Minimum Error Rate Training Semiring

Artem Sokolov LIMSI-CNRS
François Yvon LIMSI-CNRs & Uni. Paris Sud
BP-133, 91 403 Orsay, France
{artem.sokolov, francois.yvon}@limsi.fr

Abstract

Modern Statistical Machine Translation (SMT) systems make their decisions based on multiple information sources, which assess various aspects of the match between a source sentence and its possible translation(s). Tuning a SMT system consists in finding the right balance between these sources so as to produce the best possible output, and is usually achieved through Minimum Error Rate Training (MERT) (Och, 2003). In this paper, we recast the operations implied in MERT in the terms of operations over a specific semiring, which, in particular, enables us to derive a simple and generic implementation of MERT over word lattices.

1 Introduction

Inference (decoding) in phrase-based statistical machine translation (SMT) systems is typically based on a log-linear model of the probability $p(\mathbf{e}|\mathbf{f}) = Z(\mathbf{f})^{-1}\exp(\bar{\lambda}\cdot\bar{h}(\mathbf{e},\mathbf{f}))$ of obtaining a target sentence $\mathbf{e}$ given an input sentence $\mathbf{f}$. For such model, the MAP decision rule selects $\tilde{\mathbf{e}}_\mathbf{f}$ as :

$$\begin{aligned}\tilde{\mathbf{e}}_\mathbf{f}(\bar{\lambda}) &= \arg\max_{\mathbf{e}\in E} p(\mathbf{e}|\mathbf{f}) \\ &= \arg\max_{\mathbf{e}\in E} \bar{\lambda}\cdot\bar{h}(\mathbf{e},\mathbf{f}), \qquad (1)\end{aligned}$$

where E is the set of reachable translations, $\bar{h}(\mathbf{e},\mathbf{f})$ is the vector of *feature functions* representing various compatibility measures of $\mathbf{f}$ and $\mathbf{e}$, and $\bar{\lambda}$ is a parameter vector, each component λ_i of which regulates the influence of the feature $h_i(\mathbf{e},\mathbf{f})$.

The set of reachable translations E (also referred to as the *search space*) in modern decoders is based on a set of heuristics that define the set of possible translation of each word or phrase (up to a maximum limit) and specify the range of possible reorderings of words or phrases during translation. Assuming that the components of E have been defined, the actual *tuning step* of a SMT system consists in finding $\bar{\lambda}^*$ that maximizes the empirical gain G on a development set $F = \{(\mathbf{f}, r_\mathbf{f})\}$ made of pairs of a source sentence $\mathbf{f}$ and corresponding reference translation(s) $r_\mathbf{f}$:

$$\bar{\lambda}^* = \arg\max_{\bar{\lambda}} G(F;\bar{\lambda}) \qquad (2)$$

where the computation of the gain function G, typically the BLEU score (Papineni et al., 2002) [1], depends on the actual translations $\{\tilde{\mathbf{e}}_\mathbf{f}(\bar{\lambda}), \mathbf{f}\in F\}$ achieved for a given value of $\bar{\lambda}$ according to (1).

For the sake of performing this optimization efficiently, the search space of the decoder is often approximated using an explicit list of n-best hypotheses or a directed acyclic graph (lattice) encoding a large number of potential translations.

Because of the form of the inference rule (1), the learning criterion (2) is neither convex nor differentiable. Furthermore, its exact computation is made intractable by the typical size of E, hence the recourse to various heuristic optimization strategies. The most successful to date is the proposal of (Och, 2003), usually referred to as Minimum Error Rate Training (MERT). This proposal has however been repeatedly questioned for (i) its computational cost and (ii) the instability of the resulting solutions (Cer et al., 2008; Moore and Quirk, 2008; Foster and Kuhn, 2009). The most promis-

[1] At this stage, any other metrics could be used instead of BLEU (see *e.g.*, (Zaidan, 2009)).

ing improvement consists in extending the approximation of the search space used in (2) from n-best lists to lattices, which both improves speed and reduces the variability of the final outcome (Macherey et al., 2008). Our main contribution in this paper is to recast the algorithm of (Macherey et al., 2008; Kumar et al., 2009) ("lattice-MERT") in a sound algebraic framework, using the MERT semiring[2] . Using this reformulation, we produce an efficient implementation of lattice MERT based on a generic finite-state toolbox (Allauzen et al., 2007). Preliminary experimental results confirm the main conclusions of (Macherey et al., 2008).

