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) sentence 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.

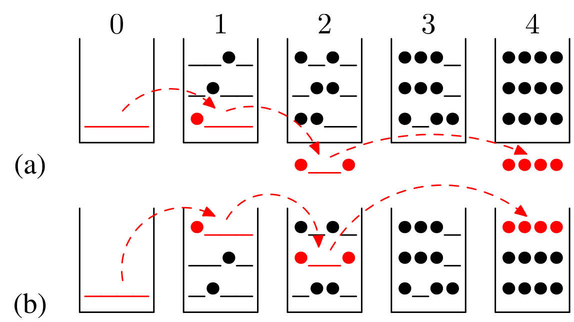

Figure 1: (a) Some potentially promising partial translations (in red) fall out of the beam (bin 2); (b) We identify such partial translations and assign them higher model scores so that they are more likely to survive the search.

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 $e(d) \triangleq e(r_1) \circ e(r_2) \ldots \circ e(r_{|d|})$ 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):

$$B_0(x) = \{\epsilon\}$$
$$B_i(x) = \mathbf{top}^k_{\mathbf{w}_0}\Big(\bigcup_{j=1..i}\{d \circ r \mid d \in B_{i-j}(x), |c(r)| = j\}\Big)$$

where r is a rule covering j Chinese words, and $\mathbf{top}^k_{\mathbf{w}_0}(\cdot)$ returns the top k derivations according to the current model $\mathbf{w}_0$. As a special case, note that $\mathbf{top}^1_{\mathbf{w}_0}(S) = \mathbf{argmax}_{d\in S}\, \mathbf{w}_0 \cdot \mathbf{\Phi}(d)$, so $\mathbf{top}^1_{\mathbf{w}_0}(B_{|x|}(x))$ 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 all **search-agnostic**: they only care about the *complete* translations from the last bin, $B_{|x|}(x)$, ignoring all *partial* ones, i.e., $B_i(x)$ for all $i < |x|$. As a result, many potentially promising partial derivations never reach the final bin (See Figure 1).

[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).

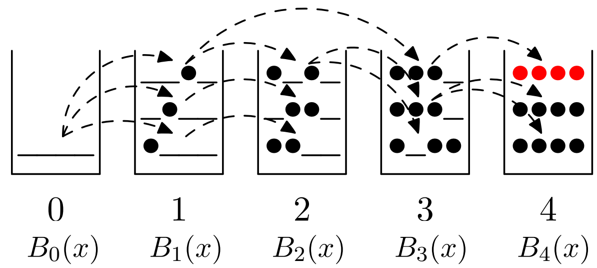

Figure 2: Beam search for phrase-based decoding. The item in red is $\mathbf{top}^1_{\mathbf{w}_0}(B_4(x))$, i.e., the 1-best result. Traditional tuning only uses the final bin $B_4(x)$ while search-aware tuning considers all bins $B_i(x)$ ($i = 1..4$).

To address this problem, our new "search-aware tuning" aims to promote not only the accurate translations in the final bin, but more importantly those potentially promising *partial derivations* in non-final bins. The key challenge, however, is how to evaluate the "promise" or "potential" of a partial derivation. In this Section, we develop two such measures, a simple "partial BLEU" (Section 3.1) and a more principled "potential BLEU" (Section 3.2). In Section 4, we will then adapt traditional tuning methods to their search-aware versions using these partial evaluation metrics.

