Lattice-based Minimum Error Rate Training for Statistical Machine Translation

Wolfgang Macherey **Franz Josef Och** **Ignacio Thayer** **Jakob Uszkoreit**

Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
{wmach,och,thayer,uszkoreit}@google.com

Abstract

Minimum Error Rate Training (MERT) is an effective means to estimate the feature function weights of a linear model such that an automated evaluation criterion for measuring system performance can directly be optimized in training. To accomplish this, the training procedure determines for each feature function its exact error surface on a given set of candidate translations. The feature function weights are then adjusted by traversing the error surface combined over all sentences and picking those values for which the resulting error count reaches a minimum. Typically, candidates in MERT are represented as N-best lists which contain the N most probable translation hypotheses produced by a decoder. In this paper, we present a novel algorithm that allows for efficiently constructing and representing the exact error surface of *all* translations that are encoded in a phrase lattice. Compared to N-best MERT, the number of candidate translations thus taken into account increases by several orders of magnitudes. The proposed method is used to train the feature function weights of a phrase-based statistical machine translation system. Experiments conducted on the NIST 2008 translation tasks show significant runtime improvements and moderate BLEU score gains over N-best MERT.

1 Introduction

Many statistical methods in natural language processing aim at minimizing the probability of sentence errors. In practice, however, system quality is often measured based on error metrics that assign non-uniform costs to classification errors and thus go far beyond counting the number of wrong decisions. Examples are the mean average precision for ranked retrieval, the F-measure for parsing, and the BLEU score for *statistical machine translation* (SMT). A class of training criteria that provides a tighter connection between the decision rule and the final error metric is known as *Minimum Error Rate Training* (MERT) and has been suggested for SMT in (Och, 2003).

MERT aims at estimating the model parameters such that the decision under the zero-one loss function maximizes some end-to-end performance measure on a development corpus. In combination with log-linear models, the training procedure allows for a direct optimization of the unsmoothed error count. The criterion can be derived from Bayes' decision rule as follows: Let $\mathbf{f} = f_1, ..., f_J$ denote a source sentence ('French') which is to be translated into a target sentence ('English') $\mathbf{e} = e_1, ..., e_I$. Under the zero-one loss function, the translation which maximizes the *a posteriori* probability is chosen:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \Big\{ \Pr(\mathbf{e}|\mathbf{f}) \Big\} \tag{1}$$

Since the true posterior distribution is unknown, $\Pr(\mathbf{e}|\mathbf{f})$ is modeled via a log-linear translation model which combines some feature functions $h_m(\mathbf{e}, \mathbf{f})$ with feature function weights λ_m, $m = 1, ..., M$:

$$\begin{aligned} \Pr(\mathbf{e}|\mathbf{f}) &= p_{\lambda_1^M}(\mathbf{e}|\mathbf{f}) \\ &= \frac{\exp\Big[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f})\Big]}{\sum_{\mathbf{e}'} \exp\Big[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}', \mathbf{f})\Big]} \end{aligned} \tag{2}$$

The feature function weights are the parameters of the model, and the objective of the MERT criterion is to find a parameter set λ_1^M that minimizes the error count on a representative set of training sentences. More precisely, let $\mathbf{f}_1^S$ denote the source sentences of a training corpus with given reference translations

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 725–734,
Honolulu, October 2008.

$\mathbf{r}_1^S$, and let $\mathbf{C}_s = \{\mathbf{e}_{s,1}, ..., \mathbf{e}_{s,K}\}$ denote a set of K candidate translations. Assuming that the corpus-based error count for some translations $\mathbf{e}_1^S$ is additively decomposable into the error counts of the individual sentences, i.e., $E(\mathbf{r}_1^S, \mathbf{e}_1^S) = \sum_{s=1}^{S} E(\mathbf{r}_s, \mathbf{e}_s)$, the MERT criterion is given as:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} E\big(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M)\big) \right\} \quad (3)$$

$$= \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} \sum_{k=1}^{K} E(\mathbf{r}_s, \mathbf{e}_{s,k}) \delta\big(\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M), \mathbf{e}_{s,k}\big) \right\}$$

with

$$\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M) = \arg\max_{\mathbf{e}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f}_s) \right\} \quad (4)$$

In (Och, 2003), it was shown that linear models can effectively be trained under the MERT criterion using a special line optimization algorithm. This line optimization determines for each feature function h_m and sentence $\mathbf{f}_s$ the exact error surface on a set of candidate translations $\mathbf{C}_s$. The feature function weights are then adjusted by traversing the error surface combined over all sentences in the training corpus and moving the weights to a point where the resulting error reaches a minimum.

Candidate translations in MERT are typically represented as N-best lists which contain the N most probable translation hypotheses. A downside of this approach is, however, that N-best lists can only capture a very small fraction of the search space. As a consequence, the line optimization algorithm needs to repeatedly translate the development corpus and enlarge the candidate repositories with newly found hypotheses in order to avoid overfitting on $\mathbf{C}_s$ and preventing the optimization procedure from stopping in a poor local optimum.

In this paper, we present a novel algorithm that allows for efficiently constructing and representing the unsmoothed error surface for *all* translations that are encoded in a phrase lattice. The number of candidate translations thus taken into account increases by several orders of magnitudes compared to N-best MERT. Lattice MERT is shown to yield significantly faster convergence rates while it explores a much larger space of candidate translations which is exponential in the lattice size. Despite this vast search space, we show that the suggested algorithm is always efficient in both running time and memory.

