Regularized Minimum Error Rate Training

Michel Galley
Microsoft Research
mgalley@microsoft.com

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Colin Cherry
National Research Council
colin.cherry@nrc-cnrc.gc.ca

Kristina Toutanova
Microsoft Research
kristout@microsoft.com

Abstract

Minimum Error Rate Training (MERT) remains one of the preferred methods for tuning linear parameters in machine translation systems, yet it faces significant issues. First, MERT is an unregularized learner and is therefore prone to overfitting. Second, it is commonly used on a noisy, non-convex loss function that becomes more difficult to optimize as the number of parameters increases. To address these issues, we study the addition of a regularization term to the MERT objective function. Since standard regularizers such as ℓ_2 are inapplicable to MERT due to the scale invariance of its objective function, we turn to two regularizers—ℓ_0 and a modification of ℓ_2—and present methods for efficiently integrating them during search. To improve search in large parameter spaces, we also present a new direction finding algorithm that uses the gradient of expected BLEU to orient MERT's exact line searches. Experiments with up to 3600 features show that these extensions of MERT yield results comparable to PRO, a learner often used with large feature sets.

1 Introduction

Minimum Error Rate Training emerged a decade ago (Och, 2003) as a superior training method for small numbers of linear model parameters of machine translation systems, improving over prior work using maximum likelihood criteria (Och and Ney, 2002). This technique quickly rose to prominence, becoming standard in many research and commercial MT systems. Variants operating over lattices (Macherey et al., 2008) or hypergraphs (Kumar et al., 2009) were subsequently developed, with the benefit of reducing the approximation error from n-best lists.

The primary advantages of MERT are twofold. It directly optimizes the evaluation metric under consideration (e.g., BLEU) instead of some surrogate loss. Secondly, it offers a globally optimal line search. Unfortunately, there are several potential difficulties in scaling MERT to larger numbers of features, due to its non-convex loss function and its lack of regularization. These challenges have prompted some researchers to move away from MERT, in favor of linearly decomposable approximations of the evaluation metric (Chiang et al., 2009; Hopkins and May, 2011; Cherry and Foster, 2012), which correspond to easier optimization problems and which naturally incorporate regularization. In particular, recent work (Chiang et al., 2009) has shown that adding thousands or tens of thousands of features can improve MT quality when weights are optimized using a margin-based approximation. On simulated datasets, Hopkins and May (2011) found that conventional MERT struggles to find reasonable parameter vectors, where a smooth loss function based on Pairwise Ranking Optimization (PRO) performs much better; on real data, this PRO method appears at least as good as MERT on small feature sets, and also scales better as the number of features increases.

In this paper, we seek to preserve the advantages of MERT while addressing its shortcomings in terms of regularization and search. The idea of adding a regularization term to the MERT objective function can be perplexing at first, because the most common regularizers, such as ℓ_1 and ℓ_2, are not directly applicable to MERT. Indeed, these regularizers are *scale sensitive*, while the MERT objective function is not: scaling the weight vector neither changes the predictions of the linear model nor affects the error count. Hence, MERT can hedge any regularization penalty by maximally scaling down linear model weights.

The first contribution of this paper is to analyze various forms of regularization that are not susceptible to this scaling problem. We analyze and experiment with ℓ_0, a form of regularization that is *scale insensitive*. We also present new parameterizations of ℓ_2

regularization, where we apply ℓ_2 regularization to scale-senstive linear transforms of the original linear model. In addition, we introduce efficient methods of incorporating regularization in Och (2003)'s exact line searches. For all of these regularizers, our methods let us find the true optimum of the regularized objective function along the line.

Finally, we address the issue of searching in a high-dimensional space by using the gradient of expected BLEU (Smith and Eisner, 2006) to find better search directions for our line searches. This direction finder addresses one of the serious concerns raised by Hopkins and May (2011): MERT widely failed to reach the optimum of a synthetic linear objective function. In replicating Hopkins and May's experiments, we confirm that existing search algorithms for MERT—including coordinate ascent, Powell's algorithm (Powell, 1964), and random direction sets (Cer et al., 2008)—perform poorly in this experimental condition. However, when using our gradient-based direction finder, MERT has no problem finding the true optimum even in a 1000-dimensional space.

Our results suggest that the combination of a regularized objective function and a gradient-informed line search algorithm enables MERT to scale well with a large number of features. Experiments with up to 3600 features show that these extensions of MERT yield results comparable to PRO (Hopkins and May, 2011), a parameter tuning method known to be effective with large feature sets.

2 Unregularized MERT

Prior to introducing regularized MERT, we briefly review standard unregularized MERT (Och, 2003). We use $\mathbf{f}_1^S = \{\mathbf{f}_1 \dots \mathbf{f}_S\}$ to denote the S input sentences of a given tuning set. For each sentence $\mathbf{f}_s$, let $\mathbf{C}_s = \{\mathbf{e}_{s,1} \dots \mathbf{e}_{s,M}\}$ denote the list of M-best candidate translations. Each input and output sentence pair $(\mathbf{f}_s, \mathbf{e}_{s,m})$ is weighted using a linear model that applies model parameters $\mathbf{w} = (w_1 \dots w_D) \in \mathbb{R}^D$ to D feature functions $h_1(\mathbf{f}, \mathbf{e}, \sim) \dots h_D(\mathbf{f}, \mathbf{e}, \sim)$, where $\sim$ is the hidden state associated with the derivation from $\mathbf{f}$ to $\mathbf{e}$, such as phrase segmentation and alignment. Furthermore, let $\mathbf{h}_{s,m} \in \mathbb{R}^D$ denote the feature vector representing the translation pair $(\mathbf{f}_s, \mathbf{e}_{s,m})$.

