Search-Aware Tuning for Hierarchical Phrase-based Decoding

Feifei Zhai

Queens College
City University of New York
ffzhai2012@gmail.com

Liang Huang **Kai Zhao**

Queens College & Graduate Center
City University of New York
{lianghuang.sh,kzhao.hf}@gmail.com

Abstract

Parameter tuning is a key problem for statistical machine translation (SMT). Most popular parameter tuning algorithms for SMT are agnostic of decoding, resulting in parameters vulnerable to search errors in decoding. The recent research of "search-aware tuning" (Liu and Huang, 2014) addresses this problem by considering the partial derivations in every decoding step so that the promising ones are more likely to survive the inexact decoding beam. We extend this approach from phrase-based translation to syntax-based translation by generalizing the evaluation metrics for partial translations to handle tree-structured derivations in a way inspired by inside-outside algorithm. Our approach is simple to use and can be applied to most of the conventional parameter tuning methods as a plugin. Extensive experiments on Chinese-to-English translation show significant BLEU improvements on MERT, MIRA and PRO.

1 Introduction

Efforts in parameter tuning algorithms for SMT, such as MERT (Och, 2003; Galley et al., 2013), MIRA (Watanabe et al., 2007; Chiang et al., 2009; Cherry and Foster, 2012; Chiang, 2012), and PRO (Hopkins and May, 2011) have improved the translation quality considerably in the past decade. These tuning algorithms share the same characteristic that they treat the decoder as a black box. This decoding insensitiveness has two effects: 1) the parameter tuning algorithm can be more general to choose the most effective decoding paradigm for different language pairs; 2) however, it also means that the learned parameters may not fit the decoding algorithm well, so that promising partial translations might be pruned out due to the beam search decoding.

Recent researches reveal that the parameter tuning algorithms tailored for specific decoding algorithms can be beneficial in general structured prediction problems (Huang et al., 2012), and in machine translation (Yu et al., 2013; Zhao et al., 2014; Liu and Huang, 2014). Particularly, Liu and Huang (2014) show that by requiring the conventional parameter tuning algorithms to consider the final decoding results (full translations) as well as the intermediate decoding states (partial translations) at the same time, the inexact decoding can be significantly improved so that correct intermediate partial translations are more likely to survive the beam. However, the underlying phrase-based decoding model suffers from limited distortion, and thus, may not be flexible enough for translating language pairs that are syntactically different, which require long distance reordering.

In order to better handle long distance reordering which beyonds the capability of phrase-based MT, we extend the search-aware tuning framework from phrase-based MT to syntax-based MT, in particular the hierarchical phrase-based translation model (HIERO) (Chiang, 2007).

One key advantage of search-aware tuning for previous phrase-based MT is the minimal change to existing parameter tuning algorithms, which is achieved by defining BLEU-like metrics for the intermediate decoding states with sequence-structured derivations. To keep our approach simple, we generalize these BLEU-like metrics to handle intermediate decoding states with tree-structured derivations in HIERO, which are calculated by dynamic programming algorithms inspired by the inside-outside algorithm.

We make the following contributions:

1. We extend the framework of search-aware tuning methods from phrase-based translation to syntax-based translation. This extension is simple to be applied to most conventional parameter tuning methods, requiring minimal extra changes to existing algorithms.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1294–1303,
Lisbon, Portugal, 17-21 September 2015. ©2015 Association for Computational Linguistics.

2. We propose two BLEU metrics and their variants to evaluate partial derivations. In order to efficiently compute the new BLEU metrics, we investigate dynamic programming based algorithms to recover the good partial derivations for search-aware tuning.

3. Our method obtains significant improvements on large-scale Chinese-to-English translation on top of MERT, MIRA and PRO baselines.

2 Hierarchical Phrase-based Decoding

We first briefly describe the hierarchical phrase-based translation (Chiang, 2007), HIERO, by a Chinese-English translation example (Figure 1).

A HIERO decoder translates a source sentence by using a synchronous grammar (Figure 1 (a)) to parse it into a bilingual derivation tree, as shown in Figure 1 (b) and Figure 1 (c). An example rule of the synchronous grammar is

$$r_8 : \mathrm{X} \rightarrow \langle \mathrm{X}_1 \text{ qian } \mathrm{X}_2, \mathrm{X}_2 \text{ before } \mathrm{X}_1 \rangle,$$

where the left-hand-side is a non-terminal symbol, and the right-hand-side is a pair of source and target strings, each of which consists a sequence of lexical terminals and non-terminals. Specifically, the same subscript on both sides denotes a one-to-one correspondence between their non-terminals. We use $|r|$ to denote the arity of rule r, i.e., the number of non-terminals. Usually, the rule used in HIERO system has a maximum arity 2.