3.1 Solution 1: Simple and Naive Partial BLEU

Inspired by a trick in (Li and Khudanpur, 2009) and (Chiang, 2012) for oracle or hope extraction, we use a very simple metric to evaluate partial translations for tuning. For a given derivation d, the basic idea is to evaluate the (short) partial translation $e(d)$ against the (full) reference y, but using a "prorated" reference length proportional to $c(d)$ which is the number of Chinese words covered so far in d:

$$|y| \cdot |c(d)|/|x|$$

For example, if d has covered 2 words on a 8-word Chinese sentence with a 12-word English reference, then the "effective reference length" is $12 \times 2/8 = 3$. We call this method "partial BLEU" since it does not complete the translation, and denote it by

$$\bar{\delta}^{|x|}_y(d) = -\delta\big(y, e(d); \text{reflen} = |y| \cdot |c(d)|/|x|\big). \quad (1)$$

$\delta(y, y') = -\mathrm{Bleu}^{+1}(y, y')$		string distance metric
$\delta_y(d)$	$= \delta(y, e(d))$	full derivations eval
$\delta_y^x(d)$	$= \begin{cases} \bar{\delta}_y^{\lvert x \rvert}(d) \\ \delta(y, \bar{e}_x(d)) \end{cases}$	partial bleu (Sec. 3.1) potential bleu (Sec. 3.2)

Table 1: Notations for evaluating full and partial derivations. Functions $\bar{\delta}_y^{|x|}(\cdot)$ 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 mathces 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^{+1} 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 $\mathrm{best}_{\mathbf{w}}(x_{[i:j]})$ denote the best monotonic derivation for span $[i : j]$, then

$$\mathit{future}(d, x) = \circ_{[i:j] \in \mathit{uncov}(d,x)} \; e(\mathrm{best}_{\mathbf{w}}(x_{[i:j]}))$$

where $\circ$ is the concatenation operator and $\mathit{uncov}(d, x)$ returns an ordered list of uncovered

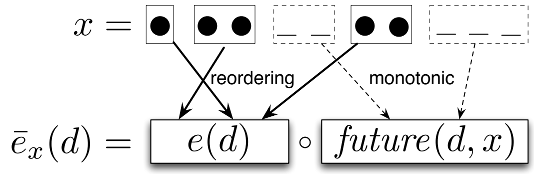

Figure 3: Example of the extension function $\bar{e}_x(\cdot)$ (and future string) on an incomplete derivation d.

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 $\mathit{future}(d, x)$:

$$\bar{e}_x(d) = e(d) \circ \mathit{future}(d, x). \quad (3)$$

Instead of calculating $\mathrm{best}_{\mathbf{w}}(x_{[i:j]})$ on-the-fly for each derivation d, we can precompute it for each span $[i : j]$ during future-cost computation, since the score of $\mathrm{best}_{\mathbf{w}}(x_{[i:j]})$ is context-free (Koehn, 2004). Algorithm 1 shows the pseudo-code of computing $\mathrm{best}_{\mathbf{w}}(x_{[i:j]})$. 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 $\delta_y^x(d)$ to denote a generic evaluation function for partial derivation d, which could be instantiated in two ways, partial bleu ($\bar{\delta}_y^{|x|}(d)$) or potential bleu ($\delta(y, \bar{e}_x(d))$). See Table 1 for details. The next Section will only use the generic notation $\delta_y^x(d)$.

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^{+1} 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 $\mathbf{w}$ so that the rankings based on model score defined by $\mathbf{w}$ is positively correlated with those based

Algorithm 1 Computation of best Translations for Potential BLEU.

Input: Source sentence x, a rule set $\Re$ for x, and $\mathbf{w}$.
Output: Best translations $e(\text{best}_{\mathbf{w}}(x[i:j]))$ for all spans $[i:j]$.