The remainder of this paper is organized as follows. Section 2 briefly reviews N-best MERT and introduces some basic concepts that are used in order to develop the line optimization algorithm for phrase lattices in Section 3. Section 4 presents an upper bound on the complexity of the unsmoothed error surface for the translation hypotheses represented in a phrase lattice. This upper bound is used to prove the space and runtime efficiency of the suggested algorithm. Section 5 lists some best practices for MERT. Section 6 discusses related work. Section 7 reports on experiments conducted on the NIST 2008 translation tasks. The paper concludes with a summary in Section 8.

2 Minimum Error Rate Training on N-best Lists

The goal of MERT is to find a weights set that minimizes the unsmoothed error count on a representative training corpus (cf. Eq. (3)). This can be accomplished through a sequence of line minimizations along some vector directions $\{d_1^M\}$. Starting from an initial point λ_1^M, computing the most probable sentence hypothesis out of a set of K candidate translations $\mathbf{C}_s = \{\mathbf{e}_1, ..., \mathbf{e}_K\}$ along the line $\lambda_1^M + \gamma \cdot d_1^M$ results in the following optimization problem (Och, 2003):

$$\hat{\mathbf{e}}(\mathbf{f}_s; \gamma) = \arg\max_{\mathbf{e} \in \mathbf{C}_s} \Big\{ (\lambda_1^M + \gamma \cdot d_1^M)^\top \cdot h_1^M(\mathbf{e}, \mathbf{f}_s) \Big\}$$

$$= \arg\max_{\mathbf{e} \in \mathbf{C}_s} \Big\{ \underbrace{\sum_m \lambda_m h_m(\mathbf{e}, \mathbf{f}_s)}_{=a(\mathbf{e}, \mathbf{f}_s)} + \gamma \cdot \underbrace{\sum_m d_m h_m(\mathbf{e}, \mathbf{f}_s)}_{=b(\mathbf{e}, \mathbf{f}_s)} \Big\}$$

$$= \arg\max_{\mathbf{e} \in \mathbf{C}_s} \big\{ \underbrace{a(\mathbf{e}, \mathbf{f}_s) + \gamma \cdot b(\mathbf{e}, \mathbf{f}_s)}_{(*)} \big\} \quad (5)$$

Hence, the total score $(*)$ for any candidate translation corresponds to a line in the plane with γ as the independent variable. For any particular choice of γ, the decoder seeks that translation which yields the largest score and therefore corresponds to the topmost line segment.

Overall, the candidate repository $\mathbf{C}_s$ defines K lines where each line may be divided into at most K line segments due to possible intersections with the other $K - 1$ lines. The sequence of the topmost line segments constitute the *upper envelope* which is the pointwise maximum over all lines induced by $\mathbf{C}_s$. The upper envelope is a convex hull and can be inscribed with a convex polygon whose edges are the segments of a piecewise linear function in γ (Papineni, 1999; Och, 2003):

$$\mathsf{Env}(\mathbf{f}) = \max_{\mathbf{e} \in \mathbf{C}} \big\{ a(\mathbf{e}, \mathbf{f}) + \gamma \cdot b(\mathbf{e}, \mathbf{f}) : \gamma \in \mathbb{R} \big\} \quad (6)$$

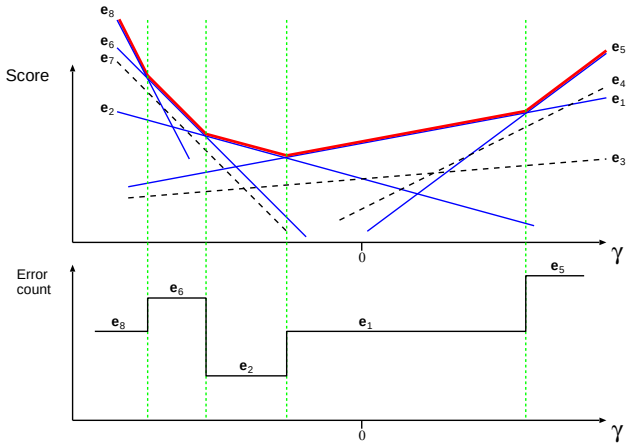


Figure 1: *The upper envelope (bold, red curve) for a set of lines is the convex hull which consists of the topmost line segments. Each line corresponds to a candidate translation and is thus related to a certain error count. Envelopes can efficiently be computed with Algorithm 1.*

The importance of the upper envelope is that it provides a compact encoding of all possible outcomes that a rescoring of $\mathbf{C}_s$ may yield if the parameter set λ_1^M is moved along the chosen direction. Once the upper envelope has been determined, we can project its constituent line segments onto the error counts of the corresponding candidate translations (cf. Figure 1). This projection is independent of *how* the envelope is generated and can therefore be applied to any set of line segments[1].

An effective means to compute the upper envelope is a *sweep line* algorithm which is often used in computational geometry to determine the intersection points of a sequence of lines or line segments (Bentley and Ottmann, 1979). The idea is to shift ("sweep") a vertical ray from $-\infty$ to $+\infty$ over the plane while keeping track of those points where two or more lines intersect. Since the upper envelope is fully specified by the topmost line segments, it suffices to store the following components for each line object ℓ: the x-intercept $\ell.x$ with the left-adjacent line, the slope $\ell.m$, and the y-intercept $\ell.y$; a fourth component, $\ell.t$, is used to store the candidate translation. Algorithm 1 shows the pseudo code for a sweep line algorithm which reduces an input array `a[0..K-1]` consisting of the K line objects of the candidate repository $\mathbf{C}_s$ to its upper envelope. By construction, the upper envelope consists of at most K line segments. The endpoints of each line segment define the interval boundaries at which the decision made by the decoder will change. Hence, as γ increases from $-\infty$ to $+\infty$, we will see that the most probable translation hypothesis will change whenever γ passes an intersection point.