Let $\langle x, y \rangle$ be a Chinese-English sentence pair in tuning set, the HIERO decoder will generate a derivation tree for it (Figure 1 (b)), which can be defined recursively in a *functional* way:

$$d = \begin{cases} r & |r| = 0 \\ r(d_1) & |r| = 1 \\ r(d_1, d_2) & |r| = 2 \,, \end{cases} \quad (1)$$

where d is a (partial) derivation, i.e., the (sub-) derivation tree. When r is a fully lexicalized rule ($|r| = 0$), the decoder generates a tree node directly (e.g., $X_{[3,5]}$ in Figure 1 (b)). If $|r| > 0$, a new (partial) derivation will be created by applying r to its children nodes. In our notation, we denote this process as applying a function of rule r to its arguments. For example, node $X_{[1,5]}$ in Figure 1(b) is created by applying rule r_8 in Figure 1(a), where the arguments are $d_1 = X_{[1,2]}$ and $d_2 = X_{[3,5]}$ respectively.

Practically, we organize the partial derivations based on spans in HIERO decoder, and use a beam $B_{[i,j]}$ to keep the k-best partial derivations for each span $[i, j]$:

$$B_{[i,j]} = \mathbf{top}^k_{\mathbf{w}} \left(D_{[i,j]} \right),$$

where $D_{[i,j]}$ is the set of all possible partial derivations for span $[i, j]$, and $\mathbf{top}^k_{\mathbf{w}}(\cdot)$ returns the top k ones according to the current model $\mathbf{w}$. Figure 2 shows an example of the HIERO beam-search decoding.

3 Search-Aware Tuning for HIERO

As we discussed in Section 1, current tuning methods for HIERO system (MERT, MIRA, or PRO) are mostly *search-agnostic*. They only consider the *complete* translations in final beam $B_{[1,|x|]}$, but ignore the partial ones in the intermediate beams. However, because MT decoding is inexact (beam-search), many potentially promising derivations might be pruned before reaching the final beam (e.g., the partial derivation $X_{[1,5]}$ of Figure 1(b) is pruned in beam $B_{[1,5]}$ in Figure 2). Consequently, once we lose these good partial derivations in decoding, it is hard to promote them by these search-agnostic tuning methods.

In order to address this problem, search-aware tuning (Liu and Huang, 2014) aims to promote not only the accurate complete translations in the final beam, but more importantly those promising partial derivations in non-final beams.

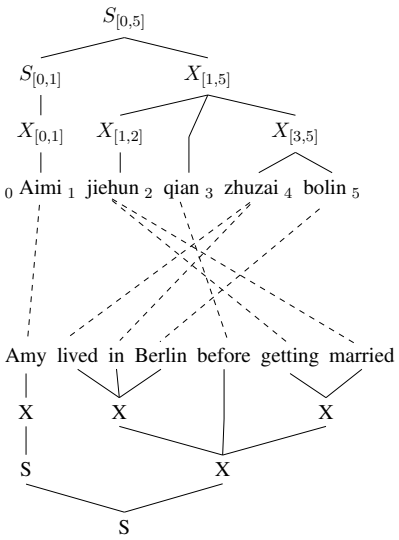
In this section, we apply search-aware tuning to HIERO system. Similar to the phrase-based one, the key challenge here is also the evaluation of partial derivations. Following (Liu and Huang, 2014), we design two different evaluation metrics, *partial* BLEU and *potential* BLEU respectively for HIERO decoding.

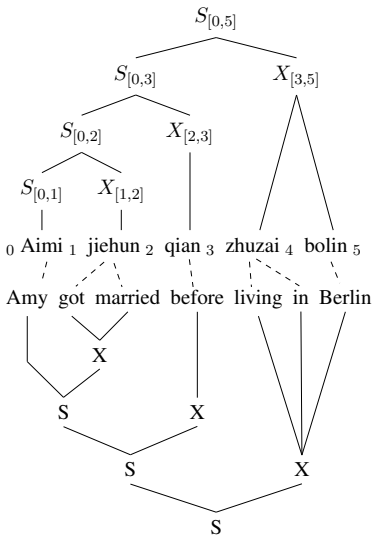
3.1 Partial BLEU

Given a partial derivation d of span $[i, j]$, *partial* BLEU evaluates it by comparing its partial translation $e(d)$ directly with the (full) reference. We explore two different partial BLEU measures here: *full partial* BLEU and *span partial* BLEU.

