

# Minimum Error Rate Training by Sampling the Translation Lattice

**Samidh Chatterjee\***

Department of Computer Science  
Florida State University, USA  
chatterj@cs.fsu.edu

**Nicola Cancedda**

Xerox Research Centre Europe  
6 Chemin de Maupertuis, 38240 Meylan, France  
nicola.cancedda@xrce.xerox.com

## Abstract

Minimum Error Rate Training is the algorithm for log-linear model parameter training most used in state-of-the-art Statistical Machine Translation systems. In its original formulation, the algorithm uses N-best lists output by the decoder to grow the *Translation Pool* that shapes the surface on which the actual optimization is performed. Recent work has been done to extend the algorithm to use the entire translation lattice built by the decoder, instead of N-best lists. We propose here a third, intermediate way, consisting in growing the translation pool using samples randomly drawn from the translation lattice. We empirically measure a systematic improvement in the BLEU scores compared to training using N-best lists, without suffering the increase in computational complexity associated with operating with the whole lattice.

## 1 Introduction

Most state-of-the-art Statistical Machine Translation (SMT) systems are based on a log-linear model of the conditional probability of generating a certain translation given a specific source sentence. More specifically, the conditional probability of a translation  $\mathbf{e}$  and a word alignment  $\mathbf{a}$  given a source sentence  $\mathbf{f}$  is modeled as:

---

\*The work behind this paper was done during an internship at the Xerox Research Centre Europe. The author was partially supported by NSF through Grant CCF-0643593 and the AFOSR Young Investigator Research Program.

$$P(\mathbf{e}, \mathbf{a}|\mathbf{f}) \propto \exp\left(\sum_{k=1}^K \lambda_k h_k(\mathbf{e}, \mathbf{a}, \mathbf{f})\right) \quad (1)$$

where the  $h_k(\mathbf{e}, \mathbf{a}, \mathbf{f})$  are *feature functions* providing complementary sources of information on the quality of the produced translation (and alignment). Once such a model is known: the *decoder* (i.e. the actual translation program), which builds a translation by searching in the space of all possible translations the one that maximizes the conditional probability:

$$(\mathbf{e}^*, \mathbf{a}^*) = \arg \max_{\mathbf{e}, \mathbf{a}} \sum_{k=1}^K \lambda_k h_k(\mathbf{e}, \mathbf{a}, \mathbf{f}) \quad (2)$$

where we have taken into account that the exponential is monotonic.

The parameters  $\lambda_k$  determine the relative importance of the different feature functions in the global score. Best results are typically obtained by searching in the space of all possible parameter vectors  $\bar{\lambda}$  for the one that minimizes the error on a held-out development dataset for which one or more reference human translations are available, as measured by some automatic measure. This procedure is referred to as *Minimum Error Rate Training (MERT)*.

### 1.1 Minimum Error Rate Training on N-best Lists

The most widespread MERT algorithm is the one described in (Och, 2003). This algorithm starts by initializing the parameter vector  $\bar{\lambda}$ . For each source sentence in the development set, the decoder

is used to initialize a translation pool with a list of N-best scoring candidate translations according to the model. Using this pool and the corresponding reference translations then, an optimization procedure is run to update the parameter vector to a  $\bar{\lambda}$  with reduced error. The decoder is then invoked again, the new output N-best list is merged into the translation pool, and the procedure is iterated. The algorithm stops either after a predefined number of iterations or upon convergence, which is reached when no new element is added to the translation pool of any sentence, or when the size of the update in the parameter vector is below a threshold.

The error measure minimized at each iteration is usually BLEU (Papineni et al., 2002). BLEU essentially measures the precision with which the translation produced by a system recovers n-grams of different orders from the available reference translation(s), used as a gold standard.

