Batch Tuning Strategies for Statistical Machine Translation

Colin Cherry and **George Foster**
National Research Council Canada
{Colin.Cherry,George.Foster}@nrc-cnrc.gc.ca

Abstract

There has been a proliferation of recent work on SMT tuning algorithms capable of handling larger feature sets than the traditional MERT approach. We analyze a number of these algorithms in terms of their sentence-level loss functions, which motivates several new approaches, including a Structured SVM. We perform empirical comparisons of eight different tuning strategies, including MERT, in a variety of settings. Among other results, we find that a simple and efficient batch version of MIRA performs at least as well as training online, and consistently outperforms other options.

1 Introduction

The availability of linear models and discriminative tuning algorithms has been a huge boon to statistical machine translation (SMT), allowing the field to move beyond the constraints of generative noisy channels (Och and Ney, 2002). The ability to optimize these models according to an error metric has become a standard assumption in SMT, due to the wide-spread adoption of Minimum Error Rate Training or MERT (Och, 2003). However, MERT has trouble scaling to more than 30 features, which has led to a surge in research on tuning schemes that can handle high-dimensional feature spaces.

These methods fall into a number of broad categories. *Minimum risk* approaches (Och, 2003; Smith and Eisner, 2006) have been quietly capable of handling many features for some time, but have yet to see widespread adoption. *Online* methods (Liang et al., 2006; Watanabe et al., 2007), are recognized to be effective, but require substantial implementation efforts due to difficulties with parallelization. *Pairwise ranking* (Shen et al., 2004; Hopkins and May, 2011) recasts tuning as classification, and can be very easy to implement, as it fits nicely into the established MERT infrastructure.

The MERT algorithm optimizes linear weights relative to a collection of k-best lists or lattices, which provide an approximation to the true search space. This optimization is wrapped in an outer loop that iterates between optimizing weights and re-decoding with those weights to enhance the approximation. Our primary contribution is to empirically compare eight tuning algorithms and variants, focusing on methods that work within MERT's established outer loop. This is the first comparison to include all three categories of optimizer.

Furthermore, we introduce three tuners that have not been previously tested. In particular, we test variants of Chiang et al.'s (2008) hope-fear MIRA that use k-best or lattice-approximated search spaces, producing a *Batch MIRA* that outperforms a popular mechanism for parallelizing online learners. We also investigate the direct optimization of hinge loss on k-best lists, through the use of a *Structured SVM* (Tsochantaridis et al., 2004). We review and organize the existing tuning literature, providing sentence-level loss functions for minimum risk, online and pairwise training. Finally, since randomization plays a different role in each tuner, we also suggest a new method for testing an optimizer's stability (Clark et al., 2011), which sub-samples the tuning set instead of varying a random seed.

2 Background

We begin by establishing some notation. We view our training set as a list of triples $[f, R, \mathcal{E}]_{i=1}^{n}$, where f is a source-language sentence, R is a set of target-language reference sentences, and $\mathcal{E}$ is the set of

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 427–436,
Montréal, Canada, June 3-8, 2012.

all reachable hypotheses; that is, each $e \in \mathcal{E}_i$ is a target-language derivation that can be decoded from f_i. The function $\vec{h}_i(e)$ describes e's relationship to its source f_i using features that decompose into the decoder. A linear model $\vec{w}$ scores derivations according to their features, meaning that the decoder solves:

$$e_i(\vec{w}) = \arg\max_{e \in \mathcal{E}_i} \vec{w} \cdot \vec{h}_i(e) \quad (1)$$

Assuming we wish to optimize our decoder's BLEU score (Papineni et al., 2002), the natural objective of learning would be to find a $\vec{w}$ such that $\text{BLEU}([e(\vec{w}), R]_1^n)$ is maximal. In most machine learning papers, this would be the point where we would say, "unfortunately, this objective is unfeasible." But in SMT, we have been happily optimizing exactly this objective for years using MERT.

However, it is now acknowledged that the MERT approach is not feasible for more than 30 or so features. This is due to two main factors:

1. MERT's parameter search slows and becomes less effective as the number of features rises, stopping it from finding good training scores.
2. BLEU is a scale invariant objective: one can scale $\vec{w}$ by any positive constant and receive the same BLEU score.[1] This causes MERT to resist standard mechanisms of regularization that aim to keep $||\vec{w}||$ small.

The problems with MERT can be addressed through the use of surrogate loss functions. In this paper, we focus on linear losses that decompose over training examples. Using R_i and $\mathcal{E}_i$, each loss $\ell_i(\vec{w})$ indicates how poorly $\vec{w}$ performs on the i^{th} training example. This requires a sentence-level approximation of BLEU, which we re-encode into a cost $\Delta_i(e)$ on derivations, where a high cost indicates that e receives a low BLEU score. Unless otherwise stated, we will assume the use of sentence BLEU with add-1 smoothing (Lin and Och, 2004). The learners differ in their definition of ℓ and Δ, and in how they employ their loss functions to tune their weights.