In MERT, the goal is to minimize a loss function $E(\mathbf{r}, \mathbf{e})$ that scores translation hypotheses against a set of reference translations $\mathbf{r}_1^S = \{\mathbf{r}_1 \dots \mathbf{r}_S\}$. This yields the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} \sum_{m=1}^{M} E(\mathbf{r}_s, \mathbf{e}_{s,m}) \delta(\mathbf{e}_{s,m}, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\} \quad (1)$$

where

$$\hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w}) = \arg\max_{m \in \{1 \dots M\}} \left\{ \mathbf{w}^\intercal \mathbf{h}_{s,m} \right\} \quad (2)$$

While the error surface of Equation 1 is only an approximation of the true error surface of the MT decoder, the quality of this approximation depends on the size of the hypothesis space represented by the M-best list. Therefore, the hypothesis list is grown iteratively: decoding with an initial parameter vector seeds the M-best lists; next, parameter estimation and M-best list gathering alternate until the cumulative M-best list no longer grows, or until changes of $\mathbf{w}$ between two decoding runs are deemed too small. To increase the size of the hypothesis space, subsequent work (Macherey et al., 2008) instead operated on lattices, but this paper focuses on M-best lists.

A crucial observation is that the unsmoothed error count represented in Equation 1 is a piecewise constant function. This enabled Och (2003) to devise a line search algorithm guaranteed to find the optimum point along the line. To extend the search from one to multiple dimensions, MERT applies a sequence of line optimizations along some fixed or variable set of search directions $\{\mathbf{d}_t\}$ until some convergence criteria are met. Considering a given point $\mathbf{w}_t$ and a given direction $\mathbf{d}_t$ at iteration t, finding the most probable translation hypothesis in the set of candidates translations $\mathbf{C}_s = \{\mathbf{e}_{s,1} \dots \mathbf{e}_{s,M}\}$ corresponds to solving the following optimization problem:

$$\hat{\mathbf{e}}(\mathbf{f}_s; \gamma) = \arg\max_{m \in \{1 \dots M\}} \left\{ (\mathbf{w}_t + \gamma \cdot \mathbf{d}_t)^\intercal \mathbf{h}_{s,m} \right\} \quad (3)$$

The function in this equation is piecewise linear (Papineni, 1999), which enables an efficient exhaustive computation. Specifically, this function is optimized by enumerating the up to M hypotheses that form the *upper envelope* of the model score function. The error count, then, is a piecewise constant function

defined by the points $\gamma_1^{\mathbf{f}_s} < \cdots < \gamma_M^{\mathbf{f}_s}$ at which an increase in γ causes a change of optimum in Equation 3. Error counts for the whole corpus are simply the sums of sentence-level piecewise constant functions aggregated over all sentences of the corpus.[1] The optimal γ is finally computed by enumerating all piecewise constant intervals of the corpus-level error function, and by selecting the one that has the lowest error count (or, correspondingly, highest BLEU score). Assuming the optimum is found in the interval $[\gamma_{k-1}, \gamma_k]$, we define $\gamma_{\text{opt}} = (\gamma_{k-1} + \gamma_k)/2$ and change the parameters using the update $\mathbf{w}_{t+1} = \mathbf{w}_t + \gamma_{\text{opt}} \cdot \mathbf{d}_t$.

Finally, this method is turned into a global D-dimensional search using algorithms that repeatedly use the aforementioned exact line search algorithm. Och (2003) first advocated the use of Powell's method (Powell, 1964; Press et al., 2007). Pharaoh (Koehn, 2004) and subsequently Moses (Koehn et al., 2007) instead use coordinate ascent, and more recent work often uses random search directions (Cer et al., 2008; Macherey et al., 2008). In Section 4, we will present a novel direction finder for maximum-BLEU optimization, which uses the gradient of expected BLEU to find directions where the BLEU score is most likely to increase.

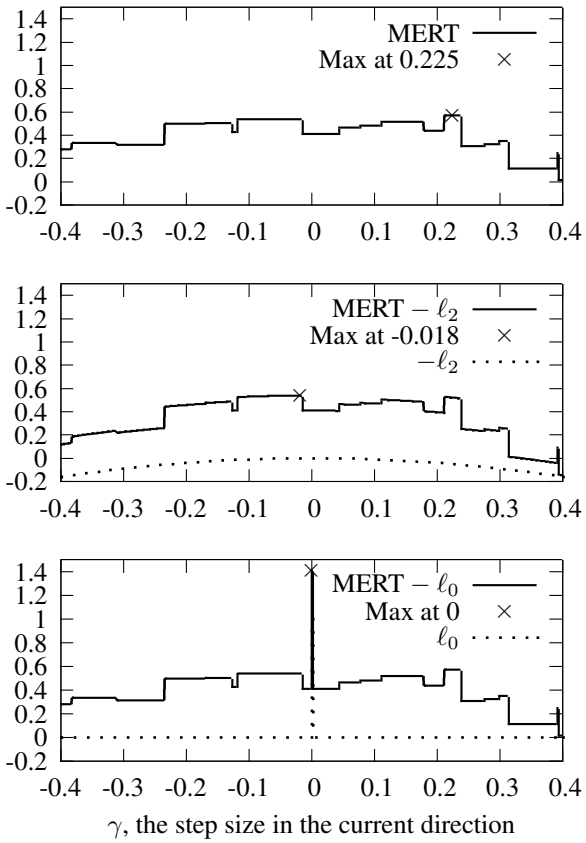

Figure 1: Example MERT values along one coordinate, first unregularized. When regularized with ℓ_2, the piecewise constant function becomes piecewise quadratic. When using ℓ_0, the function remains piecewise constant with a point discontinuity at 0.