The rest of this paper is organized as follows. In Section 2, we recall the main principles of the basic algorithm of (Och, 2003), and some of the improvements that have been proposed in the literature. Section 3 is where we introduce the MERT semiring and its main properties. We then describe (Section 4) our own implementation of lattice MERT using a generic shortest distance algorithm in the appropriate semiring, and discuss various possible speed-ups. We then report machine translation experiments that demonstrate the effectiveness of our proposal (Section 5).

2 MERT and Lattice MERT

In the SMT literature, the MERT optimization cycle is used to refer to the development phase, where the weights of the various features/models involved in equation (1) are to be tuned over some development data. The whole procedure (Och, 2003) is sketched in algorithm 1.

Algorithm 1: The MERT optimization cycle

Input: initial value $\bar{\lambda}_0$ for $\bar{\lambda}$, development data F, required minimum improvement ϵ
Output: optimal value $\bar{\lambda}^*$ for $\bar{\lambda}$
repeat
  for *(*$\mathbf{f} \in F$*)* **do** $H_t(\mathbf{f}, \lambda_t) \leftarrow \texttt{Translate}(\mathbf{f})$
  $\bar{\lambda}_{t+1} \leftarrow \texttt{Optimize}(\{H_t(\mathbf{f}, \bar{\lambda}_t), \mathbf{f} \in F\}, \bar{\lambda}_t)$
  $t \leftarrow t + 1$
until *(*$|\bar{\lambda}_{t+1} - \bar{\lambda}_t| < \epsilon$*)*
$\bar{\lambda}^* \leftarrow \bar{\lambda}_t$

MERT thus implies two different kinds of operations: *decoding*, which basically implements the inference procedure and returns a set $H_t(\mathbf{f}, \lambda_t)$ of hypotheses, and *optimization*, which we now describe. The `Optimize()` function relies on optimization techniques for non-differentiable functions, such as the Powell's search algorithm (Powell, 1964). This requires to perform a series of minimizations of (2) along lines $\bar{\lambda} = \bar{\lambda}_0 + \gamma\bar{r}$ for some directions $\bar{r}$.

Due to the log-linear form of the probability in (1), the optimal hypothesis $\tilde{\mathbf{e}}_\mathbf{f}$ is given by:

$$\tilde{\mathbf{e}}_\mathbf{f}(\gamma) = \arg\max_{\mathbf{e} \in E} y_\mathbf{e} + \gamma s_\mathbf{e}$$

where $y_\mathbf{e} = \bar{\lambda}_0 \cdot \bar{h}(\mathbf{e}, \mathbf{f})$ and $s_\mathbf{e} = \bar{r} \cdot \bar{h}(\mathbf{e}, \mathbf{f})$. Each translation hypothesis is thus associated with a line in $\mathbb{R}^2$, and the most probable hypothesis for a given γ is the one whose line dominates all the others. The sequence of line segments that dominate all other lines for some value of γ is called the *upper envelope* (see Figure 1 and Definition 3.2).

The upper envelope identifies hypotheses that can be selected when $\bar{\lambda}$ is moved along the considered line. Projections of the intersections of the envelope's lines onto the γ-axis define the interval boundaries; over each such interval, the optimal hypothesis is constant. After merging the intervals computed separately for each sentence in F, it is possible to find γ^* maximizing gain G by computing it on each interval and setting γ^* to the middle of the best interval.

