Fast and Adaptive Online Training of Feature-Rich Translation Models

Spence Green, Sida Wang, Daniel Cer, and **Christopher D. Manning**
Computer Science Department, Stanford University
{spenceg,sidaw,danielcer,manning}@stanford.edu

Abstract

We present a fast and scalable online method for tuning statistical machine translation models with large feature sets. The standard tuning algorithm—MERT—only scales to tens of features. Recent discriminative algorithms that accommodate sparse features have produced smaller than expected translation quality gains in large systems. Our method, which is based on stochastic gradient descent with an adaptive learning rate, scales to millions of features and tuning sets with tens of thousands of sentences, while still converging after only a few epochs. Large-scale experiments on Arabic-English and Chinese-English show that our method produces significant translation quality gains by exploiting sparse features. Equally important is our analysis, which suggests techniques for mitigating overfitting and domain mismatch, and applies to other recent discriminative methods for machine translation.

1 Introduction

Sparse, overlapping features such as words and n-gram contexts improve many NLP systems such as parsers and taggers. Adaptation of discriminative learning methods for these types of features to statistical machine translation (MT) systems, which have historically used idiosyncratic learning techniques for a few dense features, has been an active research area for the past half-decade. However, despite some research successes, feature-rich models are rarely used in annual MT evaluations. For example, among all submissions to the WMT and IWSLT 2012 shared tasks, just one participant tuned more than 30 features (Hasler et al., 2012a). Slow uptake of these methods may be due to implementation complexities, or to practical difficulties of configuring them for specific translation tasks (Gimpel and Smith, 2012; Simianer et al., 2012, *inter alia*).

We introduce a new method for training feature-rich MT systems that is effective yet comparatively easy to implement. The algorithm scales to millions of features and large tuning sets. It optimizes a logistic objective identical to that of PRO (Hopkins and May, 2011) with stochastic gradient descent, although other objectives are possible. The learning rate is set adaptively using AdaGrad (Duchi et al., 2011), which is particularly effective for the mixture of dense and sparse features present in MT models. Finally, feature selection is implemented as efficient L_1 regularization in the forward-backward splitting (FOBOS) framework (Duchi and Singer, 2009). Experiments show that our algorithm converges faster than batch alternatives.

To learn good weights for the sparse features, most algorithms—including ours—benefit from more tuning data, and the natural source is the training bitext. However, the bitext presents two problems. First, it has a single reference, sometimes of lower quality than the multiple references in tuning sets from MT competitions. Second, large bitexts often comprise many text genres (Haddow and Koehn, 2012), a virtue for classical dense MT models but a curse for high dimensional models: bitext tuning can lead to a significant domain adaptation problem when evaluating on standard test sets. Our analysis separates and quantifies these two issues.

We conduct large-scale translation quality experiments on Arabic-English and Chinese-English. As baselines we use MERT (Och, 2003), PRO, and the Moses (Koehn et al., 2007) implementation of k-best MIRA, which Cherry and Foster (2012) recently showed to work as well as online MIRA (Chiang, 2012) for feature-rich models. The first experiment uses standard tuning and test sets from the NIST OpenMT competitions. The second experiment uses tuning and test sets sampled from the large bitexts. The new method yields significant improvements in both experiments. Our code is included in the Phrasal (Cer et al., 2010) toolkit, which is freely available.

2 Adaptive Online Algorithms

Machine translation is an unusual machine learning setting because multiple correct translations exist and decoding is comparatively expensive. When we have a large feature set and therefore want to tune on a large data set, batch methods are infeasible. Online methods can converge faster, and in practice they often find *better* solutions (Liang and Klein, 2009; Bottou and Bousquet, 2011, *inter alia*).

Recall that stochastic gradient descent (SGD), a fundamental online method, updates weights w according to

$$w_t = w_{t-1} - \eta \nabla \ell_t(w_{t-1}) \tag{1}$$

with loss function[1] $\ell_t(w)$ of the t^{th} example, (sub)gradient of the loss with respect to the parameters $\nabla \ell_t(w_{t-1})$, and learning rate η.

SGD is sensitive to the learning rate η, which is difficult to set in an MT system that mixes frequent "dense" features (like the language model) with sparse features (e.g., for translation rules). Furthermore, η applies to each coordinate in the gradient, an undesirable property in MT where good sparse features may fire very infrequently. We would instead like to take larger steps for sparse features and smaller steps for dense features.

2.1 AdaGrad

AdaGrad is a method for setting an *adaptive learning rate* that comes with good theoretical guarantees. The theoretical improvement over SGD is most significant for high-dimensional, sparse features. AdaGrad makes the following update:

$$w_t = w_{t-1} - \eta \Sigma_t^{1/2} \nabla \ell_t(w_{t-1}) \tag{2}$$

$$\begin{aligned}\Sigma_t^{-1} &= \Sigma_{t-1}^{-1} + \nabla \ell_t(w_{t-1}) \nabla \ell_t(w_{t-1})^T \\ &= \sum_{i=1}^{t} \nabla \ell_i(w_{i-1}) \nabla \ell_i(w_{i-1})^T \end{aligned} \tag{3}$$

A diagonal approximation to Σ can be used for a high-dimensional vector w_t. In this case, AdaGrad is simple to implement and computationally cheap. Consider a single dimension j, and let scalars $v_t = w_{t,j}$, $g_t = \nabla_j \ell_t(w_{t-1})$, $G_t = \sum_{i=1}^{t} g_i^2$, then the update rule is

$$v_t = v_{t-1} - \eta \, G_t^{-1/2} g_t \tag{4}$$

$$G_t = G_{t-1} + g_t^2 \tag{5}$$

Compared to SGD, we just need to store $G_t = \Sigma_{t,jj}^{-1}$ for each dimension j.