- **for** l **in** $(0..|x|)$ **do**
  - **for** i **in** $(0..|x|-l)$ **do**
    - $j = i + l + 1$
    - best_score $= -\infty$
    - **if** $\Re[i:j] \neq \emptyset$ **then** $\quad\triangleright$ $\Re[i:j]$ is a subset of rules $\Re$ for span $[i:j]$.
      - $\text{best}_{\mathbf{w}}(x[i:j]) = \mathbf{argmax}_{r \in \Re[i:j]}\, \mathbf{w} \cdot \mathbf{\Phi}(\{r\})$ $\quad\triangleright$ $\{r\}$ is a derivation consisting of one rule r.
      - best_score $= \mathbf{w} \cdot \mathbf{\Phi}(\text{best}_{\mathbf{w}}(x[i:j]))$
    - **for** k **in** $(i+1 \,..\, i+p)$ **do** $\quad\triangleright$ p is the phrase length limit
      - **if** best_score $< \mathbf{w} \cdot \mathbf{\Phi}\Big(\text{best}_{\mathbf{w}}(x[i:k]) \circ \text{best}_{\mathbf{w}}(x[k:j])\Big)$ **then**
        - $\text{best}_{\mathbf{w}}(x[i:j]) = \text{best}_{\mathbf{w}}(x[i:k]) \circ \text{best}_{\mathbf{w}}(x[k:j])$
        - best_score $= \mathbf{w} \cdot \mathbf{\Phi}(\text{best}_{\mathbf{w}}(x[i:j]))$

on some translation metric (such as BLEU (Papineni et al., 2002)). In other words, for a training sentence pair $\langle x, y\rangle$, if a pair of its translations $y_1 = e(d_1)$ and $y_2 = e(d_2)$ satisfies $\text{BLEU}(y, y_1) > \text{BLEU}(y, y_2)$, then we expect $\mathbf{w} \cdot \mathbf{\Phi}(d_1) > \mathbf{w} \cdot \mathbf{\Phi}(d_2)$ to hold after tuning.

4.1 From MERT to Search-Aware MERT

Suppose D is a tuning set of $\langle x, y\rangle$ 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:

$$\ell_{\text{MERT}}(D, \mathbf{w}) = \bigoplus_{\langle x,y\rangle \in D} \delta_y\Big(\mathbf{top}^1_{\mathbf{w}}(B_{|x|}(x))\Big), \tag{4}$$

where $\mathbf{top}^1_{\mathbf{w}}(S)$ is the best derivation in S under model $\mathbf{w}$, and $\delta_{\cdot}(\cdot)$ 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:

$$\ell_{\text{SA-MERT}}(D, \mathbf{w}) = \bigoplus_{\langle x,y\rangle \in D} \boxed{\bigoplus_{i=1..|x|}}\ \boxed{\delta^x_y}\Big(\mathbf{top}^1_{\mathbf{w}}(\boxed{B_i}(x))\Big) \tag{5}$$

where $\mathbf{top}^1_{\mathbf{w}}(\cdot)$ is defined in Eq. (4), and $\delta^x_y(d)$, 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-MERT^{par} and SA-MERT^{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):

$$d^+(x, y) = \mathop{\mathbf{argmax}}_{d \in B_{|x|}(x)} \mathbf{w}_0 \cdot \mathbf{\Phi}(d) - \delta_y(d) \tag{10}$$

$$d^-(x, y) = \mathop{\mathbf{argmax}}_{d \in B_{|x|}(x)} \mathbf{w}_0 \cdot \mathbf{\Phi}(d) + \delta_y(d) \tag{11}$$

where $\mathbf{w}_0$ 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

$$\ell_{\text{MIRA}}(D,\mathbf{w}) = \frac{1}{2C}\|\mathbf{w}-\mathbf{w}_0\|^2 + \sum_{\langle x,y\rangle\in D} \left[\Delta\delta_y\big(d^+(x,y),d^-(x,y)\big) - \mathbf{w}\cdot\Delta\mathbf{\Phi}\big(d^+(x,y),d^-(x,y)\big)\right]_+ \tag{6}$$

$$\ell_{\text{SA-MIRA}}(D,\mathbf{w}) = \frac{1}{2C}\|\mathbf{w}-\mathbf{w}_0\|^2 + \sum_{\langle x,y\rangle\in D} \boxed{\sum_{i=1}^{|x|}} \left[\Delta\boxed{\delta_y^x}\big(\boxed{d_i^+}(x,y),\ \boxed{d_i^-}(x,y)\big) - \mathbf{w}\cdot\Delta\mathbf{\Phi}\big(\boxed{d_i^+}(x,y),\ \boxed{d_i^-}(x,y)\big)\right]_+ \tag{7}$$