[1] This is true of any evaluation metric that considers only the ranking of hypotheses and not their model scores; ie, it is true of all common MT metrics.

2.1 Margin Infused Relaxed Algorithm

First employed in SMT by Watanabe et al. (2007), and refined by Chiang et al. (2008; 2009), the Margin Infused Relaxed Algorithm (MIRA) employs a structured hinge loss:

$$\ell_i(\vec{w}) = \max_{e \in \mathcal{E}_i} \left[\Delta_i(e) + \vec{w} \cdot \left(\vec{h}_i(e) - \vec{h}_i(e_i^*) \right) \right] \quad (2)$$

where e_i^* is an oracle derivation, and cost is defined as $\Delta_i(e) = \text{BLEU}_i(e_i^*) - \text{BLEU}_i(e)$, so that $\Delta_i(e_i^*) = 0$. The loss $\ell_i(\vec{w})$ is 0 only if $\vec{w}$ separates each $e \in \mathcal{E}_i$ from e_i^* by a margin proportional to their BLEU differentials.

MIRA is an instance of online learning, repeating the following steps: visit an example i, decode according to $\vec{w}$, and update $\vec{w}$ to reduce $\ell_i(\vec{w})$. Each update makes the smallest change to $\vec{w}$ (subject to a step-size cap C) that will separate the oracle from a number of negative hypotheses. The work of Crammer et al. (2006) shows that updating away from a single "fear" hypothesis that maximizes (2) admits a closed-form update that performs well. Let e_i' be the $e \in \mathcal{E}_i$ that maximizes $\ell_i(\vec{w})$; the update can be performed in two steps:

$$\begin{aligned} \eta_t &= \min \left[C, \frac{\ell_i(\vec{w}_t)}{||\vec{h}_i(e_i^*) - \vec{h}_i(e_i')||^2} \right] \\ \vec{w}_{t+1} &= \vec{w}_t + \eta_t \left(\vec{h}_i(e_i^*) - \vec{h}_i(e_i') \right) \end{aligned} \quad (3)$$

To improve generalization, the average of all weights seen during learning is used on unseen data.

Chiang et al. (2008) take advantage of MIRA's online nature to modify each update to better suit SMT. The cost Δ_i is defined using a pseudo-corpus BLEU that tracks the n-gram statistics of the model-best derivations from the last few updates. This modified cost matches corpus BLEU better than add-1 smoothing, but it also makes Δ_i time-dependent: each update for an example i will be in the context of a different pseudo-corpus. The oracle e_i^* also shifts with each update to $\vec{w}$, as it is defined as a "hope" derivation, which maximizes $\vec{w} \cdot \vec{h}_i(e) + \text{BLEU}_i(e)$. Hope updating ensures that MIRA aims for ambitious, reachable derivations.

In our implementation, we make a number of small, empirically verified deviations from Chiang et al. (2008). These include the above-mentioned use of a single hope and fear hypothesis, and the use

of hope hypotheses (as opposed to model-best hypotheses) to build the pseudo-corpus for calculating BLEU_i. These changes were observed to be neutral with respect to translation quality, but resulted in faster running time and simplified implementation.

2.2 Direct Optimization

With the exception of MIRA, the tuning approaches discussed in this paper are direct optimizers. That is, each solves the following optimization problem:

$$\vec{w}* = \arg\min_{\vec{w}} \frac{\lambda}{2}||\vec{w}||^2 + \sum_i \ell_i(\vec{w}) \qquad (4)$$

where the first term provides regularization, weighted by λ. Throughout this paper, (4) is optimized with respect to a fixed approximation of the decoder's true search space, represented as a collection of k-best lists. The various methods differ in their definition of loss and in how they optimize their objective.

Without the complications added by hope decoding and a time-dependent cost function, unmodified MIRA can be shown to be carrying out dual coordinate descent for an SVM training objective (Martins et al., 2010). However, exactly what objective hope-fear MIRA is optimizing remains an open question. Gimpel and Smith (2012) discuss these issues in greater detail, while also providing an interpretable alternative to MIRA.

2.3 Pairwise Ranking Optimization

Introduced by Hopkins and May (2011), Pairwise Ranking Optimization (PRO) aims to handle large feature sets inside the traditional MERT architecture. That is, PRO employs a growing approximation of $\mathcal{E}_i$ by aggregating the k-best hypotheses from a series of increasingly refined models. This architecture is desirable, as most groups have infrastructure to k-best decode their tuning sets in parallel.