[1] We specify the loss function for MT in section 3.1.

2.2 Prior Online Algorithms in MT

AdaGrad is related to two previous online learning methods for MT.

MIRA Chiang et al. (2008) described an adaption of MIRA (Crammer et al., 2006) to MT. MIRA makes the following update:

$$w_t = \arg\min_w \frac{1}{2\eta} \|w - w_{t-1}\|_2^2 + \ell_t(w) \tag{6}$$

The first term expresses *conservativity*: the weight should change as little as possible based on a single example, ensuring that it is never beneficial to overshoot the minimum.

The relationship to SGD can be seen by linearizing the loss function $\ell_t(w) \approx \ell_t(w_{t-1}) + (w - w_{t-1})^T \nabla \ell_t(w_{t-1})$ and taking the derivative of (6). The result is exactly (1).

AROW Chiang (2012) adapted AROW (Crammer et al., 2009) to MT. AROW models the current weight as a Gaussian centered at w_{t-1} with covariance Σ_{t-1}, and does the following update upon seeing training example x_t:

$$\begin{aligned} w_t, \Sigma_t = \\ \arg\min_{w,\Sigma} \; & \frac{1}{\eta} D_{\text{KL}}(\mathcal{N}(w, \Sigma) \parallel \mathcal{N}(w_{t-1}, \Sigma_{t-1})) \\ & + \ell_t(w) + \frac{1}{2\eta} x_t^T \Sigma x_t \end{aligned} \tag{7}$$

The KL-divergence term expresses a more general, directionally sensitive conservativity. Ignoring the third term, the Σ that minimizes the KL is actually Σ_{t-1}. As a result, the first two terms of (7) generalize MIRA so that we may be more conservative in some directions specified by Σ. To see this, we can write out the KL-divergence between two Gaussians in closed form, and observe that the terms involving w do not interact with the terms involving Σ:

$$\begin{aligned} w_t = \arg\min_w \; & \frac{1}{2\eta} (w - w_{t-1})^T \Sigma_{t-1}^{-1} (w - w_{t-1}) \\ & + \ell_t(w) \end{aligned} \tag{8}$$

$$\begin{aligned} \Sigma_t = \arg\min_\Sigma \; & \frac{1}{2\eta} \log \left(\frac{|\Sigma_{t-1}|}{|\Sigma|} \right) + \frac{1}{2\eta} \text{Tr}\left(\Sigma_{t-1}^{-1} \Sigma\right) \\ & + \frac{1}{2\eta} x_t^T \Sigma x_t \end{aligned} \tag{9}$$

The third term in (7), called the confidence term, gives us *adaptivity*, the notion that we should have smaller variance in the direction v as more data x_t

is seen in direction v. For example, if Σ is diagonal and x_t are indicator features, the confidence term then says that the weight for a rarer feature should have more variance and vice-versa. Recall that for generalized linear models $\nabla \ell_t(w) \propto x_t$; if we substitute $x_t = \alpha_t \nabla \ell_t(w)$ into (9), differentiate and solve, we get:

$$\begin{aligned}\Sigma_t^{-1} &= \Sigma_{t-1}^{-1} + x_t x_t^T \\ &= \Sigma_0^{-1} + \sum_{i=1}^{t} \alpha_i^2 \nabla \ell_i(w_{i-1}) \nabla \ell_i(w_{i-1})^T \end{aligned} \tag{10}$$

The precision Σ_t^{-1} generally grows as more data is seen. Frequently updated features receive an especially high precision, whereas the model maintains large variance for rarely seen features.

If we substitute (10) into (8), linearize the loss $\ell_t(w)$ as before, and solve, then we have the linearized AROW update

$$w_t = w_{t-1} - \eta \Sigma_t \nabla \ell_t(w_{t-1}) \tag{11}$$

which is also an adaptive update with per-coordinate learning rates specified by Σ_t (as opposed to $\Sigma_t^{1/2}$ in AdaGrad).

2.3 Comparing AdaGrad, MIRA, AROW

Compare (3) to (10) and observe that if we set $\Sigma_0^{-1} = 0$ and $\alpha_t = 1$, then the only difference between the AROW update (11) and the AdaGrad update (2) is a square root. Under a constant gradient, AROW decays the step size more aggressively ($1/t$) compared to AdaGrad ($1/\sqrt{t}$), and it is sensitive to the specification of Σ_0^{-1}.

Informally, SGD can be improved in the conservativity direction using MIRA so the updates do not overshoot. Second, SGD can be improved in the adaptivity direction using AdaGrad where the decaying stepsize is more robust and the adaptive stepsize allows better weight updates to features differing in sparsity and scale. Finally, AROW combines both adaptivity and conservativity. For MT, adaptivity allows us to deal with mixed dense/sparse features effectively without specific normalization.