During each round of optimization, MERT explores several directions $\bar{r}_i$ and updates $\bar{\lambda} = \lambda_0 + \gamma^*_{i^*}\bar{r}_{i^*}$, where i^* is the index of the direction yielding the highest increase of G.

The procedure sketched in algorithm 1 has repeatedly been criticized for its computational cost and its lack of stability, which often implies the finding of a suboptimal solution. There are two main reasons why MERT can be very time consuming. The first is due to the total number of iterations that need to be performed to attain convergence: folklore wisdom is that the number of iterations shall be approximately proportional to total the number of dimensions. Speed is also dependent on the time required to perform one iteration, which is dominated by the translation phase. Even when distributed over several CPUs translation takes much more time and resource than optimization. It is thus expected that the most significant speed improvements will be obtained by reducing the number of iterations. The other main

[2]The notion of a MERT semiring has been alluded to in the literature (Dyer et al., 2010). To the best of our knowledge, this semiring has never been formally described, neither from the algebraic, nor from the implementation standpoint. This is a gap that we intend to fill in this work.

issue is the stability of the results, which, in practice, is addressed by running the optimize several times, with different starting points. These inefficiencies have stimulated the development of alternative approaches[3]. Attempts at improving MERT can be split into two categories: works that try to fix the optimization procedure and works that consider alternative, arguably easier to optimize or better suited training criteria (Smith and Eisner, 2006; Zens et al., 2007; Watanabe et al., 2007). In the sequel, we only discuss the former approaches, which are more relevant to this work.

(Cer et al., 2008) provides a thorough analysis of the optimization procedure, and suggests that improvements can be attained by (i) considering multiple random search directions instead of the Powell algorithm, and (ii) ensuring, through regularization, that the optimal $\bar{\lambda}^*$ have the ability to generalize well. This analysis is completed by the works of (Foster and Kuhn, 2009), which also suggests to improve the exploration of the search space by using well chosen multiple restarting points at each iteration (see also (Moore and Quirk, 2008)).

Initially proposed for n-best lists, MERT was also generalized to the case when E is approximated by a phrase lattice (Macherey et al., 2008), and more recently, to hypergraphs (Kumar et al., 2009). These generalizations take advantage of the decomposability of the feature functions $\bar{h}(\mathbf{e}, \mathbf{f})$, which are computed as a sum of *local* feature functions. When this property holds, rather than constructing upper envelopes for each hypothesis in the lattice[4], the envelopes are distributed over nodes in the lattice. Working with much better approximations of the complete search spaces not only allows to converge in less iterations, but also to achieve better generalization, a finding that was recently confirmed by (Larkin et al., 2010). Our work is a continuation of this line of research, driven by the intuition that recasting MERT in a clear algebraic framework, as we do in the next section, can help develop faster, and even more efficient, implementations of MERT for complex hypotheses set.

3 The MERT Semiring

Recall that a semiring $\mathbb{K}$ over a set K is a system $\langle K, \oplus, \otimes, \bar{0}, \bar{1}\rangle$, where $\langle K, \oplus, \bar{0}\rangle$ is a commutative monoid with identity element $\bar{0}$, meaning that $a \oplus (b \oplus c) = (a \oplus b) \oplus c$, $a \oplus b = b \oplus a$ and $\forall a, a \oplus \bar{0} = \bar{0} \oplus a = a$. Additionally, $\langle K, \otimes, \bar{1}\rangle$ is a monoid with identity element $\bar{1}$; $\otimes$ distributes over $\oplus$ so that $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ and $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$ and element $\bar{0}$ annihilates K ($a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$). A semiring is called commutative if the operation $\otimes$ is commutative.

In this section, we characterize the algebraic structure of the set of upper envelopes of a set of lines in the plane, and show that a set of envelopes equipped with certain operations of addition ($\oplus$) and multiplication ($\otimes$) defines a commutative semiring.

