Optimizing for Sentence-Level BLEU+1 Yields Short Translations

Preslav NAKOV Francisco GUZMÁN Stephan VOGEL
Qatar Computing Research Institute, Qatar Foundation
Tornado Tower, floor 10, P.O. Box 5825, Doha, Qatar
{pnakov,fherrera,svogel}@qf.org.qa

ABSTRACT
We study a problem with *pairwise ranking optimization* (*PRO*): that it tends to yield too short translations. We find that this is partially due to the inadequate smoothing in PRO's BLEU+1, which boosts the precision component of BLEU but leaves the brevity penalty unchanged, thus destroying the balance between the two, compared to BLEU. It is also partially due to PRO optimizing for a sentence-level score without a global view on the overall length, which introducing a bias towards short translations; we show that letting PRO optimize a *corpus-level* BLEU yields a perfect length. Finally, we find some residual bias due to the interaction of PRO with BLEU+1: such a bias does not exist for a version of MIRA with sentence-level BLEU+1. We propose several ways to fix the length problem of PRO, including smoothing the brevity penalty, scaling the effective reference length, grounding the precision component, and unclipping the brevity penalty, which yield sizable improvements in test BLEU on two Arabic-English datasets: IWSLT (+0.65) and NIST (+0.37).

KEYWORDS: Statistical machine translation, parameter optimization, MERT, PRO, MIRA.

Proceedings of COLING 2012: Technical Papers, pages 1979–1994,
COLING 2012, Mumbai, December 2012.

1 Introduction

Early work on statistical machine translation (SMT) has relied on generative training using maximum likelihood parameter estimation. This was inspired by the noisy channel model (Brown et al., 1993), which asked for calculating the product of two components, a language model and a translation model, giving them equal weights. As mainstream research has moved towards combining multiple scores, the field has switched to discriminative tuning in a log-linear fashion. The standard approach has been to maximize BLEU (Papineni et al., 2002) on a tuning dataset using a coordinate descent optimization algorithm known as *minimum error rate training* (*MERT*), as proposed by Och (2003).

MERT has dominated the SMT field for years, until the number of parameters in the log-linear model has gradually increased, in some cases to hundreds and even to hundreds of thousands of scores, which has called for new tuning algorithms since MERT was unable to scale beyond just a handful of parameters. Many alternatives to MERT have been proposed over the years, but it is only recently that some of them have gained popularity in the community, most notably, the *margin infused relaxed algorithm* (MIRA) (Chiang et al., 2008) and *pairwise ranking optimization* (*PRO*) (Hopkins and May, 2011).

While the number of parameters that an optimizer can handle has become a major concern recently, there are many other important aspects that researchers have paid attention to, e.g., the performance of parameters when translating unseen test data, the speed of convergence, the stability across multiple reruns, the objective function being optimized (e.g., BLEU vs. an approximation of BLEU), the mode of learning (e.g., online vs. batch).

Here we study a different, and so far neglected, aspect: the characteristics of the translations generated using weights found by different optimizers. More specifically, we focus on the length with respect to the reference translations. It has been observed that while optimizers like MERT and MIRA typically yield translation hypotheses with the right level of verbosity, other optimizers such as PRO tend to generate translations that are too short.[1] This phenomenon has not been analyzed in sufficient detail so far; yet, it is important since generating short translation hypotheses is penalized by BLEU, which could mean missed opportunities for better BLEU scores. Our contributions can be summarized as follows:

- We analyze the length issue with PRO in detail, we explore possible reasons, and we propose several fixes. We achieve small but consistent improvements in BLEU for fixes that yield longer hypotheses: up to 0.65 BLEU points absolute. We also find that the better a fix approximates the target length, the higher the testing BLEU score.
- We find that the core of the problem is the function being optimized: sentence-level BLEU+1 yields short translations, while corpus-level BLEU yields the right verbosity.
- We demonstrate that what matters is the objective function, not the optimization algorithm. We show that similar translations – in terms of BLEU score and length – can be generated using weights from different optimizers, e.g., PRO vs. MIRA, when they are given the same objective function, e.g., sentence-level vs. corpus-level BLEU.

The remainder of this paper is organized as follows: Section 2 describes related work, Section 3 explains the length issue with PRO and several methods we propose to fix it, Section 4 describes our experiments and evaluation results, and Section 5 offers a more general analysis and discussion. Finally, Section 6 concludes with directions for future work.

[1] That is why the Moses toolkit has an option to run a few iterations of MERT after PRO – to get the length right.

2 Related Work

The dominant approach for parameters optimization in SMT is to use MERT (Och, 2003), a batch tuning algorithm that iterates between two modes: (i) generating a k-best list of translation hypotheses using the current parameters values, and (ii) parameter optimization using the k-best lists from all previous iterations. MERT optimizes expected BLEU. It works well for a small number of parameters, but suffers from scalability and stability issues (Foster and Kuhn, 2009). Most importantly for our discussion, it tends not to have length biases; this is also confirmed by our own experiments (see Table 4).

Various alternatives to MERT have been proposed, motivated primarily by scalability considerations. One popular alternative is MIRA (Watanabe et al., 2007; Chiang et al., 2008, 2009), which is a perceptron-like online tuning algorithm with passive-aggressive updates. It uses an approximation to BLEU, where a sentence is scored in the context of a pseudo-document formed from the n-gram statistics for the last few updates. MIRA can scale to thousands of parameters and generally has no length bias (see Table 4).

Another recent, but already popular alternative to MERT is PRO (Hopkins and May, 2011), which models parameter tuning as pairwise ranking optimization. This is a batch tuning algorithm, which iterates between translation and optimization, just like MERT, but scales to thousands of parameters. It uses an add-one smoothed sentence-level version of BLEU, known as BLEU+1 (Lin and Och, 2004), and suffers from a length bias: the parameters it finds yield translations that are too short compared to the references (see Table 4). Exploring the reasons for this bias and proposing ways to solve it is the main focus of this paper.