$$\ell_{\text{PRO}}(D,\mathbf{w}) = \sum_{\langle x,y\rangle\in D} \quad \sum_{d_1,d_2\in B_{|x|}(x),\ \Delta\delta_y(d_1,d_2)>0} \log\Big(1+\exp(-\mathbf{w}\cdot\Delta\mathbf{\Phi}(d_1,d_2))\Big) \tag{8}$$

$$\ell_{\text{SA-PRO}}(D,\mathbf{w}) = \sum_{\langle x,y\rangle\in D} \boxed{\sum_{i=1}^{|x|}} \quad \sum_{d_1,d_2\in \boxed{B_i}(x),\ \boxed{\Delta\delta_y^x}(d_1,d_2)>0} \log\Big(1+\exp(-\mathbf{w}\cdot\Delta\mathbf{\Phi}(d_1,d_2))\Big) \tag{9}$$

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:

$$\boxed{d_i^+}(x,y) = \underset{d\in \boxed{B_i}(x)}{\mathbf{argmax}}\ \mathbf{w}_0\cdot\mathbf{\Phi}(d) - \delta_y(d) \tag{12}$$

$$\boxed{d_i^-}(x,y) = \underset{d\in \boxed{B_i}(x)}{\mathbf{argmax}}\ \mathbf{w}_0\cdot\mathbf{\Phi}(d) + \delta_y(d) \tag{13}$$

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 $\mathbf{\Phi}(d_1) - \mathbf{\Phi}(d_2)$ and one negative example $\mathbf{\Phi}(d_2) - \mathbf{\Phi}(d_1)$.

Now to adapt it to search-aware PRO (SA-**PRO**), we will have many more examples to consider: besides the final bin, we will include all preference-pairs in the non-final bins as well. For each bin $B_i(x)$, for each preference-pairs $d_1, d_2 \in B_i(x)$ such that d_1 has a higher partial or potential BLEU score than d_2 (i.e., $\delta_y^x(d_1) < \delta_y^x(d_2)$), we add one positive example $\mathbf{\Phi}(d_1) - \mathbf{\Phi}(d_2)$ and one negative example $\mathbf{\Phi}(d_2) - \mathbf{\Phi}(d_1)$. 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-

[2] http://www.cis.upenn.edu/~lhuang3/cubit/

Methods	nist03	nist04	nist05	nist06	nist08	avg
MERT	33.6	35.1	33.4	31.6	27.9	–
SA-MERT^{par}	-0.2	+0.0	+0.1	-0.1	-0.1	–
SA-MERT^{pot}	**+0.8**	**+1.1**	**+0.9**	**+1.7**	**+1.5**	+1.2
MIRA	33.5	35.2	33.5	31.6	27.6	–
SA-MIRA^{par}	+0.3	+0.3	+0.4	+0.4	+0.6	–
SA-MIRA^{pot}	**+1.3**	**+1.6**	**+1.4**	**+2.2**	**+2.6**	+1.8
PRO	33.3	35.1	33.3	31.1	27.5	–
*SA-PRO^{par}	-2.0	-2.7	-2.2	-1.0	-1.7	–
*SA-PRO^{pot}	**+0.8**	**+0.5**	**+1.0**	**+1.6**	**+1.6**	+1.1

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.

	Final bin	All bins
MERT	35.5	28.2
SA-MERT	-0.1	+3.1

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.

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). 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.

[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).