The optimization procedure that is run within each iteration on the growing translation pools is based on the key observation that BLEU only depends on the single translation receiving the highest score by the translation model (which would be the one shown to the recipient) in the translation pool. This in turn means that, for any given sentence, its contribution to BLEU changes only when the value of the parameters change in such a way that the sentence ranking first according to the model switches from one to another. This situation does not change when one considers all the sentences in a development set instead of just one: while varying the  $\bar{\lambda}$  vector, the BLEU score changes only when there is a change at the top of the ranking of the alternatives for at least one sentence in the set. In other words, BLEU is piece-wise constant in  $\bar{\lambda}$ . MERT then proceeds by performing an iterative line search by fixing each time the value of all components of  $\bar{\lambda}$  but one<sup>1</sup>: for such a free parameter a global optimum can be identified by enumerating all the points that cause a change in BLEU. The value of the component is then fixed at the middle of an interval with maximum BLEU, and the procedure is iterated until convergence. Since the error function is highly irregular, and the iterative line search is not guaran-

<sup>1</sup>More generally, one can select each time a combination of coordinates identifying a line in the parameter space, and is not restricted to a coordinate direction.

teed to converge to a global optimum, the procedure is repeated many times with different initializations, and the best convergence point is retained.

The MERT algorithm suffers from the following problem: it assumes at each iteration that the set of candidates with a chance to make it to the top (for some value of the parameter vector) is well represented in the translation pool. If the translation pool is formed in the standard way by merging N-best lists, this assumption is easily violated in practice. Indeed, the N-best list often contains only candidates displaying minor differences, and represents only a very small sample of alternative possible translations, strongly biased by the current parameter setting.

Recognizing this shortcoming, Macherey et al. (2008) extended the MERT algorithm so as to use the whole set of candidate translations compactly represented in the search lattice produced by the decoder, instead of only a N-best list of candidates extracted from it. This is achieved via an elegant but relatively heavy dynamic programming algorithm that propagates sufficient statistics (called *envelopes*) throughout the whole search graph. The reported theoretical worst-case complexity of this algorithm is  $O(|V||\mathcal{E}| \log |\mathcal{E}|)$ , where  $V$  and  $\mathcal{E}$  are the vertex set and the edge set of the lattice respectively.

We propose here an alternative method consisting in sampling a list of candidate translations from the probability distribution induced by the translation lattice. This simple method produces a list of candidates more representative of the complete distribution than an N-best list, side-stepping the intricacies of propagating envelopes throughout the lattice. Computational complexity increases only marginally over the N-best list approach, while still yielding significant improvements in final translation quality.

## 1.2 The translation lattice

Finding the optimal translation according to Equation 1 is NP-complete (Knight, 1999). Most phrase-based SMT systems resort then to beam-search heuristic algorithms for solving the problem approximately. In their most widespread version, PBSMT decoders proceed by progressively extending translation prefixes by adding one new phrase at a time, and correspondingly “consuming” portions of the

source sentence. Each prefix is associated with a node in a graph, and receives a score according to the model. Whenever two prefixes having exactly the same possible extensions are detected, the lower-scoring one is *merged* into the other, thus creating a re-entrancy in the directed graph, which has then the characteristics of a *lattice* (Figure 1). Edges in the lattice are labelled with the phrase-pair that was used to perform the corresponding extension, the source word positions that were covered in doing the extension, and the corresponding increment in model score.

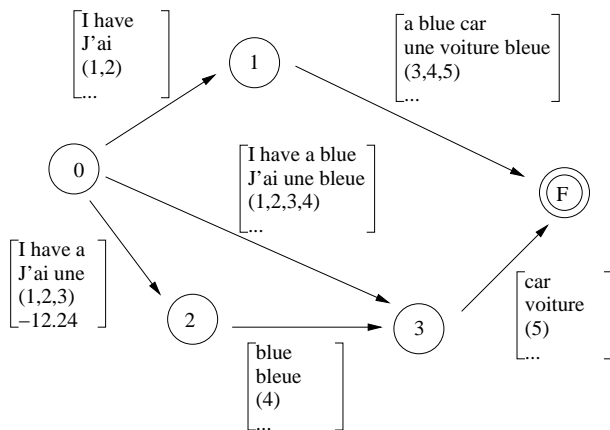


Figure 1: A lattice showing some possible translations of the English sentence: *I have a blue car*. The state with ID 0 is the start state and the one with *F* is the final state.