For a given approximate $\tilde{\mathcal{E}}_i$, PRO creates a sample S_i of (e_g, e_b) pairs, such that $\mathrm{BLEU}_i(e_g) > \mathrm{BLEU}_i(e_b)$. It then uses a binary classifier to separate each pair. We describe the resulting loss in terms of an SVM classifier, to highlight similarities with MIRA. In terms of (4), PRO defines

$$\ell_i(\vec{w}) = \sum_{(e_g,e_b)\in S_i} 2\left(1 + \vec{w}\cdot\left(\vec{h}_i(e_b) - \vec{h}_i(e_g)\right)\right)^+$$

where $(x)^+ = \max(0, x)$. The hinge loss is multiplied by 2 to account for PRO's use of two examples (positive and negative) for each sampled pair. This sum of hinge-losses is 0 only if each pair is separated by a model score of 1. Given $[S]_{i=1}^n$, this convex objective can be optimized using any binary SVM.[2] Unlike MIRA, the margin here is fixed to 1; cost enters into PRO through its sampling routine, which performs a large uniform sample and then selects a subset of pairs with large BLEU differentials.

The PRO loss uses a sum over pairs in place of MIRA's max, which allows PRO to bypass oracle selection, and to optimize with off-the-shelf classifiers. This sum is potentially a weakness, as PRO receives credit for each correctly ordered pair in its sample, and these pairs are not equally relevant to the final BLEU score.

2.4 Minimum Risk Training

Minimum risk training (MR) interprets $\vec{w}$ as a probabilistic model, and optimizes expected BLEU. We focus on expected sentence costs (Och, 2003; Zens et al., 2007; Li and Eisner, 2009), as this risk is simple to optimize and fits nicely into our mathematical framework. Variants that use the expected sufficient statistics of BLEU also exist (Smith and Eisner, 2006; Pauls et al., 2009; Rosti et al., 2011).

We again assume a MERT-like tuning architecture. Let $\Delta_i(e) = -\mathrm{BLEU}_i(e)$ and let

$$\ell_i(\vec{w}) = E_{P_{\vec{w}}}[\Delta_i(e)] = \frac{\sum_{e\in\tilde{\mathcal{E}}_i}\left[\exp(\vec{w}\cdot\vec{h}_i(e))\Delta_i(e)\right]}{\sum_{e'\in\tilde{\mathcal{E}}_i}\exp(\vec{w}\cdot\vec{h}_i(e'))}$$

This expected cost becomes increasingly small as greater probability mass is placed on derivations with high BLEU scores. This smooth, non-convex objective can be solved to a local minimum using gradient-based optimizers; we have found stochastic gradient descent to be quite effective (Bottou, 2010).

Like PRO, MR requires no oracle derivation, and fits nicely into the established MERT architecture. The expectations needed to calculate the gradient

$$E_{P_{\vec{w}}}\left[\vec{h}_i(e)\Delta_i(e)\right] - E_{P_{\vec{w}}}\left[\Delta_i(e)\right]E_{P_{\vec{w}}}\left[\vec{h}_i(e)\right]$$

[2] Hopkins and May (2011) advocate a maximum-entropy version of PRO, which is what we evaluate in our empirical comparison. It can be obtained using a logit loss $\ell_i(\vec{w}) = \sum_{g,b} 2\log\left(1+\exp\left(\vec{w}\cdot\left(\vec{h}_i(e_b) - \vec{h}_i(e_g)\right)\right)\right)$.

are trivial to extract from a k-best list of derivations. Each downward step along this gradient moves the model toward likely derivations, and away from likely derivations that incur high costs.

3 Novel Methods

We have reviewed three tuning methods, all of which address MERT's weakness with large features by using surrogate loss functions. Additionally, MIRA has the following advantages over PRO and MR:

1. Loss is optimized using the true $\mathcal{E}_i$, as opposed to an approximate search space $\tilde{\mathcal{E}}_i$.
2. Sentence BLEU is calculated with a fluid pseudo-corpus, instead of add-1 smoothing.

Both of these advantages come at a cost: operating on the true $\mathcal{E}_i$ sacrifices easy parallelization, while using a fluid pseudo-corpus creates an unstable learning objective. We develop two large-margin tuners that explore these trade-offs.

3.1 Batch MIRA

Online training makes it possible to learn with the decoder in the loop, forgoing the need to approximate the search space, but it is not necessarily convenient to do so. Online algorithms are notoriously difficult to parallelize, as they assume each example is visited in sequence. Parallelization is important for efficient SMT tuning, as decoding is still relatively expensive.

The parallel online updates suggested by Chiang et al. (2008) involve substantial inter-process communication, which may not be easily supported by all clusters. McDonald et al. (2010) suggest a simpler distributed strategy that is amenable to map-reduce-like frameworks, which interleaves online training on shards with weight averaging across shards. This strategy has been adopted by Moses (Hasler et al., 2011), and it is the one we adopt in our MIRA implementation.

