Search-Aware Tuning for Machine Translation

Lemao Liu
Queens College
City University of New York
lemaoliu@gmail.com

Liang Huang
Queens College and Graduate Center
City University of New York
liang.huang.sh@gmail.com

Abstract

Parameter tuning is an important problem in statistical machine translation, but surprisingly, most existing methods such as MERT, MIRA and PRO are agnostic about search, while search errors could severely degrade translation quality. We propose a search-aware framework to promote promising partial translations, preventing them from being pruned. To do so we develop two metrics to evaluate partial derivations. Our technique can be applied to all of the three above-mentioned tuning methods, and extensive experiments on Chinese-to-English and English-to-Chinese translation show up to +2.6 BLEU gains over search-agnostic baselines.

1 Introduction

Parameter tuning has been a key problem for machine translation since the statistical revolution. However, most existing tuning algorithms treat the decoder as a black box (Och, 2003; Hopkins and May, 2011; Chiang, 2012), ignoring the fact that many potentially promising partial translations are pruned by the decoder due to the prohibitively large search space. For example, the popular beam-search decoding algorithm for phrase-based MT (Koehn, 2004) only explores $O(nb)$ items for a sentence of $n$ words (with a beam width of $b$), while the full search space is $O(2^n n^2)$ or worse (Knight, 1999).

As one of the very few exceptions to the “search-agnostic” majority, Yu et al. (2013) and Zhao et al. (2014) propose a variant of the perceptron algorithm that learns to keep the reference translations in the beam or chart. However, there are several obstacles that prevent their method from becoming popular: First of all, they rely on “forced decoding” to track gold derivations that lead to the reference translation, but in practice only a small portion of (mostly very short) sen-

Figure 1: (a) Some potentially promising partial translations (in red) fall out of the beam (bin 2); (b) We identify such partial translations and assign them higher model scores so that they are more likely to survive the search.

tence pairs have at least one such derivation. Secondly, they learn the model on the training set, and while this does enable a sparse feature set, it is orders of magnitude slower compared to MERT and PRO.

We instead propose a very simple framework, search-aware tuning, which does not depend on forced decoding, and thus can be trained on all sentence pairs of any dataset. The key idea is that, besides caring about the rankings of the complete translations, we also promote potentially promising partial translations so that they are more likely to survive throughout the search, see Figure 1 for illustration. We make the following contributions:

- Our idea of search-aware tuning can be applied (as a patch) to all of the three most popular tuning methods (MERT, PRO, and MIRA) by defining a modified objective function (Section 4).
- To measure the “promise” or “potential” of a partial translation, we define a new concept “potential BLEU” inspired by future cost in MT decoding (Koehn, 2004) and heuristics in A* search (Hart et al., 1968) (Section 3.2). This work is the first study of evaluating metrics for partial translations.
- Our method obtains substantial and consistent
improvements on both the large-scale NIST Chinese-to-English and English-to-Chinese translation tasks on top of MERT, MIRA, and PRO baselines. This is the first time that consistent improvements can be achieved with a new learning algorithm under dense feature settings (Section 5).

For simplicity reasons, in this paper we use phrase-based translation, but our work has the potential to be applied to other translation paradigms.

2 Review: Beam Search for PBMT Decoding

We review beam search for phrase-based decoding in our notations which will facilitate the discussion of search-aware tuning in Section 4. Following Yu et al. (2013), let \( \langle x, y \rangle \) be a Chinese-English sentence pair in the tuning set \( D \), 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 phrase-pair. Let \( |c(r)| \) be the number of Chinese words in rule \( r \), and \( e(d) \overset{\Delta}{=} e(r_1) \circ e(r_2) \circ \ldots \circ e(r_{|d|}) \) be the English prefix (i.e., partial translation) generated so far.

In beam search, each bin \( B_i(x) \) contains the best \( k \) derivations covering exactly \( i \) Chinese words, based on items in previous bins (see Figures 1 and 2):

\[
B_0(x) = \{ \epsilon \}
\]

\[
B_i(x) = \text{top}^k_{w_0}(\bigcup_{j=1 \ldots i} \{d \circ r \mid d \in B_{i-j}(x), |c(r)| = j \})
\]

where \( r \) is a rule covering \( j \) Chinese words, and \( \text{top}^k_{w_0}(\cdot) \) returns the top \( k \) derivations according to the current model \( w_0 \). As a special case, note that \( \text{top}^1_{w_0}(S) = \arg \max_{d \in S} w_0 \cdot \Phi(d) \), so \( \text{top}^1_{w_0}(B_{|x|}(x)) \) is the final 1-best result.\(^1\) See Figure 2 for an illustration.

