SampleRank Training for Phrase-Based Machine Translation

Barry Haddow
School of Informatics
University of Edinburgh
bhaddow@inf.ed.ac.uk

Abhishek Arun
Microsoft UK
abarun@microsoft.com

Philipp Koehn
School of Informatics
University of Edinburgh
pkoehn@inf.ed.ac.uk

Abstract

Statistical machine translation systems are normally optimised for a chosen gain function (metric) by using MERT to find the best model weights. This algorithm suffers from stability problems and cannot scale beyond 20-30 features. We present an alternative algorithm for discriminative training of phrase-based MT systems, SampleRank, which scales to hundreds of features, equals or beats MERT on both small and medium sized systems, and permits the use of sentence or document level features. SampleRank proceeds by repeatedly updating the model weights to ensure that the ranking of output sentences induced by the model is the same as that induced by the gain function.

1 Introduction

In phrase-based machine translation (PBMT), the standard approach is to express the probability distribution $p(a, e|f)$ (where f is the source sentence and (a, e) is the aligned target sentence) in terms of a linear model based on a small set of feature functions

$$p(a, e|f) \propto \exp\left(\sum_{i=1}^{n} w_i h_i(a, e, f)\right) \qquad (1)$$

The feature functions $\{h_i\}$ typically include log probabilities of generative models such as translation, language and reordering, as well as non-probabilistic features such as word, phrase and distortion penalties. The feature weights $\mathbf{w} = \{w_i\}$ are normally trained using MERT (minimum error rate training) (Och, 2003), to maximise performance as measured by an automated metric such as BLEU (Papineni et al., 2002). MERT training uses a parallel data set (known as the tuning set) consisting of about 1000-2000 sentences, distinct from the data set used to build the generative models. Optimising the weights in Equation (1) is often referred to as *tuning* the MT system, to differentiate it from the process of training the generative models.

MERT's inability to scale beyond 20-30 features, as well as its instability (Foster and Kuhn, 2009) have led to investigation into alternative ways of tuning MT systems. The development of tuning methods is complicated, however by, the use of BLEU as an objective function. This objective in its usual form is not differentiable, and has a highly non-convex error surface (Och, 2003). Furthermore BLEU is evaluated at the corpus level rather than at the sentence level, so tuning methods either have to consider the entire corpus, or resort to a sentence-level approximation of BLEU. It is unlikely, however, that the difficulties in discriminative MT tuning are due solely to the use of BLEU as a metric – because evaluation of translation is so difficult, any reasonable gain function is likely to have a complex relationship with the model parameters.

Gradient-based tuning methods, such as minimum risk training, have been investigated as possible alternatives to MERT. Expected BLEU is normally adopted as the objective since it is differentiable and so can be optimised by a form of stochastic gradient ascent. The feature expectations required for the gradient calculation can be obtained from n-best lists or lattices (Smith and Eisner, 2006; Li and Eisner, 2009), or using sampling (Arun et al., 2010), both of which can be computationally expensive.

Margin-based techniques such as perceptron training (Liang et al., 2006) and MIRA (Chiang et al., 2008; Watanabe et al., 2007) have also been shown to be able to tune MT systems and scale to large numbers of features, but these generally involve repeatedly decoding the tuning set (and so are expensive) and require sentence-level approximations to the BLEU objective.

In this paper we present an alternative method of tuning MT systems known as *SampleRank*, which has certain advantages over other methods in use today. SampleRank operates by repeatedly sampling pairs of translation hypotheses (for a given source sentence) and updating the feature weights if the ranking induced by the MT model (1) is different from the ranking induced by the gain function (i.e. BLEU). By considering the translation hypotheses in batches, it is possible to directly optimise corpus level metrics like BLEU without resorting to sentence level approximations.

Tuning using SampleRank does not limit the size of the feature set in the same way as MERT does, and indeed it will be shown that SampleRank can successfully train a model with several hundred features. Using just the core PBMT features and training using SampleRank will be shown to achieve BLEU scores which equal or exceed those produced by MERT trained models.

Since SampleRank does not require repeated decoding of the tuning set, and is easily parallelisable, it can run at an acceptable speed, and since it always maintains a complete translation hypothesis, it opens up the possibility of sentence or document level features[1].

2 Method

2.1 SampleRank Training

SampleRank (Culotta, 2008; Wick et al., 2009) is an online training algorithm that was introduced for parameter learning in weighted logics, and has been applied to complex graphical models (Wick et al., 2011). Assume a probabilistic model $p(y|x)$ admitting a log-linear parametrisation

$$p(y|x) \propto \exp \sum_i (w_i \phi_i(x, y)) \qquad (2)$$

[1]As long as the batches described in Section 2.2.1 respect document boundaries.

where $\{\phi_i\}$ are a set of feature functions and $\{w_i\}$ are corresponding feature weights. SampleRank can be used to optimise the feature weights to maximise a given gain function.