There are many other tuning strategies, which fall outside of the scope of the current study, but to many of which some of our general finding and conclusions should apply. This includes improved versions of some of the above-mentioned algorithms, e.g., a batch version of MIRA (Cherry and Foster, 2012), or a linear regression version of PRO (Bazrafshan et al., 2012), but also many original algorithms that use a variety of machine learning methods and loss functions. We refer the interested reader to some excellent recent overviews: (McAllester and Keshet, 2011; Cherry and Foster, 2012; Gimpel and Smith, 2012).

To the best of our knowledge, no prior work has tried to study the reasons for the length bias of optimizers like PRO. However, researchers have previously expressed concerns about sentence-level BLEU+1, and some have proposed improvements, e.g., He and Deng (2012) used different smoothing for higher-order n-grams, unclipped brevity penalty, and scaled reference length. However, this was not done for the purpose of studying the length bias of PRO; moreover, as we will see below, the use of BLEU+1 is not the only reason for this bias.

3 The Length Bias with PRO

We explore the following hypotheses about the length bias with PRO:

- **PRO's optimization:** The bias could be due to the optimization mechanism of PRO.
- **BLEU+1:** PRO uses BLEU+1, where the add-one smoothing is applied to the precision component but does not touch the brevity penalty, which introduces an imbalance.
- **Sentence-level optimization:** PRO uses a (smoothed) sentence-level BLEU instead of corpus-level BLEU, which could make it hard to get the global corpus length right.

3.1 Possible Reason 1: The Optimization Mechanism of PRO

PRO is a batch optimizer that iterates between (i) *translation*: using the current parameter values, generate k-best translations, and (ii) *optimization*: using the translations from all previous iterations, find new parameter values. The optimization step has four substeps:

1. **Sampling:** For each sentence, sample uniformly at random $\Gamma = 5000$ pairs from the set of all candidate translations for that sentence from all previous iterations.
2. **Selection:** From these sampled pairs, select those for which the absolute difference in their BLEU+1 scores is higher than $\alpha = 0.05$.
3. **Acceptance:** For each sentence, accept the $\Xi = 50$ selected pairs with the highest absolute difference in their BLEU+1 scores.
4. **Learning:** Assemble the accepted pairs for all sentences into a single set and use it to train a ranker to prefer the higher-scoring sentence in each pair.

While sampling is unlikely to introduce a bias, selection and acceptance could do so.

Selection could filter too many pairs for some of the sentences, leaving for them less than Ξ pairs for the acceptance step; such sentences would be under-represented compared to those for which there are Ξ pairs accepted. If these under-represented sentences are generally longer then their references, this could yield a bias towards shorter translations.

Similarly, by focusing on the pairs with the highest differentials, the acceptance step would over-represent pairs with big differences in sentence lengths, e.g., a sentence that is extremely long/short compared to the reference would have a very low BLEU+1 score, and thus it will have a high differential when paired with any other sentence. If at some iteration, the k-best list is populated with overly long sentences, they would keep getting accepted in subsequent iterations of PRO, and thus the classifier will keep learning that being shorter is better.

Given the above discussion, we propose the following experiments with PRO:

- **PRO, no threshold.** At the selection step, keep everything, i.e., no selection step.
- **PRO, random accept.** At the acceptance step, accept the pairs at random, as opposed to accepting those with the highest differentials in BLEU+1, i.e., no acceptance step.
- **PRO, no threshold, random accept.** Combination of the previous two.

3.2 Possible Reason 2: BLEU+1

The other possible reasons are related to BLEU; thus, we should have a look at its definition:

$$\text{BLEU} = BP \cdot \left(\prod_{n=1}^{N} p_n \right)^{\frac{1}{N}} \tag{1}$$

BLEU has two components: (1) brevity penalty (BP), and (2) precision component (PC), the geometric mean of n-gram precisions p_n, $1 \leq n \leq N$. The BP is defined as follows:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \tag{2}$$

where c is the length of the candidate, and r is the effective reference corpus length.

BLEU is defined at the corpus-level, and p_n, r and c are sums over all corpus sentences:

- $p_n = \frac{\sum_i m_{in}}{\sum_i h_{in}}$, where m_{in} is the number of n-gram matches between a translation and the references for sentence i, and h_{in} is the number of n-grams in the hypothesis;[2]
- $c = \sum_i c_i$, where c_i is the length of the candidate translation for sentence i;
- $r = \sum_i r_i$, where r_i is the reference length for sentence i; in case of multiple reference translations, this is the closest reference sentence length.

While BLEU is defined at the corpus level, PRO works at the sentence level and thus needs to optimize sentence-level BLEU for each sentence i. Such a version of BLEU can be obtained by redefining p_n, r and c to look at sentence i only: $p_n = \frac{m_{in}}{h_{in}}$, $c = c_i$, and $r = r_i$.

Using such a sentence-level version of BLEU is problematic though since it can easily become zero. This is because of the product of n-gram precisions in the geometric mean of the precision component of BLEU: if some p_n is zero, the whole product will be zero. In particular, it is easy to see that BLEU will be zero for any hypothesis without 4-gram matches. This is undesirable for optimization purposes since it does not allow to distinguish (a) a hypothesis translation that has no matches at all from (b) one that has unigram, bigram and trigram matches but no 4-gram matches; intuitively, the latter should be preferred over the former.

A popular strategy to solve the problem, which is also adopted by PRO, is to use an add-one smoothed version of BLEU, called BLEU+1, where p_n is redefined as follows: $p_n = \frac{m_{in}+1}{h_{in}+1}$.