3 Challenge: Evaluating Partial Derivations

As mentioned in Section 1, the current mainstream tuning methods such as MERT, MIRA, and PRO are

\[
\delta_{y|x}^d(d) = -\delta(y, e(d); \text{reflen} = |y| \cdot |c(d)|/|x|).
\]

\(^1\)Actually \( B_{|x|}(x) \) is an approximation to the \( k \)-best list since some derivations are merged by dynamic programming; to recover those we can use Alg. 3 of Huang and Chiang (2005).
\[
\begin{align*}
\delta(y, y') &= -\text{Bleu}^+ (y, y') \quad \text{string distance metric} \\
\delta_y(d) &= \delta(y, e(d)) \quad \text{full derivations eval} \\
\delta^x_y(d) &= \begin{cases} \delta_y^{|x|} (d) & \text{partial bleu (Sec. 3.1)} \\
\delta(y, \bar{e}_x(d)) & \text{potential bleu (Sec. 3.2)} \end{cases}
\end{align*}
\]

Table 1: Notations for evaluating full and partial derivations. Functions \(\delta_y^{|x|} (\cdot)\) and \(\bar{e}_x (\cdot)\) are defined by Equations 1 and 3, respectively.

where reflen is the effective length of reference translations, see (Papineni et al., 2002) for details.

### 3.1.1 Problem with Partial BLEU

Simple as it is, this method does not work well in practice because comparison of partial derivations might be unfair for different derivations covering different set of Chinese words, as it will naturally favor those covering “easier” portions of the input sentence (which we do observe empirically). For instance, consider the following Chinese-to-English example which involves a reordering of the Chinese PP:

(2) wǒ cóng Shànghǎi fēi dào Běijīng
I from Shanghai fly to Beijing
“I flew from Shanghai to Beijing”

Partial BLEU will prefer subtranslation “I from” to “I fly” in bin 2 (covering 2 Chinese words) because the former has 2 unigram matches while the latter only 1, even though the latter is almost identical to the reference and will eventually lead to a complete translation with substantially higher Bleu\(^+\) score (matching a 4-gram “from Shanghai to Beijing”). Similarly, it will prefer “I from Shanghai” to “I fly from” in bin 3, without knowing that the former will eventually pay the price of word-order difference. This example suggests that we need a more “global” or less greedy metric (see below).

### 3.2 Solution 2: Potential BLEU via Extension

Inspired by future cost computation in MT decoding (Koehn, 2004), we define a very simple future string by simply concatenating the best model-score translation (with no reorderings) in each uncovered span. Let \(\text{best}_w(x_{[i:j]})\) denote the best monotonic derivation for span \([i : j]\), then

\[
\text{future}(d, x) = \circ_{[i:j] \in \text{uncov}(d, x)} \text{e}(\text{best}_w(x_{[i:j]}))
\]

where \(\circ\) is the concatenation operator and \(\text{uncov}(d, x)\) returns an ordered list of uncovered spans of \(x\). See Figure 3 for an example. This future string resembles (inadmissible) heuristic function (Hart et al., 1968). Now the “extended translation” is simply a concatenation of the existing partial translation \(e(d)\) and the future string \(\text{future}(d, x)\):

\[
\bar{e}_x(d) = e(d) \circ \text{future}(d, x). \tag{3}
\]

Instead of calculating \(\text{best}_w(x_{[i:j]})\) on-the-fly for each derivation \(d\), we can precompute it for each span \([i : j]\) during future-cost computation, since the score of \(\text{best}_w(x_{[i:j]})\) is context-free (Koehn, 2004). Algorithm 1 shows the pseudo-code of computing \(\text{best}_w(x_{[i:j]})\). In practice, since future-cost precomputation already solves the best (monotonic) model-score for each span, is the only extra work for potential BLEU is to record (for each span) the subtranslation that achieves that best score. Therefore, the extra time for potential BLEU is negligible (the time complexity is \(O(n^2)\), but just as in future cost, the constant is much smaller than real decoding). The implementation should require minimal hacking on a phrase-based decoder (such as Moses).

To summarize the notation, we use \(\delta^x_y(d)\) to denote a generic evaluation function for partial derivation \(d\), which could be instantiated in two ways, partial bleu \(\delta^{|x|}_y(d)\) or potential bleu \(\delta(y, \bar{e}_x(d))\). See Table 1 for details. The next Section will only use the generic notation \(\delta^x_y(d)\).

Finally, it is important to note that although both partial and potential metrics are not BLEU-specific, the latter is much easier to adapt to other metrics such as TER since it does not change the original Bleu\(^+\) definition. By contrast, it is not clear to us at all how to generalize partial BLEU to partial TER.