However, online training using the decoder may not be necessary for good performance. The success of MERT, PRO and MR indicates that their shared search approximation is actually quite reasonable. Therefore, we propose Batch MIRA, which sits exactly where MERT sits in the standard tuning architecture, greatly simplifying parallelization:

1. Parallel Decode: $[\tilde{\mathcal{E}}']_1^n$ = k-best($[f, \mathcal{E}]_1^n, \vec{w}$)
2. Aggregate: $[\tilde{\mathcal{E}}]_1^n = [\tilde{\mathcal{E}}]_1^n \cup [\tilde{\mathcal{E}}']_1^n$
3. Train: $\vec{w}$ = BatchMIRA($[f, R, \tilde{\mathcal{E}}]_1^n, \vec{w}$)
4. Repeat

where BatchMIRA() trains the SMT-adapted MIRA algorithm to completion on the current approximation $\tilde{\mathcal{E}}$, without parallelization.[3] The only change we make to MIRA is to replace the hope-fear decoding of sentences with the hope-fear re-ranking of k-best lists. Despite its lack of parallelization, each call to BatchMIRA() is extremely fast, as SMT tuning sets are small enough to load all of $[\tilde{\mathcal{E}}]_1^n$ into memory. We test two Batch MIRA variants, which differ in their representation of $\tilde{\mathcal{E}}$. Pseudo-code that covers both is provided in Algorithm 1. Note that if we set $\tilde{\mathcal{E}} = \mathcal{E}$, Algorithm 1 also describes online MIRA.

Batch k-best MIRA inherits all of the MERT architecture. It is very easy to implement; the hope-fear decoding steps can by carried out by simply evaluating BLEU score and model score for each hypothesis in the k-best list.

Batch Lattice MIRA replaces k-best decoding in step 1 with decoding to lattices. To enable loading all of the lattices into memory at once, we prune to a density of 50 edges per reference word. The hope-fear decoding step requires the same oracle lattice decoding algorithms as online MIRA (Chiang et al., 2008). The lattice aggregation in the outer loop can be kept reasonable by aggregating only those paths corresponding to hope or fear derivations.

3.2 Structured SVM

While MIRA takes a series of local hinge-loss reducing steps, it is also possible to directly minimize the sum of hinge-losses using a batch algorithm, creating a structured SVM (Tsochantaridis et al., 2004). To avoid fixing an oracle before optimization begins, we adapt Yu and Joachim's (2009) latent SVM to our task, which allows the oracle derivation for each sentence to vary during training. Again we assume a MERT-like architecture, which approximates $\mathcal{E}$ with an $\tilde{\mathcal{E}}$ constructed from aggregated k-best lists.

Inspired by the local oracle of Liang et al. (2006), we define $\tilde{\mathcal{E}}_i*$ to be an oracle set:

$$\tilde{\mathcal{E}}_i* = \{e | \mathrm{BLEU}_i(e) \text{ is maximal}\}.$$

[3] In our implementation, BatchMIRA() trains for $J = 30$ passes over $[\tilde{\mathcal{E}}]_1^n$.

Algorithm 1 BatchMIRA

input $[f, R, \tilde{\mathcal{E}}]_1^n$, $\vec{w}$, max epochs J, step cap C, and pseudo-corpus decay γ.
init Pseudo-corpus BG to small positive counts.
init $t = 1$; $\vec{w}_t = \vec{w}$
for j from 1 to J **do**
  for i from 1 to n in random order **do**
    `// Hope-fear decode in` $\tilde{\mathcal{E}}_i$
    $e_t^* = \arg\max_{e \in \tilde{\mathcal{E}}_i} \left[\vec{w}_t \cdot \vec{h}_i(e) + \mathrm{BLEU}_i(e)\right]$
    $e'_t = \arg\max_{e \in \tilde{\mathcal{E}}_i} \left[\vec{w}_t \cdot \vec{h}_i(e) - \mathrm{BLEU}_i(e)\right]$
    `// Update weights`
    $\Delta_t = \mathrm{BLEU}_i(e_t^*) - \mathrm{BLEU}_i(e'_t)$
    $\eta_t = \min\left[C, \frac{\Delta_t + \vec{w}_t \cdot \left(\vec{h}_i(e'_t) - \vec{h}_i(e_t^*)\right)}{||\vec{h}_i(e_t^*) - \vec{h}_i(e'_t)||^2}\right]$
    $\vec{w}_{t+1} = \vec{w}_t + \eta_t\left(\vec{h}_i(e_t^*) - \vec{h}_i(e'_i)\right)$
    `// Update statistics`
    $BG = \gamma BG +$ BLEU stats for e_t^* and R_i
    $t = t + 1$
  end for
  $\vec{w}_j^{avg} = \frac{1}{nj} \sum_{t'=1}^{nj} \vec{w}_{t'}$
end for
return $\vec{w}_j^{avg}$ that maximizes training BLEU

Cost is also defined in terms of the maximal BLEU,

$$\Delta_i(e) = \max_{e' \in \tilde{\mathcal{E}}_i} \left[\mathrm{BLEU}_i(e')\right] - \mathrm{BLEU}_i(e).$$

Finally, loss is defined as:

$$\ell_i(\vec{w}) = \max_{e \in \tilde{\mathcal{E}}_i} \left[\Delta_i(e) + \vec{w} \cdot \vec{h}_i(e) - \max_{e_i^* \in \tilde{\mathcal{E}}_i *} \left(\vec{w} \cdot \vec{h}_i(e_i^*)\right)\right]$$

This loss is 0 only if some hypothesis in the oracle set is separated from all others by a margin proportional to their BLEU_i differentials.