3.1 Lineset semiring

Consider a set D of sets d^k of n lines $\{d^k_{i.s} \cdot x + d^k_{i.y}, i = 1 \dots n\}$ in $\mathbb{R}^2$, where $d^k_{i,s}, d^k_{i,y} \in \mathbb{R}$ are, respectively, the slope and the y-intercept of the i-th line in the set d^k. For two sets $d^1, d^2 \in D$, we define the following internal operations $\oplus_D$ and $\otimes_D$[5] as follows:

$$\begin{aligned} d^1 \oplus_D d^2 =& d^1 \cup d^2, \\ d^1 \otimes_D d^2 =& \{(d^1_{i.s} + d^2_{j.s}) \cdot x + (d^1_{i.y} + d^2_{j.y}) \\ & \quad \mid \forall d^1_i \in d^1, d^2_j \in d^2\}. \end{aligned} \tag{3}$$

Proposition 3.1. *$\mathbb{D} = \langle D, \oplus_D, \otimes_D, \bar{0}_D, \bar{1}_D\rangle$, where $\bar{0}_D = \emptyset$ and $\bar{1}_D = \{0 \cdot x + 0\}$, is a commutative semiring.*

Proof. It is well known that $\langle D, \oplus_D\rangle$ is a commutative monoid with $\bar{0}_D$ as identity element. $\langle D, \otimes_D\rangle$ is also a commutative monoid with $\bar{1}_D$ as identity element. It is finally routine to check that $\otimes_D$ is distributive over $\oplus_D$. The additive identity $\bar{0}_D$ annihilates D for the $\otimes_D$ operation, because of the definition (3): as no line is contained in $\bar{0}_D$ the result of multiplication by $\bar{0}_D$ is always empty. □

3.2 Envelope semiring

Definition 3.2. *The upper envelope of a set of lines $d \in D$ is a subset $\mathrm{env}(d) \subseteq d$ consisting of lines $d_i \in d$, s.t. for each line $d_i \in \mathrm{env}(d)$, there exists an non-empty interval $I_i \in \mathbb{R}$, s.t. if $x \in I_i$, then $d_{i.s} \cdot x + d_{i.y} > d_{i'.s} \cdot x + d_{i'.y}$, for any line $d_{i'} \neq d_i$.*

Two lines d_i and d_j in $\mathrm{env}(d)$ are said to be neighbors *if their corresponding intervals I_i and I_j are adjacent.*

[3] Not to mention changes in the core optimization routines, as in *e.g.*, (Lambert and Banchs, 2006)

[4] They are too numerous to be efficiently enumerated.

[5] Formally, $\otimes$ corresponds to the *Minkowski sum* of the two sets of lines.

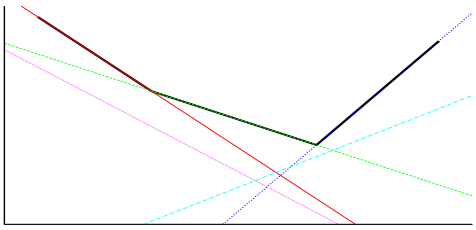

Figure 1: The upper envelope of a set of lines

For MERT, it is important to know the intersections of neighboring lines in the envelope. For this purpose, an envelope can be ordered as a list of lines with increasing slopes and each line encoded as a tuple (x, s, y) where x is the line's x-intersection with the previous line in the list. The upper envelope can be computed by algorithm 2, which is a rewrite of the sweepline algorithm from (Macherey et al., 2008). We use a separate function `Sweep` (algorithm 3), which will serve in a faster implementation of the $\oplus$ operation.