Algorithm 1 SweepLine

input: array `a[0..K-1]` containing lines
output: upper envelope of `a`

```
sort(a:m);
j = 0; K = size(a);
for (i = 0; i < K; ++i) {
  ℓ = a[i];
  ℓ.x = -∞;
  if (0 < j) {
    if (a[j-1].m == ℓ.m) {
      if (ℓ.y <= a[j-1].y) continue;
      --j;
    }
    while (0 < j) {
      ℓ.x = (ℓ.y - a[j-1].y)/
            (a[j-1].m - ℓ.m);
      if (a[j-1].x < ℓ.x) break;
      --j;
    }
    if (0 == j) ℓ.x = -∞;
    a[j++] = ℓ;
  } else a[j++] = ℓ;
}
a.resize(j);
return a;
```

Let $\gamma_1^{\mathbf{f}_s} < \gamma_2^{\mathbf{f}_s} < ... < \gamma_{N_s}^{\mathbf{f}_s}$ denote the sequence of interval boundaries and let $\Delta E_1^{\mathbf{f}_s}, \Delta E_2^{\mathbf{f}_s}, ..., \Delta E_{N_s}^{\mathbf{f}_s}$ denote the corresponding sequence of changes in the error count where $\Delta E_n^{\mathbf{f}_s}$ is the amount by which the error count will change if γ is moved from a point in $[\gamma_{n-1}^{\mathbf{f}_s}, \gamma_n^{\mathbf{f}_s})$ to a point in $[\gamma_n^{\mathbf{f}_s}, \gamma_{n+1}^{\mathbf{f}_s})$. Both sequences together provide an exhaustive representation of the unsmoothed error surface for the sentence $\mathbf{f}_s$ along the line $\lambda_1^M + \gamma \cdot d_1^M$. The error surface for the whole training corpus is obtained by merging the interval boundaries (and their corresponding error counts) over all sentences in the training corpus. The optimal γ can then be found by traversing the merged error surface and choosing a point from the interval where the total error reaches its minimum.

After the parameter update, $\hat{\lambda}_1^M = \lambda_1^M + \gamma_{\text{opt}} \cdot d_1^M$, the decoder may find new translation hypotheses which are merged into the candidate repositories if they are ranked among the top N candidates. The relation $K = N$ holds therefore only in the first iteration. From the second iteration on, K is usually larger than N. The sequence of line optimizations and decodings is repeated until (1) the candidate repositories remain unchanged and (2) $\gamma_{\text{opt}} = 0$.

[1] For lattice MERT, it will therefore suffice to find an efficient way to compute the upper envelope over all translations that are encoded in a phrase graph.

3 Minimum Error Rate Training on Lattices

In this section, the algorithm for computing the upper envelope on N-best lists is extended to phrase lattices. For a description on how to generate lattices, see (Ueffing et al., 2002).

Formally, a phrase lattice for a source sentence $\mathbf{f}$ is defined as a connected, directed acyclic graph $\mathcal{G}_\mathbf{f} = (\mathcal{V}_\mathbf{f}, \mathcal{E}_\mathbf{f})$ with vertice set $\mathcal{V}_\mathbf{f}$, unique source and sink nodes $s, t \in \mathcal{V}_\mathbf{f}$, and a set of arcs $\mathcal{E}_\mathbf{f} \subset \mathcal{V}_\mathbf{f} \times \mathcal{V}_\mathbf{f}$. Each arc is labeled with a phrase $\varphi_{ij} = e_{i_1}, ..., e_{i_j}$ and the (local) feature function values $h_1^M(\varphi_{ij}, \mathbf{f})$. A path $\pi = (v_0, \varepsilon_0, v_1, \varepsilon_1, ..., \varepsilon_{n-1}, v_n)$ in $\mathcal{G}_\mathbf{f}$ (with $\varepsilon_i \in \mathcal{E}_\mathbf{f}$ and $v_i, v_{i+1} \in \mathcal{V}_\mathbf{f}$ as the tail and head of ε_i, $0 \leqslant i < n$) defines a partial translation $\mathbf{e}_\pi$ of $\mathbf{f}$ which is the concatenation of all phrases along this path. The corresponding feature function values are obtained by summing over the arc-specific feature function values:

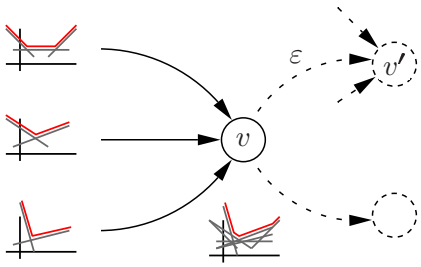
$$\pi : \bullet_{v_0} \xrightarrow[h_1^M(\varphi_{0,1}, \mathbf{f})]{\varphi_{0,1}} \bullet_{v_1} \xrightarrow[h_1^M(\varphi_{1,2}, \mathbf{f})]{\varphi_{1,2}} \cdots \xrightarrow[h_1^M(\varphi_{n-1,n}, \mathbf{f})]{\varphi_{n-1,n}} \bullet_{v_n}$$

$$\mathbf{e}_\pi = \bigcirc_{\substack{i,j\,:\\ v_i \to v_j \in \pi}} \varphi_{ij} = \varphi_{0,1} \circ ... \circ \varphi_{n-1,n}$$

$$h_1^M(\mathbf{e}_\pi, \mathbf{f}) = \sum_{\substack{i,j\,:\\ v_i \to v_j \in \pi}} h_1^M(\varphi_{ij}, \mathbf{f})$$

In the following, we use the notation $\mathsf{in}(v)$ and $\mathsf{out}(v)$ to refer to the set of incoming and outgoing arcs for a node $v \in \mathcal{V}_\mathbf{f}$. Similarly, $\mathsf{head}(\varepsilon)$ and $\mathsf{tail}(\varepsilon)$ denote the head and tail of $\varepsilon \in \mathcal{E}_\mathbf{f}$.