SampleRank is a supervised training algorithm, requiring a set of labelled training data $D = \{(x^1, y^1\}, \ldots, (x^n, y^n)\}$, where the x^i are the inputs and the y^i the outputs. The algorithm works by considering each training example (x^i, y^i) in turn, and repeatedly sampling pairs of outputs from a neighbourhood defined in the space of all possible outputs, updating the weights when the ranking of the pair due to the model scores is different from the ranking due to the gain function. So if the sampled pair of outputs for x^i is (y, y'), where $p(y'|x^i) > p(y|x^i)$, the weights are updated iff $gain(y', y^i) < gain(y, y^i)$.

The sampled pairs are drawn from a chain which can be constructed in a similar way to an MCMC (Markov Chain Monte Carlo) chain.

In (Culotta, 2008) different strategies are explored for building the chain, choosing the neighbourhood and updating the weights.

2.2 SampleRank Training for Machine Translation

We adapted SampleRank for the tuning of PBMT systems, as summarised in Algorithm 1. The definitions of the functions in the algorithm (described in the following subsections) draw inspiration from work on MIRA training for MT (Watanabe et al., 2007; Chiang et al., 2008). SampleRank is used to optimise the parameter weights in (1) using the tuning set.

2.2.1 Gain Function

The first thing that needs to be defined in Algorithm 1 is the gain function. For this we use BLEU, the most popular gain function for automated MT evaluation, although the procedure described here will work with any gain function that can be evaluated quickly. Using BLEU, however, creates a problem, as BLEU is defined at the corpus level rather than the sentence level, and in previous work on SampleRank, the training data is processed one example at a time. In other work on online training for SMT, (Liang et al., 2006; Chiang et al., 2008), sentence-level approximations to BLEU were

Algorithm 1 The SampleRank algorithm for tuning phrase-based MT systems.

```
Require: Tuning data:
    $D = \{(f^1, e^1), \ldots, (f^n, e^n)\}$
Require: $gain(y, y')$: A function which scores a
    set of hypotheses ($y'$) against a set of references ($y$).
Require: $score(x, y)$: A function which computes
    a model score for a set of hypotheses $y$ and
    source sentences $x$.
 1: for $epoch = 1$ to number of epochs do
 2:   $A \leftarrow D$
 3:   while $A$ is non-empty do
 4:     Pick $(x, y)$, a batch of sentence pairs, randomly from $A$, and remove.
 5:     Initialise $y_0$, a set of translation hypotheses for $x$.
 6:     for $s = 1$ to number of samples do
 7:       $N \leftarrow ChooseNeighbourhood(y_{s-1})$
 8:       $y' \leftarrow ChooseSample(N)$
 9:       $y^+ \leftarrow ChooseOracle(N)$
10:       if $\frac{gain(y,y') - gain(y,y^+)}{score(x,y') - score(x,y^+)} < 0$ then
11:         $UpdateWeights()$
12:       end if
13:       $y_s \leftarrow y'$
14:     end for
15:   end while
16: end for
```

employed, however in this work we directly optimise corpus BLEU by processing the data in small batches. Using batches was found to work better than processing the data sentence by sentence.

So the while loop in Algorithm 1 iterates through the tuning data in batches of parallel sentences, rather than single sentences. One complete pass through the tuning data is known as an *epoch*, and normally SampleRank training is run for several epochs. The gain on a particular batch is calculated by scoring the current set of hypotheses for the whole batch against the references for that batch. When calculating BLEU, a smoothing constant of 0.01 is added to all counts in order to avoid zero counts.

2.2.2 Sample Generation

For each iteration of the while loop in Algorithm 1, a new batch of parallel sentences is chosen from the tuning set, and a corresponding new set of translation hypotheses must be generated (the y_0 in line 5 of Algorithm 1). These initial hypotheses are generated by glossing. For each word in the source, the most likely translation option (according to the weighted phrase-internal score) is selected, and these translations are joined together monotonically. This method of initialisation was chosen because it was simple and fast, and experiments with an alternative method of initialisation (where the decoder was run with random scores assigned to hypotheses) showed very little difference in performance.

Once the initial set of hypotheses for the new batch is created, the SampleRank innermost loop (lines 6-14 in Algorithm 1) proceeds by repeatedly choosing a sample hypothesis set (y') and an oracle hypothesis set (y^+), corresponding to the source side of the batch (x).