3 Regularization for MERT

Because MERT is prone to overfitting when a large number of parameters must be optimized, we study the addition of a regularization term to the objective function. One conventional approach is to regularize the objective function with a penalty based on the Euclidean norm $||\mathbf{w}||_2 = \sqrt{\sum_i w_i^2}$, also known as ℓ_2 regularization. In the case of MERT, this yields the following objective function:[2]

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) + \frac{||\mathbf{w}||_2^2}{2\sigma^2} \right\} \quad (4)$$

where the regularization term $1/2\sigma^2$ is a free parameter that controls the strength of the regularization penalty. Similar regularizers have also been used in conjunction with other norms, such as ℓ_1 and ℓ_0 norms. The ℓ_1 norm, defined as $||\mathbf{w}||_1 = \sum_i |w_i|$, applies a constant force toward zero, preferring vectors with fewer non-zero components; ℓ_0, defined as $||\mathbf{w}||_0 = |\{i \mid w_i \neq 0\}|$, simply counts the number of non-zero components of the weight vector, encoding a preference for sparse vectors.

Geometrically, ℓ_2 is a parabola, ℓ_1 is the wedge-shaped absolute value function, and ℓ_0 is an impulse function with a spike at 0. The original formulation (Equation 1) of MERT consists of a piecewise constant representation of the loss, as a function of the step size in a given direction. But with these three reg-

[1] This assumes that the sufficient statistics of the metric under consideration are additively decomposable by sentence, which is the case with most popular evaluation metrics such as BLEU (Papineni et al., 2001).

[2] The ℓ_2 regularizer is often used in conjunction with log-likelihood objectives. The regularization term of Equation 4 could similarly be added to the log of an objective—e.g., log(BLEU) instead of BLEU—but we found that the distinction doesn't have much of an impact in practice.

ularization terms, the function respectively becomes piecewise quadratic, piecewise linear, or piecewise constant with a potential impulse jump for each distinct choice of regularizer. Figure 1 demonstrates this effect graphically.

As discussed in (McAllester and Keshet, 2011), the problem with optimizing Equation 4 directly is that the output of the underlying linear classifier, and therefore the error count, are not sensitive to the scale of $\mathbf{w}$. Moreover, ℓ_2 regularization (as well as ℓ_1 regularization) is scale sensitive, which means any optimizer of this function can drive the regularization term down to zero by scaling down $\mathbf{w}$. As special treatments for ℓ_2, we evaluate three linear transforms of the weight vector, where the vector $\mathbf{w}$ of the regularization term $||\mathbf{w}||_2^2/2\sigma^2$ is replaced with either:

1. an affine transform: $\mathbf{w} - \mathbf{w}_0$
2. a vector with only $(D-1)$ free parameters, e.g., $(1, w'_2, \cdots, w'_D)$
3. an ℓ_1 renormalization: $\mathbf{w}/||\mathbf{w}||_1$

In (1), regularization is biased towards $\mathbf{w}_0$, a weight vector previously optimized using a competitive yet much smaller feature set, such as core features of a phrase-based (Koehn et al., 2007) or hierarchical (Chiang, 2007) system. The requirement that this feature set be small is to prevent overfitting. Otherwise, any regularization toward an overfit parameter vector $\mathbf{w}_0$ would defeat the purpose of introducing a regularization term in the first place.[3] In (2), the transformation is motivated by the observation that the D-parameter linear model of Equation 2 only needs $(D-1)$ degrees of freedom. Fixing one of the components of $\mathbf{w}$ to any non-zero constant and allowing the others to vary, the new linear model retains the same modeling power, but the $(D-1)$ free parameters are no longer scale invariant, i.e., scaling the $(D-1)$-dimensional vector now has an effect on linear model predictions. In (3), the weight vector is normalized as to have an ℓ_1-norm equal to 1. In contrast, the ℓ_0 norm is scale insensitive, thus not affected by this problem.

3.1 Exact line search with regularization

Optimizing with a regularized error surface requires a change in the line search algorithm presented in Section 2, but the other aspects of MERT remain the same, and we can still use global search algorithms such as coordinate ascent, Powell, and random directions exactly the same way as with unregularized MERT. Line search with a regularization term is still as efficient as in (Och, 2003), and it is still guaranteed to find the optimum of the (now regularized) objective function along the line. Considering again a given point $\mathbf{w}_t$ and a given direction $\mathbf{d}_t$ at line search iteration t, finding the optimum γ_{opt} corresponds to finding γ that minimizes:

$$\sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \gamma)) + \frac{||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2}{2\sigma^2} \qquad (5)$$

Since regularization does not affect the points at which $\hat{\mathbf{e}}(\mathbf{f}_s; \gamma)$ changes its optimum, the points $\gamma_1^{\mathbf{f}_s} < \cdots < \gamma_M^{\mathbf{f}_s}$ of intersection in the upper envelope remain the same, so the points of discontinuity in the error surface remain the same. The difference now is that the error count on each segment $[\gamma_{i-1}, \gamma_i]$ is no longer constant. This means we need to adjust the final step of line search, which consists of enumerating all $[\gamma_{i-1}, \gamma_i]$, and keeping the optimum of Equation 5 for each segment. $\hat{\mathbf{e}}(\mathbf{f}_s; \gamma)$ remains constant within the segment, so we only need to consider the expression $||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2$ to select a segment point. The optimum is either at the left edge, the right edge, or in the middle if the vertex of the parabola happens to lie within that segment.[4] We compute this optimum by finding the value γ for which the derivative of the regularization term is zero. There is an easy closed-form solution:

$$\frac{\mathrm{d}}{\mathrm{d}\gamma}\left[\frac{||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2}{2\sigma^2}\right] = 0$$

$$\frac{\mathrm{d}}{\mathrm{d}\gamma}\left[\sum_i (w_{t,i}^2 + 2 \cdot \gamma \cdot w_{t,i} \cdot d_{t,i} + \gamma^2 \cdot d_{t,i}^2)\right] = 0$$

$$\sum_i (2 \cdot w_{t,i} \cdot d_{t,i} + 2 \cdot \gamma \cdot d_{t,i}^2) = 0$$

$$\gamma = -\left(\sum_i w_{t,i} \cdot d_{t,i}\right) / \left(\sum_i d_{t,i}^2\right) = -\frac{\mathbf{w}_t{}^\intercal \mathbf{d}_t}{\mathbf{d}_t{}^\intercal \mathbf{d}_t}$$