Algorithm 2: The `SweepLine` algorithm

Input: sorted array S containing lines
Output: upper envelope of S
$j = 0$
for $(i = 0; i < |S|\,; ++i)$ **do**
  `Sweep` $(S, S[i], j)$
end
S.`resize` (j)

Algorithm 3: The `Sweep`(S, ℓ, j) function

Input: upper envelope S, line ℓ, current value of j
Output: upper envelope of $S \cup \ell$, updated value for j
if $(0 < j)$ **then**
  if $(S[j-1].s = \ell.s)$ **then**
    if $(\ell.y <= S[j-1].y)$ **then** continue
    $j \leftarrow j - 1$
  while $(0 < j)$ **do**
    $\ell.x = (\ell.y - S[j-1].y)/(S[j-1].s - \ell.s)$
    if $(S[j-1].x < \ell.x)$ **then** break
    $j \leftarrow j - 1$
  end
  if $(0 = j)$ **then** $\ell.x = -\infty$
$S[j] = \ell$
$j \leftarrow j + 1$

Let E be a subset of D such that $\mathrm{env}(d) = d$ for each $d \in E$, and define the operations $\oplus_E$ and $\otimes_E$ as the projections of the respective operations in D on the set E:

$$d^1 \oplus_E d^2 = \mathrm{env}(d^1 \oplus_D d^2),$$
$$d^1 \otimes_E d^2 = \mathrm{env}(d^1 \otimes_D d^2).$$

Proposition 3.3. *The tuple* $\mathbb{E} = \langle E, \oplus_E, \otimes_E, \bar{0}_E, \bar{1}_E \rangle$, *where* $\bar{0}_E = \emptyset$ *and* $\bar{1}_E = \{0 \cdot x + 0\}$, *is a commutative semiring.*

Proof. It is routine to check both associative, commutative and distributive properties, and that $\bar{0}_E$ is a multiplicative annihilator of E. □

A semiring is called weakly divisible if for any pair of elements d^1 and d^2 such that $d^1 \oplus d^2 \neq \bar{0}$, there exists at least one d such that $d^1 = (d^1 \oplus d^2) \otimes d$. The concept of divisibility is important as it is a crucial requirement for optimizing transducers by determinization (Mohri, 2009). However, one can observe that the MERT semiring is not divisible, which has its implication on the minimization of the finite-state automata between iterations of algorithm 1 (see subsection 4.2).

3.3 Shortest Distance and MERT

Let $A = (\Sigma, Q, I, F, E)$ be a weighted finite state acceptor with weights in $\mathbb{E}$, meaning that the transitions (q, a, q') in A carry a weight w in $\mathbb{E}$. Formally, E is a mapping from $(Q \times \Sigma \times Q)$ into $\mathbb{E}$; likewise I and F are mappings from Q into $\mathbb{E}$. We use the notations of (Mohri, 2009): if $e = (q, a, q')$ is a transition in $\mathrm{domain}(E)$, $p(e) = q$ (resp. $n(e) = q'$) denotes its origin (resp. destination) state, $i(e) = a$ its label, and $w(e) = E(e)$ its weight. These notations extend to paths: if π is a path in A, $p(\pi)$ (resp. $n(\pi)$) is its initial (resp. ending) state and $i(\pi)$ is the label along the path.

In our setting, A is derived from a word lattice L as follows. We assume that L has a single start and end state, denoted respectively q_0 and q_F. Each arc in L labeled with a target word a carries a vector $\bar{h}(a, \mathbf{f})$ of local features associated with a. Given a starting point $\bar{\lambda}_0$ and a search direction $\bar{r}$, L is turned into a weighted acceptor over $\mathbb{E}$ by associating with (q_0) (resp. q_F) the weight $\bar{1}$ and replacing $\bar{h}$ with a singleton containing line d_i with slope $d_{i.s} = (\bar{r} \cdot \bar{h})x$ and y-intercept $d_{i.y} = (\bar{\lambda}_0 \cdot \bar{h})$.