Given the current hypothesis set $y_{s-1} = (e_1, \ldots, e_k)$, the sample and oracle are chosen as follows. Firstly, a hypothesis e_j is selected randomly from y_{s-1} , and a neighbourhood of alternate hypotheses $N \ni e_j$ generated using operators from Arun et al. (2009) (explained shortly). Model scores are calculated for all the hypotheses in N, converted to probabilities using Equation (1), and a sample e'_j taken from N using these probabilities. The sample hypothesis set (y') is then the current hypothesis set (y_{s-1}) with e_j replaced by e'_j. The oracle is created, analogously Chiang et al. (2008), by choosing $e^+_j \in N$ to maximise the sum of gain (calculated on the batch) and model score. The oracle hypothesis set (y^+) is then y_{s-1} with e_j replaced by e^+_j.

We now describe how the neighbourhood is chosen. Given a single hypothesis e_j, a neighbourhood is generated by first randomly choosing one of the two operators MERGE-SPLIT or REORDER, then randomly choosing a point of application for the operator, then applying it to generate the neighbourhood. The MERGE-SPLIT operator can be applied at any inter-word position, and generates its neighbourhood by listing all hypotheses obtained by optionally merging or splitting the phrases(s) touching that position, and retranslating them. The REORDER operator applies at a pair of target phrases (subject to distortion limits) and generates a neighbourhood containing two hypotheses, one with the original order and one with the chosen phrases swapped. The distortion limits and translation option pruning used by the operators matches those used in decoding, so together they are able to explore the same hypothesis space as the decoder. A fuller explanation of the two operators is give in Arun et al. (2009).

2.2.3 Weight Updates

After choosing the sample and oracle hypothesis set (y' and y^+), the weight update may be performed. The weights of the model are updated if the relative ranking of the sample hypothesis set and the oracle hypothesis set provided by the model score is different from that provided by the gain. The model score function $score(x, y)$ is defined for a hypothesis set $y = e_1, \dots e_k$ as follows:

$$score(x,y) = \sum_{j=1}^{k} \left(\sum_{i=1}^{n} w_i h_i(a_j, e_j, f_j) \right) \tag{3}$$

where $x = f_1, \dots f_k$ are the corresponding source sentences. The weight update is performed iff $score(x, y') \neq score(x, y^+)$ and the following condition is satisfied:

$$\frac{gain(y, y') - gain(y, y^+)}{score(x, y') - score(x, y^+)} < 0 \tag{4}$$

where the $gain()$ function is just the BLEU score.

The weight update used in this work is a MIRA-like update from $\mathbf{w}_{s-1}$ to $\mathbf{w}_s$ defined as follows:

$$\mathbf{w}_s = \arg\min_{\mathbf{w}} \left(\|\mathbf{w} - \mathbf{w}_{s-1}\| + C\xi \right) \tag{5}$$

subject to

$$\begin{aligned} &score_{\mathbf{w}}(x, y^+) - score_{\mathbf{w}}(x, y') + \xi \\ &\qquad \geq M \cdot (gain(y, y^+) - gain(y, y')) \end{aligned} \tag{6}$$

The margin scaling M is set to be $gain(y, y^+)$, so that ranking violations of low BLEU solutions are assigned a lower importance than ranking violations of high BLEU solutions. The ξ in (5) is a slack variable, whose influence is controlled by C (set to 0.01), and which has the effect of "clipping" the magnitude of the weight updates. Since there is only one constraint, there is no need to use an iterative method such as Hildreth's, because it is straightforward to solve the optimisation in (5) and (6) exactly using its Lagrangian dual, following (Crammer et al., 2006). The weight update is then given by

$$\mathbf{w}_s = \mathbf{w}_{s-1} + \min\left(\frac{b}{\|\mathbf{a}\|^2}, C \right) \mathbf{a}$$

$$\begin{aligned} \text{where} \quad & \mathbf{a} = \mathbf{h}(a_j^+, e_j^+, f_j) - \mathbf{h}(a_j', e_j', f_j) \\ \text{and} \quad & b = M\left(gain(y, y^+) - gain(y, y')\right) \\ & \qquad - \left(score(x, y^+) - gain(y, y')\right) \end{aligned}$$

After updating the weights, the current hypothesis set (y_s) is updated to be the sample hypothesis set (y'), as in line 13 of Algorithm 1, and then the next sample is generated.

2.2.4 Implementation Considerations

After each iteration of the inner loop of Algorithm 1, the weights are collected, and the overall weights output by the tuning algorithm are the average of all these collected weights. When each new batch is loaded at the start of the inner loop, a period of burn-in is run, analogous to the burn-in used in MCMC sampling, where no weight updates are performed and weights are not collected.

In order to help the stability of the tuning algorithm, and to enable it to process the tuning data more quickly, several chains are run in parallel, each with their own set of current weights, and each processing a distinct subset of the tuning data. The weights are mixed (averaged) after each epoch. The same technique is frequently adopted for the averaged perceptron (McDonald et al., 2010).