This closed-form solution is computed in time proportional to D, which doesn't slow down the com-

[3] (Gimpel and Smith, 2012, footnote 6) briefly mentions the use of such a regularizer with its ramp loss objective function.

[4] When the optimum is either at the left edge γ_{i-1} or right edge γ_i of a segment, we select a point at a small relative distance within the segment ($.999\gamma_{i-1} + .001\gamma_i$, in the former case) to avoid ties in objective values.

putation of Equation 5 for each segment (the construction of each segment of the upper envelope is proportional to D anyway).

We also use ℓ_0 regularization. While minimization of the ℓ_0-norm is known to be NP-hard in general (Hyder and Mahata, 2009), this optimization is relatively trivial in the case of a line search. Indeed, for a given segment, the value in Equation 5 is constant everywhere except where we intersect any of the coordinate hyperplanes, i.e., where one of the coordinates is zero. Thus, our method consists of evaluating Equation 5 at the intersection points between the line and coordinate hyperplanes, returning the optimal point within the given segment. For any segment that doesn't cross any of these hyperplanes, we evaluate the objective function at any point of the segment (since the value is constant across the entire segment).

4 Direction finding

4.1 A Gradient-based direction finder

Perhaps the greatest obstacle in scaling MERT to many dimensions is finding good search directions. In problems of lower dimensions, iterating through all the coordinates is computationally feasible, though not guaranteed to find a global maximum even in the case of a perfect line search. As the number of dimensions increases by orders of magnitude, this coordinate direction approach becomes less and less tractable, and the quality of the search also suffers (Hopkins and May, 2011).

Optimization has traditionally relied on finding the direction of steepest ascent: the gradient. Unfortunately, the objective function optimized by MERT is piecewise constant; while it may admit a subgradient, this direction is generally not very informative. Instead we may consider a smoothed variation of the original approximation. While some variants have been considered (Och, 2003; Flanigan et al., 2013), we use an expected BLEU approximation, assuming hypotheses are drawn from a log-linear distribution according to their parameter values (Smith and Eisner, 2006). That is, we assume the probability of a translation candidate $\mathbf{e}_{s,m}$ is proportional to $(\exp(\mathbf{w}^\intercal \mathbf{h}_{s,m}))^\mu$, where $\mathbf{w}$ are the parameters being optimized, $\mathbf{h}_{s,m}$ is the vector of the features for $\mathbf{e}_{s,m}$, and μ is a scaling parameter. As μ approaches infinity, the distribution places all its weight on the highest scoring candidate.

The log of the BLEU score may be written as:

$$\min\left(1-\frac{R}{C},0\right)+\frac{1}{N}\sum_{n=1}^{N}(\log m_n-\log c_n)$$

where R is the sum of reference lengths across the corpus, C is the sum of candidate lengths, m_n is the number of matched n-grams (potentially clipped), and c_n is the number of n-grams in all candidates.

Given a distribution over candidates, we can use the expected value of the log of the BLEU score. This is a smooth approximation to the BLEU score, which asymptotically approaches the true BLEU score as the scaling parameter μ approaches infinity. While this expectation is difficult to compute exactly, we can compute approximations thereof using Taylor series. Although prior work demonstrates that a second-order Taylor approximation is feasible to compute (Smith and Eisner, 2006), we find that a first-order approximation is faster and very close to the second-order approximation.[5] The first order Taylor approximation is as follows:

$$\min\left(1-\frac{R}{\mathbb{E}[C]},0\right)+\frac{1}{N}\sum_{n=1}^{N}(\log\mathbb{E}[m_n]-\log\mathbb{E}[c_n])$$

where $\mathbb{E}$ is the expectation operator using the probability distribution $P(\mathbf{h};\mathbf{w},\mu)$.

First we note that the gradient $\frac{\partial}{\partial w_i}P(\mathbf{h};\mathbf{w},\mu)$ is

$$P(\mathbf{h};\mathbf{w},\mu)\left(h_i-\sum_{\mathbf{h}'}h_i'P(\mathbf{h}';\mathbf{w},\mu)\right)$$

Using the chain rule, the gradient of the first order approximation to BLEU is as follows:

$$\frac{1}{N}\sum_{n=1}^{N}\Big(\frac{1}{\mathbb{E}[m_n]}\sum_{\mathbf{h}}m_n(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i} - \frac{1}{\mathbb{E}[c_n]}\sum_{\mathbf{h}}c_n(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i}\Big)$$

$$+\begin{cases}0 & \text{if } \mathbb{E}[C]>R\\ \frac{R}{\mathbb{E}[C]^2}\sum_{\mathbf{h}}c_1(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i} & \text{otherwise}\end{cases}$$

[5] Experimentally, we compared our analytical gradient of the first-order Taylor approximation with the finite-difference gradients of the first- and second-order approximations, and we found these three gradients to be very close in terms of cosine similarity (> 0.99). We performed these measurements both at arbitrary points and at points of convergence of MERT.