Full partial BLEU is similar to the one used in (Liu and Huang, 2014), which compares $e(d)$ with the full reference y and computes BLEU score using an adjusted effective reference length proportional to the length of the source span $[i, j]$ (i.e.,

id	rule
r_0	S → ⟨X$_1$, X$_1$⟩
r_1	S → ⟨S$_1$X$_1$, S$_1$X$_1$⟩
r_2	X → ⟨Aimi, Amy⟩
r_2	X → ⟨Aimi, Amy has⟩
r_3	X → ⟨zhuzai bolin, lived in Berlin⟩
r_3	X → ⟨zhuzai bolin, living in Berlin⟩
r_4	X → ⟨qian, before⟩
r_5	X → ⟨qian, former⟩
r_6	X → ⟨jiehun, got married⟩
r_7	X → ⟨jiehun, getting married⟩
r_8	X → ⟨X$_1$ qian X$_2$,X$_2$ before X$_1$⟩
r_9	X → ⟨X$_1$ qian X$_2$,X$_1$ before X$_2$⟩





(a) HIERO rules (b) good derivation (c) bad derivation

Figure 1: An example of HIERO translation.

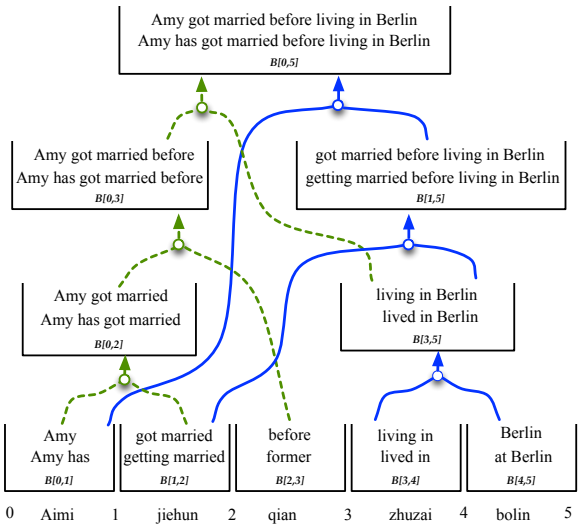


Figure 2: An example of HIERO beam search decoding (beam size = 2). We can see the good derivation (Figure 1(b)) is pruned very early, while the bad derivation (Figure 1(c)) ranks first at the final beam.

the number of translated source words). See (Liu and Huang, 2014) for more details.

Span partial BLEU differs from full partial BLEU by creating a *span reference* for span $[i, j]$ to evaluate its partial derivations, rather than using the full reference. We use word alignment to determine the span references of different spans. Specifically, we first add the tuning set into training data, and then run GIZA++ to get the word alignment. For a pair of source and target span which are consistent with word alignment[1], the string of target span is used as the span reference for the source span.[2] For example, in Figure 1(b), the span reference of source span $[1, 5]$ is "lived in Berlin before getting married".

Partial BLEU is quite an intuitive choice for evaluating partial derivations. However, as Liu and Huang (2014) discussed, partial BLEU only evaluates the partial derivation itself without considering any of its context information, leading to a performance degradation. Thus, in order to take the context information into account, we investigate a more reasonable metric, potential BLEU, in the next section.

3.2 Potential BLEU

Potential BLEU evaluates the partial derivation d by assessing its *potential*, which is a complete translation generated from d, rather than $e(d)$ as in partial BLEU. In phrase-based decoding, Liu

[1]Two spans are consistent with word alignment means that words in one span only align to words in the other span via word alignment, and vice versa.

[2]In practice, there might be several different consistent target spans for a source span due to the unaligned words. We use the longest target span as the span reference which shows the best performance in our experiments.

and Huang (2014) introduce a future string $f(d)$ to denote the translation of the untranslated part, and get the potential of d by concatenating $e(d)$ and $f(d)$.

Different from their work, we define an *outside string* for d in HIERO system. Suppose a complete translation generated from d is $\bar{e}(d)$, it can be decomposed as follows:

$$\bar{e}(d) = \eta(d) \circ e(d) \circ \xi(d) \quad (2)$$

where $\circ$ is an operator that concatenates strings monotonically, $\eta(d)$ and $\xi(d)$ are the left and right parts of $\bar{e}(d)$ apart from $e(d)$, which we call *left* and *right future strings* of d. The pair of $\eta(d)$ and $\xi(d)$ is defined as the outside string of d:

$$O(d) = \langle \eta(d),\ \xi(d) \rangle$$

An example of the outside string is shown in Figure 3. Assume we are evaluating the partial derivation which translates Chinese word "qian" to English word "before", and get a complete translation "Amy lived in Berlin before getting married" from it, then its outside string would be:

⟨Amy lived in Berlin, getting married⟩

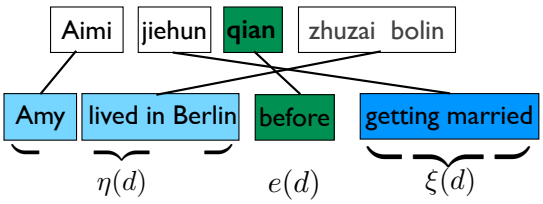


Figure 3: Example of outside string when we use"Amy lived in Berlin before getting married" as the potential for the partial derivation which translates word "qian" to "before".