## 2 Related Work

Since its introduction, (Och, 2003) there has been various suggestions for optimizing the MERT criterion. Zens et al. (2007) use the MERT criterion to optimize the N-best lists using the Downhill Simplex Algorithm (Press, 2007). But the Downhill Simplex Algorithm loses its robustness as the dimension goes up by more than 10 (Macherey et al., 2008). Deterministic Annealing was suggested by Smith and Eisner (2006) where the authors propose to minimize the *expected loss* or *risk*. They define the expectation using a probability distribution over hypotheses that they gradually anneal to focus on the 1-best hypothesis. Different search strategies were investigated by Cer et al. (2008). Work has been done to investigate a perceptron-like online margin training for statistical machine translation (Watanabe et al., 2007).

Building on this paper, the most recent work to our knowledge has been done by Chiang et al. (2008). They explore the use of the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006) algorithm instead of MERT. Macherey et al. (2008) propose a new variation of MERT where the algorithm is tuned to work on the whole phrase lattice instead of N-best list only. The new algorithm constructs the error surface of all translations that are encoded in the phrase lattice. They report significant convergence improvements and BLEU score gains over N-best MERT when trained on NIST 2008 translation tasks. More recently, this algorithm was extended to work with hypergraphs encoding a huge number of translations produced by MT systems based on Synchronous Context Free Grammars (Kumar et al., 2009). All the methods cited here work on either N-best lists or from whole translation lattices built by the decoder. To our knowledge, none of them proposes sampling translations from the lattice.

## 3 Sampling candidate translations from the lattice

In this section we first start by providing an intuition of why we believe it is a good idea to sample from the translation lattice, and then describe in detail how we do it.

### 3.1 An intuitive explanation

The limited scope of n-best lists rules out many alternative translations that would receive the highest score for some values of the parameter vector. The complete set of translations that can be produced using a fixed phrase table (also called *reachable* translations) for a given source sentence can be represented as a set of vectors in the space spanned by the feature functions (Fig. 2). Not all such translations stand a chance to receive the highest score for any value of the parameter vector, though. Indeed, if translations  $\mathbf{h}$ ,  $\mathbf{h}'$  and  $\mathbf{h}''$  are such that  $h_k \leq h'_k \leq h''_k$  for all feature  $k$ , then there is no value of  $\bar{\lambda}$  that will give to  $\mathbf{h}'$  a score higher than both  $\mathbf{h}$  and  $\mathbf{h}''$ . The candidates that would rank first for some value of the  $\bar{\lambda}$  parameter vector are those on the convex envelope of the overall candidate set. We know of no effective way to generate this convex envelope in

polynomial time. The set of candidates represented by the decoder lattice is a subset (enclosed in the larger dashed polygon in the figure) of this set. This subset is biased to contain translations ranking high according to the values of the parameter vector (the direction labelled with  $\lambda$ ) used to produce it, because of the pruning strategies that guide the construction of the translation lattice. Both the N-best list and our proposed random sample are further subsets of the set of translations encoded in the lattice. The N-best list is very biased towards translations that score high with the current choice of parameters: its convex envelope (the smaller dashed polygon) is very different from the one of the complete set of translations, and also from that of the translations in the lattice. The convex envelope of a random sample from the translation lattice (the dotted polygon in the figure), will generally be somewhat closer to the envelope of the whole lattice itself.

The curves in the figure indicate regions of constant loss (e.g. iso-BLEU score, much more irregularly shaped in reality than in the drawing). For this sentence, then, the optimal choice of the parameters would be around  $\lambda^*$ . Performing an optimization step based on the random sample envelope would result in a more marked update ( $\lambda'_{\text{sample}}$ ) in the direction of the best parameter vector than if an N-best list is used ( $\lambda'_{\text{N-best}}$ ).

Notice that Figure 2 portrays a situation with only two features, for obvious reasons. In practice the number of features will be substantially larger, with values between five and twenty being common practice. In real cases, then, a substantially larger fraction of reachable translations will tend to lie on the convex envelope of the set, and not inside the convex hull.