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 supurise, 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 principled "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-

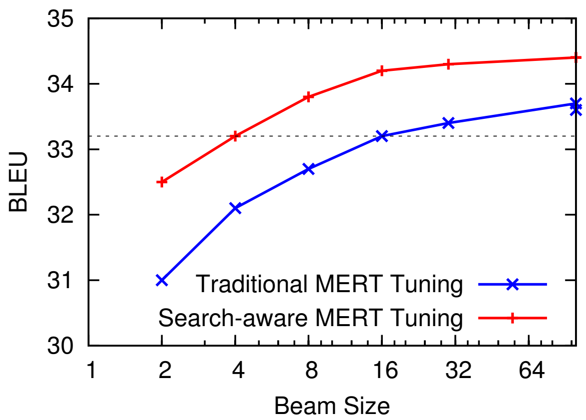

Figure 5: BLEU scores against beam size on nist05. Our search-aware tuning can achieve (almost) the same BLEU scores with much smaller beam size (beam of 4 vs. 16).

	methods	nist02	nist05
1-best	MERT	35.5	33.4
	SA-MERT	-0.1	+0.9
Oracle	MERT	44.3	41.1
	SA-MERT	+0.5	+1.6

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

PRO^{par}). In Table 2, we can see that they may achieve slight improvements over tradition tuning on some datasets, but SA-MERT^{pot}, SA-MIRA^{pot}, and SA-PRO^{pot} using potential BLEU consistently outperform them on all the datasets.

Even though our search-aware tuning gains substantially on all test sets, it does not gain significantly on nist02 tuning set. The main reason is that, search-aware tuning optimizes an objective (i.e. BLEU for all bins) which is different from the objective for evaluation (i.e. BLEU for the final bin), and thus it is not quite fair to evaluate the complete translations for search-aware tuning as the same done for traditional tuning on the tuning set. Actally, if we evaluate the potential BLEU for all partial translations, we find that search-aware tuning gains about 3.0 BLEU on nist02 tuning set, as shown in Table 3.

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 mehtods achieve better trade-off between performance and efficiey. 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.

Diversity	nist02	nist05
MERT	0.216	0.204
SA-MERT	0.227	0.213

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.

Oracle BLEU. In addition, we examine the BLEU ponits 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 tesing 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:

$$d(y, y') = -\sum_{i=1}^{|y|-q} \sum_{j=1}^{|y'|-q} [\![y_{i:i+q} = y'_{j:j+q}]\!]$$

Methods	tuning set				test sets (4-refs)				
	set	# refs	# sents	# words	nist03	nist04	nist05	nist06	nist08
MERT	nist02	4	878	23181	33.6	35.1	33.4	31.6	27.9
SA-MERT^{pot}	nist02	4	878	23181	**34.4**	**36.2**	**34.3**	**33.3**	**29.4**
MAXFORCE	nist02-px	1	434	6227	29.0	30.3	28.7	26.8	24.1
MAXFORCE	train-r-part	1	1225	22684	31.7	33.5	31.5	30.3	26.7
MERT	nist02-r	1	92	1173	31.6	32.7	31.3	29.3	25.9
SA-MERT^{pot}	nist02-r	1	92	1173	33.5	35.0	33.4	31.5	28.0

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.

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 = "the the the the" and y_2 = "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 simlar to MAXFORCE

Bùshí yǔ Shālóng jǔxíng huìtán	Bush and Sharon held a meeting Bush held talks with Sharon
qiāngshǒu bèi jǐngfāng jībì	police killed the gunman the gunman was shot dead

⇓

Bùshí yǔ Shālóng jǔxíng huìtán	Bush and Sharon held a meeting
Bùshí yǔ Shālóng jǔxíng huìtán	Bush held talks with Sharon
qiāngshǒu bèi jǐngfāng jībì	police killed the gunman
qiāngshǒu bèi jǐngfāng jībì	the gunman was shot dead

Figure 6: Transformation of a tuning set in forced decoding for MAXFORCE: the original tuning set (on the top) contains 2 source sentences with 2 references for each; while the transformed set (on the bottom) contains 4 source sentences with one reference for each.

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^{pot} tuning on nist02 with about 5 BLEU points.

For more direct comparisons, we run MERT and SA-MERT^{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-

Methods	tuning-set	nist08
MERT	ssmt07	31.3
MAXFORCE	train-r-part	29.9
SA-MERT^{par}	ssmt07	31.3
SA-MERT^{pot}	ssmt07	**31.7**

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.