Theoretically, different partial derivations of the same span could have different outside strings. To simplify the problem, we use the same outside string for all partial derivations of the same span. The outside string for span $[i, j]$ is defined as:

$$O([i, j]) = \langle \eta([i, j]),\ \xi([i, j]) \rangle$$

For a specific partial derivation d of span $[i, j]$, by combining $O([i, j])$ and $e(d)$, we can compute its potential BLEU against the reference. Apparently, different outside string will lead to different potential BLEU score. In the rest of this section, we will explore three different methods to generate outside strings.

3.2.1 Concatenation

In order to generate outside string, one simple and straightforward way is concatenation. For a specific span $[i, j]$, we first get the best translation of its adjacent spans $[0, i]$ and $[j, |x|]$ ($e([0, i])$ and $e([j, |x|])$ respectively).[3] Then for a partial derivation d of span $[i, j]$, we generate the outside string of d by *concatenation*

$$\eta([i, j]) = e([0, i])$$
$$\xi([i, j]) = e([j, |x|])$$
$$\bar{e}_x(d) = e([0, i]) \circ e(d) \circ e([j, |x|])$$

Also take the example of Figure 3, if we perform concatenation on it, the outside string of the corresponding partial derivation is:

⟨Amy getting married, lived in Berlin⟩.

Obviously, this outside string is not good, since it does not consider the reordering between spans. In order to incorporate reordering into outside string, we propose a new top-down algorithm in the next subsection.

3.2.2 Top-Down

Top-down method (Algorithm 1) is defined over the derivation tree, and takes the complete translations in the final beam as the potential for computing potential BLEU.

Suppose we have a partial derivation $d = r(d_1, d_2)$ as shown in Formula 1, and the target side string of rule r is:

$$w_1 \cdots w_p\ X_1\ w_q \cdots w_m\ X_2\ w_n \cdots w_l$$

The corresponding *(partial) translation* $e(d)$ of d could be generated by applying an r-based function $e_r(\cdot)$ with d_1 and d_2 as the arguments:

$$e_r(d_1, d_2) = w_1 \cdots w_p\ e(d_1)\ w_q \cdots w_m\ e(d_2)\ w_n \cdots w_l$$

where $e(d_1)$ and $e(d_2)$ are the partial translations of d_1 and d_2. We can further decompose $e_r(d_1, d_2)$ based on $e(d_1)$:

$$\underbrace{w_1 \cdots w_p}_{\eta_d(d_1)}\ e(d_1)\ \underbrace{w_q \cdots w_m\ e(d_2)\ w_n \cdots w_l}_{\xi_d(d_1)}.$$

where we call $\eta_d(d_1)$ and $\xi_d(d_1)$ the *partial left and right future strings* of d_1 under d. Similar

[3] For the spans not having translations, we compute a best possible translation for each of them by concatenating the translations of its children spans monotonically.

Algorithm 1 Top-Down Outside String.

```
1: function TopDown(d)
2:   r ← the rule that generates d
3:   for each non-terminal x in rule r do
4:     η(d_x) = η(d) ∘ η_d(d_x)
5:     ξ(d_x) = ξ_d(d_x) ∘ ξ(d)
6:     TopDown(d_x)
7:   return
```

to the outside cost of inside-outside algorithm, we compute the outside string $\langle \eta(d_1), \xi(d_1) \rangle$ for d_1 based on the decomposition:

$$\eta(d_1) = \eta(d) \circ \eta_d(d_1)$$
$$\xi(d_1) = \xi_d(d_1) \circ \xi(d).$$

Similarly, if we decompose $e_r(d_1, d_2)$ based on $e(d_2)$ we have:

$$\underbrace{w_1 \cdots w_p \;\; e(d_1) \;\; w_q \cdots w_m}_{\eta_d(d_2)} \;\; e(d_2) \;\; \underbrace{w_n \cdots w_l}_{\xi_d(d_2)},$$

and the outside string $\langle \eta(d_2), \xi(d_2) \rangle$ for d_2 is:

$$\eta(d_2) = \eta(d) \circ \eta_d(d_2)$$
$$\xi(d_2) = \xi_d(d_2) \circ \xi(d).$$

The top-down method works based on the above decompositions. For a complete translation in the final beam $B_{[0,|x|]}$, the algorithm tracebacks on its derivation tree, and gets the outside string for all spans in the derivation. Taking the span $[1, 2]$ in Figure 1(b) as an example, if we do top-down, its outside string would be:

$$\langle \text{Amy lived in Berlin before}, \;\; \epsilon \rangle.$$

where ϵ denotes the empty string. Compared to the concatenation method, top-down is able to consider the reordering between spans, and thus would be much better. However, since it bases on the k-best list in the final beam, it can only handle the spans appearing in the final k-best derivations. In our experiments, top-down algorithm only covers about 30% of all spans.[4] In order to incorporate more spans into search-aware tuning, we enhance the top-down algorithm and propose guided backtrace algorithm in the next subsection.