The total weight of a successful path $\pi = e_1 ... e_l$ in A is thus computed as:

$$w(\pi) = I(e_1) \otimes \big[\bigotimes_{i=1}^{l} w(e_i) \big] \otimes F(e_l)$$
$$= \{ \bar{\lambda}_0 \cdot \sum_{i=1}^{l} \bar{h}(i(e_i), \mathbf{f}) + (\bar{r} \cdot \sum_{i=1}^{l} \bar{h}(i(e_i), \mathbf{f}))x \}.$$

This is because each weight in A contains a single line, which means that the total weight of a path is also a singleton line set, which corresponds to the complete translation hypothesis read along π as $\mathbf{e} = i(e_1) \dots i(e_l)$.

The computation of the complete upper envelope of all the lines corresponding to translation hypotheses in the lattice L thus corresponds to the envelope of the union of all these lines:

$$\text{env}(\bigcup_{\pi \in A} w(\pi)) = \bigoplus_{\pi \in A} w(\pi). \quad (4)$$

Quantities such as (4) can be efficiently computed by generic shortest distance algorithms over acyclic graphs (Mohri, 2002).

As an additional note, it is interesting to realize that all the previous considerations hold if we make the elements in E *vectors* of set of lines, instead of just set of lines; and define the semiring operations to be performed componentwise. This means that the computation of (4) can be done simultaneously in any number of directions.

4 Implementing MERT with semiring operations

4.1 Basic operations

Given that $\mathbb{E}$ is a semiring, our implementation relies on the general finite-state transducer library OpenFst (Allauzen et al., 2007) to perform the computation of the quantities that are required by MERT on lattices. The main benefit of adopting this framework is that we only need to implement the basic semiring operations to get the full power of proven and well optimized algorithms.

We first detail the $\otimes$ operation, implemented as specified in algorithm 4. In our application, the automata derived from translation lattices are acyclic which means that, if we process states in topological order, the right-hand argument d^2 in algorithm 4 always contains only one line, what removes the need for the inner for-loop. As multiplication by a single line does not change the relative order of lines in the envelope, the final call to `SweepLine()` is also not required (algorithm 5).

Algorithm 4: $\otimes$ operation

Input: two envelopes d^1, d^2
Output: $d^1 \otimes d^2$
$S = \emptyset$
for $d_i^1 \in d^1$ **do**
  for $d_j^2 \in d^2$ **do**
    $S \leftarrow S \cup \{(d_{i.s}^1 + d_{j.s}^1) \cdot x + (d_{i.y}^1 + d_{j.y}^1)\}$
  end
end
`SweepLine` (S);

Algorithm 5: $\otimes$ operation for acyclic lattices

Input: two envelopes d^1, d^2
Output: $d^1 \otimes d^2$
$S = \emptyset$
for $d_i^1 \in d^1$ **do**
  $S \leftarrow S \cup \{(d_{i.s}^1 + d_{0.s}^1) \cdot x + (d_{i.y}^1 + d_{0.y}^1)\}$
end

In (Macherey et al., 2008; Kumar et al., 2009), the $\oplus$ operation was straight-forwardly defined as `SweepLine`$(d^1 \cup d^2)$. However, for the sake of efficiency, one can use the fact that both arguments are sorted, which means that the envelop of their union can be performed in linear time, by processing them simultaneously (see algorithm 6).

4.2 Lattice optimization

Performing the full MERT cycle involves repeatedly solving the optimization problem (2) on approximations of the translation search space of increasing quality. Standard implementation of MERT alternate between the optimization of $\bar{\lambda}$ and the computation of updated lattices L (translation hypotheses). To ensure the convergence of this procedure and to avoid overfitting, we need to ensure that the lattices used at step t actually contain all the hypotheses that have served to optimize $\bar{\lambda}$ at iteration $t-1$. This requirement is typically met by merging decoder's outputs (lattice L_t) produced during all the previous iterations $L_{t-1} \dots L_1$.