To develop the algorithm for computing the upper envelope of *all* translation hypotheses that are encoded in a phrase lattice, we first consider a node $v \in \mathcal{V}_\mathbf{f}$ with some incoming and outgoing arcs:



Each path that starts at the source node s and ends in v defines a partial translation hypothesis which can be represented as a line (cf. Eq. (5)). We now assume that the upper envelope for these partial translation hypotheses is known. The lines that constitute this envelope shall be denoted by $f_1, ..., f_N$. Next we consider continuations of these partial translation candidates by following one of the outgoing arcs

Algorithm 2 Lattice Envelope

input: a phrase lattice $\mathcal{G}_\mathbf{f} = (\mathcal{V}_\mathbf{f}, \mathcal{E}_\mathbf{f})$
output: upper envelope of $\mathcal{G}_\mathbf{f}$

```
a = ∅;
L = ∅;
TopSort(G_f);
for v = s to t do {
  a = SweepLine( ⋃_{ε∈in(v)} L[ε]);
  foreach (ε ∈ in(v))
    L.delete(ε);
  foreach (ε ∈ out(v)) {
    L[ε] = a;
    for (i = 0; i < a.size(); ++i) {
      L[ε][i].m = a[i].m + Σ_m d_m h_m(ε, f);
      L[ε][i].y = a[i].y + Σ_m λ_m h_m(ε, f);
      L[ε][i].p = a[i].p ∘ φ_{v,head(ε)};
    }
  }
}
return a;
```

$\varepsilon \in \mathsf{out}(v)$. Each such arc defines another line denoted by $g(\varepsilon)$. If we add the slope and y-intercept of $g(\varepsilon)$ to each line in the set $\{f_1, ..., f_N\}$, then the upper envelope will be constituted by segments of $f_1 + g(\varepsilon), ..., f_N + g(\varepsilon)$. This operation neither changes the number of line segments nor their relative order in the envelope, and therefore it preserves the structure of the convex hull. As a consequence, we can propagate the resulting envelope over an outgoing arc ε to a successor node $v' = \mathsf{head}(\varepsilon)$. Other incoming arcs for v' may be associated with different upper envelopes, and all that remains is to merge these envelopes into a single combined envelope. This is, however, easy to accomplish since the combined envelope is simply the convex hull of the union over the line sets which constitute the individual envelopes. Thus, by merging the arrays that store the line segments for the incoming arcs and applying Algorithm 1 to the resulting array we obtain the combined upper envelope for *all* partial translation candidates that are associated with paths starting at the source node s and ending in v'. The correctness of this procedure is based on the following two observations:

(1) A single translation hypothesis cannot constitute multiple line segments of the same envelope. This is because translations associated with different line segments are path-disjoint.

(2) Once a partial translation has been discarded from an envelope because its associated line $\tilde{f}$ is completely covered by the topmost line segments of the convex hull, there is no path continuation that could bring back $\tilde{f}$ into the upper envelope

again. Proof: Suppose that such a continuation exists, then this continuation can be represented as a line g, and since $\tilde{f}$ has been discarded from the envelope, the path associated with g must also be a valid continuation for the line segments $f_1, ..., f_N$ that constitute the envelope. Thus it follows that $\max(f_1 + g, ..., f_N + g) = \max(f_1, ..., f_N) + g < \tilde{f} + g$ for some $\gamma \in \mathbb{R}$. This, however, is in contradiction with the premise that $\tilde{f} < \max(f_1, ..., f_N)$ for all $\gamma \in \mathbb{R}$.

To keep track of the phrase expansions when propagating an envelope over an outgoing arc $\varepsilon \in \mathsf{tail}(v)$, the phrase label $\varphi_{v,\mathsf{head}(\varepsilon)}$ has to be appended from the right to all partial translation hypotheses in the envelope. The complete algorithm then works as follows: First, all nodes in the phrase lattice are sorted in topological order. Starting with the source node, we combine for each node v the upper envelopes that are associated with v's incoming arcs by merging their respective line arrays and reducing the merged array into a combined upper envelope using Algorithm 1. The combined envelope is then propagated over the outgoing arcs by associating each $\varepsilon \in \mathsf{out}(v)$ with a copy of the combined envelope. This copy is modified by adding the parameters (slope and y-intercept) of the line $g(\varepsilon)$ to the envelope's constituent line segments. The envelopes of the incoming arcs are no longer needed and can be deleted in order to release memory. The envelope computed at the sink node is by construction the convex hull over all translation hypotheses represented in the lattice, and it compactly encodes those candidates which maximize the decision rule Eq. (1) for any point along the line $\lambda_1^M + \gamma \cdot d_1^M$. Algorithm 2 shows the pseudo code. Note that the component $\ell.x$ does not change and therefore requires no update.

It remains to verify that the suggested algorithm is efficient in both running time and memory. For this purpose, we first analyze the complexity of Algorithm 1 and derive from it the running time of Algorithm 2.

After sorting, each line object in Algorithm 1 is visited at most three times. The first time is when it is picked by the outer loop. The second time is when it either gets discarded or when it terminates the inner loop. Whenever a line object is visited for the third time, it is irrevocably removed from the envelope. The runtime complexity is therefore dominated by the initial sorting and amounts to $\mathcal{O}(K \log K)$

Topological sort on a phrase lattice $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be performed in time $\Theta(|\mathcal{V}| + |\mathcal{E}|)$. As will be shown in Section 4, the size of the upper envelope for $\mathcal{G}$ can never exceed the size of the arc set $\mathcal{E}$. The same holds for any subgraph $\mathcal{G}_{[s,v]}$ of $\mathcal{G}$ which is induced by the paths that connect the source node s with $v \in \mathcal{V}$. Since the envelopes propagated from the source to the sink node can only increase linearly in the number of previously processed arcs, the total running time amounts to a worst case complexity of $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{E}| \log |\mathcal{E}|)$.