Why do we choose AdaGrad over AROW? MIRA/AROW requires selecting the loss function $\ell(w)$ so that w_t can be solved in closed-form, by a quadratic program (QP), or in some other way that is better than linearizing. This usually means choosing a hinge loss. On the other hand, AdaGrad/linearized AROW only requires that the gradient of the loss function can be computed efficiently.

Algorithm 1 Adaptive online tuning for MT.

Require: Tuning set $\{f_i, e_i^{1:k}\}_{i=1:M}$
1: Set $w_0 = 0$
2: Set $t = 1$
3: **repeat**
4: **for** i in $1 \ldots M$ in random order **do**
5: Decode n-best list N_i for f_i
6: Sample pairs $\{d_{j,+}, d_{j,-}\}_{j=1:s}$ from N_i
7: Compute $\mathcal{D}_t = \{\phi(d_{j,+}) - \phi(d_{j,-})\}_{j=1:s}$
8: Set $g_t = \nabla \ell(\mathcal{D}_t;\ w_{t-1})\}$
9: Set $\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + g_t g_t^T$ ▷ Eq. (3)
10: Update $w_t = w_{t-1} - \eta \Sigma_t^{1/2} g_t$ ▷ Eq. (2)
11: Regularize w_t ▷ Eq. (15)
12: Set $t = t + 1$
13: **end for**
14: **until** convergence

Linearized AROW, however, is less robust than AdaGrad empirically[2] and lacks known theoretical guarantees. Finally, by using AdaGrad, we separate adaptivity from conservativity. Our experiments suggest that adaptivity is actually more important.

3 Adaptive Online MT

Algorithm 1 shows the full algorithm introduced in this paper. AdaGrad (lines 9–10) is a crucial piece, but the loss function, regularization technique, and parallelization strategy described in this section are equally important in the MT setting.

3.1 Pairwise Logistic Loss Function

Algorithm 1 lines 5–8 describe the gradient computation. We cast MT tuning as *pairwise ranking* (Herbrich et al., 1999, *inter alia*), which Hopkins and May (2011) applied to MT. The pairwise approach results in simple, convex loss functions suitable for online learning. The idea is that for any two derivations, the ranking predicted by the model should be consistent with the ranking predicted by a gold sentence-level metric G like BLEU+1 (Lin and Och, 2004).

Consider a single source sentence f with associated references $e^{1:k}$. Let d be a derivation in an n-best list of f that has the target $e = e(d)$ and the feature map $\phi(d)$. Let $M(d) = w \cdot \phi(d)$ be the model score. For any derivation d_+ that is better than d_- under G, we desire pairwise agreement such that

$$\begin{aligned} G\left(e(d_+), e^{1:k}\right) > G\left(e(d_-), e^{1:k}\right) \\ \iff M(d_+) > M(d_-) \end{aligned}$$

[2] According to experiments not reported in this paper.

Ensuring pairwise agreement is the same as ensuring $w \cdot [\phi(d_+) - \phi(d_-)] > 0$.

For learning, we need to select derivation pairs (d_+, d_-) to compute difference vectors $x_+ = \phi(d_+) - \phi(d_-)$. Then we have a 1-class separation problem trying to ensure $w \cdot x_+ > 0$. The derivation pairs are sampled with the algorithm of Hopkins and May (2011).

We compute difference vectors $\mathcal{D}_t = \{x_+^{1:s}\}$ (Algorithm 1 line 7) from s pairs (d_+, d_-) for source sentence f_t. We use the familiar logistic loss:

$$\ell_t(w) = \ell(\mathcal{D}_t, w) = -\sum_{x_+ \in \mathcal{D}_t} \log \frac{1}{1 + e^{-w \cdot x_+}} \tag{12}$$

Choosing the hinge loss instead of the logistic loss results in the 1-class SVM problem. The 1-class separation problem is equivalent to the binary classification problem with $x_+ = \phi(d_+) - \phi(d_-)$ as positive data and $x_- = -x_+$ as negative data, which may be plugged into an existing logistic regression solver.

We find that Algorithm 1 works best with mini-batches instead of single examples. In line 4 we simply partition the tuning set so that i becomes a mini-batch of examples.

3.2 Updating and Regularization

Algorithm 1 lines 9–11 compute the adaptive learning rate, update the weights, and apply regularization. Section 2.1 explained the AdaGrad learning rate computation. To update and regularize the weights we apply the Forward-Backward Splitting (FOBOS) (Duchi and Singer, 2009) framework, which separates the two operations. The two-step FOBOS update is

$$w_{t-\frac{1}{2}} = w_{t-1} - \eta_{t-1} \nabla \ell_{t-1}(w_{t-1}) \tag{13}$$

$$w_t = \arg\min_w \frac{1}{2} \|w - w_{t-\frac{1}{2}}\|_2^2 + \eta_{t-1} r(w) \tag{14}$$

where (13) is just an unregularized gradient descent step and (14) balances the regularization term $r(w)$ with staying close to the gradient step.