With loss defined in this manner, we can minimize (4) to local minimum by using an alternating training procedure. For each example i, we select a fixed $e_i^* \in \tilde{\mathcal{E}}_i *$ that maximizes model score; that is, $\vec{w}$ is used to break ties in BLEU for oracle selection. With the oracle fixed, the objective becomes a standard structured SVM objective, which can be minimized using a cutting-plane algorithm, as described by Tsochantaridis et al. (2004). After doing so, we can drive the loss lower still by iterating this process: re-select each oracle (breaking ties with the new $\vec{w}$), then re-optimize $\vec{w}$. We do so 10 times. We were surprised by the impact of these additional iterations on the final loss; for some sentences, $\tilde{\mathcal{E}}_i *$ can be quite large.

Despite the fact that both algorithms use a structured hinge loss, there are several differences between our SVM and MIRA. The SVM has an explicit regularization term λ that is factored into its global objective, while MIRA regularizes implicitly by taking small steps. The SVM requires a stable objective to optimize, meaning that it must forgo the pseudo-corpus used by MIRA to calculate Δ_i; instead, the SVM uses an interpolated sentence-level BLEU (Liang et al., 2006).[4] Finally, MIRA's oracle is selected with hope decoding. With a sufficiently large $\vec{w}$, any $e \in \tilde{\mathcal{E}}$ can potentially become the oracle. In contrast, the SVM's local oracle is selected from a small set $\tilde{\mathcal{E}} *$, which was done to more closely match the assumptions of the Latent SVM.

To solve the necessary quadratic programming sub-problems, we use a multiclass SVM similar to LIBLINEAR (Hsieh et al., 2008). Like Batch MIRA and PRO, the actual optimization is very fast, as the cutting plane converges quickly and all of $[\tilde{\mathcal{E}}]_1^n$ can be loaded into memory at once.

3.3 Qualitative Summary

We have reviewed three tuning methods and introduced three tuning methods. All six methods employ sentence-level loss functions, which in turn employ sentence-level BLEU approximations. Except for online MIRA, all methods plug nicely into the existing MERT architecture. These methods can be split into two groups: MIRA variants (online, batch k-best, batch lattice), and direct optimizers (PRO, MR and SVM). The MIRA variants use pseudo-corpus BLEU in place of smoothed BLEU, and provide access to richer hypothesis spaces through the use of online training or lattices.[5] The direct optimizers have access to a tunable regularization parameter λ, and do not require special purpose code for hope and fear lattice decoding. Batch

[4] SVM training with interpolated BLEU outperformed add-1 BLEU in preliminary testing. A comparison of different BLEU approximations under different tuning objectives would be an interesting path for future work.

[5] MR approaches that use lattices (Li and Eisner, 2009; Pauls et al., 2009; Rosti et al., 2011) or the complete search space (Arun et al., 2010) exist, but are not tested here.

k-best MIRA straddles the two groups, benefiting from pseudo-corpus BLEU and easy implementation, while being restricted to a k-best list.

4 Experimental Design

We evaluated the six tuning strategies described in this paper, along with two MERT baselines, on three language pairs (French-English (Fr-En), English-French (En-Fr) and Chinese-English (Zh-En)), across three different feature-set sizes. Each setting was run five times over randomized variants to improve reliability. To cope with the resulting large number of configurations, we ran all experiments using an efficient phrase-based decoder similar to Moses (Koehn et al., 2007).

All tuning methods that use an approximate $\tilde{\mathcal{E}}$ perform 15 iterations of the outer loop and return the weights that achieve the best development BLEU score. When present, λ was coarsely tuned (trying 3 values differing by magnitudes of 10) in our large-feature Chinese-English setting.

- **kb-mert** : k-best MERT with 20 random restarts. All k-best methods use $k = 100$.
- **lb-mert** : Lattice MERT (Machery et al., 2008) using unpruned lattices and aggregating only those paths on the line search's upper envelope.
- **mira** : Online MIRA (§2.1). All MIRA variants use a pseudo-corpus decay $\gamma = 0.999$ and $C = 0.01$. Online parallelization follows McDonald et al. (2010), using 8 shards. We tested 20, 15, 10, 8 and 5 shards during development.
- **lb-mira** : Batch Lattice MIRA (§3.1).
- **kb-mira** : Batch k-best MIRA (§3.1).
- **pro** : PRO (§2.3) follows Hopkins and May (2011); however, we were unable to find settings that performed well in general. Reported results use MegaM[6] with a maximum of 30 iterations (as is done in Moses; the early stopping provides a form of regularization) for our six English/French tests, and MegaM with 100 iterations and a reduced initial uniform sample (50 pairs instead of 5000) for our three English/Chinese tests.
- **mr** : MR as described in §2.4. We employ a learning rate of $\eta_0/(1 + \eta_0\lambda t)$ for stochastic gradient descent, with η_0 tuned to optimize the training loss achieved after one epoch (Bottou, 2010). Upon reaching a local optimum, we reshuffle our data, re-tune our learning rate, and re-start from the optimum, repeating this process 5 times. We do not sharpen our distribution with a temperature or otherwise control for entropy; instead, we trust $\lambda = 50$ to maintain a reasonable distribution.
- **svm** : Structured SVM (§3.2) with $\lambda = 1000$.