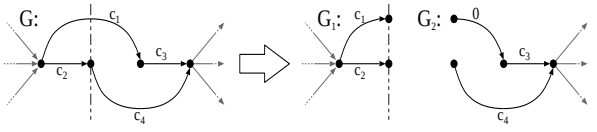
4 Upper Bound for Size of Envelopes

The memory efficiency of the suggested algorithm results from the following theorem which provides a novel upper bound for the number of cost minimizing paths in a directed acyclic graph with arc-specific affine cost functions. The bound is not only meaningful for proving the space efficiency of lattice MERT, but it also provides deeper insight into the structure and complexity of the unsmoothed error surface induced by log-linear models. Since we are examining a special class of shortest paths problems, we will invert the sign of each local feature function value in order to turn the feature scores into corresponding costs. Hence, the objective of finding the best translation hypotheses in a phrase lattice becomes the problem of finding all cost-minimizing paths in a graph with affine cost functions.

Theorem: *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected directed acyclic graph with vertex set $\mathcal{V}$, unique source and sink nodes $s, t \in \mathcal{V}$, and an arc set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ in which each arc $\varepsilon \in \mathcal{E}$ is associated with an affine cost function $c_\varepsilon(\gamma) = a_\varepsilon \cdot \gamma + b_\varepsilon$, $a_\varepsilon, b_\varepsilon \in \mathbb{R}$. Counting ties only once, the cardinality of the union over the sets of all cost-minimizing paths for all $\gamma \in \mathbb{R}$ is then upper-bounded by $|\mathcal{E}|$:*

$$\Big| \bigcup_{\gamma \in \mathbb{R}} \big\{ \pi \,:\, \pi = \pi(\mathcal{G}; \gamma) \text{ is a cost-minimizing path in } \mathcal{G} \text{ given } \gamma \big\} \Big| \leqslant |\mathcal{E}| \quad (7)$$

Proof: The proposition holds for the empty graph as well as for the case that $\mathcal{V} = \{s, t\}$ with all arcs $\varepsilon \in \mathcal{E}$ joining the source and sink node. Let $\mathcal{G}$ therefore be a larger graph. Then we perform an s-t cut and split $\mathcal{G}$ into two subgraphs $\mathcal{G}_1$ (left subgraph) and $\mathcal{G}_2$ (right subgraph). Arcs spanning the section boundary are duplicated (with the costs of the copied arcs in $\mathcal{G}_2$ being set to zero) and connected with a newly added head or tail node:



The zero-cost arcs in $\mathcal{G}_2$ that emerged from the duplication process are contracted, which can be done without loss of generality because zero-cost arcs do not affect the total costs of paths in the lattice. The contraction essentially amounts to a removal of arcs and is required in order to ensure that the sum of edges in both subgraphs does not exceed the number of edges in $\mathcal{G}$. All nodes in $\mathcal{G}_1$ with out-degree zero are then combined into a single sink node t_1. Similarly, nodes in $\mathcal{G}_2$ whose in-degree is zero are combined into a single source node s_2. Let N_1 and N_2 denote the number of arcs in $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. By construction, $N_1 + N_2 = |\mathcal{E}|$. Both subgraphs are smaller than $\mathcal{G}$ and thus, due to the induction hypothesis, their lower envelopes consist of at most N_1 and N_2 line segments, respectively. We further notice that either envelope is a convex hull whose constituent line segments inscribe a convex polygon, in the following denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$. Now, we combine both subgraphs into a single graph $\mathcal{G}'$ by merging the sink node t_1 in $\mathcal{G}_1$ with the source node s_2 in $\mathcal{G}_2$. The merged node is an *articulation point* whose removal would disconnect both subgraphs, and hence, all paths in $\mathcal{G}'$ that start at the source node s and stop in the sink node t lead through this articulation point. The graph $\mathcal{G}'$ has at least as many cost minimizing paths as $\mathcal{G}$, although these paths as well as their associated costs might be different from those in $\mathcal{G}$. The additivity of the cost function and the articulation point allow us to split the costs for any path from s to t into two portions: the first portion can be attributed to $\mathcal{G}_1$ and must be a line inside $\mathcal{P}_1$; the remainder can be attributed to $\mathcal{G}_2$ and must therefore be a line inside $\mathcal{P}_2$. Hence, the total costs for any path in $\mathcal{G}'$ can be bounded by the convex hull of the superposition of $\mathcal{P}_1$ and $\mathcal{P}_2$. This convex hull is again a convex polygon which consists of at most $N_1 + N_2$ edges, and therefore, the number of cost minimizing paths in $\mathcal{G}'$ (and thus also in $\mathcal{G}$) is upper bounded by $N_1 + N_2$. □

Corollary: *The upper envelope for a phrase lattice* $\mathcal{G}_\mathbf{f} = (\mathcal{V}_\mathbf{f}, \mathcal{E}_\mathbf{f})$ *consists of at most* $|\mathcal{E}_\mathbf{f}|$ *line segments.* This bound can even be refined and one obtains (proof omitted) $|\mathcal{E}| - |\mathcal{V}| + 2$. Both bounds are tight.