Equation (14) permits efficient L_1 regularization, which is well-suited for selecting good features from exponentially many irrelevant features (Ng, 2004). It is well-known that feature selection is very important for feature-rich MT. For example, simple indicator features like lexicalized re-ordering classes are potentially useful yet bloat the the feature set and, in the worst case, can negatively impact search. Some of the features generalize, but many do not. This was well understood in previous work, so heuristic filtering was usually applied (Chiang et al., 2009, *inter alia*). In contrast, we need only select an appropriate regularization strength λ.

Specifically, when $r(w) = \lambda \|w\|_1$, the closed-form solution to (14) is

$$w_t = \text{sign}(w_{t-\frac{1}{2}}) \Big[|w_{t-\frac{1}{2}}| - \eta_{t-1} \lambda \Big]_+ \tag{15}$$

where $[x]_+ = \max(x, 0)$ is the clipping function that in this case sets a weight to 0 when it falls below the threshold $\eta_{t-1}\lambda$. It is straightforward to adapt this to AdaGrad with diagonal Σ by setting each dimension of $\eta_{t-1,j} = \eta \Sigma_{t,jj}^{-\frac{1}{2}}$ and by taking element-wise products.

We find that $\nabla \ell_{t-1}(w_{t-1})$ only involves several hundred active features for the current example (or mini-batch). However, naively following the FOBOS framework requires updating millions of weights. But a practical benefit of FOBOS is that we can do lazy updates on just the active dimensions without any approximations.

Algorithm 2 "Stale gradient" parallelization method for Algorithm 1.

Require: Tuning set $\{f_i, e_i^{1:k}\}_{i=1:M}$
1: Initialize threadpool $p_1, \ldots, p_j$
2: Set $t = 1$
3: **repeat**
4: **for** i in $1 \ldots M$ in random order **do**
5: Wait until any thread p is idle
6: Send $(f_i, e_i^{1:k}, t)$ to p ▷ Alg. 1 lines 5–8
7: **while** $\exists\ p'$ done with gradient $g_{t'}$ **do** ▷ $t' \leq t$
8: Update $w_t = w_{t-1} - \eta g_{t'}$ ▷ Alg. 1 lines 9–11
9: Set $t = t + 1$
10: **end while**
11: **end for**
12: **until** convergence

3.3 Parallelization

Algorithm 1 is inherently sequential like standard online learning. This is undesirable in MT where decoding is costly. We therefore parallelize the algorithm with the "stale gradient" method of Langford et al. (2009) (Algorithm 2). A fixed threadpool of workers computes gradients in parallel and sends them to a master thread, which updates a central weight vector. Crucially, the weight updates need not be applied in order, so synchronization is unnecessary; the workers only idle at the end of an epoch. The consequence is that the update in line 8 of Algorithm 2 is with respect to gradient $g_{t'}$ with $t' \leq t$. Langford et al. (2009) gave convergence results for

stale updating, but the bounds do not apply to our setting since we use L_1 regularization. Nevertheless, Gimpel et al. (2010) applied this framework to other non-convex objectives and obtained good empirical results.

Our asynchronous, stochastic method has practical appeal for MT. During a tuning run, the online method decodes the tuning set under many more weight vectors than a MERT-style batch method. This characteristic may result in broader exploration of the search space, and make the learner more robust to local optima local optima (Liang and Klein, 2009; Bottou and Bousquet, 2011, *inter alia*). The adaptive algorithm identifies appropriate learning rates for the mixture of dense and sparse features. Finally, large data structures such as the language model (LM) and phrase table exist in shared memory, obviating the need for remote queries.

4 Experiments

We built Arabic-English and Chinese-English MT systems with Phrasal (Cer et al., 2010), a phrase-based system based on alignment templates (Och and Ney, 2004). The corpora[3] in our experiments (Table 1) derive from several LDC sources from 2012 and earlier. We de-duplicated each bitext according to exact string match, and ensured that no overlap existed with the test sets. We produced alignments with the Berkeley aligner (Liang et al., 2006b) with standard settings and symmetrized via the grow-diag heuristic.

For each language we used SRILM (Stolcke, 2002) to estimate 5-gram LMs with modified Kneser-Ney smoothing. We included the monolingual English data and the respective target bitexts.

4.1 Feature Templates

The baseline "dense" model contains 19 features: the nine Moses baseline features, the hierarchical lexicalized re-ordering model of Galley and Manning (2008), the (log) count of each rule, and an indicator for unique rules.

To the dense features we add three high dimensional "sparse" feature sets. **Discriminative phrase table** (PT): indicators for each rule in the phrase table. **Alignments** (AL): indicators for phrase-internal alignments and deleted (unaligned) source words. **Discriminative reordering** (LO): indicators for eight lexicalized reordering classes, including the six standard monotone/swap/discontinuous classes plus the two simpler Moses monotone/non-monotone classes.

[3]We tokenized the English with packages from the Stanford Parser (Klein and Manning, 2003) according to the Penn Treebank standard (Marcus et al., 1993), the Arabic with the Stanford Arabic segmenter (Green and DeNero, 2012) according to the Penn Arabic Treebank standard (Maamouri et al., 2008), and the Chinese with the Stanford Chinese segmenter (Chang et al., 2008) according to the Penn Chinese Treebank standard (Xue et al., 2005).