corpus	sentences	words (en)	words (fr)
train	2,928,759	60,482,232	68,578,459
dev	2,002	40,094	44,603
test1	2,148	42,503	48,064
test2	2,166	44,701	49,986

Table 1: Hansard Corpus (English/French)

corpus	sentences	words (zh)	words (en)
train1	6,677,729	200,706,469	213,175,586
train2	3,378,230	69,232,098	66,510,420
dev	1,506	38,233	40,260
nist04	1,788	53,439	59,944
nist06	1,664	41,782	46,140
nist08	1,357	35,369	42,039

Table 2: NIST09 Corpus (Chinese-English). Train1 corresponds to the UN and Hong Kong sub-corpora; train2 to all others.

4.1 Data

Systems for English/French were trained on Canadian Hansard data (years 2001–2009) summarized in table 1.[7] The dev and test sets were chosen randomly from among the most recent 5 days of Hansard transcripts.

The system for Zh-En was trained on data from the NIST 2009 Chinese MT evaluation, summarized in table 2. The dev set was taken from the NIST 05 evaluation set, augmented with some material reserved from other NIST corpora. The NIST 04, 06, and 08 evaluation sets were used for testing.

4.2 SMT Features

For all language pairs, phrases were extracted with a length limit of 7 from separate word alignments

[6] Available at `www.cs.utah.edu/~hal/megam/`

[7] This corpus will be distributed on request.

template	max	fren	enfr	zhen
tgt unal	50	50	50	31
count bin	11	11	11	11
word pair	6724	1298	1291	1664
length bin	63	63	63	63
total	6848	1422	1415	1769

Table 3: Sparse feature templates used in Big.

performed by IBM2 and HMM models and symmetrized using diag-and (Koehn et al., 2003). Conditional phrase probabilities in both directions were estimated from relative frequencies, and from lexical probabilities (Zens and Ney, 2004). Language models were estimated with Kneser-Ney smoothing using SRILM. Six-feature lexicalized distortion models were estimated and applied as in Moses.

For each language pair, we defined roughly equivalent systems (exactly equivalent for En-Fr and Fr-En, which are mirror images) for each of three nested feature sets: Small, Medium, and Big.

The Small set defines a minimal 7-feature system intended to be within easy reach of all tuning strategies. It comprises 4 TM features, one LM, and length and distortion features. For the Chinese system, the LM is a 5-gram trained on the NIST09 Gigaword corpus; for English/French, it is a 4-gram trained on the target half of the parallel Hansard.

The Medium set is a more competitive 18-feature system. It adds 4 TM features, one LM, and 6 lexicalized distortion features. For Zh-En, Small's TM (trained on both *train1* and *train2* in table 2) is replaced by 2 separate TMs from these sub-corpora; for En/Fr, the extra TM (4 features) comes from a forced-decoding alignment of the training corpus, as proposed by Wuebker et al. (2010). For Zh-En, the extra LM is a 4-gram trained on the target half of the parallel corpus; for En/Fr, it is a 4-gram trained on 5m sentences of similar parliamentary data.

The Big set adds sparse Boolean features to Medium, for a maximum of 6,848 features. We used sparse feature templates that are equivalent to the PBMT set described in (Hopkins and May, 2011): *tgt unal* picks out each of the 50 most frequent target words to appear unaligned in the phrase table; *count bin* uniquely bins joint phrase pair counts with upper bounds 1,2,4,8,16,32,64,128,1k,10k,∞; *word pair* fires when each of the 80 most frequent words in each language appear aligned 1-1 to each other, to some other word, or not 1-1; and *length bin* captures each possible phrase length and length pair. Table 3 summarizes the feature templates, showing the maximum number of features each can generate, and the number of features that received non-zero weights in the final model tuned by MR for each language pair.

Feature weights are initialized to 1.0 for each of the TM, LM and distortion penalty features. All other weights are initialized to 0.0.

4.3 Stability Testing

We follow Clark et al (2011), and perform multiple randomized replications of each experiment. However, their method of using different random seeds is not applicable in our context, since randomization does not play the same role for all tuning methods. Our solution was to randomly draw and fix four different sub-samples of each dev set, retaining each sentence with a probability of 0.9. For each tuning method and setting, we then optimize on the original dev and all sub-samples. The resulting standard deviations provide an indication of stability.