Merging lattices is readily implemented using the OpenFst `Union` operation. However, the resulting acceptor might still contain a large number of duplicated paths, corresponding to identical hypotheses produced for different values of t, even after $\bar{\lambda}$ update. This leads to a waste of time while

Algorithm 6: $\oplus$ operation

Input: two envelopes d^1, d^2
Output: $d^1 \oplus d^2$
$j = 0$; $i_1 = i_2 = 0$; $S = \emptyset$
while $(i_1 < \left|d^1\right|$ **and** $i_2 < \left|d^2\right|)$ **do**
  if $(d^1_{i_1.s} < d^2_{i_2.s})$ **then**
    `Sweep` $(S, d^1_{i_1}, j)$
    $i_1 \leftarrow i_1 + 1$
  else
    `Sweep` $(S, d^2_{i_2}, j)$
    $i_2 \leftarrow i_2 + 1$
  if $(i_1 = \left|d^1\right|)$ **then**
    for $(; i_2 < \left|d^2\right| ; i_2 + +)$ **do** `Sweep` $(S, d^2_{i_2}, j)$
    break
  if $(i_2 = \left|d^2\right|)$ **then**
    for $(; i_1 < \left|d^1\right| ; i_1 + +)$ **do** `Sweep` $(S, d^1_{i_1}, j)$
    break
end
S.`resize` (j)

Algorithm 7: MERT workflow

Input: initial $\bar{\lambda}_0$, FST-archive $\mathcal{A}$, set of restart points P
Output: optimal $\bar{\lambda}^*$
for *all restart points* $p \in P$ **do**
  for *each lattice* $L \in \mathcal{A}$ **do**
    for *direction* $\bar{r} \in \mathbb{R}^n$ **do**
      init arcs $a \in L$ with singleton $\{(\bar{r} \cdot \bar{F}_a) \cdot x + \bar{\lambda} \cdot \bar{F}_a\}$
    end
    run `ShortestDistance` (L)
    get envelope of the final state
    collect its intersections and BLEU statistics
  end
  merge intersections from all the lattices
  find interval with maximum BLEU
  set $\bar{\lambda}^*$ as the middle of the winning interval
end
return $\bar{\lambda}^*$

performing shortest distance calculation. Using the OpenFst operation `Determinize` over the MERT semiring is ruled out by the fact that the determinization of a transducer requires (weak) divisibility of weights (Mohri, 2009), a property that does not hold in the MERT semiring.

To circumvent the problem, we perform the required operations `Union` and `Determinize` in the $(\min, +)$ (tropical) semiring as follows. Each input lattice is first converted into an intermediate automaton with identical arcs and states. In the new automaton, the output label of an arc compactly encodes the original output phrase and all the model scores, and the weights of all arcs are set to one. The tropical semiring being divisible, so the resulting automaton can be optimized using the standard library operations. We then restore the original encoding so as to recover a transducer with proper labels and weights.

4.3 Additional speed-ups and improvements

Finding the optimal $\bar{\lambda}^*$ for a set of translation lattices and the corresponding references is an iterative procedure detailed in algorithm 7.

Our implementation uses the optimization strategy known as Koehn's coordinate descent (Cer et al., 2008), which optimizes $\bar{\lambda}$ separately for each feature (dimension). This is a difference with the approach of (Och, 2003) which uses Powell's search algorithm. This basic approach is extended as follows: optimization can be restarted from a number of randomly chosen points, search is also performed in several random dimensions, which are regenerated after each improvement of $\bar{\lambda}$.

At each step of the MERT workflow (algorithm 7), all directions are processed simultaneously *in one single traversal of the lattice*, as explained in section 3.3. As reported in (Cer et al., 2008; Macherey et al., 2008), we have observed that using more random directions is a simple and effective means to gain up to 0.3-0.5 BLEU points.

In comparison, the improvements obtained with random restarts remained limited, except than during the first iteration. Given the time needed to reinitialize each lattice in the archive with each new starting point $\bar{\lambda}_0$ before recomputing the shortest distance, we use random restart only for the first iterations; from the second iteration on, we initialize search only with the previous best point.