	Bilingual		Monolingual
	Sentences	*Tokens*	*Tokens*
Ar-En	6.6M	375M	990M
Zh-En	9.3M	538M	

Table 1: Bilingual and monolingual corpora used in these experiments. The monolingual English data comes from the AFP and Xinhua sections of English Gigaword 4 (LDC2009T13).

4.2 Tuning Algorithms

The primary baseline is the dense feature set tuned with MERT (Och, 2003). The Phrasal implementation uses the line search algorithm of Cer et al. (2008), uniform initialization, and 20 random starting points.[4] We tuned according to BLEU-4 (Papineni et al., 2002).

We built high dimensional baselines with two different algorithms. First, we tuned with batch PRO using the default settings in Phrasal (L_2 regularization with σ=0.1). Second, we ran the k-best batch MIRA (kb-MIRA) (Cherry and Foster, 2012) implementation in Moses. We did implement an online version of MIRA, and in small-scale experiments found that the batch variant worked just as well. Cherry and Foster (2012) reported the same result, and their implementation is available in Moses. We ran their code with standard settings.

Moses[5] also contains the discriminative phrase table implementation of (Hasler et al., 2012b), which is identical to our implementation using Phrasal. Moses and Phrasal accept the same phrase table and LM formats, so we kept those data structures in common. The two decoders also use the same multi-stack beam search (Och and Ney, 2004).

For our method, we used uniform initialization, 16 threads, and a mini-batch size of 20. We found that η=0.02 and λ=0.1 worked well on development sets for both languages. To compute the gradients

[4]Other system settings for all experiments: distortion limit of 5, a maximum phrase length of 7, and an n-best size of 200.

[5]v1.0 (28 January 2013)

Model	#features	Algorithm	Tuning Set		MT02	MT03	MT04	MT09
Dense	19	MERT	MT06	45.08	51.32	52.26	51.42	48.44
Dense	19	This paper	MT06	44.19	51.42	52.52	50.16	48.13
+PT	151k	kb-MIRA	MT06	42.08	47.25	48.98	47.08	45.64
+PT	23k	PRO	MT06	44.31	51.06	52.18	50.23	47.52
+PT	50k	This paper	MT06	50.61	***51.71***	52.89	50.42	**48.74**
+PT+AL+LO	109k	PRO	MT06	44.87	51.25	52.43	50.05	47.76
+PT+AL+LO	242k	This paper	MT06	57.84	52.45	53.18	51.38	49.37
Dense	19	MERT	MT05/6/8	49.63	51.60	52.29	51.73	48.68
+PT+AL+LO	390k	This paper	MT05/6/8	58.20	**53.61**	**54.99**	**52.79**	**49.94**
(Chiang, 2012)*	10-20k	MIRA	MT04/6	–	–	–	–	45.90
(Chiang, 2012)*	10-20k	AROW	MT04/6	–	–	–	–	47.60
				#sentences	*728*	*663*	*1,075*	*1,313*

Table 2: Ar-En results [BLEU-4 % uncased] for the NIST tuning experiment. The tuning and test sets each have four references. MT06 has 1,717 sentences, while the concatenated MT05/6/8 set has 4,213 sentences. **Bold** indicates statistical significance relative to the *best* baseline in each block at $p < 0.001$; *bold-italic* at $p < 0.05$. We assessed significance with the permutation test of Riezler and Maxwell (2005). (*) Chiang (2012) used a similar-sized bitext, but two LMs trained on twice as much monolingual data.

Model	#features	Algorithm	Tuning Set		MT02	MT03	MT04
Dense	19	MERT	MT06	33.90	35.72	33.71	34.26
Dense	19	This paper	MT06	32.60	36.23	**35.14**	***34.78***
+PT	105k	kb-MIRA	MT06	29.46	30.67	28.96	30.05
+PT	26k	PRO	MT06	33.70	36.87	34.62	34.80
+PT	66k	This paper	MT06	33.90	36.09	34.86	34.73
+PT+AL+LO	148k	PRO	MT06	34.81	36.31	33.81	34.41
+PT+AL+LO	344k	This paper	MT06	38.99	36.40	35.07	34.84
Dense	19	MERT	MT05/6/8	32.36	35.69	33.83	34.33
+PT+AL+LO	487k	This paper	MT05/6/8	37.64	**37.81**	**36.26**	**36.15**
				#sentences	*878*	*919*	*1,597*

Table 3: Zh-En results [BLEU-4 % uncased] for the NIST tuning experiment. MT05/6/8 has 4,103 sentences. OpenMT 2009 did not include Zh-En, hence the asymmetry with Table 2.

we sampled 15 derivation pairs for each tuning example and scored them with BLEU+1.

4.3 NIST OpenMT Experiment

The first experiment evaluates our algorithm when tuning and testing on standard test sets, each with four references. When we add features, our algorithm tends to overfit to a standard-sized tuning set like MT06. We thus concatenated MT05, MT06, and MT08 to create a larger tuning set.

Table 2 shows the Ar-En results. Our algorithm is competitive with MERT in the low dimensional "dense" setting. The discriminative phrase table (PT) results in models that are a thousand times larger. Our algorithm compares favorably to both PRO and kb-MIRA. Finally, we add all of the sparse features. As expected, our algorithm overfits badly to MT06. The addition of more in-domain tuning data yields the best results. For comparison, we also ran the Moses version of PRO with the discriminative phrase table on MT05/6/8 but found that it underperformed, e.g. 44.96 BLEU on MT09.