In the case of ℓ_2-regularized MERT, the final gradient also includes the partial derivative of the regularization penalty of Equation 4, which is w_i/σ^2 for a given component i of the gradient. We do not update the gradient in the case of ℓ_0 regularization since the ℓ_0-norm is not differentiable.

4.2 Search

Our search strategy consists of looking at the directions of steepest increase of expected BLEU, which is similar to that of Smith and Eisner (2006), but with the difference that we do so in the context of MERT. We think this difference provides two benefits. First, while the smooth approximation of BLEU reduces the likelihood of remaining trapped in a local optimum, we avoid approximation error by retaining the original objective function. Second, the benefit of exact line searches in MERT is that there is no need to be concerned about step size, since step size in MERT line searches is guaranteed to be optimal with respect to the direction under consideration.

Finally, our gradient-based search algorithm operates as follows. Considering the current point $\mathbf{w}_t$, we compute the gradient $\mathbf{g}_t$ of the first order Taylor approximation at that point, using the current scaling parameter μ. (We initialize the search with $\mu = 0.01$.) We find the optimum along the line $\mathbf{w}_t + \gamma \cdot \mathbf{g}_t$. Whenever any given line search yields no improvement larger than a small tolerance threshold, we multiply μ by two and perform a new line search. The increase of this parameter μ corresponds to a cooling schedule (Smith and Eisner, 2006), which progressively sharpens the objective function to get a better estimate of BLEU as the search converges to an optimum. We repeatedly perform new line searches until μ exceeds 1000. The inability to improve the current optimum with a sharp approximation ($\mu > 1000$) doesn't mean line searches would fail with smaller values, so we find it helpful to repeat the above procedure until a full pass of updates of μ from 0.01 to 1000 yields no improvement.

4.3 Computational complexity

Computing the gradient increases the computational cost of MERT, though not its asymptotic complexity. The cost of a single exhaustive line search is

$$\mathcal{O}\left(SM(D + \log M + \log S)\right)$$

where S is the number of sentences, each with M possible translations, and D is the number of features. For each sentence, we first identify the model score as a linear function of the step size, requiring two dot products for an overall cost of $\mathcal{O}(SMD)$.[6] Next we construct the upper envelope for each sentence: first the equations are sorted in increasing order of slope, and then they are merged in linear time to form an envelope, with an overall cost of $\mathcal{O}(SM \log M)$. A linear pass through the envelope converts these into piecewise constant (or linear, or quadratic) representations of the (regularized) loss function. Finally the per-sentence envelopes are merged into a global representation of the loss along that direction. Our implementation successively merges adjacent pairs of piecewise smooth loss function representations until a single list remains. These $\log S$ passes lead to a merging runtime of $\mathcal{O}(SM \log S)$.

The time required to compute a gradient is proportional to $\mathcal{O}(SMD)$. For each sentence, we first gather the probability and its gradient, then use this to compute expected n-gram counts and matches as well as those gradients in time $\mathcal{O}(MD)$. A constant number of arithmetic operations suffice to compute the final expected loss value and its gradient. Therefore, computing the gradient does not increase the algorithmic complexity when compared to conventional approaches using coordinate ascent and random directions. Likewise the runtime of a single iteration is competitive with PRO, given that gradient finding is generally the most expensive part of convex optimization. Of course, it is difficult to compare overall runtime of convex optimization with that of MERT, as we know of no way to bound the number of gradient evaluations required for convergence with MERT. Therefore, we resort to empirical comparison later in the paper, and find that the two methods appear to have comparable runtime.

[6] In the special case where the difference between the prior direction and the current direction is sparse, we may update the individual linear functions in time proportional to the number of changed dimensions. Coordinate ascent in particular can update the linear functions in time $\mathcal{O}(SM)$: to the intercept of the equation for each translation, we may add the prior step size multiplied by the feature value in the prior coordinate, and the slope becomes the feature value in the new coordinate. However, this optimization does not appear to be widely adopted, likely because it does not lead to any speedup when random vectors, conjugate directions, or other non-sparse directions are used.

	Language pair	Train	Tune	Dev	Test
GBM	Chinese-English	0.99M	1,797 (mt02+03)	1,000	1,082 (mt05)
	Finnish-English	2.20M	11,935	2,001	4,855
SparseHRM	Chinese-English	3.51M	1,894 (mt05)	1,664 (mt06)	1,357 (mt08)
	Arabic-English	1.49M	1,663 (mt06)	1,360 (mt08)	1,313 (mt09)

Table 1: Datasets for the two experimental conditions.

5 Experimental Design

Following Hopkins and May (2011), our experimental setup utilizes both real and synthetic data. The motivation for using synthetic data is that it is a way of gauging the quality of optimization methods, since the data is constructed knowing the global optimum. Hopkins and May also note that the use of an objective function that is linear in some gold weight vector makes the search much simpler than in a real translation setting, and they suggest that a learner that performs poorly in such a simple scenario has little hope of succeeding in a more complex one.

The setup of our synthetic data experiment is almost the same as that performed by Hopkins and May (2011). We generate feature vectors of dimensionality ranging from 10 to 1000. These features are generated by drawing random numbers uniformly in the interval $[0, 500]$. This synthetic dataset consists of S=1000 source "sentences", and M=500 "translation" hypotheses for each sentence. A pseudo "BLEU" score is then computed for each hypothesis, by computing the dot product between a predefined gold weight vector $\mathbf{w}^*$ and each feature vector $\mathbf{h}_{s,m}$. By this linear construction, $\mathbf{w}^*$ is guaranteed to be a global optimum.[7] The pseudo-BLEU score is normalized for each M-best list, so that the translation with highest model score according to $\mathbf{w}^*$ has a BLEU score of 1, and so that the translation with lowest model score for the sentence gets a BLEU of zero. This normalization has no impact on search, but makes results more interpretable.