3.2.3 Guided Backtrace

The *guided backtrace* algorithm is a variation of top-down method. In guided backtrace, we first introduce a container $s_{[i,j]}$ to keep a best partial derivation for each split point of $[i, j]$ during decoding.[5] Hence, there will be at most $j - i$ partial derivations in $s_{[i,j]}$.[6] For instance, suppose the span is $[2, 5]$, we will keep three partial derivations in $s_{[i,j]}$ for split point 2, 3, and 4 respectively. Different from the similar k-best list in the decoding beam, $s_{[i,j]}$ introduces more diverse partial derivations for backtracing, and thus could help to cover more spans.

After decoding, we employ algorithm 2 to do backtrace. The algorithm starts from the best translation in the final beam. At first, we get the corresponding span of the input partial derivation d (line 2), and the outside string for this span (line 5-6). We then sort the partial derivations in $s_{[i,j]}$ based on their potential Bleu^{+1} (line 12). Thus, the sorting will guide us to first backtrace the partial derivations with better potential Bleu^{+1} scores. Then we traverse all these partial derivations (line 13-14), and do guided backtrace recursively on its child partial derivations (line 16-19). In this process, about 90% of all spans are visited in our experiments. We demand each span will only be visited once (line 3-4),

During the above process, guided backtrace will collect good partial derivations which have better potential Bleu^{+1} scores than the best Bleu^{+1} score *MaxSenBleu* of the final beam (line 10-11) into *goodcands*. Meanwhile, we also collect the good partial derivations from descendant nodes (line 19), and apply rule r to them to form good partial derivations for span $[i, j]$ (line 20-21). At last, we add the top 50 good partial derivations to the beam $B_{[i,j]}$ for tuning (line 22).

The purpose of adding good partial derivations for tuning is to recover the good ones pruned out of the beam. For example, the partial derivation $X_{[1,5]}$ in Figure 1 has been pruned out of the beam $B_{[1,5]}$ in Figure 2. If we only consider the partial derivations in the beam, it is still hard to promote it. After adding good partial derivations, we will have more good targets to do better tuning.

From Algorithm 2, we can also get a new way to compute oracle BLEU score of a translation system. We memorize the string that has the maximum potential Bleu^{+1} score of all strings (line 9). We then collect the best string of all source sen-

[4] We do not consider the hypothesis recombination in top-down algorithm.

[5] If the partial derivation is generated by HIERO rules, we set the split point as the last position of the first non-terminal.

[6] Some split points might not have corresponding partial derivations.

Algorithm 2 Guided Backtrace Outside String.

function GUIDED(d)
  $[i, j] \leftarrow$ the source span that d belongs to
  if $[i, j]$ has been visited **then**
    return $\emptyset$
  $\eta([i, j]) = \eta(d)$
  $\xi([i, j]) = \xi(d)$
  goodcands $\leftarrow \emptyset$
  for each partial derivation d in $B_{[i,j]}$ **do**
    $bleu = \text{Bleu}^{+1}(\eta([i, j]) \circ e(d) \circ \xi([i, j]))$
    if $bleu > MaxSenBleu$ **then**
      add d to goodcands
  sort $s_{[i,j]}$ based on potential Bleu^{+1}
  for each partial derivation d of span $[i, j]$ **do**
    $r \leftarrow$ the rule that generates d
    $D_d \leftarrow$ the set of partial derivations r use to form d
    for each non-terminal x in rule r **do**
      $\eta(d_x) = \eta([i, j]) \circ \eta_d(d_x)$
      $\xi(d_x) = \xi_d(d_x) \circ \xi([i, j])$
      cands $\leftarrow$ GUIDED(d_x)
      for each cand d_g in cands **do**
        add $r(d_g, D_k \backslash d_x)$ to goodcands
  add top 50 partial derivations of goodcands to $B_{[i,j]}$
  return goodcands

tences in the tuning or test set, and use them to compute BLEU score of the set. This can be seen as an approximation upper bound of the current model and decoder, which we call **guided oracle**.