The full feature set PT+AL+LO does help. With the PT feature set alone, our algorithm tuned on MT05/6/8 scores well below the best model, e.g. 48.56 BLEU on MT09. For Ar-En, our algorithm

Model	#features	Algorithm	Tuning Set		#refs	bitext5k-test	MT04
Dense	19	MERT	MT06	45.08	4	39.28	51.42
+PT	72k	This paper	MT05/6/8	51.29	4	39.50	50.60
+PT	79k	This paper	bitext5k	44.79	1	43.85	45.73
+PT+AL+LO	647k	This paper	bitext15k	45.68	1	**43.93**	45.24

Table 4: Ar-En results [BLEU-4 % uncased] for the bitext tuning experiment. Statistical significance is relative to the Dense baseline. We include MT04 for comparison to the NIST genre.

Model	#features	Algorithm	Tuning Set		#refs	bitext5k-test	MT04
Dense	19	MERT	MT06	33.90	4	33.44	34.26
+PT	97k	This paper	MT05/6/8	34.45	4	35.08	35.19
+PT	67k	This paper	bitext5k	36.26	1	36.01	33.76
+PT+AL+LO	536k	This paper	bitext15k	37.57	1	**36.30**	34.05

Table 5: Zh-En results [BLEU-4 % uncased] for the bitext tuning experiment.

thus has the desirable property of benefiting from more and better features, and more data.

Table 3 shows Zh-En results. Somewhat surprisingly our algorithm improves over MERT in the dense setting. When we add the discriminative phrase table, our algorithm improves over kb-MIRA, and over batch PRO on two evaluation sets. With all features and the MT05/6/8 tuning set, we improve significantly over all other models. PRO learns a smaller model with the PT+AL+LO feature set which is surprising given that it applies L_2 regularization (AdaGrad uses L_1). We speculate that this may be an consequence of stochastic learning. Our algorithm decodes each example with a new weight vector, thus exploring more of the search space for the same tuning set.

4.4 Bitext Tuning Experiment

Tables 2 and 3 show that adding tuning examples improves translation quality. Nevertheless, even the larger tuning set is small relative to the bitext from which rules were extracted. He and Deng (2012) and Simianer et al. (2012) showed significant translation quality gains by tuning on the bitext. However, their bitexts matched the genre of their test sets. Our bitexts, like those of most large-scale systems, do not. Domain mismatch matters for the dense feature set (Haddow and Koehn, 2012). We show that it also matters for feature-rich MT.

Before aligning each bitext, we randomly sampled and sequestered 5k and 15k sentence tuning sets, and a 5k test set. We prevented overlap between the tuning sets and the test set. We then tuned a dense model with MERT on MT06, and feature-rich models on both MT05/6/8 and the bitext tuning set. Table 4 shows the Ar-En results. When tuned on bitext5k the translation quality gains are significant for bitext5k-test relative to tuning on MT05/6/8, which has multiple references. However, the bitext5k models do not generalize as well to the NIST evaluation sets as represented by the MT04 result. Table 5 shows similar trends for Zh-En.

$\mathcal{D}_A$	$\mathcal{D}_B$	$\lvert A\rvert$	$\lvert B\rvert$	$\lvert A \cap B\rvert$
MT04	MT06	70k	72k	5.9k
MT04	MT568	70k	96k	7.6k
MT04	bitext5k	70k	67k	4.4k
MT04	bitext15k	70k	310k	10.5k
5ktest	bitext5k	82k	67k	5.6k
5ktest	bitext15k	82k	310k	14k

Table 6: Number of overlapping phrase table (+PT) features on various Zh-En dataset pairs.

5 Analysis

5.1 Feature Overlap Analysis

How many sparse features appear in both the tuning and test sets? In Table 6, A is the set of phrase table features that received a non-zero weight when tuned on dataset $\mathcal{D}_A$ (same for B). Column $\mathcal{D}_A$ lists several Zh-En test sets used and column $\mathcal{D}_B$ lists tuning sets. Our experiments showed that tuning on MT06 generalizes better to MT04 than tuning on bitext5k, whereas tuning on bitext5k general-

izes better to bitext5k-test than tuning on MT06. These trends are consistent with the level of feature overlap. Phrase table features in $A \cap B$ are overwhelmingly short, simple, and correct phrases, suggesting L_1 regularization is effective for feature selection. It is also important to balance the number of features with how well weights can be learned for those features, as tuning on bitext15k produced higher coverage for MT04 but worse generalization than tuning on MT06.

5.2 Domain Adaptation Analysis

To understand the domain adaptation issue we compared the non-zero weights in the discriminative phrase table (PT) for Ar-En models tuned on bitext5k and MT05/6/8. Table 7 illustrates a statistical idiosyncrasy in the data for the American and British spellings of program/programme. The mass is concentrated along the diagonal, probably because MT05/6/8 was prepared by NIST, an American agency, while the bitext was collected from many sources including Agence France Presse.