For our translation experiments, we use multi-stack phrase-based decoding (Koehn et al., 2007). We report results for two feature sets: non-linear features induced using Gradient Boosting Machines (Toutanova and Ahn, 2013) and sparse lexicalized reordering features (Cherry, 2013). We exploit these feature sets (GBM and SparseHRM, respectively) in two distinct experimental conditions, which we detail in the two next paragraphs. Both GBM and SparseHRM augment baseline features similar to Moses': relative frequency and lexicalized phrase translation scores for both translation directions; one or two language model features, depending on the language pair; distortion penalty; word and phrase count; six lexicalized reordering features. For both experimental conditions, phrase tables have maximum phrase length of 7 words on either side. In reference to Table 1, we used the training set (Train) for extracting phrase tables and language models; the Tune set for optimization with MERT or PRO; the Dev set for selecting hyperparameters of PRO and regularized MERT; and the Test set for reporting final results. In each experimental condition, we first trained weights for the base feature sets, and then decoded the Tune, Dev, and Test datasets, generating 500-best lists for each set. All results report reranking performance on these lists with different feature sets and optimization methods, based on lower-cased BLEU (Papineni et al., 2001).

The GBM feature set (Toutanova and Ahn, 2013) consists of about 230 features automatically induced using decision tree weak learners, which derive features using various word-level, phrase-level, and morphological attributes. For Chinese-English, the training corpus consists of approximately one million sentence pairs from the FBIS and Hong Kong portions of the LDC data for the NIST MT evaluation and the Tune and Test sets are from NIST competitions. A 4-gram language model was trained on the Xinhua portion of the English Gigaword corpus and on the target side of the bitext. For Finnish-English we used a dataset from a technical domain of software manuals. For this language pair we used two language models: one very large model trained on billions of words, and another language model trained from the target side of the parallel training set.

The SparseHRM set (Cherry, 2013) contains 3600 sparse reordering features. For each phrase, the features take the form of indicators describing its orientation in the derivation, and its lexical content in terms of word clusters or frequent words. For both Chinese-English and Arabic-English, systems are trained on data from the NIST 2012 MT evaluation. 4-gram

[7] The objective function remains piecewise constant, and the plateau containing $\mathbf{w}^*$ maps to the optimal *value* of the function.

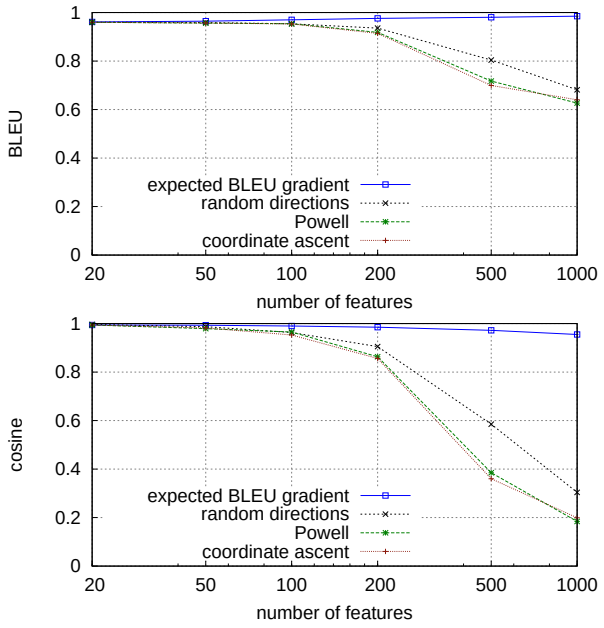

Figure 2: Change in BLEU score and cosine similarity to the gold weight vector $\mathbf{w}^*$ as the number of features increases, using the **noisy** synthetic experiments. The gradient-based direction finding method is barely affected by the noise. The increase of the number of dimensions enables our direction finder to find a slightly better optimum, which moved away from $\mathbf{w}^*$ due to noise.

language models were trained on the target side of the parallel training data for both Arabic and Chinese. The Chinese systems development set is taken from the NIST mt05 evaluation set, augmented with some material reserved from our NIST training corpora in order to better cover newsgroup and weblog domains.

6 Results

We conducted experiments with the synthetic data scenario described in the previous section, as well as with noise added to the data (Hopkins and May, 2011). The purpose of adding noise is to make the optimization task more realistic. Specifically, after computing all pseudo-BLEU scores, we added noise to each feature vector $\mathbf{h}_{s,m}$ by drawing from a zero-mean Gaussian with standard deviation 200. Our results with both noiseless and noisy data yield the same conclusion as Hopkins and May: standard MERT struggles with many dimensions, and fails to recover $\mathbf{w}^*$. However, our experiments with the gradient direction finder of Section 4 are much more positive. This direction finder not only recovers $\mathbf{w}^*$

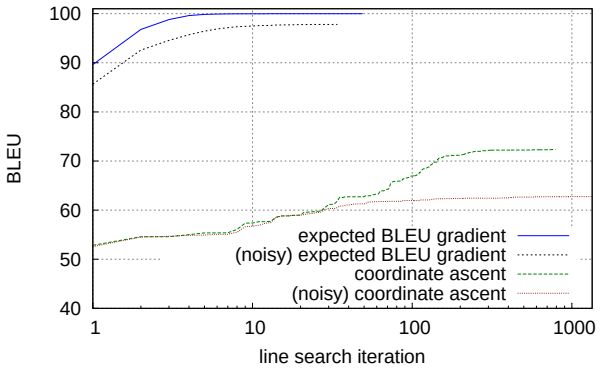

Figure 3: Comparison of rate of convergence between coordinate ascent and our expected BLEU direction finder ($D = 500$). **Noisy** refers to the noisy experimental setting.