### 3.2 The sampling procedure

We propose to modify the standard MERT algorithm and sample N candidates from the translation lattice according to the probability distribution over paths induced by the model, given the current setting of the  $\bar{\lambda}$  parameters, instead of using an N-best list. The sampling proceeds from the root node of the lattice, corresponding to an empty translation candidate covering no words of the source, by choosing step by step the next edge to follow. The probability

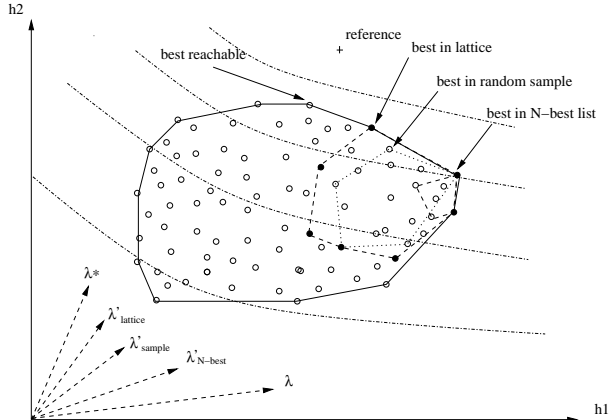


Figure 2: Envelope of the set of reachable translations where the model has two feature functions  $h_1$  and  $h_2$ . The envelope of the lattice is the outer dashed polygon, while the envelope of the N-best list is the inner one. Using the whole lattice as translation pool will result in a more marked update towards the optimal parameters. The random sample from the lattice is enclosed by the dotted line. If we use it, we can intuitively expect updates towards the optimum of intermediate effectiveness between those of the N-best list method and those of the lattice method.

distribution for each possible follow-up is the posterior probability of following the edge given the path prefix derived from the lattice: it is obtained via a preliminary backward sweep.

Since feature functions are incremental over the edges by design, the non-normalized probability of a path is given by:

$$P(e_1, \dots, e_m) = e^{\sum_{i=1}^m \sigma(e_i)} \quad (3)$$

where

$$\sigma(e_i) = \sum_{k=1}^K \lambda_k h_k(e_i) \quad (4)$$

is the *score* of edge  $e_i$ . With a small abuse of notation we will also denote it as  $\sigma(n_{j,k})$ , where it is intended that  $e_i$  goes from node  $n_j$  to node  $n_k$ . Let's denote with  $\sigma(n_i)$  the score of node  $n_i$ , i.e. the logarithm of the cumulative unnormalized probability of all the paths in the lattice that go from node  $n_i$  to a final node. The unnormalized probability of selecting node  $n_j$  starting from  $n_i$  can then be expressed recursively as follows:

$$S(n_j|n_i) \approx e^{(\sigma(n_j)+\sigma(n_{i,j}))} \quad (5)$$

The scores required to compute this sampling probabilities can be obtained by a simple backward pass in the lattice. Let  $P_i$  be the set of successors of  $n_i$ . So the total unnormalized log-probability of reaching a final state (i.e. with a complete translation) from  $n_i$  is given by the equation below.

$$\sigma(n_i) = \log\left(\sum_{n_j \in P_i} e^{(\sigma(n_j)+\sigma(n_{i,j}))}\right) \quad (6)$$

where we set  $\sigma(n_i) = 0$  if  $P_i = \emptyset$ , that is if  $n_i$  is a final node. At the end of the backward sweep,  $\sigma(n_0)$  contains the unnormalized cumulative probability of all paths, i.e. the partition function. Notice that this normalising constant cancels out when computing local sampling probabilities for traversed nodes in the lattice.

Once we know the transition probability (Eq. 5) for each node, we sample by starting in the root node of the lattice and at each step randomly selecting among its successors, until we end in the final node. The whole sampling procedure is repeated as many times as the number of samples sought. After collecting samples for each sentence, the whole list is used to grow the translation pool.

Notice that when using this sampling method it is no longer possible to use the stability of the translation pool as a stopping criterion. The MERT algorithm must thus be run either for a fixed number of iterations, or until the norm of the update to the parameter vector goes below a threshold.