There are two observations we can make about BLEU+1: (i) while it alters the precision component, it leaves the brevity penalty unchanged, and (ii) it is strictly positive.[3]

Let us start with the first observation. Adding one to m_{in} implies the need to add an extra n-gram to the reference translation – an n-gram that matches the candidate translation. This can be seen in the extreme case of a perfect match between the hypothesis and the reference: then, the additional match between the hypothesis and the reference, which is accounted for in m_{in}, would also require an additional word in the reference. However, the brevity penalty is not altered to account for this additional n-gram. Consider the special case of unigrams: if we assume that the reference contains an extra unigram, then it should be longer by one word. And vice versa, adding one extra word to the reference would increase the number of n-grams it contains by one for each n, $1 \leq n \leq N$, which is exactly what BLEU+1 assumes in its precision smoothing.

Let us try to analyze the impact of BLEU+1 on the hypothesis length. We have seen that BLEU+1 sees the precision component of a longer reference, but it pays the brevity penalty with respect to a shorter one. Because BLEU+1 boosts the precision component while leaving the BP intact, the relative weight of BP decreases compared to the original BLEU. This means that there could be potential gain from staying shorter if this can boost precision even further: the BP price to be paid for this would be relatively lower than it was in BLEU. Thus, high-scoring shorter hypotheses are more likely with BLEU+1 than with BLEU.

[2] More precisely, let g_n be an arbitrary n-gram, and let $\#_c(i, g_n)$ be the number of times g_n occurs in the candidate translation for sentence i. Let also $\#_r(i, g_n)$ be the maximum number of occurrences of g_n in any reference translation for sentence i. Then, we can define $m_{in} = \sum_{g_n} \min\left(\#_c(i, g_n), \#_r(i, g_n)\right)$ and $h_{in} = \sum_{g_n} \#_c(i, g_n)$.

[3] While BLEU+1 in PRO is strictly positive, there are definitions in the literature that allow it to become zero, e.g., Lin and Och (2004) do not smooth the unigram counts, which makes BLEU+1 zero in case of no matches at all. Experimenting with their version of BLEU+1 is left for future work; it would still need a fix for BP though.

Let us now consider the second observation: that BLEU+1 is strictly positive. While there is nothing wrong with this per se, it suggests a different way to restore the balance between BP and PC: by "grounding" BLEU+1 so that it is zero when there are no matches at all. This can be achieved by subtracting from the precision component of BLEU+1 the score for that component when there are no matches. Thus, the "grounded" precision component of BLEU+1 changes from $PC = \left(\prod_{n=1}^{N} \frac{m_{in}+1}{h_{in}+1}\right)^{\frac{1}{N}}$ to $PC = \left(\prod_{n=1}^{N} \frac{m_{in}+1}{h_{in}+1}\right)^{\frac{1}{N}} - \left(\prod_{n=1}^{N} \frac{1}{h_{in}+1}\right)^{\frac{1}{N}}$.

Given the above discussion, we propose the following fixes for the sentence-level BLEU+1:

- **BLEU, PC-unsmoothed**. Just use BLEU. The idea is that if add-one smoothing in BLEU+1 is causing length problems, they should go away if unsmoothed BLEU is used instead. While unsmoothed BLEU will make many pairs of hypotheses indistinguishable, both having a score of zero, many other pairs with at least one non-zero-score hypothesis will still remain, and they should be enough to train the classifier of PRO.
- **BLEU+1, grounded.** "Ground" the PC of BLEU+1 by subtracting from it its value when there are no matches. Given the asymmetry of add-one smoothing in BLEU+1, which boosts the PC for shorter sentences more than for longer ones for the same number of matches, "grounded" BLEU is slightly more biased towards longer sentences.
- **BLEU+1, BP-smoothed.** Use add-one smoothing not only for the precision component but also for the length of the reference translation, i.e., use $r = r_i + 1$, in addition to $p_n = \frac{m_{in}+1}{h_{in}+1}$. The idea here is to smooth the PC and the BP consistently, thus maintaining the balance between them: if we assume an extra n-gram in the reference, then we should also assume that the reference contains an extra word.
- **BLEU+1, BP-smoothed, grounded.** Combination of the previous two: smooth the reference length in BP and also ground the precision component of BLEU+1 .

3.3 Possible Reason 3: Sentence-Level Optimization

Another possible reason for the length issue with PRO could be a bias due to a sentence-level version of BLEU being optimized as opposed to using a corpus-level BLEU. If, for whatever reason, being a bit shorter than the reference is preferable to being a bit longer, e.g., because sentence-level BLEU+1 penalizes longer sentences more than shorter ones, then the candidate translation for each sentence will try to stay on the short side as opposed to getting longer than the reference translation. These length differences might be minimal at the sentence level, but they would accumulate and would eventually yield a larger difference at the corpus level. The crucial observation is that if each and every sentence looks at its length in isolation, it would be reluctant to getting longer than its reference (since this would harm its sentence-level BLEU), even if this could improve corpus-level BLEU.

One possible way to address the problem is to introduce in the sentence-level brevity penalty information about the ratio of the corpus-level length and the effective reference corpus length from the previous iteration of PRO. For example, if the corpus length was shorter than the effective reference corpus length, then we could scale the sentence-level reference lengths r_i on the current iteration accordingly, so that proportionally higher brevity penalty be paid for being too short; and vice versa, if on the previous iteration the corpus length was too long, we could scale the reference length so that no penalty be paid for staying proportionally shorter; this idea was previously explored by He and Deng (2012).

Another possibility is to unclip the brevity penalty, as proposed by He and Deng (2012), by allowing it to get bigger than 1, thus effectively becoming a "bonus" for longer sentences. If there is a bias in sentence-level BLEU+1 towards shorter translations, i.e., longer hypotheses are penalized by the precision component more heavily than shorter hypotheses are penalized by BP, then turning the brevity penalty into a bonus for longer hypothesis should encourage them; note, however, that they will still be discouraged by the decrease in the score for the precision component, which should act as a counter-balance.