3 Experiments

3.1 Corpora and Baselines

The experiments in this section were conducted with French-English and German-English sections of the WMT2011[2] shared task data. In particular, we used News-Commentary data (`nc11`), and Europarl data (`ep11`) for training the generative models. Phrase tables were built from lowercased versions of the

[2] `http://www.statmt.org/wmt11/`

parallel texts using the standard Moses[3] training pipeline, with the target side of the texts used to build Kneser-Ney smoothed language models using the SRILM toolkit[4]. These data sets were used to build two phrase-based translation systems: WMT-SMALL and WMT-LARGE.

The WMT-SMALL translation system uses a translation model built from just the `nc11` data (about 115,000 sentences), and a 3-gram language model built from the target side of this data set. The features used in the WMT-SMALL translation system were the five Moses translation features, a language model feature, a word penalty feature and a distortion distance feature.

To build the WMT-LARGE translation system, both the `ep11` data set and the `nc11` data set were concatenated together before building the translation model out of the resulting corpus of about 2 million sentences. Separate 5-gram language models were built from the target side of the two data sets and then they were interpolated using weights chosen to minimise the perplexity on the tuning set (Koehn and Schroeder, 2007). In the WMT-LARGE system, the eight core features were supplemented with the six features of the lexicalised reordering model, which was trained on the same data as was used to build the translation model. Whilst a training set size of 2 million sentences would not normally be sufficient to build a competitive system for an MT shared task, it is sufficient to show that how SampleRank training performs on a realistic sized system, whilst still allowing for plenty of experimenation with the algorithm's parameters.

For tuning, the `nc-devtest2007` was used, with the first half of `nc-test2007` corpus used for heldout testing and `nc-test2008` and `newstest2010` reserved for final testing. The tuning and heldout sets are about 1000 sentences in size, whereas the final test sets are approximately 2000 sentences each.

In Table 1, the performance (in BLEU[5]) of untrained and MERT-tuned models on the heldout set is shown[6]. The untuned models use the default weights output by the Moses `train-model.perl` script, whereas the performance of the tuned models is the mean across five different MERT runs.

All decoding in this paper is with Moses, using default settings.

Pair	System	untuned	MERT-tuned
fr-en	WMT-SMALL	28.0	29.2 (0.2)
	WMT-LARGE	29.4	32.5 (0.1)
de-en	WMT-SMALL	25.0	25.3 (0.1)
	WMT-LARGE	26.6	26.8 (0.2)

Table 1: Untrained and MERT-trained performance on heldout. MERT training is repeated five times, with the table showing the mean BLEU, and standard deviation in brackets.

3.2 SampleRank Training For Small Models

First we look at how SampleRank training compares to MERT training using the WMT-SMALL models. Using the smaller models allows reasonably quick experimentation with a large range of different parameter settings.

For these experiments, the epoch size is set at 1024, and we vary both the number of cores and the number of samples used in training. The number of cores n is set to either 1,2,4,8 or 16, meaning that each epoch we split the tuning data into n different, non-overlapping shards, passing a different shard to each process, so the shard size k is set to $1024/n$. In each process, a burn of $100 * k$ samples is run (without updating the weights), followed by either $100 * k$ or $500 * k$ samples with weight updates, using the algorithm described in Section 2.2. After an epoch is completed, the current weights are averaged across all processes to give the new current weights in each process. At intervals of 50000 samples in each core, weights are averaged across all samples so far, and across all cores, and used to decode the heldout set to measure performance.

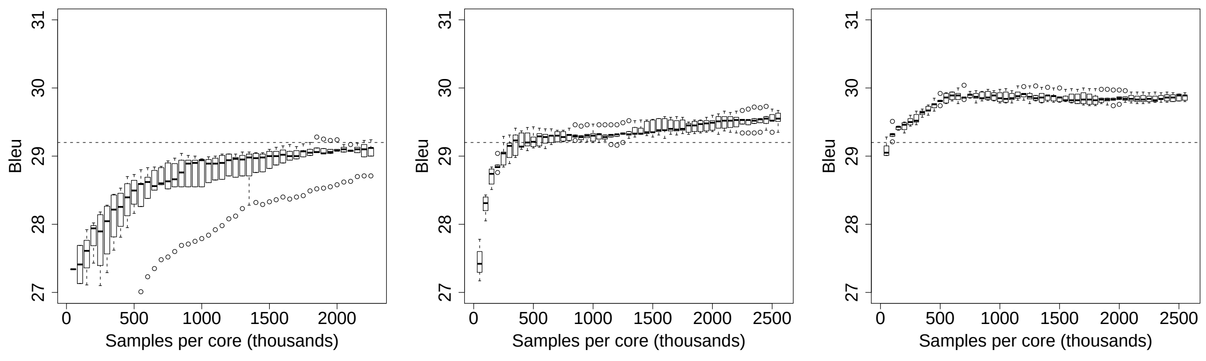
In Figure 1, learning curves are shown for the 100 sample-per-sentence case, for 1, 4 and 16 cores, for French-English. The training is repeated five times and the error bars in the graph indicate the

[3] `http://www.statmt.org/moses/`

[4] `http://www-speech.sri.com/projects/srilm/`

[5] Calculated with `multi-bleu.perl`

[6] All BLEU scores and standard deviations are rounded to one decimal place.