### 3.3 Time Complexity Analysis

For each line search in the inner loop of the MERT algorithm, all methods considered here need to compute the projection of the convex envelope that can be scanned by leaving all components unchanged but one<sup>2</sup>. If we use either N-best lists or random samples to form the translation pool, and  $M$  is the size of the translation pool, then computing the envelope can be done in time  $O(M \log M)$  using the *SweepLine* algorithm reproduced as Algorithm 1 in (Macherey et al., 2008). As shown in the same article, the lattice method for computing the envelope

<sup>2</sup>In general, moving along a 1-dimensional subspace of the parameter space.

is  $O(|V||\mathcal{E}| \log |\mathcal{E}|)$ , where  $V$  is the vertex set of the lattice, and  $\mathcal{E}$  is its edge set. In standard decoders there is a maximum limit  $D$  to the allowed distortion, and lattice vertices are organized in  $J$  priority queues<sup>3</sup> of size at most  $a$ , where  $J$  is the length of the source sentence and  $a$  is a parameter of the decoder set by the user. Also, there is a limit  $K$  to the maximum number of source words spanned by a phrase, and only up to  $c$  alternative translations for a same source phrase are kept in the phrase table. Under these standard conditions, the number of outgoing edges  $E'$  from each lattice vertex can be bounded by a constant. A way to see this is by considering that if an hypothesis is extended with a phrase, then the extended hypothesis must end up in a stack at most  $K$  stacks to the right of the original one. There are only  $aK$  places in these stacks, so it must be  $|E'| \leq aK$ .

Since the number of edges leaving each node is bounded by a constant, it is  $|E| = \Theta(|V|)$ , and the lattice method is  $O(|V|^2 \log(|V|))$ . The maximum number of vertices in the lattice is limited by the capacity of the stacks:  $|V| \leq aJ$ . This eventually leads to a complexity of  $O(J^2 \log J)$  for the inner loop of the lattice method.

It is interesting to observe that the complexity is driven by the length of the source sentence in the case of the lattice method, and by the size of the translation pool in the case of both the N-best list method and the random sampling method. The latter two methods are asymptotically more effective as long as the size of the sample/N-best list grows sub-quadratically in the length of the sentence. In most of our experiments we keep the size of the sample constant, independent of the length of the sentence, but other choices can be considered. Since the number of reachable translations grows with the length of the source sentence, length-independent samples explore a smaller fraction of the reachable space. Generating samples (or n-best lists) of size increasing with the length of the source sentence could thus lead to more homogeneous sampling, and possibly a better use of CPU time.

We have so far compared methods in term of the complexity of the innermost loop: the search for a global optimum along a line in the parameter space.

<sup>3</sup>Traditionally referred to as *stacks*.

This is indeed the most important analysis, since the line search is repeated many times. In order to complete the analysis, we also compare the different methods in terms of the operations that need be performed as part of the outer iteration, that is upon redecoding the development set with a new parameter vector.

The N-best list method requires simply constructing an N-best list from the lattice. This can be done in time linear in the size  $J$  of the sentence and in  $N$  with a backward sweep in the lattice.

The sampling method requires sampling  $N$  times the lattice according to the probability distribution induced by the weights on its edges. We use a dynamic programming approach for computing the posterior probabilities of traversing edges. In this phase we visit each edge of the lattice exactly once, hence this phase is linear in the number of edges in the lattice, hence under the standard assumptions above in the length  $J$  of the sentence. Once posterior probabilities are computed for the lattice, we need to sample  $N$  paths from it, each of which is composed of at most  $J$  edges<sup>4</sup>. Under standard assumptions, randomly selecting the next edge to follow at each lattice node can be done in constant time, so the whole sampling is also  $O(NJ)$ , like extracting the N-best list.

No operation at all is required by the lattice method in the outer loop, since the whole lattice is passed over for envelope propagation to the inner loop.

#### 4 Experimental Results

Experiments were conducted on the Europarl corpus with the split used for the WMT-08 shared task (Europarl training and test condition) for the language pairs English-French (En-Fr), English-Spanish (En-Es) and English-German (En-De), each in both directions. Training corpora contain between 1.2 and 1.3 million sentence pairs each, development and test datasets are of size 2,000. Detailed token and type statistics can be found in Callison-Burch et al. (2008). The Moses decoder (Koehn et al., 2007) was used for generating lattices and n-best lists. The maximum number of decoding iterations was set to twelve. Since Moses was run with its lexicalised dis-

<sup>4</sup>We assume all phrase pairs cover at least one source word.