Finally, if optimizing sentence-level BLEU causes problems, we could just get rid of it and use some kind of corpus-level BLEU instead. This is what MIRA does (Chiang et al., 2008): even though it is an online algorithm and makes updates on a per-sentence basis, it does not use a sentence-level BLEU, but instead it calculates the BLEU score for the current sentence in the context of a pseudo-corpus that tracks the n-gram statistics of the model-best derivations from the last few updates. We could adopt a similar strategy here as well.

Given the above, we propose the following fixes/substitutes for the sentence-level BLEU+1:

- **BLEU+1, unclipped.** Use an unclipped brevity penalty: $BP = \exp\left(1 - \frac{r}{c}\right)$.
- **BLEU+1, scaled.** Scale the reference length with the inverse length ratio (ILR) from the previous iteration of PRO, i.e., define $r = r_i \cdot ILR$, where $ILR = c'/r' = \sum_i c'_i / \sum_i r'_i$, where c' and r' are the corpus-level candidate and reference lengths from the previous iteration, respectively; we define $LR = 1$ for the first iteration of PRO.
- **BLEU+1, unclipped, scaled.** Combination of the previous two: BLEU+1 with unclipped brevity penalty and also with scaled reference length. This combination is advocated by He and Deng (2012).[4]
- **Corpus-level BLEU, MIRA-style.** This is BLEU calculated using a background pseudo-corpus, similar to the way this is done in MIRA. In our case, this pseudo-corpus is initialized with an exponentially decaying sum over the sufficient statistics for the one-best hypotheses from all previous iterations of PRO: taken with a weight of 0.9 for the last iteration,[5] with a weight of 0.9^2 for the iteration before it, etc. Then, the BLEU score for a candidate sentence is calculated by adding its sufficient statistics to the sufficient statistics for the background pseudo-corpus. Each time PRO samples, selects, and accepts a pair of hypotheses to use for training its classifier, we immediately update the pseudo-corpus by adding to it the sufficient statistics for the higher-scoring sentence, i.e., the positive example, in the pair. Thus, while the pseudo-corpus is initialized with the statistics from previous iterations, it soon gets dominated by statistics for the positive examples that are being accepted as training sentence pairs at the current iteration, since there are more of them than there are one-best hypotheses from previous iterations.[6] As in MIRA, our pseudo-document is time-dependent and thus can adjust dynamically to changing sentence lengths by encouraging or discouraging longer translations depending on what kinds of sentences have been accepted already.

[4] He and Deng (2012) further add to the combination a sophisticated smoothing for the precision component; however, in our experiments, the combination of all three changes to BLEU of theirs did not work well with PRO, yielding extremely long translations and an absolute loss of seven BLEU points on the NIST datasets.

[5] The value of 0.9 was found to work well in general, but many other values yielded a similar result since the BLEU score gets dominated by examples from the current iteration very quickly, making this value irrelevant.

[6] We only allow up to 25 iterations, which means there could be up to 24 accumulated one-best hypotheses per sentence, while we accept up to $\Xi = 50$ pairs per sentence.

4 Experiments and Evaluation

We compare variations of three parameter optimization algorithms: MERT, PRO, and MIRA. In all experiments, we use the phrase-based SMT model (Koehn et al., 2003) as implemented in the Moses toolkit (Koehn et al., 2007), and we report evaluation results over two datasets: NIST, which has four reference translations, and IWSLT, with a single reference translation.

In order to be able to directly compare the candidate/reference length ratios on the development and on the testing datasets, we need to make sure that we use the same tokenization when calculating BLEU on tuning and on testing. Such differences can arise because many standard scoring tools, e.g., those of NIST, work on detokenized text, which they retokenize again internally; this retokenization typically differs from the one used by the SMT system. As a result, the optimizer and the final scorer would most certainly see different tokenizations, and more crucially, this could affect the length ratios and thus the brevity penalty. Thus, we report BLEU scores calculated using the *multi-bleu* scoring tool, which uses the supplied tokenization and does no retokenization; we supplied it with references that were tokenized and truecased using the same models that were used for training and tuning.[7]

Finally, in order to avoid stability issues, we report results averaged over three runs.

4.1 Experimental Setup

Preprocessing: We tokenized the English side of all bi-texts and the monolingual data for language modeling using the standard tokenizer of Moses. We further truecased this data by changing the casing of each sentence-initial word to its most frequent casing in the training corpus; for lines containing ALL CAPS, we did this for each word. We segmented the words on the Arabic side using the ATB segmentation scheme: we used MADA (Roth et al., 2008) for NIST, and the Stanford word segmenter (Green and DeNero, 2012) for IWSLT.

Training. We built separate directed word alignments using IBM model 4 (Brown et al., 1993), we symmetrized them with the *grow-diag-final-and* heuristic of Moses, and we extracted phrase pairs of length up to seven. We scored these pairs using maximum likelihood with Kneser-Ney smoothing, to build a phrase table with five standard scores: forward and reverse phrase translation probabilities, forward and reverse lexical translation probabilities, and phrase penalty. We also built a lexicalized reordering model (Koehn et al., 2005): *msd-bidirectional-fe*. For language modeling, we trained a separate 5-gram Kneser-Ney smoothed model on each corpus (target side of a training bi-text or monolingual dataset); we then interpolated these models minimizing the perplexity on the target side of the tuning dataset. Finally, we built a log-linear model including the language model probability, the word penalty, and the parameters from the phrase and the reordering tables.

Tuning. We tuned the weights in the log-linear model by optimizing BLEU (Papineni et al., 2002) on the tuning dataset, using MERT, PRO, or MIRA. We allowed optimizers to run for up to 25 iterations, and we allowed them to extract 1000-best lists for each iteration.

Decoding. On tuning and testing, we dropped unknown words (this has yielded slightly shorter translations) and we used monotone-at-punctuation decoding (this had no impact on the translation length). On testing, we further used cube pruning and minimum Bayes Risk decoding (the latter yielded slightly longer translations).

[7] Still, for comparison purposes, we also report BLEU calculated with respect of the original references using NIST v13a, after detokenization and recasing of the system's output (shown in small script in the tables).