(a) 1 core (b) 4 cores (c) 16 cores

Figure 1: SampleRank learning curves for the WMT-SMALL French-English system, for 1, 4 and 16 cores. The dashed line shows the mean MERT performance, which has a standard deviation of 0.2.

spread across the different training runs. Increasing the number of cores makes a clear difference to the training, with the single core training run failing to reach the the level of MERT, and the 16 core training run exceeding the mean MERT performance by more than 0.5 BLEU. Using a single core also results in a much bigger training variance, which makes sense as using more cores and averaging weights reduces the adverse effect of a single chain going astray. The higher BLEU score achieved when using the larger number of cores is probably because a larger portion of the parameter space is being explored.

In one sense, the x axes of the graphs in Figure 1 are not comparable, since increasing the number of cores and keeping the number of samples per core increases the total computing time. However even if the single core training was run for much longer, it did not reach the level of performance obtained by multi-core training. Limited experimentation with increasing the core count to 32 did not show any appreciable gain, despite greatly increasing the computing resources required.

The training runs shown in Figure 1 take between 21 hours (for 16 cores) and 35 hours (for a single core)[7]. In the 16 core runs each core is doing the same amount of work as in the single core runs, so the difference in time is due to the extra effort involved in dealing with larger batches. These times are for the 100 samples-per-sentence condition, and increasing to 500 samples-per-sentence provides a speed-up of about 25%, since proportionally less time is spent on burn-in. Most of the time is spent in BLEU evaluation, so improved memoisation and incremental evaluation would reduce training time.

In Table 2 the mean maximum BLEU achieved on the heldout set at each parameter setting is shown. By this it is meant that for each of the five training runs at each (samples,cores) setting, the maximum BLEU on heldout data is observed, and these maxima are averaged across the five runs. It can be seen that changing the samples-per-sentence makes little difference, but there is a definite effect of increasing the core count.

Cores	100 Samples	500 Samples
1	29.1 (0.2)	29.2 (0.1)
2	29.3 (0.1)	29.3 (0.1)
4	29.6 (0.1)	29.5 (0.1)
8	30.0 (0.0)	29.9 (0.1)
16	30.0 (0.1)	29.8 (0.1)

Table 2: Mean maximum heldout performance for SampleRank training of the French-English WMT-SMALL model. Standard deviations are shown in brackets.

The learning curves for the equivalent German-English model are shown in Figure 2 and show a fairly different behaviour to their French-English counterparts. Again, using more cores helps to im-

[7]The processors are Intel Xeon 5450 (3GHz)

prove and stabilise the performance, but there is little if any improvement throughout training. As with MERT training, SampleRank training of the model weights makes little difference to the BLEU score, suggesting a fairly flat error surface.

Table 3 shows the mean maximum BLEU score on heldout data, the equivalent of Table 2, but for German-English. The results show very little variation as the samples-per-sentence and core counts are changed.

Cores	100 Samples	500 Samples
1	25.2 (0.0)	25.3 (0.1)
2	25.4 (0.1)	25.4 (0.1)
4	25.4 (0.1)	25.4 (0.1)
8	25.4 (0.1)	25.4 (0.1)
16	25.3 (0.1)	25.4 (0.1)

Table 3: Mean maximum heldout performance for SampleRank training of the German-English WMT-SMALL model. Standard deviations are shown in brackets

3.3 SampleRank Training for Larger Models

For the training of the WMT-LARGE systems with SampleRank, similar experiments to those in Section 3.2 were run, although only for 8 and 16 cores. The learning curves for the two language pairs (Figure 3) show roughly similar patterns to those in the previous section, in that the French-English system gradually increases performance through training to reach a maximum, as opposed to the German-English system with its fairly flat learning curve. Training times are around 27 hours for the 500 sample curve shown in Figure 3, increasing to 64 hours for 100 samples-per-sentence.

In Table 4, the mean maximum BLEU scores are shown for each configuration. of each language pair, calculated in the manner described in the previous section. For the larger system, SampleRank shows a smaller advantage over MERT for French-English, and little if any gain for German-English. For both large and small German-English models, neither of the parameter tuning algorithms are able to lift BLEU scores very far above the scores obtained from the untuned weights set by the Moses training script.