### 4 Search-Aware MERT, MIRA, and PRO

Parameter tuning aims to optimize the weight vector \(w\) so that the rankings based on model score defined by \(w\) is positively correlated with those based
on some translation metric (such as BLEU (Papineni et al., 2002)). In other words, for a training sentence pair \((x, y)\), if a pair of its translations \(y_1 = e(d_1)\) and \(y_2 = e(d_2)\) satisfies \(\text{BLEU}(y_1, y_1) > \text{BLEU}(y_2, y_2)\), then we expect \(w \cdot \Phi(d_1) \geq w \cdot \Phi(d_2)\) to hold after tuning.

### 4.1 From MERT to Search-Aware MERT

Suppose \(D\) is a tuning set of \(\{(x, y)\}\) pairs. Traditional MERT learns the weight by iteratively reranking the complete translations towards those with higher BLEU in the final bin \(B_{[x]}(x)\) for each \(x\) in \(D\). Formally, it tries to minimize the document-level error of 1-best translations:

\[
\ell_{\text{MERT}}(D, w) = \sum_{(x, y) \in D} \delta_y \left( \text{top}_w^1(B_{[x]}(x)) \right),
\]  

where \(\text{top}_w^1(S)\) is the best derivation in \(S\) under model \(w\), and \(\delta(\cdot)\) is the full derivation metric as defined in Table 1; in this paper we use \(\delta_y(y') = -\text{BLEU}(y, y')\). Here we follow Och (2003) and Lopez (2008) to simplify the notations, where the \(\oplus\) operator (similar to \(\sum\)) is an over-simplification for BLEU which, as a document-level metric, is actually not factorizable across sentences.

Besides reranking the complete translations as traditional MERT, our search-aware MERT (SA-MERT) also reranks the partial translations such that potential translations may survive in the middle bins during search. Formally, its objective function is defined as follows:

\[
\ell_{\text{SA-MERT}}(D, w) = \sum_{(x, y) \in D} \left( \sum_{i=1}^{x} \delta_y \left( \text{top}_w^1(B_{[i]}(x)) \right) \right),
\]  

where \(\text{top}_w^1(\cdot)\) is defined in Eq. (4), and \(\delta_y(\cdot)\) defined in Table 1, is the generic metric for evaluating a partial derivation \(d\) which has two implementations (partial bleu or potential bleu). In other words we can obtain two implementations of search-aware MERT methods, SA-MERT\text{par} and SA-MERT\text{pot}.

Notice that the traditional MERT is a special case of SA-MERT where \(i\) is fixed to \(|x|\).

### 4.2 From MIRA to Search-Aware MIRA

MIRA is another popular tuning method for SMT. It firstly introduced in (Watanabe et al., 2007), and then was improved in (Chiang et al., 2008; Chiang, 2012; Cherry and Foster, 2012). Its main idea is to optimize a weight such that the model score difference of a pair of derivations is greater than their loss difference.

In this paper, we follow the objective function in (Chiang, 2012; Cherry and Foster, 2012), where only the violation between hope and fear derivations is concerned. Formally, we define \(d^+(x, y)\) and \(d^-(x, y)\) as the hope and fear derivations in the final bin (i.e., complete derivations):

\[
d^+(x, y) = \argmax_{d \in B_{[x]}(x)} w_0 \cdot \Phi(d) - \delta_y(d)\)  
\]

\[
d^-(x, y) = \argmax_{d \in B_{[x]}(x)} w_0 \cdot \Phi(d) + \delta_y(d)\)  
\]

where \(w_0\) is the current model. The loss function of MIRA is in Figure 4. The update will be between \(d^+(x, y)\) and \(d^-(x, y)\).

To adapt MIRA to search-aware MIRA (SA-MIRA), we need to extend the definitions of hope

\[
\text{top}_w^1(\cdot)\) is defined in Eq. (4), and \(\delta_y(\cdot)\) defined in Table 1, is the generic metric for evaluating a partial derivation \(d\) which has two implementations (partial bleu or potential bleu). In order words we can obtain two implementations of search-aware MERT methods, SA-MERTpar and SA-MERTpot.

Notice that the traditional MERT is a special case of SA-MERT where \(i\) is fixed to \(|x|\).

### 4.2 From MIRA to Search-Aware MIRA

MIRA is another popular tuning method for SMT. It firstly introduced in (Watanabe et al., 2007), and then was improved in (Chiang et al., 2008; Chiang, 2012; Cherry and Foster, 2012). Its main idea is to optimize a weight such that the model score difference of a pair of derivations is greater than their loss difference.