3.3 Implementation

The major process of search-aware tuning is similar to traditional tuning pipeline. We first decode the source sentence, and then output both the complete translations in the final beam and the partial derivations from the shorter spans as training instances. For potential BLEU, the outputs of partial derivations would be the corresponding complete translations generated by Equation (2). If we use partial BLEU, the outputs are the corresponding partial translations. Each span will serve as a single tuning instance. Different from (Liu and Huang, 2014), we only use the features from partial derivations for tuning.

For the spans that top-down or guided backtrace algorithm cannot get outside strings, we use concatenation for them to maintain consistent number of tuning instances between different tuning iterations. However, since we do not want to spent much effort on them, we only use the one-best partial derivation for each of them.

4 Experiments

To evaluate our method, we conduct experiments on Chinese-to-English translation. The training data includes 1.8M bilingual sentence pairs, with about 40M Chinese words and 48M English words. We generate symmetric word alignment using GIZA++ and the *grow-diag-final-and* strategy. We train a 4-gram language model on the Xinhua portion of English Gigaword corpus by SRILM toolkit (Stolcke, 2002). We use BLEU 4 with "average reference length" to evaluate the translation performance for all experiments.

We use the NIST MT 2002 evaluation data (878 sentences) as the tuning set, and adopt NIST MT04 (1,789 sentences), MT05 (1,082 sentences), MT06 (616 sentences from new portion) and MT08 (691 sentences from new portion) data as the test set.

Our baseline system is an in-house hierarchical phrase-based system. The translation rules are extracted with Moses toolkit (Koehn et al., 2007) by default settings. For the decoder, we set the beam size to 30, nbest list to 50, and 20 as the maximum length of spans for using non-glue rules.

The baseline tuning methods are batch tuning methods based on k-best translations, including MERT (Och, 2003), MIRA (Cherry and Foster, 2012) and PRO (Hopkins and May, 2011) from Moses. Another baseline tuning method is hypergraph-MERT from `cdec` toolkit (Dyer et al., 2010). To guarantee the tuning efficiency, we constrain the minimum length of spans for search-aware tuning to restrict the number of training instances. For the sentences with less than 20 words, we only use the spans longer than 0.75 times sentence length. For the ones with more than 20 words, the minimum span length is set to 18.[7] All results are achieved by averaging three independent runs for fair comparison (Clark et al., 2011).

4.1 Translation Results

Table 1 compares the main results of our MERT-based search-aware tuning with traditional tuning: MERT and hypergraph-MERT.

From the results, we can see that hypergraph-MERT is better than MERT by 0.5 BLEU points, verifying the result of (Kumar et al., 2009). For search-aware tuning, partial BLEU (both full and span one) only gets comparable results with baseline tuning method, confirming our previous analysis in section 3.1, and the results are also consistent with (Liu and Huang, 2014).

[7] As the decoder demands that spans longer than 20 can only be translated by glue rule, for these spans, we only need to consider the ones beginning with 0 in search-aware tuning.

	nist03	nist04	nist05	nist06	nist08	avg.
MERT	34.4	35.9	34.2	31.9	28.2	32.9
MERT-S	34.3	36.1	34.3	32.3	28.5	33.1
hypergraph-MERT	34.5	36.4	34.4	33.0	28.9	33.4
full partial BLEU	34.3	35.9	34.1	32.4	28.1	33.0
span partial BLEU	34.1	36.1	34.5	32.5	28.4	33.1
concatenation	34.3	36.4	34.6	33.1	28.5	33.4
top-down	34.5	36.6	34.5	33.1	28.9	33.5
guided backtrace	**34.9**	**36.9**	**35.2**	**33.7**	**29.1**	**34.0**

Table 1: MERT results: BLEU scores the test sets (nist03, nist04, nist05, nist06, and nist08). MERT-S is an enhanced version of MERT, which uses a beam size 200 and nbest list 200 for tuning, and beam size 30 for testing.

	methods	nist02	nist04
1-best	MERT	36.1	35.9
	guided backtrace	**+0.5**	**+1.0**
k-best Oracle	MERT	43.3	42.8
	guided backtrace	**+1.1**	**+1.2**
Guided Oracle	MERT	48.0	47.2
	guided backtrace	**+1.4**	**+1.4**

Table 2: The oracle BLEU comparison between baseline MERT and guided backtrace.

Compared to partial BLEU, potential BLEU is more helpful. Both concatenation and top-down method are better than MERT on all five test sets. Guided backtrace gets the best performance over all methods. It outperforms traditional MERT by 1.1 BLEU points on average, and better than hypergraph-MERT by 0.6 BLEU points.

4.2 Analysis

In this section, we analyze the results of search-aware tuning by comparing MERT-based baseline and guided backtrace in detail.