Pair	Cores	100 Samples	500 Samples
fr-en	8	32.6 (0.1)	32.7 (0.1)
	16	32.8 (0.1)	32.9 (0.1)
de-en	8	26.9 (0.0)	27.0 (0.1)
	16	26.8 (0.1)	26.9 (0.1)

Table 4: Mean (and standard deviation) of maximum heldout performance for SampleRank training of the WMT-LARGE model.

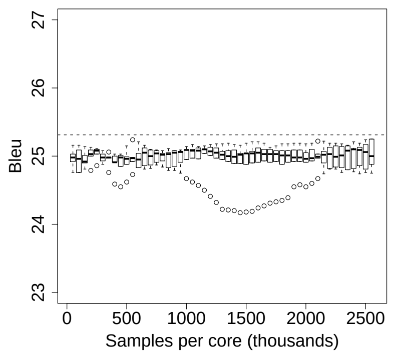
3.4 SampleRank Training for Larger Feature Sets

The final set of experiments are concerned with using SampleRank training for larger feature sets than the 10-20 typically used in MERT-trained models. The models considered in this section are based on the WMT-SMALL systems, but also include a family of part-of-speech tag based phrase boundary features.

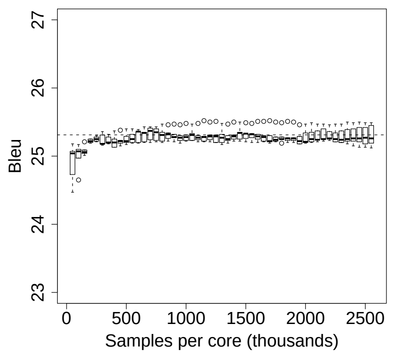
The phrase boundary features are defined by considering the target-side part-of-speech tag bigrams spanning each phrase boundary in the hypothesis, and allowing a separate feature to fire for each bigram. Dummy phrases with parts-of-speech <S> and </S> are inserted at the start and end of the sentence, and also used to construct phrase boundary features. The example in Figure 4 shows the phrase-boundary features from a typical hypothesis. The idea is similar to a part-of-speech language model, but discriminatively trained, and targeted at how phrases are joined together in the hypothesis.

The target-side part-of-speech tags are added using the Brill tagger, and incorporated into the phrase table using the factored translation modelling capabilities of Moses (Koehn and Hoang, 2007).

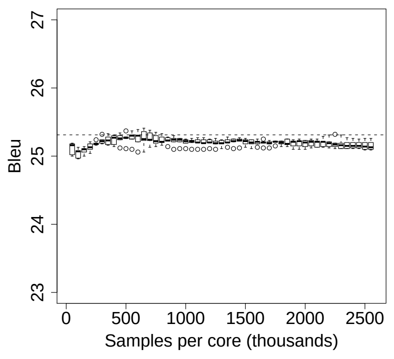
Adding the phrase boundary features to the WMT-SMALL system increased the feature count from 8 to around 800. Training experiments were run for both the French-English and German-English models, using the same configuration as in Section 3.2, varying the number of cores (8 or 16) and the number of samples per sentence (100 or 500). Training times were similar to those for the WMT-SMALL system. The mean maximum scores on heldout are shown in Table 5. We suspect that these features are fixing some short range reordering problems which



(a) 1 core

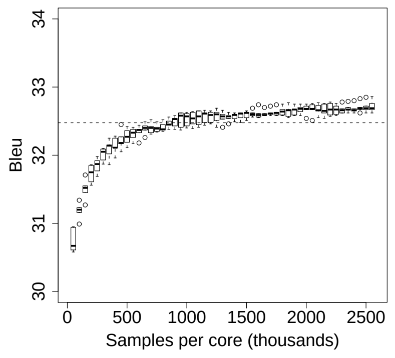


(b) 4 cores

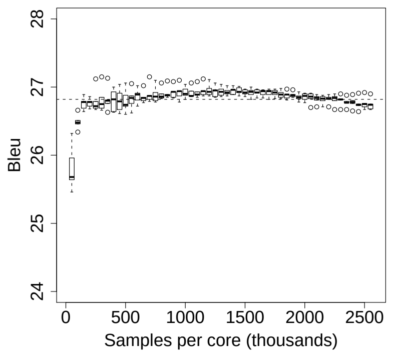


(c) 16 cores

Figure 2: SampleRank learning curves for the WMT-SMALL German-English system, for 1, 4 and 16 cores. The dashed line shows the mean MERT performance, which has a standard deviation of 0.1.



(a) French-English



(b) German-English

Figure 3: SampleRank learning curves for the WMT-LARGE French-English and German-English systems, using 8 cores and 500 samples per sentence. The dashed line shows the mean MERT performance, which has a standard deviation of 0.07 (fr-en) and 0.2 (de-en).

occur in the former language pair, but since the reordering problems in the latter language pair tend to be longer range, adding these extra features just tend to add extra noise to the model.