In this paper, we follow the objective function in (Chiang, 2012; Cherry and Foster, 2012), where only the violation between hope and fear derivations is concerned. Formally, we define \(d^+(x, y)\) and \(d^-(x, y)\) as the hope and fear derivations in the final bin (i.e., complete derivations):

\[
d^+(x, y) = \argmax_{d \in B_{[x]}(x)} w_0 \cdot \Phi(d) - \delta_y(d)\)  
\]

\[
d^-(x, y) = \argmax_{d \in B_{[x]}(x)} w_0 \cdot \Phi(d) + \delta_y(d)\)  
\]

where \(w_0\) is the current model. The loss function of MIRA is in Figure 4. The update will be between \(d^+(x, y)\) and \(d^-(x, y)\).

To adapt MIRA to search-aware MIRA (SA-MIRA), we need to extend the definitions of hope
\[ \ell_{\text{MIRA}}(D, w) = \frac{1}{2C} \| w - w_0 \|^2 + \sum_{(x, y) \in D} \left[ \Delta \delta_y(d^+(x, y), d^-(x, y)) - w \cdot \Delta \Phi(d^+(x, y), d^-(x, y)) \right]_+ \]

\[ \ell_{\text{SA-MIRA}}(D, w) = \frac{1}{2C} \| w - w_0 \|^2 + \sum_{(x, y) \in D} \left[ \sum_{i=1}^{\left| x \right|} \Delta \delta_y^i(d^+_i(x, y), d^-_i(x, y)) - w \cdot \Delta \Phi(d^+_i(x, y), d^-_i(x, y)) \right]_+ \]

\[ \ell_{\text{PRO}}(D, w) = \sum_{(x, y) \in D} \sum_{d_1, d_2 \in B_{|x|}(x), \Delta \delta_y(d_1, d_2) > 0} \log \left( 1 + \exp(-w \cdot \Delta \Phi(d_1, d_2)) \right) \]

\[ \ell_{\text{SA-PRO}}(D, w) = \sum_{(x, y) \in D} \sum_{d_1, d_2 \in B_{|x|}(x), \Delta \delta_y^x(d_1, d_2) > 0} \log \left( 1 + \exp(-w \cdot \Delta \Phi(d_1, d_2)) \right) \]

Figure 4: Loss functions of MIRA, SA-MIRA, PRO, and SA-PRO. The differences between traditional and search-aware versions are highlighted in gray. The hope and fear derivations are defined in Equations 10–13, and we define \( \Delta \delta_y(d_1, d_2) = \delta_y(d_1) - \delta_y(d_2) \), and \( \Delta \delta_y^x(d_1, d_2) = \delta_y^x(d_1) - \delta_y^x(d_2) \). In addition, \( [\theta]_+ = \max\{\theta, 0\} \).

and fear derivations from the final bin to all bins:
\[ d^+_i(x, y) = \arg\max_{d \in B_i(x)} w_0 \cdot \Phi(d) - \delta_y(d) \] (12)

\[ d^-_i(x, y) = \arg\max_{d \in B_i(x)} w_0 \cdot \Phi(d) + \delta_y(d) \] (13)

The new loss function for SA-MIRA is Eq. 7 in Figure 4. Now instead of one update per sentence, we will perform \( |x| \) updates, each based on a pair \( d^+_i(x, y) \) and \( d^-_i(x, y) \).

4.3 From PRO to Search-Aware PRO

Finally, the PRO algorithm (Hopkins and May, 2011; Green et al., 2013) aims to correlate the ranking under model score and the ranking under BLEU score, among all complete derivations in the final bin. For each preference-pair \( d_1, d_2 \in B_{|x|}(x) \) such that \( d_1 \) has a higher BLEU score than \( d_2 \) (i.e., \( \delta_y(d_1) < \delta_y(d_2) \)), we add one positive example \( \Phi(d_1) - \Phi(d_2) \) and one negative example \( \Phi(d_2) - \Phi(d_1) \). In sum, search-aware PRO has \( |x| \) times more examples than traditional PRO. The loss functions of PRO and search-aware PRO are defined in Figure 4.

5 Experiments

We evaluate our new tuning methods on two large scale NIST translation tasks: Chinese-to-English (CH-EN) and English-to-Chinese (EN-CH) tasks.