Oracle BLEU We first compare the oracle BLEU scores of baseline and guided backtrace in Table 2. In order to get the k-best oracle, we first look for the best Bleu^{+1} translation in the k-best list for each source sentence, and then use these best translations to compute the BLEU score of the entire set. To get the guided oracle, we use the weights from baseline MERT to run Algorithm 2 on tuning set (nist02) and nist04 test set, and generate the best oracle translation (section 3.2.3) for each source sentence for evaluation.

The final oracle BLEU comparison is shown in Table 2. On both nist02 tuning set and nist04 test set, guided backtrace method gains at lease 1.0 BLEU points improvements over traditional MERT on both k-best oracle and guided oracle. Moreover, k-best and guided oracle get more improvements than 1-best, indicating that by search-aware tuning, the decoder could generate a better k-best list, and has a higher upper bound.

	nist02	nist04
MERT	0.3960	0.3791
guided backtrace	0.4098	0.3932

Table 3: The diversity comparison based on k-best list in the final beam on both tuning set (nist02) and nist04 test set. The higher the value is, the more diverse the k-best list is.

Diversity As shown in (Gimpel et al., 2013; Liu and Huang, 2014), diversity is important for MT tuning. An k-best list with higher diversity can better represent the entire decoding space, and thus tuning on it will lead to a better performance.

Similar as (Liu and Huang, 2014), our method encourages the diversity of partial translations in each beam, including the ones in the final beam. We use the metric in (Liu and Huang, 2014) to compare the diversity of traditional MERT and guided backtrace. The metric is based on the n-gram matches between two sentences y and y':

$$\Delta(y, y') = -\sum_{i=1}^{|y|-q} \sum_{j=1}^{|y'|-q} \langle y_{i:i+q} = y'_{j:j+q} \rangle$$

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

where $q = n - 1$ and $\langle x \rangle$ equals to 1 if x is true, 0 otherwise. The final diversity results are shown in Table 3. We can see guided backtrace gets a better diversity than traditional MERT.

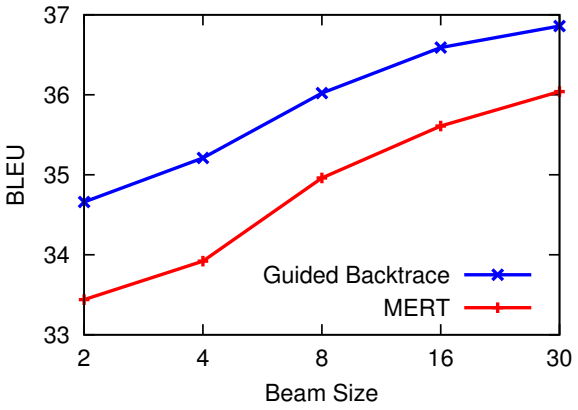


Figure 4: The BLEU comparison between MERT and guided backtrace on nist04 test set over different beam sizes.

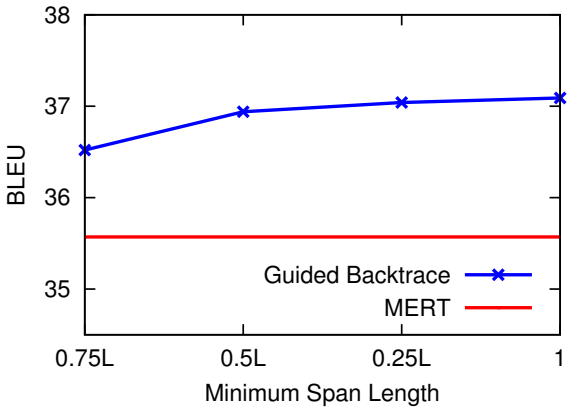


Figure 5: The BLEU comparison between MERT and guided backtrace on nist04 test set over different span sizes. L denotes the sentence length. The value of x-axis refers to the minimum length of spans used for search-aware tuning. More spans will be used in tuning with smaller minimum span length.

Beam Size As we discussed before, search-aware tuning helps to accommodate search errors in decoding, and promotes good partial derivations. Thus, we believe that even with a small beam, these good partial derivations can still survive with search-aware tuning, resulting in a good translation quality. Figure 4 compares the results of different beam sizes $(2, 4, 8, 16, 30)$ between traditional MERT and guided backtrace. The comparison shows that guided backtrace achieves better result than baseline MERT, and when the beam is smaller, the improvement is bigger. Moreover, guided backtrace method with a beam size 8 could achieve comparable BLEU score to traditional MERT with beam size 30.

Span Size For a big tuning set, in order to make the tuning tractable, we constrain the length of spans for search-aware tuning. Intuitively, towards search-aware tuning, more spans will get better results because we have more training instances to guide the tuning process, and accommodate search errors in early decoding stage (shorter spans). To verify this intuition, we perform experiments on a small tuning set, which is generated by selecting the sentences with less than 20 words from the original NIST MT 02 tuning set.