3.5 Comparison of MERT and SampleRank on Test Data

Final testing was performed on the `nc-test2008` and `newstest2010` data sets. The former is quite similar to the tuning and heldout data, whilst the latter can be considered to be "out-of-domain", so provides a check to see whether the model weights are being tuned too heavily towards the domain.

For the SampleRank experiments on the test set, the best training configurations were chosen from the results in Tables 2, 3, 4 and 5, and the best performing weight sets for each of the five runs for this configuration. For the MERT trained models, the same five models from Table 1 were used. The test set results are shown in Table 6.

The patterns observed on the heldout data carry over, to a large extent, to the test data. This is especially true for the WMT-SMALL system, where similar improvements (for French-English) over the MERT trained system are observed on the SampleRank trained system. For the WMT-LARGE system, the slightly improved performance that SampleRank offered on the in-domain data is no longer there, al-

Hypothesis		[europe	's]	[after]	[racial]	[house	divided	against	itself]	
Tags	<S>	NNP	POS	IN	JJ	NN	VBN	IN	PRP	</S>

This produces five phrase boundary features: <S>:NNP, POS:IN, IN:JJ, JJ:NN and PRP:</S>.

Figure 4: The definition of the phrase boundary feature from part-of-speech tags

		fr-en		de-en	
Training	System	nc-test2008	newstest2010	nc-test2008	newstest2010
MERT	WMT-SMALL	28.1 (0.1)	19.6 (0.1)	25.9 (0.1)	16.4 (0.2)
SampleRank	WMT-SMALL	28.7 (0.0)	20.1 (0.1)	25.9 (0.1)	16.6 (0.1)
SampleRank	WMT-SMALL+pb	28.8 (0.1)	19.8 (0.1)	25.9 (0.1)	16.7 (0.1)
MERT	WMT-LARGE	30.1 (0.1)	22.9 (0.1)	28.0 (0.2)	19.1 (0.2)
SampleRank	WMT-LARGE	30.0 (0.1)	23.6 (0.3)	28.1 (0.1)	19.5 (0.2)

Table 6: Comparison of MERT trained and SampleRank trained models on the test sets. The WMT-SMALL+pb model is the model with phrase boundary features, as described in Section 3.4

Pair	Cores	100 Samples	500 Samples
fr-en	8	30.2 (0.0)	30.2 (0.0)
	16	30.3 (0.0)	30.3 (0.00)
de-en	8	25.1 (0.1)	25.1 (0.0)
	16	25.0 (0.1)	25.0 (0.0)

Table 5: Mean (and standard deviation) of maximum heldout performance for SampleRank training of the WMT-SMALL model, with the phrase boundary feature.

though interestingly there is a reasonable improvement on out-of-domain, over the MERT trained model, similar to the effect observed in (Arun et al., 2010). Finally, the improvements offered by the phrase boundary feature are reduced, perhaps an indication of some over-fitting.

4 Related Work

Whilst MERT (Och, 2003) is still the dominant algorithm used for discriminative training (tuning) of SMT systems, research into improving on MERT's line search has tended to focus either on gradient-based or margin-based techniques.

Gradient-based techniques require a differentiable objective, and expected sentence BLEU is the most popular choice, beginning with Smith and Eisner (2006). They used n-best lists to calculate the feature expectations required for the gradient, optimising a second order Taylor approximation of expected sentence BLEU. They also introduced the idea of deterministic annealing to the SMT community, where an entropy term is added to the objective in training, and has its temperature progressively lowered in order to sharpen the model probability distribution. The work of Smith and Eisner was extended by Li and Eisner (2009) who were able to obtain much better estimates of feature expectations by using a packed chart instead of an n-best list. They also demonstrated that their method could extend to large feature sets, although their experiments were only run on small data sets.

An alternative method of calculating the feature expectations for expected BLEU training is Monte-Carlo Markov Chain (MCMC) approximation, and this was explored in (Arun et al., 2009) and (Arun et al., 2010). The sampling methods introduced in this earlier work form the basis of the current work, although in using the sampler for expected BLEU training, many samples must be collected before making a parameter weight update, as opposed to the current work where weights may be updated after every sample. One novel feature of Arun et al. (2010) is that they were able to train to directly maximise corpus BLEU, instead of its sentence-based approximation, although this only made a small difference to the results. The training methods in (Arun et al.,

2010) are very resource intensive, with the experiments running for 48 hours on around 40 cores, on a pruned phrase table derived from Europarl, and a 3-gram language model.

Instead of using expected BLEU as a training objective, Blunsom et al. (2008) trained their model to directly maximise the log-likelihood of the discriminative model, estimating feature expectations from a packed chart. Their model treats derivations as a latent variable, directly modelling the translation probability.