5 Results

The results of our survey of tuning methods can be seen in Tables 4, 5 and 6. Results are averaged over test sets (2 for Fr/En, 3 for Zh/En), and over 5 sub-sampled runs per test set. The SD column reports the standard deviation of the average test score across the 5 sub-samples.

It may be dismaying to see only small score improvements when transitioning from Medium to Big. This is partially due to the fact that our Big feature set affects only phrase-table scores. Our phrase tables are already strong, through our use of large data or leave-one-out forced decoding. The important baseline when assessing the utility of a method is Medium k-best MERT. In all language pairs, our Big systems generally outperform this baseline by 0.4 BLEU points. It is interesting to note that most methods achieve the bulk of this improvement on the Medium feature set.[8] This indicates that MERT begins to show some problems even in an 18-feature

[8] One can see the same phenomenon in the results of Hopkins and May (2011) as well.

Table 4: French to English Translation (Fr-En)

	Small			Medium			Big		
	Tune	Test	SD	Tune	Test	SD	Tune	Test	SD
kb-mert	40.50	39.94	0.04	40.75	40.29	0.13	n/a	n/a	n/a
lb-mert	**40.52**	39.93	0.11	40.93	40.39	0.08	n/a	n/a	n/a
mira	40.38	39.94	0.04	40.64	40.59	0.06	41.02	40.74	0.05
kb-mira	40.46	39.97	0.05	40.92	40.64	0.12	41.46	40.75	0.08
lb-mira	40.44	39.98	0.06	**40.94**	**40.65**	0.06	**41.59**	**40.78**	0.09
pro	40.11	40.05	0.05	40.16	40.07	0.08	40.55	40.21	0.24
mr	40.24	39.88	0.05	40.70	40.57	0.14	41.18	40.60	0.08
svm	40.05	**40.20**	0.03	40.60	40.56	0.08	41.32	40.52	0.07

Table 5: English to French Translation (En-Fr)

	Small			Medium			Big		
	Tune	Test	SD	Tune	Test	SD	Tune	Test	SD
kb-mert	**40.47**	39.72	0.06	40.70	40.02	0.11	n/a	n/a	n/a
lb-mert	40.45	39.76	0.08	40.90	40.13	0.10	n/a	n/a	n/a
mira	40.36	**39.83**	0.03	40.78	40.44	0.02	40.89	40.45	0.05
kb-mira	40.44	39.83	0.02	40.94	40.35	0.06	41.48	40.52	0.06
lb-mira	40.45	39.83	0.02	**41.05**	**40.45**	0.04	**41.65**	**40.59**	0.07
pro	40.17	39.57	0.15	40.30	40.01	0.04	40.75	40.22	0.17
mr	40.31	39.65	0.04	40.94	40.30	0.13	41.45	40.47	0.10
svm	39.99	39.55	0.03	40.40	39.96	0.05	41.00	40.21	0.03

Table 6: Chinese to English Translation (Zh-En)

	Small			Medium			Big		
	Tune	Test	SD	Tune	Test	SD	Tune	Test	SD
kb-mert	23.97	**29.65**	0.06	25.74	31.58	0.42	n/a	n/a	n/a
lb-mert	**24.18**	29.48	0.15	26.42	32.39	0.22	n/a	n/a	n/a
mira	23.98	29.54	0.01	26.23	32.58	0.08	25.99	32.52	0.08
kb-mira	24.10	29.51	0.06	26.28	32.50	0.12	26.18	32.61	0.14
lb-mira	24.13	29.59	0.05	**26.43**	**32.77**	0.06	26.40	32.82	0.18
pro	23.25	28.74	0.24	25.80	32.42	0.20	26.49	32.18	0.40
mr	23.87	29.55	0.09	26.26	32.52	0.12	26.42	32.79	0.15
svm	23.59	28.91	0.05	26.26	32.70	0.05	**27.23**	**33.04**	0.12

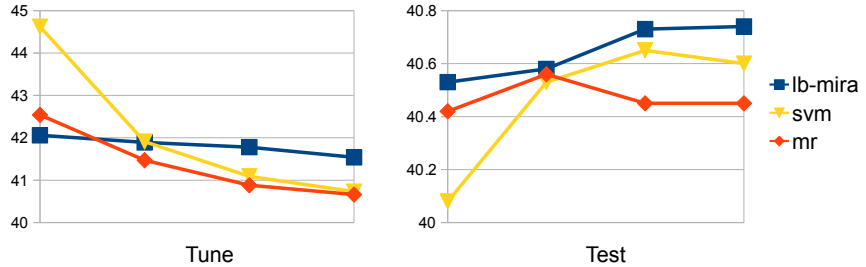

Figure 1: French-English test of regularization with an over-fitting feature set. lb-mira varies C ={1, 1e-1, 1e-2, 1e-3}, its default C is 1e-2; svm varies λ ={1e2, 1e3, 1e4, 1e5}, its default λ is 1e3; mr varies λ ={5, 5e1, 5e2, 5e3}, its default λ is 5e1.

setting, which can be mitigated through the use of Lattice MERT.