(cosine > 0.999) even with 1000 dimensions, but its effectiveness is also visible with noisy data, as seen in Figure 2. The decrease of its cosine is relatively small compared to other search algorithms, and this decrease is not necessarily a sign of search errors since the addition of noise causes the true optimum to be different from $\mathbf{w}^*$. Finally, Figure 3 shows our rate of convergence compared to coordinate ascent.

Our experimental results with the GBM feature set data are shown in Table 2. Each table is divided into three sections corresponding respectively to MERT (Och, 2003) with Koehn-style coordinate ascent (Koehn, 2004), PRO, and our optimizer featuring both regularization and the gradient-based direction finder. All variants of MERT are initialized with a single starting point, which is either uniform weight or $\mathbf{w}_0$. Instead of providing MERT with additional random starting points as in Moses, we use random walks as in (Moore and Quirk, 2008) to attempt to move out of local optima.[8] Since PRO and our optimizer have hyperparameters, we use a held-out set (Dev) for adjusting them. For PRO, we adjust three parameters: a regularization penalty for ℓ_2, the parameter α in the add-α smoothed sentence-level version of BLEU (Lin and Och, 2004), and a parameter for scaling the corpus-level length of the references. The latter scaling parameter is discussed in (He and

[8] In the case of the gradient-based direction finder, we also use the following strategy whenever optimization converges to a (possibly local) optimum. We run one round of coordinate ascent, and continue with the gradient direction finder as soon as the optimum improves. If the none of the coordinate directions helped, we stop the search.

Method	Starting pt.	Chinese-English # feat.	Tune	Dev	Test	Finnish-English # feat.	Tune	Dev	Test
MERT	uniform	14	33.2	19.9	32.9	15	53.0	52.6	54.8
MERT	uniform	224	33.0	19.2	32.1	232	53.2	51.7	53.8
MERT	$\mathbf{w}_0$	224	34.1	20.1	33.0	232	53.9	52.5	54.7
PRO	$\mathbf{w}_0$	224	33.4	20.1	33.3	232	53.3	52.9	55.3
ℓ_2 MERT (v1: $\|\|\mathbf{w} - \mathbf{w}_0\|\|$)	$\mathbf{w}_0$	224	33.2	20.3	33.5	232	53.2	52.7	55.2
ℓ_2 MERT (v2: $D-1$ dimensions)	$\mathbf{w}_0$	224	33.0	20.4	33.2	232	52.9	52.6	55.0
ℓ_2 MERT (v3: ℓ_1-renormalized)	$\mathbf{w}_0$	224	33.1	20.0	33.3	232	53.1	52.5	55.1
ℓ_0 MERT	$\mathbf{w}_0$	224	33.4	20.3	33.2	232	53.2	52.6	55.1

Table 2: BLEU scores for GBM features. Model parameters were optimized on the Tune set. For PRO and regularized MERT, we optimized with different hyperparameters (regularization weight, etc.), and retained for each experimental condition the model that worked best on Dev. The table shows the performance of these retained models.

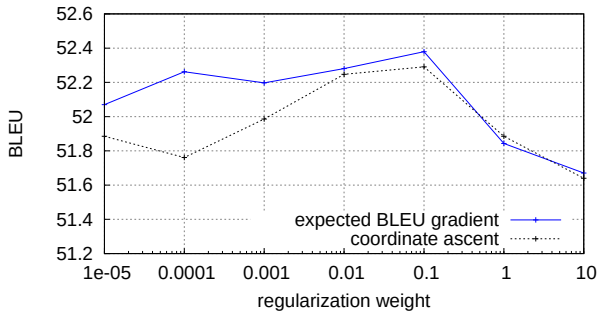

Figure 4: BLEU score on the Finnish Dev set (GBM) with different values for the $1/2\sigma^2$ regularization weight. To enable comparable results, the other hyperparameter (length) is kept fixed.

Deng, 2012; Nakov et al., 2012) and addresses the problem that systems tuned with PRO tend to produce sentences that are too short. On the other hand, regularized MERT only requires one hyperparameter to tune: a regularization penalty for ℓ_2 or ℓ_0. However, since PRO optimizes translation length on the Dev dataset and MERT does so using the Tune set, a comparison of the two systems would yield a discrepancy in length that would be undesirable. Therefore, we add another hyperparameter to regularized MERT to tune length in the same manner using the Dev set.

Table 2 offers several findings. First, unregularized MERT can achieve competitive results with a small set of highly engineered features, but adding a large set of more than 200 features causes MERT to perform poorly, particularly on the test set. However, unregularized MERT can recover much of this drop of performance if it is given a good sparse initializer $\mathbf{w}_0$. Regularized MERT (v1) provides an increase in the order of 0.5 BLEU on the test set compared to the best results with unregularized MERT. Regularized MERT is competitive with PRO, even though the number of features is relatively large. Using the same GBM experimental setting, Figure 4 compares regularized MERT using the gradient direction finder and coordinate ascent. At the best regularization setting, the two algorithms are comparable in terms of BLEU (though coordinate ascent is slower due to its lack of a good direction finder), but our method seems more robust with suboptimal regularization parameters.

Our results with the SparseHRM feature set data are shown in Table 3. As with the GBM feature set, we find again that the version of ℓ_2 MERT regularized towards $||\mathbf{w} - \mathbf{w}_0||$ is competitive with PRO, even though we train MERT with a large set of 3601 features.[9] One remaining question is whether MERT remains practical with large feature sets. As noted in the complexity analysis of Section 4.3, MERT has a dependence on the number of features that is comparable to PRO, i.e., it is linear in both cases. Practically, we find that optimization time is comparable between the two systems. In the case of Chinese-English for the GBM feature set, one run of the PRO optimizer took 26 minutes on average, while regularized MERT with the gradient direction finder took 37 minutes on average, taking into account the time to compute $\mathbf{w}_0$. In the case of Chinese-English for the SparseHRM feature set, average optimization times for PRO and our method were 3.10 hours and 3.84 hours on average, respectively.