Margin-based techniques have the advantage that they do not have to employ expensive and complex algorithms to calculate the feature expectations. Typically, either perceptron ((Liang et al., 2006), (Arun and Koehn, 2007)) or MIRA ((Watanabe et al., 2007), (Chiang et al., 2008)) is employed, but in both cases the idea is to repeatedly decode sentences from the tuning set, and update the parameter weights if the best hypothesis according to the model differs from some "oracle" sentence. The approaches differ in the way they compute the oracle sentence, as well as the way the weights are updated. Normally sentences are processed one-by-one, with a weight update after considering each sentence, and sentence BLEU is used as the objective. However Chiang et al. (2008) introduced an approximation to corpus BLEU by using a rolling history. Both papers on MIRA demonstrated its ability to extend to large numbers of features.

In the only known application of SampleRank to SMT, Roth et al. (2010) deploys quite a different translation model to the usual phrase-based model, allowing overlapping phrases and implemented as a factor graph. Decoding is with a rather slow stochastic search and performance is quite poor, but this model, in common with the training algorithm presented in the current work, permits features which depend on the whole sentence.

5 Discussion and Conclusions

The results presented in Table 6 show that SampleRank is a viable method of parameter tuning for phrase-based MT systems, beating MERT in many cases, and equalling it in others. It is also able to do what MERT cannot do, and scale to a large number of features, with the phrase boundary feature of Section 3.4 providing a "proof-of-concept".

A further potential advantage of SampleRank is that it allows training with features which depend on the whole sentence, or even the whole document, since a full set of hypotheses is retained throughout training. Of course adding these features precludes decoding with the usual dynamic programming based decoders, and would require an alternative method, such as MCMC (Arun et al., 2009).

As with the other alternatives to MERT mentioned in this paper, SampleRank training presents the problem of determining convergence. With MERT this is straightforward, since training (normally) comes to a halt when the estimated tuning BLEU stops increasing and the weights stop changing. With methods such as minimum risk training, MIRA and SampleRank, some kind of early stopping criterion is usually employed, which lengthens training unnecessarily, and adds costly decodes to the training process. Building up sufficient practical experience with each of these methods will offset these problems somewhat.

Another important item for future work is to compare SampleRank training with MIRA training, in terms of performance, speed and ability to handle large feature sets.

The code used for the experiments in this paper is available under an open source license[8].

Acknowledgements

This research was supported by the EuroMatrixPlus project funded by the European Commission (7th Framework Programme) and by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. The project made use of the resources provided by the Edinburgh Compute and Data Facility (`http://www.ecdf.ed.ac.uk/`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk/`).

The authors would like to thank Sebastian Riedel for helpful discussions related to this work.

References

Abhishek Arun and Philipp Koehn. 2007. Online Learning Methods For Discriminative Training of Phrase

[8]`https://mosesdecoder.svn.sourceforge.net/svnroot/mosesdecoder/branches/samplerank`

Based Statistical Machine Translation. In *Proceedings of MT Summit*.

Abhishek Arun, Chris Dyer, Barry Haddow, Phil Blunsom, Adam Lopez, and Philipp Koehn. 2009. Monte Carlo inference and maximization for phrase-based translation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 102–110, Boulder, Colorado, June. Association for Computational Linguistics.

Abhishek Arun, Barry Haddow, and Philipp Koehn. 2010. A Unified Approach to Minimum Risk Training and Decoding. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics-MATR*, pages 365–374, Uppsala, Sweden, July. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A Discriminative Latent Variable Model for Statistical Machine Translation. In *Proceedings of ACL*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of EMNLP*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, March.

Aron Culotta. 2008. *Learning and inference in weighted logic with application to natural language processing*. Ph.D. thesis, University of Massachusetts, May.

George Foster and Roland Kuhn. 2009. Stabilizing Minimum Error Rate Training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249, Athens, Greece, March. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.

Philipp Koehn and Josh Schroeder. 2007. Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

Zhifei Li and Jason Eisner. 2009. First- and Second-order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of EMNLP*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An End-to-End Discriminative Approach to Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July. Association for Computational Linguistics.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed Training Strategies for the Structured Perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464, Los Angeles, California, June. Association for Computational Linguistics.

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Benjamin Roth, Andrew McCallum, Marc Dymetman, and Nicola Cancedda. 2010. Machine Translation Using Overlapping Alignments and SampleRank. In *Proceedings of AMTA*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of COLING/ACL*, pages 787–794, Morristown, NJ, USA. Association for Computational Linguistics.

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, Prague, Czech Republic, June. Association for Computational Linguistics.

Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew McCallum. 2009. SampleRank: Learning Preferences from Atomic Gradients. In *Proceedings of NIPS Workshop on Advances in Ranking*.

Michael Wick, Khashayar Rohanimanesh, Kedare Bellare, Aron Culotta, and Andrew McCallum. 2011. SampleRank: training factor graphs with atomic gradients. In *Proceedings of ICML*.