4.2 Datasets

We experimented with the Arabic-English datasets from two machine translation evaluation campaigns: (1) the NIST 2012 Open Machine Translation Evaluation[8], and (2) the IWSLT 2011 Evaluation Campaign on Automatic Talk Translation (Federico et al., 2012).

1. NIST: We trained the phrase and the reordering tables on all training datasets from NIST 2012 (except for UN), we tuned on MT06 and tested on MT09. For language modeling, we built a separate LM from the English side of each training dataset, and from each year of the English GigaWord; we then interpolated them into a single LM.
2. IWSLT: We trained the phrase and the reordering tables on the TED training dataset, we tuned on dev2010, and we tested on tst2010. Since there was a small mismatch in the source/reference length ratios between dev2010 and tst2010, we also experimented with reversed tuning/testing, tuning on tst2010 and testing on dev2010; this is to see the impact of the test set length ratio being a bit longer and also being a bit shorter than the tuning dataset ratio. We used two LMs: one trained on the English side of TED, and another one that is an interpolation of Europarl and WMT11 News.

4.3 Evaluation Results

We experimented with MERT, MIRA, PRO, and the various fixes thereof that were introduced in Section 3. The results of these experiments are shown in Tables 1, 2, 3, and 4. In these tables, the first column describes the method, followed by (1) the BLEU scores and (2) the length ratio of the candidate corpus-level translation to the effective reference corpus length, which are calculated (i) for the tuning dataset using multi-bleu,[9] (ii) for the test dataset using multi-bleu, and (iii) for the test dataset using the NIST scoring tool v13a.

The most important column in these tables is column three, which reports the candidate/reference length ratio for the tuning dataset as calculated using multi-bleu. We will be observing the impact of the various fixes we propose on this ratio: the closer it gets to 1.0, the better the fix should be; this will be verified by the value in the following column four, which shows the BLEU score on the test dataset calculated using multi-bleu.

4.3.1 Testing Possible Reason 1: The Optimization Mechanism of PRO

Table 1 explores possible reason 1 from Section 3, i.e., that there may be a bias in the optimization mechanism of PRO. It compares the original PRO to (a) PRO with no threshold for the selection step (default: filter pairs whose difference in BLEU+1 is less than a threshold: $\alpha = 0.05$), (b) PRO with random acceptance for the acceptance step (default: accept the pairs with the highest absolute difference in BLEU+1), and (c) combination of (a) and (b).

We can see in Table 1 that the length ratio in column three stays relatively stable for the different versions of PRO for all testing datasets. Note, however, the notable exception of IWSLT for (c), which exhibits a large increase in the length ratio: from 0.958 to 0.967.

[8] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

[9] Note that the results for the tuning dataset are obtained not using the BLEU score for the last iteration of the respective optimizer, but by translating the development set one more time – using the weights that the tuning algorithm has found. This is because, for some of the optimizers, the final tuning weight calculation might use weights from previous iterations, e.g., PRO accumulates and interpolates weights, while our version of MIRA, Batch-MIRA (Cherry and Foster, 2012), returns the weights from the iteration that yielded the highest BLEU score (using this option was suggested by the authors of Batch-MIRA; we found that it also worked best for our datasets).

Method	Tune: multi-bleu BLEU	Tune: multi-bleu len ratio	Test: multi-bleu BLEU	Test: multi-bleu len ratio	Test: NIST v13a BLEU	Test: NIST v13a len ratio
NIST dataset (tune: MT06, eval: MT09)						
PRO	45.65	**0.981**	48.45	0.976	47.80	0.978
PRO, no threshold	45.62	**0.981**	$48.39_{(-0.06)}$	0.975	47.74	0.977
PRO, accept random	45.55	**0.979**	$48.11_{(-0.33)}$	0.973	47.48	0.976
PRO, no threshold, accept random	45.58	**0.981**	$48.23_{(-0.22)}$	0.970	47.59	0.973
IWSLT dataset (tune: dev2010, eval: tst2010)						
PRO	26.96	**0.958**	26.34	0.965	25.30	0.973
PRO, no threshold	26.94	**0.959**	$26.24_{(-0.10)}$	0.966	25.18	0.975
PRO, accept random	26.97	**0.958**	$26.29_{(-0.05)}$	0.965	25.26	0.973
PRO, no threshold, accept random	26.87	**0.967**	$26.29_{(-0.05)}$	0.975	25.24	0.983
IWSLT dataset – reversed (tune: tst2010, eval: dev2010)						
PRO	26.08	**0.952**	26.45	0.945	25.43	0.946
PRO, no threshold	26.05	**0.952**	$26.51_{(+0.06)}$	0.945	25.49	0.947
PRO, accept random	26.11	**0.953**	$26.48_{(+0.03)}$	0.946	25.46	0.949
PRO, no threshold, accept random	26.06	**0.954**	$26.36_{(-0.09)}$	0.946	25.35	0.948

Table 1: Testing possible reason 1: bias in the optimization mechanism of PRO.

These results are inconclusive but they show that, at least for one dataset, the length bias in PRO was influenced by its selection and acceptance steps.[10] Overall, multi-bleu in column 4 stays stable, with a slight degradation in some cases, which is not surprising and can be attributed to PRO training its classifier on less reliable examples in these cases – ones with smaller BLEU+1 differentials, which risks focusing on tiny, unimportant distinctions.

4.3.2 Testing possible Reason 2: Bias in the Smoothing of BLEU+1

Table 2 explores possible reason 2 from Section 3, i.e., that the length issue may be caused by the smoothing in BLEU+1; see also Section 3.2 for more detail. It compares the results for the original PRO to a version (a) without smoothing of the precision component, (b) with grounding of the precision component, (c) with smoothing of the brevity penalty, and (d) with both BP-smoothing and PC-grounding.