This result may seem somewhat surprising as it states that, independent of the choice of the direction along which the line optimization is performed, the structure of the error surface is far less complex than one might expect based on the huge number of alternative translation candidates that are represented in the lattice and thus contribute to the error surface. In fact, this result is a consequence of using a log-linear model which constrains how costs (or scores, respectively) can evolve due to hypothesis expansion. If instead quadratic cost functions were used, the size of the envelopes could not be limited in the same way. The above theorem does not, however, provide any additional guidance that would help to choose more promising directions in the line optimization algorithm to find better local optima. To alleviate this problem, the following section lists some best practices that we found to be useful in the context of MERT.

5 Practical Aspects

This section addresses some techniques that we found to be beneficial in order to improve the performance of MERT.

(1) **Random Starting Points:** To prevent the line optimization algorithm from stopping in a poor local optimum, MERT explores additional starting points that are randomly chosen by sampling the parameter space.

(2) **Constrained Optimization:** This technique allows for limiting the range of some or all feature function weights by defining *weights restrictions*. The weight restriction for a feature function h_m is specified as an interval $\mathcal{R}_m = [l_m, r_m]$, $l_m, r_m \in \mathbb{R} \cup \{-\infty, +\infty\}$ which defines the admissible region from which the feature function weight λ_m can be chosen. If the line optimization is performed under the presence of weights restrictions, γ needs to be chosen such that the following constraint holds:

$$l_1^M \leqslant \lambda_1^M + \gamma \cdot d_1^M \leqslant r_1^M \tag{8}$$

(3) **Weight Priors:** Weight priors give a small (positive or negative) boost ω on the objective function if the new weight is chosen such that it matches a certain target value λ_m^*:

$$\gamma_{\text{opt}} = \arg\min_\gamma \Big\{ \sum_s E\big(\mathbf{r}_s, \hat{e}(\mathbf{f}_s; \gamma)\big) + \sum_m \delta(\lambda_m + \gamma \cdot d_m, \lambda_m^*) \cdot \omega \Big\} \tag{9}$$

A *zero-weights prior* ($\lambda_m^* = 0$) provides a means of doing feature selection since the weight of a feature function which is not discriminative will be set to zero. An *initial-weights prior* ($\lambda_m^* = \lambda_m$) can be used to confine changes in the parameter update with the consequence that the new parameter may be closer to the initial weights set. Initial weights priors are useful in cases where the starting weights already yield a decent baseline.

(4) **Interval Merging:** The interval $[\gamma_i^{\mathbf{f}_s}, \gamma_{i+1}^{\mathbf{f}_s})$ of a translation hypothesis can be merged with the interval $[\gamma_{i-1}^{\mathbf{f}_s}, \gamma_i^{\mathbf{f}_s})$ of its left-adjacent translation hypothesis if the corresponding change in the error count $\Delta E_i^{\mathbf{f}_s} = 0$. The resulting interval $[\gamma_{i-1}^{\mathbf{f}_s}, \gamma_{i+1}^{\mathbf{f}_s})$ has a larger range, and the choice of γ_{opt} may be more reliable.
(5) **Random Directions:** If the directions chosen in the line optimization algorithm are the coordinate axes of the M-dimensional parameter space, each iteration will result in the update of a single feature function only. While this update scheme provides a ranking of the feature functions according to their discriminative power (each iteration picks the feature function for which changing the corresponding weight yields the highest gain), it does not take possible correlations between the feature functions into account. As a consequence, the optimization procedure may stop in a poor local optimum. On the other hand, it is difficult to compute a direction that decorrelates two or more correlated feature functions. This problem can be alleviated by exploring a large number of random directions which update many feature weights simultaneously. The random directions are chosen as the lines which connect some randomly distributed points on the surface of an M-dimensional hypersphere with the hypersphere's center. The center of the hypersphere is defined as the initial parameter set.

6 Related Work

As suggested in (Och, 2003), an alternative method for the optimization of the unsmoothed error count is Powell's algorithm combined with a grid-based line optimization (Press et al., 2007, p. 509). In (Zens et al., 2007), the MERT criterion is optimized on N-best lists using the Downhill Simplex algorithm (Press et al., 2007, p. 503). The optimization procedure allows for optimizing other objective function as, e.g., the expected BLEU score. A weakness of the Downhill Simplex algorithm is, however, its decreasing robustness for optimization problems in more than 10 dimensions. A different approach to minimize the expected BLEU score is suggested in (Smith and Eisner, 2006) who use deterministic annealing to gradually turn the objective function from a convex entropy surface into the more complex risk surface. A large variety of different search strategies for MERT are investigated in (Cer et al., 2008), which provides many fruitful insights into the optimization process. In (Duh and Kirchhoff, 2008), MERT is used to boost the BLEU score on N-best re-ranking tasks. The incorporation of a large number of sparse feature functions is described in (Watanabe et al., 2007). The paper investigates a perceptron-like online large-margin training for statistical machine translation. The described approach is reported to yield significant improvements on top of a baseline system which employs a small number of feature functions whose weights are optimized under the MERT criterion. A study which is complementary to the upper bound on the size of envelopes derived in Section 4 is provided in (Elizalde and Woods, 2006) which shows that the number of inference functions of any graphical model as, for instance, Bayesian networks and Markov random fields is polynomial in the size of the model if the number of parameters is fixed.

Table 1: *Corpus statistics for three text translation sets: Arabic-to-English (aren), Chinese-to-English (zhen), and English-to-Chinese (enzh). Development and test data are compiled from evaluation data used in past NIST Machine Translation Evaluations.*

data set	collection	# of sentences		
		aren	zhen	enzh
dev1	nist02	1043	878	–
dev2	nist04	1353	1788	–
blind	nist08	1360	1357	1859

7 Experiments

Experiments were conducted on the NIST 2008 translation tasks under the conditions of the constrained data track for the language pairs Arabic-to-English (aren), English-to-Chinese (enzh), and Chinese-to-English (zhen). The development corpora were compiled from test data used in the 2002 and 2004 NIST evaluations. Each corpus set provides 4 reference translations per source sentence. Table 1 summarizes some corpus statistics.