Optimization time	MERT	MIRA	PRO
basline	3	2	2
search-aware	50	7	6

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.

ter than MAXFORCE,[4] and SA-MERT^{pot} are much better than MERT on all the test sets. In addition, we select about $1.2k$ fully reachable sentence pairs from training data, and run the forced decoding on this new tuning data (denoted as **train-r-part**), which is with similar size to nist02. [5] With more tuning data, the performance of max-violation is improved largely, but it is still underperformed by SA-MERT^{pot}.

5.5 Results on EN-CH Translation Task

We also run our search-aware tuning method on EN-CH task. We use SA-MERT as the representative of search-aware tuning methods, and compare its two versions with other tuning methods MERT, MAXFORCE. For MAXFORCE, we first run forced decoding on the training data and then select about $1.2k$ fully reachable sentence pairs as its tuning set (denoted as **train-r-part**). For MERT, SA-MERT^{pot}, and SA-MERT^{par}, their tuning set is ssmt07. Table 7 shows that SA-MERT^{pot} is much better than MAXFORCE, i.e. it achieves 0.4 BLEU improvements over MERT. Finally, comparison between SA-MERT^{pot} and SA-MERT^{par} shows that the potential BLEU is better for evaluation of partial derivations.

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 magtitude 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 *complete* translations represented by 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 *partial* translations may be pruned during decoding, even if they agree with

[4] Under the dense feature setting, MAXFORCE is worse than standard MERT. This result is consistent with that in Figure 12 of (Yu et al., 2013).

[5] We run MAXFORCE on train-r-part, i.e. a part of reachable data instead of the entire reachable data, as we found that more tuning data does not necessarily lead to better testing performance under dense feature setting in our internal experiments.

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

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of NAACL-HLT*, pages 427–436, Montréal, Canada, 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.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL 2011*.

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

Vladimir Eidelman, Yuval Marton, and Philip Resnik. 2013. Online relative margin maximization for statistical machine translation. In *Proceedings of ACL*, pages 1116–1126, Sofia, Bulgaria, August.

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-HLT*, pages 248–258, Atlanta, Georgia, June.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proceedings of EMNLP*, pages 38–49, Edinburgh, Scotland, UK., July.

Michel Galley, Chris Quirk, Colin Cherry, and Kristina Toutanova. 2013. Regularized minimum error rate training. In *Proceedings of EMNLP*, pages 1948–1959, Seattle, Washington, USA, October.

Andrea Gesmundo and James Henderson. 2014. Undirected machine translation with discriminative reinforcement learning. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, April.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL-HLT*, pages 221–231, Montréal, Canada, June.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, October.

Yoav Goldberg and Joakim Nivre. 2012. Training deterministic parsers with non-deterministic oracles. In *Proceedings of COLING 2012*.

Spence Green, Sida Wang, Daniel Cer, and Christopher Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *Proc. of ACL 2013*.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

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

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*.

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

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

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.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of ACL-IJCNLP*, Suntec, Singapore, August.

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient extraction of oracle-best translations from hypergraphs. In *Proceedings of HLT-NAACL Short Papers*.

Adam Lopez. 2008. Statistical machine translation. *ACM Comput. Surv.*, 40(3).

Preslav Nakov, Francisco Guzmn, and Stephan Voge. 2013. A tale about pro and monsters. In *Proceedings of ACL Short Papers*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA, July.

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*, pages 11–21, Jeju Island, Korea, July.

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

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629, Honolulu, Hawaii, October.

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

Taro Watanabe. 2012. Optimized online rank learning for machine translation. In *Proceedings of NAACL-HLT*, pages 253–262, Montréal, Canada, June.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable mt training. In *Proceedings of EMNLP 2013*.

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

Kai Zhao, Liang Huang, Haitao Mi, and Abe Ittycheriah. 2014. Hierarchical mt training using max-violation perceptron. In *Proceedings of ACL*, Baltimore, Maryland, June.