Several interesting observations can be made about this table. First, we can see that using unsmoothed BLEU instead of BLEU+1 yields consistent improvements of the length ratio for all datasets; it also improves multi-bleu and NIST v13a scores. This suggests that the smoothing in BLEU+1 could indeed be one of the causes for the length problem. A similar trend can be seen for grounded BLEU+1, which performs slightly better, both in terms of length ratio and in terms of multi-bleu and NIST v13a scores. This suggests that part of the problem could be the balance between the precision component and the brevity penalty in BLEU+1: grounding reduces the absolute value of the precision component, which was inflated by BLEU+1, and thus helps restore a balance between the two components that is closer to that in BLEU. Moreover, grounded BLEU+1 has the advantage of assigning a non-zero value to any sentence with at least one unigram match, while unsmoothed BLEU assigns a zero score to many sentences, which could leave PRO with less training examples.

[10] However, there could be other reasons, e.g., the kind of classifier PRO uses, the fact that it samples from all previous steps, etc.; exploring these possibilities is left for future work.

Method	Tune: multi-bleu BLEU	len ratio	Test: multi-bleu BLEU	len ratio	Test: NIST v13a BLEU	len ratio
NIST dataset (tune: MT06, eval: MT09)						
PRO	45.65	**0.981**	48.45	0.976	47.80	0.978
PRO, PC-unsmoothed	45.83	**0.987**	$48.58_{(+0.13)}$	0.985	47.87	0.986
PRO, PC-grounded	45.82	**0.986**	$48.62_{(+0.17)}$	0.985	47.96	0.986
PRO, BP-smoothed	45.76	**0.988**	$48.65_{(+0.20)}$	0.985	48.03	0.988
PRO, BP-smoothed, PC-grounded	45.70	**0.992**	$48.79_{(+0.34)}$	0.991	48.16	0.993
IWSLT dataset (tune: dev2010, eval: tst2010)						
PRO	26.96	**0.958**	26.34	0.965	25.30	0.973
PRO, PC-unsmoothed	27.05	**0.970**	$26.46_{(+0.12)}$	0.979	25.38	0.987
PRO, PC-grounded	27.13	**0.975**	$26.48_{(+0.14)}$	0.984	25.39	0.992
PRO, BP-smoothed	27.10	**0.987**	$26.38_{(+0.04)}$	0.995	25.21	1.002
PRO, BP-smoothed, PC-grounded	27.16	**0.996**	$26.33_{(-0.01)}$	1.004	25.04	1.011
IWSLT dataset – reversed (tune: tst2010, eval: dev2010)						
PRO	26.08	**0.952**	26.45	0.945	25.43	0.946
PRO, PC-unsmoothed	26.09	**0.953**	$26.49_{(+0.04)}$	0.946	25.47	0.947
PRO, PC-grounded	26.24	**0.967**	$26.75_{(+0.30)}$	0.959	25.71	0.960
PRO, BP-smoothed	26.38	**0.991**	$27.10_{(+0.65)}$	0.982	26.07	0.983
PRO, BP-smoothed, PC-grounded	26.41	**1.003**	$27.07_{(+0.62)}$	0.995	26.04	0.994

Table 2: Testing possible reason 2: bias in the smoothing of BLEU+1.

Using add-one smoothing for the brevity penalty of BLEU+1 yields even better results on NIST and IWSLT-reversed: for IWSLT-reversed, the length ratio jumps from 0.952 to 0.991, and multi-bleu on the test dataset gains +0.65 absolute. The improvements in the length ratio are comparable for IWSLT-forward (from 0.958 to 0.987), and are more modest for NIST (from 0.981 to 0.988), where there is less room for improvement because of the higher baseline and because of the multiple reference translations. Overall, BP-smoothing works better than grounding, which suggests that smoothing with add-one both the precision component and the brevity penalty is a good way to restore the balance between them.

Finally, the combination of BP-smoothing and grounding yields even further improvements in the length ratios: 0.992 for NIST, 0.996 for IWSLT, and 1.003 for IWSLT-reversed. This adds +0.14 additional multi-bleu points to NIST, but slightly hurts IWSLT in both directions. This suggests that while grounding helps improve the balance between the precision component and ultimately the brevity penalty even further, it does this at the cost of deteriorating the estimates for the precision component, and thus the result in terms of multi-bleu is mixed.

Overall, as the length ratio gets closer to 1 on tuning, it does so on testing as well; this typically also yields an improvement in both multi-bleu and NIST v13a – we have achieved absolute multi-bleu improvements of up to +0.34 for NIST and +0.65 for IWSLT-reversed.

4.3.3 Testing Possible Reason 3: Sentence-Level Optimization

Table 3 explores possible reason 3 from Section 3, i.e., that the length issue may be due to sentence-level optimization. It compares the results for the original PRO to a version that (a) scales the reference length with the inverse document-level length ratio from the previous iteration of PRO, (b) unclips the brevity penalty, and (c) a combination thereof.

Method	Tune: multi-bleu BLEU	len ratio	Test: multi-bleu BLEU	len ratio	Test: NIST v13a BLEU	len ratio
NIST dataset (tune: MT06, eval: MT09)						
PRO	45.65	**0.981**	48.45	0.976	47.80	0.978
PRO, scaled	45.72	**0.984**	$48.50_{(+0.05)}$	0.979	47.87	0.981
PRO, unclipped	45.78	**0.992**	$48.81_{(+0.36)}$	0.990	48.18	0.992
PRO, unclipped, scaled	45.83	**0.993**	$48.82_{(+0.37)}$	0.991	48.19	0.993
IWSLT dataset (tune: dev2010, eval: tst2010)						
PRO	26.96	**0.958**	26.34	0.965	25.30	0.973
PRO, scaled	27.07	**0.975**	$26.39_{(+0.05)}$	0.984	25.31	0.991
PRO, unclipped	26.54	**1.024**	$25.63_{(-0.71)}$	1.035	24.34	1.040
PRO, unclipped, scaled	26.38	**1.030**	$25.43_{(-0.92)}$	1.042	24.14	1.046
IWSLT dataset – reversed (tune: tst2010, eval: dev2010)						
PRO	26.08	**0.952**	26.45	0.945	25.43	0.946
PRO, scaled	26.29	**0.972**	$26.75_{(+0.30)}$	0.962	25.73	0.963
PRO, unclipped	25.65	**1.033**	$26.40_{(-0.05)}$	1.021	25.34	1.021
PRO, unclipped, scaled	25.53	**1.037**	$26.09_{(-0.36)}$	1.028	25.02	1.028

Table 3: Testing possible reason 3: sentence-level optimization.