Of course, this discrepancy is consequential for both dense and feature-rich models. However, we observe that the feature-rich models fit the tuning data more closely. For example, the MT05/6/8 model learns rules like يتضمن برنامج → *program includes*, برنامج → *program of*, and نافذة البرنامج → *program window*. Crucially, it does not learn the basic rule برنامج → *program*.

In contrast, the bitext5k model contains basic rules such برنامج → *programme*, هذا البرنامج → *this programme*, and ذلك البرنامج → *that programme*. It also contains more elaborate rules such as كانت نفقات البرنامج → *programme expenses were* and برامج الرحلات الفضائية المأهولة → *manned space flight programmes*. We observed similar trends for 'defense/defence', 'analyze/analyse', etc. This particular genre problem could be addressed with language-specific pre-processing, but our system solves it in a data-driven manner.

5.3 Re-ordering Analysis

We also analyzed re-ordering differences. Arabic matrix clauses tend to be verb-initial, meaning that the subject and verb must be swapped when translating to English. To assess re-ordering differences—if any—between the dense and feature-rich models, we selected all MT09 segments that began with one of seven common verbs: قال *qaal* 'said', صرح *SrH* 'declared', أشار *ashaar* 'indicated', كان *kaan* 'was', ذكر *dhkr* 'commented', أضاف *aDaaf* 'added', أعلن *a^{c}ln* 'announced'. We compared the output of the MERT Dense model to our method with the full feature set, both tuned on MT06. Of the 208 source segments, 32 of the translation pairs contained different word order in the matrix clause. Our feature-rich model was correct 18 times (56.3%), Dense was correct 4 times (12.5%), and neither method was correct 10 times (31.3%).

	# bitext5k	# MT05/6/8
programme	185	0
program	19	449
PT rules w/ *programme*	353	79
PT rules w/ *program*	9	31

Table 7: Top: comparison of token counts in two Ar-En tuning sets for *programme* and *program*. Bottom: rule counts in the discriminative phrase table (PT) for models tuned on the two tuning sets. Both spellings correspond to the Arabic برنامج.

(1) ref: lebanese prime minister , fuad siniora , announced
  a. and lebanese prime minister fuad siniora that
  b. the lebanese prime minister fouad siniora announced

(2) ref: the newspaper and television reported
  a. she said the newspaper and television
  b. television and newspaper said

In (1) the dense model (1a) drops the verb while the feature-rich model correctly re-orders and inserts it after the subject (1b). The coordinated subject in (2) becomes an embedded subject in the dense output (2a). The feature-rich model (2b) performs the correct re-ordering.

5.4 Runtime Comparison

Table 8 compares our method to standard implementations of the other algorithms. MERT parallelizes easily but runtime increases quadratically with n-best list size. PRO runs (single-threaded) L-BFGS to convergence on every epoch, a potentially slow procedure for the larger feature set. Moreover, both the Phrasal and Moses PRO implementations use L_2 regularization, which regularizes every weight

		epochs	min.
MERT	Dense	22	180
PRO	+PT	25	35
kb-MIRA*	+PT	26	25
This paper	+PT	10	10
PRO	+PT+AL+LO	13	150
This paper	+PT+AL+LO	5	15

Table 8: Epochs to convergence ("epochs") and approximate runtime per epoch in minutes ("min.") for selected Zh-En experiments tuned on MT06. All runs executed on the same dedicated system with the same number of threads. (*) Moses and kb-MIRA are written in C++, while all other rows refer to Java implementations in Phrasal.

on every update. kb-MIRA makes multiple passes through the n-best lists during each epoch. The Moses implementation parallelizes decoding but weight updating is sequential.

The core of our method is an inner product between the adaptive learning rate vector and the gradient. This is easy to implement and is very fast even for large feature sets. Since we applied lazy regularization, this inner product usually involves hundred-dimensional vectors. Finally, our method does not need to accumulate n-best lists, a practice that slows down the other algorithms.

6 Related Work

Our work relates most closely to that of Hasler et al. (2012b), who tuned models containing both sparse and dense features with Moses. A discriminative phrase table helped them improve slightly over a dense, online MIRA baseline, but their best results required initialization with MERT-tuned weights and re-tuning a single, shared weight for the discriminative phrase table with MERT. In contrast, our algorithm learned good high dimensional models from a uniform starting point.

Chiang (2012) adapted AROW to MT and extended previous work on online MIRA (Chiang et al., 2008; Watanabe et al., 2007). It was not clear if his improvements came from the novel Hope/Fear search, the conservativity gain from MIRA/AROW by solving the QP exactly, adaptivity, or sophisticated parallelization. In contrast, we show that AdaGrad, which ignores conservativity and only capturing adaptivity, is sufficient.

Simianer et al. (2012) investigated SGD with a pairwise perceptron objective. Their best algorithm used *iterative parameter mixing* (McDonald et al., 2010), which we found to be slower than the stale gradient method in section 3.3. They regularized once at the end of each epoch, whereas we regularized each weight update. An empirical comparison of these two strategies would be an interesting future contribution.

Watanabe (2012) investigated SGD and even randomly selected pairwise samples as we did. He considered both softmax and hinge losses, observing better results with the latter, which solves a QP. Their parallelization strategy required a line search at the end of each epoch.