Figure 5 compares the results of traditional MERT and guided backtrace over different minimum span length. The curve in the figure shows that by using more spans for search-aware tuning, we can achieve better translation performance.

4.3 MIRA and PRO Results

Table 4 and Table 5 shows the final results of MIRA and PRO for traditional tuning and search-aware tuning using potential BLEU. We can see that potential BLEU is helpful for tuning. Guided backtrace is also the best one, which outperforms the baseline MIRA and PRO by 0.9 and 0.8 BLEU points on average.

4.4 Efficiency

Table 6 shows the training time comparisons between search-aware tuning and traditional tuning. From this Table, we can see that search-aware tuning slows down the training speed.[8] The slow training is due to three reasons.

Similar to (Liu and Huang, 2014), as search-aware tuning considers partial translations of spans besides the complete translations, it has much more training instances than traditional tuning. In our experiments, although we have constrained the length of spans, we get a total of 38,539 instances for search-aware tuning, while it is 878 for traditional tuning.

Another time consuming factor is the computation of Bleu^{+1} scores. In guided backtrace setting, we need to compute the Bleu^{+1} scores for all candidates of spans we consider. This is why the

[8] Since the tuning set is different for traditional PRO tuning and search-aware tuning, we do not compare them here.

	nist03	nist04	nist05	nist06	nist08	avg.
MIRA	34.5	36.1	34.4	32.1	28.5	33.1
concatenation	34.5	36.5	34.6	33.6	28.9	33.6
top-down	34.4	36.6	34.6	33.5	28.8	33.6
guided backtrace	**35.0**	**36.9**	**35.1**	**33.7**	**29.1**	**34.0**

Table 4: MIRA results: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08).

	nist03	nist04	nist05	nist06	nist08	avg.
PRO	34.2	36.1	33.9	32.3	28.7	33.1
concatenation	34.6	36.2	34.7	32.3	28.5	33.2
top-down	34.8	36.4	34.7	32.7	29.0	33.5
guided backtrace	**35.0**	**36.8**	**34.7**	**33.4**	**29.6**	**33.9**

Table 5: PRO results: BLEU scores on test sets (nist03, nist04, nist05, nist06, and nist08). Similar to (Liu and Huang, 2014), there is also an monster phenomenon (Nakov et al., 2013) in our search-aware tuning setting. Therefore, here we perform search-aware tuning on only 109 short sentences (with less than 10 words) from nist02.

	MERT		MIRA	
	Total	Optim.	Total	Optim.
baseline	17	2	17	2
concatenation	59	44	38	23
top-down	31	14	23	8
guided backtrace	108	45	69	22

Table 6: Comparison on training efficiency. The time (in minutes) is measured at the last iteration of tuning. Column "Total" refers to the time for an entire iteration, while "Optim." is the time of optimization. We use a parallelized MERT for tuning by 24 cores.

decoding of guided backtrace is much slower than baseline or top-down.

Finally, adding good candidates enlarges the k-best lists of training instances, which further slows down the tuning process of guided backtrace.

It should be noted that although search-aware tuning is slower than traditional tuning method, since the decoding is all the same for them in testing time, it does not slow down the testing speed.

5 Conclusion

We have presented an extension of "search-aware tuning" to accommodate search errors in hierarchical phrase-based decoding and promote the promising partial derivations so that they are more likely to survive in the inexact beam search. In order to handle the tree-structured derivations for HIERO system, we generalize the BLEU metrics and propose corresponding partial BLEU and potential BLEU to evaluate partial derivations. Our approach can be used as a plugin for most popular parameter tuning algorithms including MERT, MIRA and PRO. Extensive experiments confirmed substantial BLEU gains from our method.

6 Acknowledgment

We thank the three anonymous reviewers for the valuable suggestions. This project was supported in part by DARPA FA8750-13-2-0041 (DEFT), NSF IIS-1449278, and a Google Faculty Research Award.

References

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of NAACL 2009*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–208.

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 *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL*.

Michel Galley, Chris Quirk, Colin Cherry, and Kristina Toutanova. 2013. Regularized minimum error rate training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1948–1959, Seattle, Washington, USA, October. Association for Computational Linguistics.

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*, pages 1100–1111, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

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

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*.

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 the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 163–171. Association for Computational Linguistics.

Lemao Liu and Liang Huang. 2014. Search-aware tuning for machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1942–1952, Doha, Qatar, October. Association for Computational Linguistics.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. 2013. A tale about pro and monsters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 12–17, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

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

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

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

Kai Zhao, Liang Huang, Haitao Mi, and Abe Ittycheriah. 2014. Hierarchical mt training using max-violation perceptron. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 785–790, Baltimore, Maryland, June. Association for Computational Linguistics.