We can see that scaling does a great job at improving the length ratio, especially for IWSLT (in both directions), where it improves from 0.958/0.952 to 0.975/0.972; the effect is more limited for NIST – from 0.981 to 0.984, probably because of the higher baseline and due to the multiple reference translations. Scaling also yields small but consistent improvements in the multi-bleu/NIST-v13a scores on the tuning and the test datasets: up to +0.30 multi-bleu for IWSLT-reversed (but only +0.05 on the other two datasets).

We can further see that unclipping the brevity penalty in BLEU+1 yields a much larger increase in the length ratio, but this has mixed effect on the multi-bleu score: for NIST, the length ratio improves from 0.981 to 0.992, which yields a gain of +0.36 multi-bleu points absolute, but for IWSLT, the ratio jumps over 1, which yields a drop in the multi-bleu score.

The combination of unclipping and scaling yields an even higher length ratio, which helps NIST a bit, but hurts IWSLT even further as its length increases even more over 1.

Next, we experimented with sentence-level vs. corpus-level BLEU for PRO. The results are shown in Table 4. We can see that switching to corpus-level BLEU (calculated with respect to a pseudo-document, similarly to MIRA; see Section 3) yields a perfect length ratio of 1 on all tuning datasets, which is on par with the length ratios of MIRA and MERT, which also optimize corpus-level BLEU. This also improves the tuning multi-bleu for all datasets, and the testing multi-bleu by +0.14 for NIST and by +0.29 for IWSLT-reverse; however, there is a drop of -0.14 for IWSLT-forward because of a test length ratio that goes over 1 – due to a difference in the source/target length ratios between the tuning and the test sets for IWSLT.

We further tested the impact of doing the reverse: plugging a sentence-level BLEU+1 score as an objective in a corpus-level optimizer that generally has no length bias – MIRA. The results are shown in Table 4. We can see that MIRA with sentence-level BLEU+1 yields short reference translations just like PRO. This suggests that optimizing for sentence-level BLEU+1 yields short translations, regardless of the optimizer that is being used.

Method	Tune: multi-bleu BLEU	len ratio	Test: multi-bleu BLEU	len ratio	Test: NIST v13a BLEU	len ratio	*Avg. sent-BLEU tune*	*test*
NIST dataset (tune: MT06, eval: MT09)								
PRO	45.65	**0.981**	48.45	0.976	47.80	0.978	*46.20*	*47.75*
MIRA, sent-BLEU+1	45.51	**0.986**	48.18	0.983	47.52	0.985	*46.07*	*47.40*
PRO, corpus-BLEU	45.83	**1.000**	48.59	0.999	47.89	1.001	*45.83*	*47.59*
MIRA	45.26	**1.006**	48.26	1.006	47.48	1.009	*45.52*	*47.36*
MERT	45.58	**0.997**	48.48	0.997	47.72	0.998	*45.56*	*47.44*
IWSLT dataset (tune: dev2010, eval: tst2010)								
PRO	26.96	**0.958**	26.34	0.965	25.30	0.973	*32.03*	*31.43*
MIRA, sent-BLEU+1	26.93	**0.983**	25.81	0.994	24.62	0.999	*31.64*	*30.69*
PRO, corpus-BLEU	27.06	**1.000**	26.20	1.011	24.90	1.017	*31.50*	*30.89*
MIRA	27.28	**1.002**	26.09	1.012	24.78	1.017	*31.62*	*30.85*
MERT	27.24	**1.005**	25.84	1.017	24.53	1.020	*31.63*	*30.68*
IWSLT dataset – reversed (tune: tst2010, eval: dev2010)								
PRO	26.08	**0.952**	26.45	0.945	25.43	0.946	*31.35*	*31.70*
MIRA, sent-BLEU+1	26.25	**0.974**	26.75	0.965	25.69	0.966	*31.37*	*31.97*
PRO, corpus-BLEU	26.15	**1.000**	26.74	0.989	25.72	0.990	*30.73*	*31.12*
MIRA	26.54	**0.998**	27.09	0.988	26.08	0.989	*31.16*	*31.71*
MERT	26.33	**1.003**	26.88	0.992	25.86	0.993	*30.94*	*31.43*

Table 4: Sentence-level vs. corpus-level BLEU: experiments with PRO, MIRA, and MERT.

Note, however, that the length ratio with sentence-level MIRA does not drop all the way down to that of PRO, which also uses sentence-level optimization; this suggests once again that part of the bias may be coming from the optimization mechanism of PRO (which we have explored above as possible reason 1). This difference is larger for IWSLT than for NIST, which means that this bias is less pronounced with multiple reference translations.

The last two columns in Table 4 show the average sentence-level BLEU score, calculated for the tuning and the testing datasets. We can see that, on the tuning dataset, sentence-level optimizers score consistently higher than corpus-based ones on sent-BLEU, but they are not very competitive on corpus-BLEU; this suggests a mismatch of objectives. Note, also the relatively big difference between corpus-BLEU (i.e., multi-bleu – columns 2 and 4) and sent-BLEU (which uses no smoothing – columns 8 and 9) for IWSLT (about 4-5 BLEU points absolute), and the much smaller difference for NIST (0-1 BLEU points). One possible explanation is that with multiple references, average sent-BLEU approximates corpus-BLEU better; this is yet another possible reason for the length issue with PRO being less pronounced with NIST compared to IWSLT.