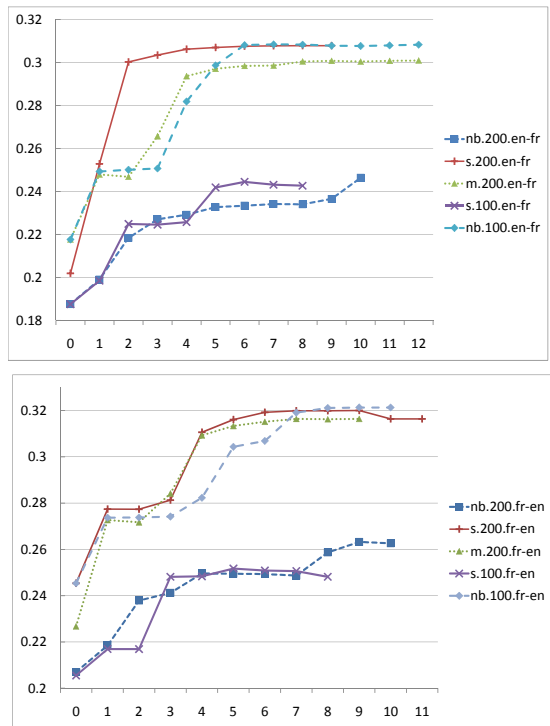


Figure 3: Learning curves (BLEU on the development set) for different tested conditions for English to French (top) and French to English (bottom).

tortion model, there were 14 features. Moses L1-normalises the parameter vector: parameter scaling only marginally affects n-best list construction (via threshold pruning during decoding), while it substantially impacts sampling.

For each of the six configurations, we compared the BLEU score on the test data when optimizing feature weights with MERT using n-best and random samples of size 100 and 200. In all cases we used 20 random restarts for MERT. Results are presented in Table 1. We also ran non systematic experiments on some of the configurations with larger samples and n-best lists, with results changing very little from the respective 200 cases: we do not report them here.

Learning curves (BLEU on the development set) are shown in Figure 3. Learning curves for the other tested language pairs follow a similar pattern.

## 5 Analysis of results

All differences of the test scores between optimizing the parameters using nbest-200 lists and from randomly sampled lists of size 200 were found to be statistically significant at 0.05 level at least. We used Approximate Randomization Test (Riezler and Maxwell, 2005) for the purpose, random sampling being done 1000 times.

S-T	NB-100	RS-100	NB-200	RS-200
En-Fr	32.47	31.36	32.32	<b>32.76</b>
Fr-En	32.43	31.77	32.46	<b>32.91</b>
En-Es	29.21	28.98	29.65	<b>30.19</b>
Es-En	30.97	30.41	31.22	<b>31.66</b>
En-De	20.36	19.92	20.55	<b>20.93</b>
De-En	27.48	26.98	27.30	<b>27.62</b>

Table 1: Test set BLEU Scores for six different “Source-Target” Pairs

Somewhat surprisingly, while random sampling with sample size of 200 yields overall the best results, random sampling with size 100 give systematically worse results than n-best lists of the same size. We conjectured that n-best lists and random samples could have complementary advantages. Indeed, it seems intuitive that a good translation pool should be sufficiently varied, as argued in Section 3.1. However it should also stand high chances to contain the best reachable translation, or translations close to the best. It might thus be that 100-best lists are unable to provide diversity, and random samples of size 100 to guarantee sufficient quality.

In order to test this conjecture we repeated our experiments, but at each iteration we used the union of a 100 random sample and a 100 n-best list. Results for this experiments are in Table 2. The corresponding results with random samples of size 200 are also repeated to ease comparison. Depending on the language pair, improvements over random sampling range from 0.17 (En-Es) to 0.44 (Fr-En) BLEU points. Improvements over 200-best lists range from 0.68 (De-En) to 0.89 (Fr-En) BLEU points. These results indicate quite clearly that N-best lists and random samples contribute complementary information to the translation pool: indeed, in most cases there is very little or no overlap between the two.