5.1 System Preparation and Data

We base our experiments on Cubit\(^2\) (Huang and Chiang, 2007), a state-of-art phrase-based system in Python. We set phrase-limit to 7, beam size to 30 and distortion limit 6. We use the 11 dense features from Moses (Koehn et al., 2007), which can lead to good performance and are widely used in almost all SMT systems. The baseline tuning methods MERT (Och, 2003), MIRA (Cherry and Foster, 2012), and PRO (Hopkins and May, 2011) are from the Moses toolkit, which are batch tuning methods based on \( k \)-best translations. The search-aware tuning methods are called SA-MERT, SA-MIRA, and SA-PRO, respectively. Their partial BLEU versions are marked with superscript \(^1\) and their potential BLEU versions are marked with superscript \(^2\), as explained in Section 3. All these search-aware tuning methods are implemented on the basis of Moses toolkit. They employ the de-
fault settings following Moses toolkit: for MERT and SA-MERT, the stop condition is defined by the weight difference threshold; for MIRA, SA-MIRA, PRO and SA-PRO, their stop condition is defined by max iteration set to 25; for all tuning methods, we use the final weight for testing.

The training data for both CH-EN and EN-CH tasks is the same, and it is collected from the NIST2008 Open Machine Translation Campaign. It consists of about 1.8M sentence pairs, including about 40M/48M words in Chinese/English sides. For CH-EN task, the tuning set is nist02 (878 sents), and test sets are nist03 (919 sents), nist04 (1788 sents), nist05 (1082 sents), nist06 (616 sents from news portion) and nist08 (691 from news portion), nist05 (1082 sents), and test sets are nist03 (919 sents), nist04 (1788 sents), nist05 (1082 sents), nist06 (616 sents from news portion) and nist08 (691 from news portion). For EN-CH task, the tuning set is ssmt07 (995 sents)\(^3\), and the test set is nist08 (1859 sents). For both tasks, all the tuning and test sets contain 4 references.

We use GIZA++ (Och and Ney, 2003) for word alignment, and SRILM (Stolcke, 2002) for 4-gram language models with the Kneser-Ney smoothing option. The LM for EN-CH is trained on its target side; and that for CH-EN is trained on the Xinhua portion of Gigaword. We use BLEU-4 (Papineni et al., 2002) with “average ref-len” to evaluate the translation performance for all experiments. In particular, the character-based BLEU-4 is employed for EN-CH task. Since all tuning methods involve randomness, all scores reported are average of three runs, as suggested by Clark et al. (2011) for fairer comparisons.

### 5.2 Main Results on CH-EN Task

Table 2 depicts the main results of our methods on CH-EN translation task. On all five test sets, our methods consistently achieve substantial improvements with two pruning options: SA-MERT\(^{pot}\) gains +1.2 BLEU points over MERT on average; and SA-MIRA\(^{pot}\) gains +1.8 BLEU points over MIRA on average as well. SA-PRO\(^{pot}\), however, does not work out of the box when we use the entire nist02 as the tuning set, which might be attributed to the “Monster” behavior (Nakov et al., 2013). To alleviate this problem, we only use the 109 short sentences with less than 10 words from nist02 as our new tuning data. To our suprise, this trick works really well (despite using much less data), and also made SA-PRO\(^{pot}\) an order of magnitude faster. This further confirms that our search-aware tuning is consistent across all tuning methods and datasets.

As discussed in Section 3, evaluation metrics of partial derivations are crucial for search-aware tuning. Besides the prinicpled “potential BLEU” version of search-aware tuning (i.e. SA-MERT\(^{pot}\), SA-MIRA\(^{pot}\), and SA-PRO\(^{pot}\)), we also run the simple “partial BLEU” version of search-aware tuning (i.e. SA-MERT\(^{par}\), SA-MIRA\(^{par}\), and SA-

<table>
<thead>
<tr>
<th>Methods</th>
<th>nist03</th>
<th>nist04</th>
<th>nist05</th>
<th>nist06</th>
<th>nist08</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>MERT</td>
<td>33.6</td>
<td>35.1</td>
<td>33.4</td>
<td>31.6</td>
<td>27.9</td>
<td>–</td>
</tr>
<tr>
<td>SA-MERT(^{par})</td>
<td>-0.2</td>
<td>+0.0</td>
<td>+0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>–</td>
</tr>
<tr>
<td>SA-MERT(^{pot})</td>
<td>+0.8</td>
<td>+1.1</td>
<td>+0.9</td>
<td>+1.7</td>
<td>+1.5</td>
<td>+1.2</td>
</tr>
<tr>
<td>MIRA</td>
<td>33.5</td>
<td>35.2</td>
<td>33.5</td>
<td>31.6</td>
<td>27.6</td>
<td>–</td>
</tr>
<tr>
<td>SA-MIRA(^{par})</td>
<td>+0.3</td>
<td>+0.3</td>
<td>+0.4</td>
<td>+0.4</td>
<td>+0.6</td>
<td>–</td>
</tr>
<tr>
<td>SA-MIRA(^{pot})</td>
<td>+1.3</td>
<td>+1.6</td>
<td>+1.4</td>
<td>+2.2</td>
<td>+2.6</td>
<td>+1.8</td>
</tr>
<tr>
<td>PRO</td>
<td>33.3</td>
<td>35.1</td>
<td>33.3</td>
<td>31.1</td>
<td>27.5</td>
<td>–</td>
</tr>
<tr>
<td>SA-PRO(^{par})</td>
<td>+0.8</td>
<td>+0.5</td>
<td>+1.0</td>
<td>+1.6</td>
<td>+1.6</td>
<td>+1.1</td>
</tr>
</tbody>
</table>