5 Discussion

Our experiments suggest that the length issue of PRO is due primarily to optimizing for sentence-level BLEU+1, which (i) cannot "see" the global length ratio, and (ii) uses a biased smoothing in BLEU+1, which boosts the precision part of the score, but leaves the brevity penalty intact, thus destroying the balance between the two. This is confirmed by an experiment where we plugged sentence-level BLEU+1 in MIRA. We have also found that part of the bias may be coming from the optimization mechanism of PRO.

We have proposed a number of ways to fix the length ratio of PRO by fixing BLEU+1, some of which have worked fairly well, e.g., smoothing the brevity penalty and scaling the effective reference length. Other fixes such as grounding the precision component and unclipping the brevity penalty were less helpful. Moreover, some combinations thereof have yielded too long length ratios, which had a negative impact on multi-bleu for IWSLT, but not for NIST.

Given our experimental results, we cannot make a universal recommendation about which fixes would work best for all datasets, but, in practical terms, we would recommend trying different fixes and choosing the one for which the length ratio is closest to 1 on the tuning dataset. Following this advice for Tables 2 and 3, we would select BLEU+1 with unclipped BP and scaled reference length for NIST (tuning length ratio of 0.993), and BLEU+1 with BP-smoothing and PC-grounding for IWSLT (tuning length ratio of 1.003), yielding +0.37 multi-bleu for NIST and +0.62 for IWSLT-reversed on the test dataset.

Coming back to our title: optimizing for sentence-level BLEU+1 yields short translations. This is true for the tuning dataset that the optimizer sees, but we can make no claims about the lengths of the test-time translations. This is because they can differ from the tuning set a lot in terms of verbosity, and the optimizer has no control over this.[11] However, if the tuning and the testing datasets have similar source/reference ratios (which is the best guess in the absence of other information), we expect that the claim would also hold for the test set.[12]

6 Conclusion and Future Work

We hope that the present study will be useful for those who want to use optimizers such as PRO and MIRA, which can handle many features, but feel frustrated that certain idiosyncrasies cause MERT to still yield the best test BLEU scores in certain cases. Here we have pointed out that one cause for this are poor length ratios in the resulting translations, which we have attributed to the use of sentence-level BLEU+1 as an objective function. We have thus suggested a number of simple modifications, which do improve the length ratio in practice, ultimately yielding better BLEU scores, while also preserving the sentence-level nature of BLEU+1, which makes optimizers simpler conceptually and implementation-wise.

In future work, we plan a more systematic study of the relationship between optimizers and objective functions with respect to the target/reference length ratio, which would be extended with other optimizers such as TER (Snover et al., 2006) and METEOR (Lavie and Denkowski, 2009). Overall, we see two promising general directions in which the present study can be extended. First, explore the relationship between sentence-level and corpus-level optimization and the possibility to combine them. Second, study the characteristics of translations generated using weights from different optimizers: while here we have only touched length, we believe there are many other important aspects that are worth exploring.

Acknowledgments

We thank the anonymous reviewers for their comments, which helped us improve the paper.

[11] For example, the average source/reference length ratio for our NIST tuning dataset MT06 is 1.248, which is very close to that for MT09, which is 1.252, and thus translation/reference length ratios are very close as well; however, this is not so for MT05, where the source/reference ratio is only 1.183; thus, tuning on MT06 and testing on MT05 yields translations that are too long even for standard PRO. There is also some imbalance in the source/reference length ratios of dev2010 vs. tst2010 for IWSLT, and thus we experimented with reversing them.

[12] The internal scoring tool segmentation and the recasing have an influence as well; though, we have seen in columns 6-7 of all tables that (1) the length ratios are quite close, and (2) the relative improvements in terms of BLEU are quite correlated for multi-bleu and for the NIST scoring tool v13a.

References

Bazrafshan, M., Chung, T., and Gildea, D. (2012). Tuning as linear regression. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, pages 543–547, Montréal, Canada.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, pages 427–436, Montréal, Canada.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '09, pages 218–226, Boulder, CO, USA.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 224–233, Honolulu, Hawaii, USA.

Federico, M., Stüker, S., Bentivogli, L., Paul, M., Cettolo, M., Herrmann, T., Niehues, J., and Moretti, G. (2012). The IWSLT 2011 evaluation campaign on automatic talk translation. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation*, LREC '12, pages 3543–3550, Istanbul, Turkey.

Foster, G. and Kuhn, R. (2009). Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 242–249, Athens, Greece.

Gimpel, K. and Smith, N. A. (2012). Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, pages 221–231, Montréal, Canada.

Green, S. and DeNero, J. (2012). A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '12, pages 146–155, Jeju Island, Korea.

He, X. and Deng, L. (2012). Maximum expected BLEU training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '12, pages 292–301, Jeju Island, Korea.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Edinburgh, Scotland, United Kingdom.

Koehn, P., Axelrod, A., Mayne, A. B., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*, IWSLT '05, Pittsburgh, PA, USA.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (Demonstration session)*., ACL '07, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (Volume 1)*, HLT-NAACL '03, pages 48–54, Edmonton, Canada.

Lavie, A. and Denkowski, M. J. (2009). The Meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23:105–115.

Lin, C.-Y. and Och, F. J. (2004). ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, pages 501–507, Geneva, Switzerland.

McAllester, D. and Keshet, J. (2011). Generalization bounds and consistency for latent structural probit and ramp loss. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, NIPS '11, pages 2205–2212, Granada, Spain.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, ACL '03, pages 160–167, Sapporo, Japan.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, PA, USA.

Roth, R., Rambow, O., Habash, N., Diab, M., and Rudin, C. (2008). Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL '08, pages 117–120, Columbus, OH, USA.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, pages 223–231, Cambridge, MA, USA.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 764–773, Prague, Czech Republic.