When examining score differentials, recall that the reported scores average over multiple test sets and sub-sampled tuning runs. Using Small features, all of the tested methods are mostly indistinguishable, but as we move to Medium and Big, Batch Lattice MIRA emerges as our method of choice. It is the top scoring system in all Medium settings, and in two of three Big settings (in Big Zh-En, the SVM comes first, with batch lattice MIRA placing second). However, all of the MIRA variants perform similarly, though our implementation of online MIRA is an order of magnitude slower, mostly due to its small number of shards. It is interesting that our batch lattice variant consistently outperforms online MIRA. We attribute this to our parallelization strategy, Chiang et al.'s (2008) more complex solution may perform better.

There may be settings where an explicit regularization parameter is desirable, thus we also make a recommendation among the direct optimizers (PRO, MR and SVM). Though these systems all tend to show a fair amount of variance across language and feature sets (likely due to their use sentence-level BLEU), MR performs the most consistently, and is always within 0.2 of batch lattice MIRA.

The SVM's performance on Big Zh-En is an intriguing outlier in our results. Note that it not only performs best on the test set, but also achieves the best tuning score by a large margin. We suspect we have simply found a setting where interpolated BLEU and our choice of λ work particularly well. We intend to investigate this case to see if this level of success can be replicated consistently, perhaps through improved sentence BLEU approximation or improved oracle selection.

5.1 Impact of Regularization

One main difference between MIRA and the direct optimizers is the availability of an explicit regularization term λ. To measure the impact of this parameter, we designed a feature set explicitly for overfitting. This set uses our Big Fr-En features, with the count bin template modified to distinguish each joint count observed in the tuning set. These new features, which expand the set to 20k+ features, should generalize poorly.

We tested MR and SVM on our Fr-En data using this feature set, varying their respective regularization parameters by factors of 10. We compared this to Batch Lattice MIRA's step-size cap C, which controls its regularization (Martins et al., 2010). The results are shown in Figure 1. Looking at the tuning scores, one can see that λ affords much greater control over tuning performance than MIRA's C. Looking at test scores, MIRA's narrow band of regularization appears to be just about right; however, there is no reason to expect this to always be the case.

6 Conclusion

We have presented three new, large-margin tuning methods for SMT that can handle thousands of features. Batch lattice and k-best MIRA carry out their online training within approximated search spaces, reducing costs in terms of both implementation and training time. The Structured SVM optimizes a sum of hinge losses directly, exposing an explicit regularization term. We have organized the literature on tuning, and carried out an extensive comparison of linear-loss SMT tuners. Our experiments show Batch Lattice MIRA to be the most consistent of the tested methods. In the future, we intend to investigate improved sentence-BLEU approximations to help narrow the gap between MIRA and the direct optimizers.

Acknowledgements

Thanks to Mark Hopkins, Zhifei Li and Jonathan May for their advice while implementing the methods in this review, and to Kevin Gimpel, Roland Kuhn and the anonymous reviewers for their valuable comments on an earlier draft.

References

Abhishek Arun, Barry Haddow, and Philipp Koehn. 2010. A unified approach to minimum risk training and decoding. In *Proceedings of the Joint Workshop on Statistical Machine Translation and MetricsMATR*, pages 365–374.

Leon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–187.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP*, pages 224–233.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *HLT-NAACL*, pages 218–226.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL*, pages 176–181.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *HLT-NAACL*, Montreal, Canada, June.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2011. Margin infused relaxed algorithm for moses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *EMNLP*, pages 1352–1362.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris 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 *ACL*, pages 177–180, Prague, Czech Republic, June.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, pages 40–51.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*, pages 761–768.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *COLING*, pages 501–507.

Wolfgang Machery, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*, pages 725–734.

André F. T. Martins, Kevin Gimpel, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Learning structured classifiers with dual coordinate descent. Technical Report CMU-ML-10-109, Carnegie Mellon University.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *ACL*, pages 456–464.

Franz Joseph Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302, Philadelphia, PA, July.

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

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

Adam Pauls, John Denero, and Dan Klein. 2009. Consensus training for consensus decoding in machine translation. In *EMNLP*, pages 1418–1427.

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.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184, Boston, Massachusetts, May 2 - May 7.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *COLING-ACL*, pages 787–794.

Ioannis Tsochantaridis, Thomas Hofman, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *ICML*, pages 823–830.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*, pages 764–773.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *ACL*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264, Boston, USA, May.

Richard Zens, Săsa Hasan, and Hermann Ney. 2007. A systematic comparison of training criteria for statistical machine translation. In *EMNLP*, pages 524–532.