Table 2: CH-EN task: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08). \(^{par}\): partial BLEU; \(^{pot}\): potential BLEU. *: SA-PRO tunes on only 109 short sentences (with less than 10 words) from nist02.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Final bin</th>
<th>All bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>MERT</td>
<td>35.5</td>
<td>28.2</td>
</tr>
<tr>
<td>SA-MERT</td>
<td>-0.1</td>
<td>+3.1</td>
</tr>
</tbody>
</table>

Table 3: Evaluation on nist02 tuning set using two methods: BLEU is used to evaluate 1-best complete translations in the final bin; while potential BLEU is used to evaluate 1-best partial translations in all bins. The search-aware objective cares about (the potential of) all bins, not just the final bin, which can explain this result.

\(^3\)On EN-CH task, there is only one test set available for us, and thus we use ssmt07 as the tuning set, which is provided at the Third Symposium on Statistical Machine Translation (http://mitlab.hit.edu.cn/ssmt2007.html).
Figure 5: BLEU scores against beam size on nist05. Our search-aware tuning can achieve (almost) the same BLEU scores with much smaller beam size (beam of 4 vs. 16).

Table 4: The $k$-best oracle BLEU comparison between MERT and SA-MERT.

<table>
<thead>
<tr>
<th>methods</th>
<th>nist02</th>
<th>nist05</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-best</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERT</td>
<td>35.5</td>
<td>33.4</td>
</tr>
<tr>
<td>SA-MERT</td>
<td>-0.1</td>
<td>+0.9</td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERT</td>
<td>44.3</td>
<td>41.1</td>
</tr>
<tr>
<td>SA-MERT</td>
<td>+0.5</td>
<td>+1.6</td>
</tr>
</tbody>
</table>

Table 5: The diversity comparison based on the $k$-best list in the final bin on both tuning and nist05 test sets by tuning methods. The higher the metric is, the more diverse the $k$-best list is.

<table>
<thead>
<tr>
<th>Diversity</th>
<th>nist02</th>
<th>nist05</th>
</tr>
</thead>
<tbody>
<tr>
<td>MERT</td>
<td>0.216</td>
<td>0.204</td>
</tr>
<tr>
<td>SA-MERT</td>
<td>0.227</td>
<td>0.213</td>
</tr>
</tbody>
</table>

5.3 Analysis on CH-EN Task

**Different beam size.** Since our search-aware tuning considers the rankings of partial derivations in the middle bins besides complete ones in the last bin, ideally, if the weight learned by search-aware tuning can exactly evaluate partial derivations, then accurate partial derivations will rank higher according to model score. In this way, even with small beam size, these accurate partial derivations may still survive in the bins. Therefore, it is expected that search-aware tuning can achieve good performance with smaller beam size. To justify our conjecture, we run SA-MERT$^{pot}$ with different beam size (2, 4, 8, 16, 30, 100), its testing results on nist05 are depicted in Figure 5. Our methods achieve better trade-off between performance and efficiency. Figure 5 shows that search-aware tuning is consistent with all beam sizes, and as a by-product, search-aware MERT with a beam of 4 can achieve almost identical BLEU scores to MERT with beam of 16.

**Oracle BLEU.** In addition, we examine the BLEU points of oracle for MERT and SA-MERT. We use the weight tuned by MERT and SA-MERT for $k$-best decoding on nist05 test set, and calculate the oracle over these two $k$-best lists. The oracle BLEU comparison is shown in Table 4. On nist05 test set, for MERT the oracle BLEU is 41.1; while for SA-MERT its oracle BLEU is 42.7, i.e. with 1.6 BLEU improvements. Although search-aware tuning employs the objective different from the objective of evaluation on nist02 tuning set, it still gains 0.5 BLEU improvements.

**Diversity.** A $k$-best list with higher diversity can better represent the entire decoding space, and thus tuning on such a $k$-best list may lead to better testing performance (Gimpel et al., 2013). Intuitively, tuning with all bins will encourage the diversity in prefix, infix and suffix of complete translations in the final bin. To testify this, we need a diversity metric.