Convergence curves show that RS-200, NB-100

Source-Target	Mixed 100 + 100	RS-200
En-Fr	33.17	32.76
Fr-En	33.35	32.91
En-Es	30.37	30.19
Es-En	32.04	31.66
En-De	21.31	20.93
De-En	27.98	27.62

Table 2: Test set BLEU Scores for the same “Source-Target” pairs using a mixed strategy combining a 100 N-best list and a random sample of size 100 after each round of decoding.

and M-200 (i.e. the hybrid combination) systematically converge to higher BLEU scores, on the development set and on their respective translation pools, than RS-100 and NB-200. Notice however that it is misleading to compare scores across different translation pools, especially if these have substantially different sizes. On the one hand adding more candidates increases the chances of adding one with high contribution to the corpus BLEU, and can thus increase the achievable value of the objective function. On the other hand, adding more candidates reduces the freedom MERT has to find parameter values selecting high-BLEU candidates for all sentences. To see this, consider the extreme case when the translation pools are all of size one and are provided by an oracle that gives the highest-BLEU reachable translation for each sentence: the objective surface is uninformatively flat, all values of the parameters are equally good, and the BLEU score on the devset is the highest achievable one. If now we add to each translation pool the second-best BLEU-scoring candidate, BLEU will be maximized in a half-space for each sentence in the development set: MERT will try to select  $\lambda$  in the intersection of all the half-spaces, if this is not empty, but will have to settle for a lower-scoring compromise otherwise. The larger the translation pools, the more difficult it becomes for MERT to “make all sentences happy”. A special case of this is when adding more candidates extends the convex envelopes in such a way that the best candidates fall in the interior of the convex hull. It is difficult to tell which of the two opposing effects (the one that tends to increase the value of the objective function or the one that tends to depress it) is stronger in any

given case, but from the convergence curves it would seem that the first prevails in the case of random samples, whereas the second wins in the case of n-best lists. In the case of random samples going from size 100 to 200 systematically leads to higher BLEU score on the devsets, as more high-BLEU candidates are drawn. In the case of n-best lists, conversely, this leads to lower BLEU scores, as lower-BLEU (in average) candidates are added to translation pools providing a sharper representation of the BLEU surface and growing MERT out of the “delusion” that a given high BLEU score is actually achievable.

In the light of this discussion, it is interesting to observe that the value achieved by the objective function on the development set is only a weak predictor of performance on the test set, e.g. M-200 never converges to values above those of NB-100, but is systematically superior on the test data.

In Macherey et al. (2008) the authors observe a dip in the value of the objective function at the first iteration when training using n-best lists. We did not observe this behaviour in our experiments. A possible explanation for this resides in the larger size of the n-best lists we use (100 or 200, compared to 50 in the cited work) and in the smaller number of dimensions (14 instead of 20-30).

We hinted in Section 3.3 that it would seem reasonable to use samples/nbest-list of size increasing with the length of the source sentence, so as to sample reachable translations with a more uniform density across development sentences. We tested this idea on the French to English condition, making samples size depend linearly on the length of the sentence, and in such a way that the average sample size is either 100 or 200. For average sample size 100 we obtained a BLEU of 31.55 (compared to 31.77 with the constant-size 100 random sample) and for average size 200 31.84 (32.46 in the corresponding constant-size condition). While partial, these results are not particularly encouraging w.r.t. using variable size samples.

Finally, in order to assess the stability of the proposed training procedure across variations in development datasets, we experimented with extracting five distinct devsets of size 2,000 each for the French to English RS-200 condition, keeping the test set fixed: the maximum difference we observed was of 0.33 BLEU points.