Table 2: *BLEU score results on the NIST-08 test set obtained after 25 iterations using N-best MERT or 5 iterations using lattice MERT, respectively.*

		dev1+dev2		blind	
task	loss	N-best	lattice	N-best	lattice
aren	MBR	56.6	57.4	42.9	43.9
	0-1	56.7	57.4	42.8	43.7
enzh	MBR	39.7	39.6	36.5	38.8
	0-1	40.4	40.5	35.1	37.6
zhen	MBR	39.5	39.7	27.5	28.2
	0-1	39.6	39.6	27.0	27.6

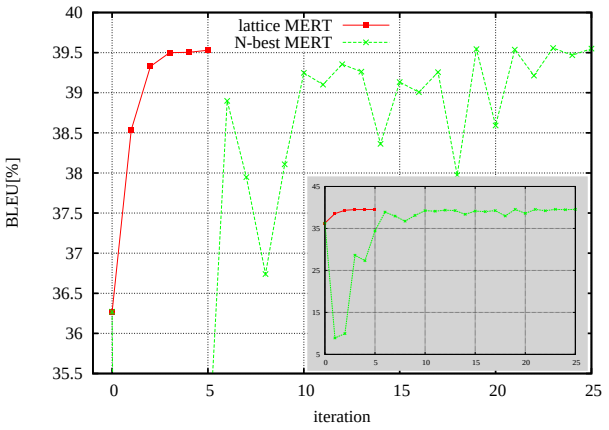


Figure 2: *BLEU scores for N-best MERT and lattice MERT after each decoding step on the zhen-dev1 corpus. The grey shaded subfigure shows the complete graph including the bottom part for N-best MERT.*

Translation results were evaluated using the mixed-case BLEU score metric in the implementation as suggested by (Papineni et al., 2001).

Translation results were produced with a state-of-the-art phrase-based SMT system which uses EM-trained word alignment models (IBM1, HMM) and a 5-gram language model built from the Web-1T collection[2]. Translation hypotheses produced on the blind test data were reranked using the *Minimum-Bayes Risk* (MBR) decision rule (Kumar and Byrne, 2004; Tromble et al., 2008). Each system uses a log-linear combination of 20 to 30 feature functions.

In a first experiment, we investigated the convergence speed of lattice MERT and N-best MERT.

[2]http://www.ldc.upenn.edu, catalog entry: LDC2006T13

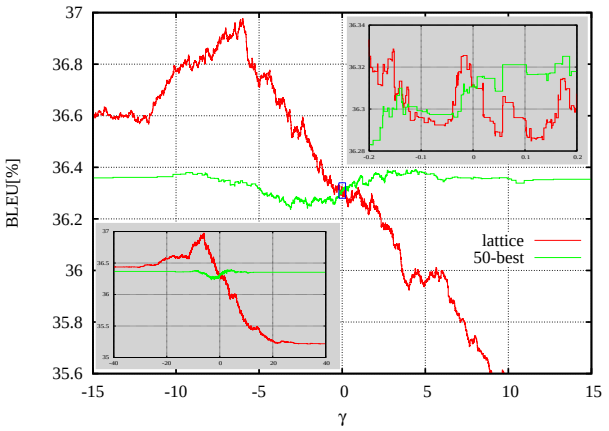


Figure 3: *Error surface of the phrase penalty feature after the first iteration on the zhen-dev1 corpus.*

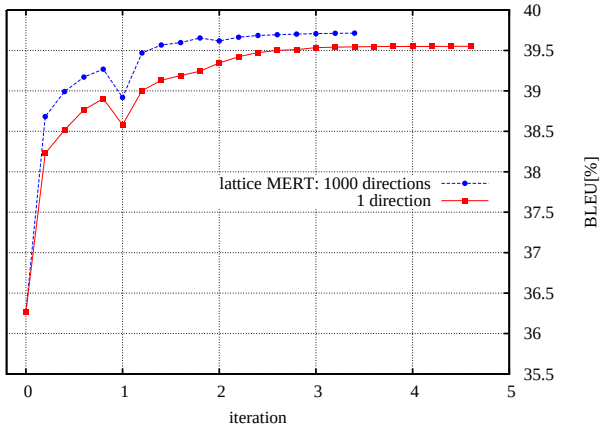


Figure 4: *BLEU scores on the zhen-dev1 corpus for lattice MERT with additional directions.*

Figure 2 shows the evolution of the BLEU score in the course of the iteration index on the zhen-dev1 corpus for either method. In each iteration, the training procedure translates the development corpus using the most recent weights set and merges the top ranked candidate translations (either represented as phrase lattices or N-best lists) into the candidate repositories before the line optimization is performed. For N-best MERT, we used $N = 50$ which yielded the best results. In contrast to lattice MERT, N-best MERT optimizes *all* dimensions in each iteration and, in addition, it also explores a large number of random starting points before it re-decodes and expands the hypothesis set. As is typical for N-best MERT, the first iteration causes a dramatic performance loss caused by overadapting the candidate repositories, which amounts to more than 27.3 BLEU points. Although this performance loss is recouped after the 5th iteration, the initial decline makes the line optimization under N-best MERT more fragile since the optimum found at the end of the training procedure is affected by the initial performance drop rather than by the choice of the initial start weights. Lattice MERT on the other hand results in a significantly faster convergence speed and reaches its optimum already in the 5th iteration. For lattice MERT, we used a graph density of 40 arcs per phrase which corresponds to an N-best size of more than two octillion $(2 \cdot 10^{27})$ entries. This huge number of alternative candidate translations makes updating the weights under lattice MERT more reliable and robust and, compared to N-best MERT, it becomes less likely that the same feature weight needs to be picked again and adjusted in subsequent iterations. Figure 4 shows the evolution of the BLEU score on the zhen-dev1 corpus using