Indeed, Gimpel et al. (2013) define a diversity metric based on the n-gram matches between two sentences $y$ and $y'$ as follows:

$$d(y, y') = - \sum_{i=1}^{|y|-q} \sum_{j=1}^{|y'|-q} \left[ y_{i:i+q} = y'_{j:j+q} \right]$$
where \( q = n - 1 \), and \( [x] \) equals to 1 if \( x \) is true, 0 otherwise. This metric, however, has the following critical problems:

- it is not length-normalized: longer strings will look as if they are more different.
- it suffers from duplicates in \( n \)-grams. After normalization, \( d(y, y') \) will exceed -1 for any \( y \). In the extreme case, consider \( y_1 = \text{"the the the the"} \) and \( y_2 = \text{"the ... the"} \) with 10 the’s then will be considered identical after normalization by length.

So we define a balanced metric based on their metric

\[
d'(y, y') = 1 - \frac{2 \times d(y, y')}{d(y, y) + d(y', y')}
\]

which satisfies the following nice properties:

- \( d'(y, y) = 0 \) for all \( y \);
- \( 0 \leq d'(y, y') \leq 1 \) for all \( y, y' \);
- \( d'(y, y') = 1 \) if \( y \) and \( y' \) is completely disjoint.
- it does not suffer from duplicates, and can differentiate \( y_1 \) and \( y_2 \) defined above.

With this new metric, we evaluate the diversity of \( k \)-best lists for both MERT and SA-MERT. As shown in Table 5, on both tuning and test sets the \( k \)-best list generated by SA-MERT is more diverse.

### 5.4 Comparison with Max-Violation Perceptron

Our method considers the rankings of partial derivations, which is similar to MAXFORCE method (Yu et al., 2013), and thus we re-implement MAXFORCE method. Since the nist02 tuning set contains 4 references and forced decoding is performed for only one reference, we enlarge the nist02 set to a variant set following the transformation in Figure 6, and obtain a variant tuning set denoted as nist02-px, which consists of 4-times sentence-pairs. On nist02-px, the non-trivial reachable prefix-data only accounts for 12% sentences and 7% words. Both these sentence-level and the word-level percentages are much lower than those on the training data as shown in Table 3 from (Yu et al., 2013). This is because there are many OOV words on a tuning set. We run the MAXFORCE with dense feature setting on nist02-px and its testing results are shown in Table 6. We can see that on all the test sets, its testing performance is lower than that of SA-MERT\textsuperscript{pot} tuning on nist02 with about 5 BLEU points.

For more direct comparisons, we run MERT and SA-MERT\textsuperscript{pot} on a data set similar to nist02-px. We pick up the fully reachable sentences from nist02-px, remove the sentence pairs with the same source side, and get a new tuning set denoted as nist02-r. When tuning on nist02-r, we find that MERT is bet-

<table>
<thead>
<tr>
<th>Methods</th>
<th>tuning set</th>
<th>test sets (4-refs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set</td>
<td># refs</td>
</tr>
<tr>
<td>MERT</td>
<td>nist02</td>
<td>4</td>
</tr>
<tr>
<td>SA-MERT\textsuperscript{pot}</td>
<td>nist02</td>
<td>4</td>
</tr>
<tr>
<td>MAXFORCE</td>
<td>nist02-px</td>
<td>1</td>
</tr>
<tr>
<td>MAXFORCE</td>
<td>train-r-part</td>
<td>1</td>
</tr>
<tr>
<td>MERT</td>
<td>nist02-r</td>
<td>1</td>
</tr>
<tr>
<td>SA-MERT\textsuperscript{pot}</td>
<td>nist02-r</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6: Comparisons with MAXFORCE in terms of BLEU. nist02-px is the non-trivial reachable prefix-data from nist02 via forced decoding; nist02-r is a subset of nist02-px consisting of the fully reachable data; train-r is a subset of fully reachable data from training data that is comparable in size to nist02. All experiments use only dense features.
Table 7: EN-Ch task: BLEU scores on nist08 test set for MERT, SA-MERT, and MAXFORCE on different tuning sets. train-r-part is a part of fully reachable data from training data via forced decoding. All the tuning methods run with dense feature set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>tuning-set</th>
<th>nist08</th>
</tr>
</thead>
<tbody>
<tr>
<td>MERT</td>
<td>ssmt07</td>
<td>31.3</td>
</tr>
<tr>
<td>MAXFORCE</td>
<td>train-r-part</td>
<td>29.9</td>
</tr>
<tr>
<td>SA-MERT\textsuperscript{par}</td>
<td>ssmt07</td>
<td>31.3</td>
</tr>
<tr>
<td>SA-MERT\textsuperscript{pot}</td>
<td>ssmt07</td>
<td>31.7</td>
</tr>
</tbody>
</table>

Table 8: Search-aware tuning slows down MERT significantly, and MIRA and PRO moderately. The time (in minutes) is for optimization only (excluding decoding) and measured at the last iteration during the entire tuning (search aware tuning does not increase the number of iterations in our experiments). The decoding time is 20 min. on a single CPU but can be parallelized.