## 6 Conclusions

We introduced a novel variant to the well-known MERT method for performing parameter estimation in Statistical Machine Translation systems based on log-linear models. This method, of straightforward implementation, is based on sampling candidates from the posterior distribution as approximated by an existing translation lattice in order to progressively expand the *translation pool* that shapes the optimization surface. This method compares favorably against existing methods on different accounts. Compared to the standard method by which N-best lists are used to grow the translation pool, it yields empirically better results as shown in our experiments, without significant penalties in terms of computational complexity. These results are in agreement with the intuition that the sampling method introduces more variety in the translation pool, and thus allows to perform more effective parameter updates towards the optimum. A hybrid strategy, consisting in combining N-best lists and random samples, brings about further significant improvements, indicating that both quality and variety are desirable in the translation pool that defines the optimization surface. A possible direction to investigate in the future consists in generalizing this hybrid strategy and combining random samples where the probability distribution induced on the lattice by the current parameters is scaled by a further temperature parameter  $\beta$ :

$$P^\beta(\mathbf{e}, \mathbf{a}|\mathbf{f}) \propto P(\mathbf{e}, \mathbf{a}|\mathbf{f})^\beta \quad (7)$$

where for  $\beta = 1$  the random samples used in this paper are obtained, for  $\beta$  tending to infinite the distribution becomes peaked around the single best path, thus producing samples similar to N-best lists, and samples from other real values of the temperature can be combined.

Compared to the method using the whole lattice, the proposed approaches have a substantially lower computational complexity under very broad and common assumptions, and yet yield translation quality improvements of comparable magnitude over the baseline N-best list method.

While the method presented in this paper operates on the translation lattices generated by Phrase-Based SMT decoders, the extension to translation

forests generated by hierarchical decoders (Chiang, 2007) seems straightforward. In that case, the backward sweep for propagating unnormalized posterior probabilities is replaced by a bottom-up sweep, and the sampling now concerns (binary) trees instead of paths, but the rest of the procedure is substantially unchanged. We conjecture however that the extension to translation forests would be less competitive compared to working with the whole packed forest (as in (Kumar et al., 2009)) than lattice sampling is compared to working with the whole lattice. The reason we believe this is that hierarchical models lead to much more spurious ambiguity than phrase-based models, so that both the N-best method and the sampling method explore a smaller portion of the candidate space compared to the compact representation of all the candidate translations in a beam.

### Acknowledgements

We would like to thank Vassilina Nikoulina, Greg Hanneman, Marc Dymetman for useful discussions and the anonymous reviewers for their suggestions and constructive criticism.

### References

- Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the ACL 2008 Workshop on Statistical Machine Translation*, Columbus, Ohio, 2008. <http://www.statmt.org/wmt08/pdf/WMT09.pdf>.
- Daniel Cer, Daniel Jurafsky, and Christopher D. Manning. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34, Columbus, Ohio, 2008. ISBN 978-1-932432-09-1.
- David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.
- David Chiang, Yuval Marton, and Philip Resnik. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 224–233, Honolulu, Hawaii, 2008.
- Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research (JMLR)*, 3:951–991, 2003. ISSN 1532-4435. doi: <http://dx.doi.org/10.1162/jmlr.2003.3.4-5.951>.
- Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585, 2006. ISSN 1532-4435.
- Kevin Knight. Decoding complexity in word-replacement translation modals. *Computational Linguistics, Squibs and Discussion*, 25(4), 1999.
- 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. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 177–180, prague, Czech republic, 2007.
- Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, Suntec, Singapore, August 2009.
- Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 725–734, Honolulu, Hawaii, 2008.
- Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL '03)*, pages 160–167, Sapporo, Japan, 2003. doi: <http://dx.doi.org/10.3115/1075096.1075117>.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic

evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*, pages 311–318, Philadelphia, Pennsylvania, 2002. doi: <http://dx.doi.org/10.3115/1073083.1073135>.

William H. Press. *Numerical recipes : the art of scientific computing*. Cambridge University Press, third edition, September 2007. ISBN 0521880688.

Stefan Riezler and John T. Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June 2005.

David A. Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the Joint International Conference on Computational Linguistics and Annual meeting of the Association for Computational Linguistics (COLING/ACL '06)*, pages 787–794, Sydney, Australia, 2006.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 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 2007.

Richard Zens, Sasa Hasan, and Hermann Ney. A systematic comparison of training criteria 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 524–532, Prague, Czech Republic, June 2007.