[9] We note that the experimental setup of (Cherry, 2013) integrates the Sparse HRM features into the decoder, while we use them in an M-best reranking scenario. The reranking setup of this paper yields smaller improvements for both PRO and MERT than those of (Cherry, 2013).

Method	Starting pt.	Chinese-English # feat.	Tune	Dev	Test	Arabic-English # feat.	Tune	Dev	Test
MERT	uniform	14	25.7	34.0	27.8	14	43.2	42.8	45.5
MERT	uniform	3601	25.4	33.1	27.3	3601	45.7	42.3	44.9
MERT	$\mathbf{w}_0$	3601	27.7	33.5	27.5	3601	46.0	42.4	45.2
PRO	$\mathbf{w}_0$	3601	25.9	34.3	28.1	3601	44.6	43.4	46.1
ℓ_2 MERT (v1: $\|\|\mathbf{w} - \mathbf{w}_0\|\|$)	$\mathbf{w}_0$	3601	26.3	34.3	28.3	3601	45.2	43.2	46.0
ℓ_2 MERT (v2: $D-1$ dimensions)	$\mathbf{w}_0$	3601	26.4	34.1	28.2	3601	45.0	43.4	45.9
ℓ_2 MERT (v3: ℓ_1-renormalized)	$\mathbf{w}_0$	3601	26.1	34.0	27.9	3601	44.9	43.3	45.7
ℓ_0 MERT	$\mathbf{w}_0$	3601	26.5	34.2	28.1	3601	45.4	43.1	46.0

Table 3: BLEU scores for SparseHRM features. Notes in Table 2 also apply here.

Finally, as shown in Table 2, we see that MERT experiments that rely on a good initial starting point $\mathbf{w}_0$ generally perform better than when starting from a uniform vector. While having to compute $\mathbf{w}_0$ in the first place is a bit of a disadvantage compared to standard MERT, the need for good initializer is hardly surprising in the context of non-convex optimization. Other non-convex problems in machine learning, such as deep neural networks (DNN) and word alignment models, commonly require such initializers in order to obtain decent performance. In the case of DNN, extensive research is devoted to the problem of finding good initializers.[10] In the case of word alignment, it is common practice to initialize search in non-convex optimization problems—such as IBM Model 3 and 4 (Brown et al., 1993)—with solutions of simpler models—such as IBM Model 1.

7 Related work

MERT and its extensions have been the target of extensive research (Och, 2003; Macherey et al., 2008; Cer et al., 2008; Moore and Quirk, 2008; Kumar et al., 2009; Galley and Quirk, 2011). More recent work has focused on replacing MERT with a linearly decomposable approximations of the evaluation metric (Smith and Eisner, 2006; Liang et al., 2006; Watanabe et al., 2007; Chiang et al., 2008; Hopkins and May, 2011; Rosti et al., 2011; Gimpel and Smith, 2012; Cherry and Foster, 2012), which generally involve a surrogate loss function incorporating a regularization term such as the ℓ_2-norm. While we are not aware of any previous work adding a penalty on the weights in the context of MERT, (Cer et al., 2008) achieves a related effect. Cer et al.'s goal is to achieve a more regular or smooth objective function, while ours is to obtain a more regular set of parameters. The two approaches may be complementary.

More recently, new research has explored direction finding using a smooth surrogate loss function (Flanigan et al., 2013). Although this method is successful in helping MERT find better directions, it also exacerbates the tendency of MERT to overfit.[11] As an indirect way of controlling overfitting on the tuning set, their line searches are performed over directions estimated over a separate dataset.

8 Conclusion

In this paper, we have shown that MERT can scale to a much larger number of features than previously thought, thanks to regularization and a direction finder that directs the search towards the greatest increase of expected BLEU score. While our best results are comparable to PRO and not significantly better, we think that this paper provides a deeper understanding of why standard MERT can fail when handling an increasingly larger number of features. Furthermore, this paper complements the analysis by Hopkins and May (2011) of the differences between MERT and optimization with a surrogate loss function.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions.

[10] For example, (Larochelle et al., 2009) presents a pre-trained DNN that outperforms a shallow network, but the performance of the DNN becomes much worse relative to the shallow network once pre-training is turned off.

[11] Indeed, in their Table 3, a comparison between HILS and HOLS suggests tuning set performance improves substantially, while held out performance degrades.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Daniel Cer, Dan Jurafsky, and Christopher D. Manning. 2008. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34.

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.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 248–258.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 38–49.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Xiaodong He and Li Deng. 2012. Maximum expected BLEU training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 292–301.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362.

M. Hyder and K. Mahata. 2009. An approximate L0 norm minimization algorithm for compressed sensing. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3365–3368.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Demonstration Session*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. of AMTA*, pages 115–124.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.*, 10:1–40.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, Stroudsburg, PA, USA.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734.

David McAllester and Joseph Keshet. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Advances in Neural Information Processing Systems 24*, pages 2205–2212.

Robert C. Moore and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, pages 585–592.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Kishore Papineni. 1999. Discriminative training via linear programming. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 561–564, Vol. 2.

M.J.D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7(2):155–162.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition.

Antti-Veikko Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected BLEU training for graphs: BBN system description for WMT11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 159–165.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794.

Kristina Toutanova and Byung-Gyu Ahn. 2013. Learning non-linear features for machine translation using gradient boosting machines. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 406–411.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 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)*, pages 764–773.