Many other discriminative techniques have been proposed based on: ramp loss (Gimpel, 2012); hinge loss (Cherry and Foster, 2012; Haddow et al., 2011; Arun and Koehn, 2007); maximum entropy (Xiang and Ittycheriah, 2011; Ittycheriah and Roukos, 2007; Och and Ney, 2002); perceptron (Liang et al., 2006a); and structured SVM (Tillmann and Zhang, 2006). These works use radically different experimental setups, and to our knowledge only (Cherry and Foster, 2012) and this work compare to at least two high dimensional baselines. Broader comparisons, though time-intensive, could help differentiate these methods.

7 Conclusion and Outlook

We introduced a new online method for tuning feature-rich translation models. The method is faster per epoch than MERT, scales to millions of features, and converges quickly. We used efficient L_1 regularization for feature selection, obviating the need for the feature scaling and heuristic filtering common in prior work. Those comfortable with implementing vanilla SGD should find our method easy to implement.

Our analysis showed that feature-rich models fit the tuning data tighter than dense models, so the test and tuning sets should be matched. We suspect that larger scale bitext tuning—perhaps in a leave-one-out scheme—might mitigate the negative effects of genre differences. Nevertheless, we showed that for two different genres our method produced significant translation quality gains over a traditional dense model. Even basic discriminative features were effective, so we believe that our work enables fresh approaches to more sophisticated feature engineering for MT.

Acknowledgments We thank John DeNero for helpful comments on this work. The first author is supported by a National Science Foundation Graduate Research Fellowship. We also acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA or the US government.

References

A. Arun and P. Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *MT Summit XI*.

L. Bottou and O. Bousquet. 2011. The tradeoffs of large scale learning. In *Optimization for Machine Learning*, pages 351–368. MIT Press.

D. Cer, D. Jurafsky, and C. D. Manning. 2008. Regularization and search for minimum error rate training. In *WMT*.

D. Cer, M. Galley, D. Jurafsky, and C. D. Manning. 2010. Phrasal: A statistical machine translation toolkit for exploring new model features. In *HLT-NAACL, Demonstration Session*.

P-C. Chang, M. Galley, and C. D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *WMT*.

C. Cherry and G. Foster. 2012. Batch tuning strategies for statistical machine translation. In *HLT-NAACL*.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP*.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *HLT-NAACL*.

D. Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *JMLR*, 13:1159–1187.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

K. Crammer, A. Kulesza, and M. Dredze. 2009. Adaptive regularization of weight vectors. In *NIPS*.

J. Duchi and Y. Singer. 2009. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

M. Galley and C. D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *EMNLP*.

K. Gimpel and N. A. Smith. 2012. Structured ramp loss minimization for machine translation. In *HLT-NAACL*.

K. Gimpel, D. Das, and N. A. Smith. 2010. Distributed asynchronous online learning for natural language processing. In *CoNLL*.

K. Gimpel. 2012. *Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation*. Ph.D. thesis, Language Technologies Institute, Carnegie Mellon University.

S. Green and J. DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *ACL*.

B. Haddow and P. Koehn. 2012. Analysing the effect of out-of-domain data on SMT systems. In *WMT*.

B. Haddow, A. Arun, and P. Koehn. 2011. SampleRank training for phrase-based machine translation. In *WMT*.

E. Hasler, P. Bell, A. Ghoshal, B. Haddow, P. Koehn, F. McInnes, et al. 2012a. The UEDIN systems for the IWSLT 2012 evaluation. In *IWSLT*.

E. Hasler, B. Haddow, and P. Koehn. 2012b. Sparse lexicalised features and topic adaptation for SMT. In *IWSLT*.

X. He and L. Deng. 2012. Maximum expected BLEU training of phrase and lexicon translation models. In *ACL*.

R. Herbrich, T. Graepel, and K. Obermayer. 1999. Support vector learning for ordinal regression. In *ICANN*.

M. Hopkins and J. May. 2011. Tuning as ranking. In *EMNLP*.

A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *HLT-NAACL*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL, Demonstration Session*.

J. Langford, A. J. Smola, and M. Zinkevich. 2009. Slow learners are fast. In *NIPS*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *HLT-NAACL*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006a. An end-to-end discriminative approach to machine translation. In *ACL*.

P. Liang, B. Taskar, and D. Klein. 2006b. Alignment by agreement. In *NAACL*.

C.-Y. Lin and F. J. Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING*.

M. Maamouri, A. Bies, and S. Kulick. 2008. Enhancing the Arabic Treebank: A collaborative effort toward new annotation guidelines. In *LREC*.

M. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

R. McDonald, K. Hall, and G. Mann. 2010. Distributed training strategies for the structured perceptron. In *NAACL-HLT*.

A. Y. Ng. 2004. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *ICML*.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

S. Riezler and J. T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing in MT. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (MTSE)*.

P. Simianer, S. Riezler, and C. Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *ACL*.

A Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *ICSLP*.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical MT. In *ACL-COLING*.

T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*.

T. Watanabe. 2012. Optimized online rank learning for machine translation. In *HLT-NAACL*. Association for Computational Linguistics.

B. Xiang and A. Ittycheriah. 2011. Discriminative feature-tied mixture modeling for statistical machine translation. In *ACL-HLT*.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.