates towards the derivation with the best BLEU in the root group \(S_{[0:\text{end}]}\). This method is about 2 BLEU points worse than max-violation.

We further investigate the contribution of sparse features in Figure 6. On the development set, max-violation update without Word-Edges features achieves BLEU similar to \(n\)-best MERT and

\(\text{MAXFORCE}\), but lower than Hypergraph MERT. Adding simple Word-Edges features improves BLEU by ~2 points, outperforming the very strong Hypergraph MERT baseline by ~1 point. See Table 2 for details. The results of \(n\)-best MERT, Hypergraph MERT, and PRO are averages from 3 runs.

**4.5 Evaluation on FBIS**

Table 3 shows BLEU scores of various training algorithms on FBIS.

<table>
<thead>
<tr>
<th>algorithm</th>
<th># feats</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)-best MERT</td>
<td>18</td>
<td>44.9</td>
<td>47.9</td>
</tr>
<tr>
<td>Hypergraph MERT</td>
<td>18</td>
<td>46.6</td>
<td>50.7</td>
</tr>
<tr>
<td>PRO</td>
<td>18</td>
<td>45.0</td>
<td>49.5</td>
</tr>
<tr>
<td>local update perc.</td>
<td>443K</td>
<td>45.6</td>
<td>49.1</td>
</tr>
<tr>
<td>MAXFORCE</td>
<td>529K</td>
<td>47.4</td>
<td>51.5</td>
</tr>
</tbody>
</table>

Table 3: BLEU scores (with 4 references) of various training algorithms on FBIS.

We have presented a latent-variable violation-fixing framework for general structured prediction problems with inexact search over hypergraphs. Its application on HIERO brings significant improvement in BLEU, compared to algorithms that are specially designed for MT tuning such as MERT and PRO.

**Acknowledgment**

Part of this work was done during K. Z.’s internship at IBM. We thank Martin Čmejrek and Lemao Liu for discussions, David Chiang for pointing us to pycdec, Dan Gildea for Footnote 2, and the anonymous reviewers for comments. This work is supported by DARPA FA8750-13-2-0041 (DEFT), DARPA HR0011-12-C-0015 (BOLT), and a Google Faculty Research Award.
References


Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*.


790