Table 3: *BLEU score results on the NIST-08 tests set obtained after 5 iterations using lattice MERT with different numbers of random directions in addition to the optimization along the coordinate axes.*

task	# random directions	dev1+dev2 0-1	dev1+dev2 MBR	blind 0-1	blind MBR
aren	–	57.4	57.4	43.7	43.9
	1000	57.6	57.7	43.9	44.5
zhen	–	39.6	39.7	27.6	28.2
	500	39.5	39.9	27.9	28.3

lattice MERT with 5 weights updates per iteration. The performance drop in iteration 1 is also attributed to overfitting the candidate repository. The decline of less than 0.5% in terms of BLEU is, however, almost negligible compared to the performance drop of more than 27% in case of N-best MERT. The vast number of alternative translation hypotheses represented in a lattice also increases the number of phase transitions in the error surface, and thus prevents MERT from selecting a low performing feature weights set at early stages in the optimization procedure. This is illustrated in Figure 3, where lattice MERT and N-best MERT find different optima for the weight of the phrase penalty feature function after the first iteration. Table 2 shows the BLEU score results on the NIST 2008 blind test using the combined dev1+dev2 corpus as training data. While only the aren task shows improvements on the development data, lattice MERT provides consistent gains over N-best MERT on all three blind test sets. The reduced performance for N-best MERT is a consequence of the performance drop in the first iteration which causes the final weights to be far off from the initial parameter set. This can impair the ability of N-best MERT to generalize to unseen data if the initial weights are already capable of producing a decent baseline. Lattice MERT on the other hand can produce weights sets which are closer to the initial weights and thus more likely to retain the ability to generalize to unseen data. It could therefore be worthwhile to investigate whether a more elaborated version of an initial-weights prior allows for alleviating this effect in case of N-best MERT. Table 3 shows the effect of optimizing the feature function weights along some randomly chosen directions in addition to the coordinate axes. The different local optima found on the development set by using random directions result in additional gains on the blind test sets and range from 0.1% to 0.6% absolute in terms of BLEU.

8 Summary

We presented a novel algorithm that allows for efficiently constructing and representing the unsmoothed error surface over all sentence hypotheses that are represented in a phrase lattice. The proposed algorithm was used to train the feature function weights of a log-linear model for a statistical machine translation system under the *Minimum Error Rate Training* (MERT) criterion. Lattice MERT was shown analytically and experimentally to be superior over N-best MERT, resulting in significantly faster convergence speed and a reduced number of decoding steps. While the approach was used to optimize the model parameters of a single machine translation system, there are many other applications in which this framework can be useful, too. One possible usecase is the computation of consensus translations from the outputs of multiple machine translation systems where this framework allows us to estimate the system prior weights directly on confusion networks (Rosti et al., 2007; Macherey and Och, 2007). It is also straightforward to extend the suggested method to hypergraphs and forests as they are used, e.g., in hierarchical and syntax-augmented systems (Chiang, 2005; Zollmann and Venugopal, 2006). Our future work will therefore focus on how much system combination and syntax-augmented machine translation can benefit from lattice MERT and to what extent feature function weights can robustly be estimated using the suggested method.

References

J. L. Bentley and T. A. Ottmann. 1979. Algorithms for reporting and counting geometric intersections. *IEEE Trans. on Computers*, C-28(9):643–647.

D. Cer, D. Jurafsky, and C. D. Manning. 2008. Regularization and Search for Minimum Error Rate Training. In *Proceedings of the Third Workshop on Statistical Machine Translation, 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-2008 HLT)*, pages 26–34, Columbus, OH, USA, June.

D. Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *ACL-2005*, pages 263–270, Ann Arbor, MI, USA, June.

K. Duh and K. Kirchhoff. 2008. Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking. In *Proceedings of the Third Workshop on Statistical Machine Translation, 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-2008 HLT)*, pages 37–40, Columbus, OH, USA, June.

S. Elizalde and K. Woods. 2006. Bounds on the Number of Inference Functions of a Graphical Model, October. arXiv:math/0610233v1.

S. Kumar and W. Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proc. HLT-NAACL*, pages 196–176, Boston, MA, USA, May.

W. Macherey and F. J. Och. 2007. An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 986–995, Prague, Czech Republic, June.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. BLEU: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, USA.

K. A. Papineni. 1999. Discriminative training via linear programming. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 561–564, Phoenix, AZ, March.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, third edition.

A. V. Rosti, N. F. Ayan, B. Xiang, S. Matsoukas, R. Schwartz, and B. Dorr. 2007. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April. Association for Computational Linguistics.

D. A. Smith and J. Eisner. 2006. Minimum Risk Annealing for Training Log-linear Models. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (Coling/ACL-2006)*, pages 787–794, Sydney, Australia, July.

R. Tromble, S. Kumar, F. J. Och, and W. Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 10, Waikiki, Honolulu, Hawaii, USA, October.

N. Ueffing, F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 156–163, Philadelphia, PE, July.

T. Watanabe, J. Suzuki, H. Tsukada, and H. 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.

R. Zens, S. Hasan, and H. Ney. 2007. A Systematic Comparison of Training Criteria for Statistical Machine Translation. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing*, Prague, Czech Republic, June. Association for Computational Linguistics.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *NAACL '06: Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 138–141, New York, NY, June. Association for Computational Linguistics.