<table>
<thead>
<tr>
<th></th>
<th>MERT</th>
<th>MIRA</th>
<th>PRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>search-aware</td>
<td>50</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

5.6 Discussions on Tuning Efficiency

As shown in Figure 2, search-aware tuning considers all partial translations in the middle bins beside all complete translations in the last bin, and thus its total number of training examples is much greater than that of the traditional tuning. In details, suppose the tuning data consists of two sentences with length 10 and 30, respectively. Then, for traditional tuning its number of training examples is 2; but for search-aware tuning, the total number is 40. More training examples makes our search-aware tuning slower than the traditional tuning.

Table 8 shows the training time comparisons between search-aware tuning and the traditional tuning. From this Table, one can see that both SA-MIRA and SA-PRO are with the same order of magnitude as MIRA and PRO; but SA-MERT is much slower than MERT. The main reason is that, as the training examples increase dramatically, the envelope calculation for exact line search (see (Och, 2003)) in MERT is less efficient than the update based on (sub-)gradient with inexact line search in MIRA and PRO.

One possible solution to speed up SA-MERT is the parallelization but we leave it for future work.

6 Related Work

Many tuning methods have been proposed for SMT so far. These methods differ by the objective function or training mode: their objective functions are based on either evaluation-directed loss (Och, 2003; Galley and Quirk, 2011; Galley et al., 2013) or surrogate loss (Hopkins and May, 2011; Gimpel and Smith, 2012; Eidelman et al., 2013); they are either batch (Och, 2003; Hopkins and May, 2011; Cherry and Foster, 2012) or online mode (Watanabe, 2012; Simianer et al., 2012; Flanigan et al., 2013; Green et al., 2013). These methods share a common characteristic: they learn a weight by iteratively reranking a set of \textit{complete} translations represented by \textit{k}-best (Och, 2003; Watanabe et al., 2007; Chiang et al., 2008) or lattice (hypergraph) (Tromble et al., 2008; Kumar et al., 2009), and they do not care about search errors that potential \textit{partial} translations may be pruned during decoding, even if they agree with...
that their decoders are built on the beam pruning based search.

On the other hand, it is well-known that search errors can undermine the standard training for many beam search based NLP systems (Huang et al., 2012). As a result, Collins and Roark (2004) and Huang et al. (2012) propose the early-update and max-violation update to deal with the search errors. Their idea is to update on prefix or partial hypotheses when the correct solution falls out of the beam. This idea has been successfully used in many NLP tasks and improves the performance over the state-of-art NLP systems (Huang and Sagae, 2010; Huang et al., 2012; Zhang et al., 2013).

Goldberg and Nivre (2012) propose the concept of “dynamic oracle” which is the absolute best potential of a partial derivation, and is more akin to a strictly admissible heuristic. This idea inspired and is closely related to our potential BLEU, except that in our case, computing an admissible heuristic is too costly, so our potential BLEU is more like an average potential.

Gesmundo and Henderson (2014) also consider the rankings between partial translation pairs as well. However, they evaluate a partial translation through extending it to a complete translation by re-decoding, and thus they need many passes of decoding for many partial translations, while ours only need one pass of decoding for all partial translations and thus is much more efficient. In summary, our tuning framework is more general and has potential to be employed over all the state-of-art tuning methods mentioned above, even though ours is only tested on three popular methods.

7 Conclusions and Future Work

We have presented a simple yet powerful approach of “search-aware tuning” by promoting promising partial derivations, and this idea can be applied to all three popular tuning methods. To solve the key challenge of evaluating partial derivations, we develop a concept of “potential BLEU” inspired by future cost in MT decoding. Extensive experiments confirmed substantial BLEU gains with only dense features. We believe our framework can be applied to sparse feature settings and other translation paradigms, and potentially to other structured prediction problems (such as incremental parsing) as well.

Acknowledgements

We thank the three anonymous reviewers for suggestions, and Kai Zhao and Feifei Zhai for discussions. In particular, we thank reviewer #3 and Chin-Yew Lin for pushing us to think about diversity. This project was supported by DARPA FA8750-13-2-0041 (DEFT), NSF IIS-1449278, a Google Faculty Research Award, and a PSC-CUNY Award.

References