We also merged two line segments of the upper envelope if generated intersection point closer than 10^{-3}, as in (Cer et al., 2008; Macherey et al., 2008), where it is claimed to make results more stable. We did not, however, notice any change in performance with or without interval merging on small datasets. Interval merging was used for the sake of decreasing the number of intervals, that saved some amount of time when computing BLEU scores for each of them. Finally note that after each round, $\bar{\lambda}^*$ was ℓ_1-normalized.

small			large		
dev	test	Δt	dev	test	Δt
15.03	16.88		20.47	20.98	
15.19	16.63	00:42	22.57	22.82	01:52
16.74	16.43	00:45	24.76	25.43	01:42
16.83	18.30	00:44	24.75	25.46	02:01
16.93	18.33	00:50	23.10	23.36	02:03
16.93	18.32	00:55	25.10	25.54	01:24
17.04	18.59	00:52	24.93	25.35	01:36
17.04	18.59	00:50	25.05	25.46	01:35
17.06	18.56	00:53	25.29	26.10	01:36
17.07	18.56	00:56	25.28	26.11	01:38
17.06	18.55	00:57	25.28	26.13	01:49
17.06	18.56	00:57			
17.06	18.56	00:58			

Table 1: n-best list MERT's dev-, test- and time-performance on both datasets.

5 Experiments

Our experiments use the n-gram approach of (Mariño et al., 2006) as implemented in the N-coder system. This implementation produces translation lattices in the form of weighted finite-state acceptors, which greatly simplifies integration with our system. Our version of N-coder uses 11 model scores. We consider a small and a larger task, both based on the data distributed for last year WMT[6] campaign: translation and language models in the former system use only the NewsCommentary dataset, the larger ones use all the data allowed in the constrained track. The smaller system is tuned on the full development set (2051 sent.), while the larger partitions it equally to optimize the language model[7] and MERT. Finally, both tasks use the official WMT'10 test set (2525 sent.).

Our baseline is the MERT distributed in the MOSES[8] toolkit with 100-best list, Koehn's coordinate descend and 20 restart points. The lattice and the baseline versions use the same $\epsilon = 10^{-5}$.

Typical runs of the baseline and our system are reported respectively, in Table 1 and Tables 2, 3 for different numbers n_r of additional random directions (0, 20 and 50). For each value of n_r, we have 3 columns: BLEU-performance on development and test sets, as well as the time (hours:minutes) taken for each iteration (includes decoding time)[9].

Our experiments showed no clear gain in term of BLEU, most probably due to the relatively small number of features used in our decoder. Papers reporting lattice MERT to increase BLEU performance typically use more features (*e.g.*, 19 in (Larkin et al., 2010)). The main benefit here seems to be speed, as convergence in lattice MERT is obtained much faster than with the baseline, which more than compensates for the increased search time. Adding random search directions seems to make a difference, but also comes at a price, as the cost increases linearly with the number of dimensions. A reasonable balance seems to be around 20-30 directions.

It may finally be noted that better stopping criteria are needed to detect convergence, as lattice MERT sometimes operates in regions where small changes in $\bar{\lambda}$ do not produce visible improvement of dev-BLEU (e.g., for $n_r = 20$ for the small set). In these regions, continuing search is highly undesirable as each subsequent iteration becomes more and more time-consuming.

$n_r = 0$			$n_r = 20$		
dev	test	Δt	dev	test	Δt
15.03	16.88	00:00	15.03	16.88	00:00
16.64	17.95	00:32	16.97	18.39	01:43
16.83	18.17	01:26	17.02	18.47	02:02
			17.02	18.46	01:20
			17.02	18.46	01:39
			17.03	18.46	02:11
			17.03	18.46	01:54
			17.03	18.46	02:02
			17.03	18.46	02:19

Table 2: Lattice MERT's dev-, test- and time-performance on the small task.

$n_r = 0$			$n_r = 20$			$n_r = 50$		
dev	test	Δt	dev	test	Δt	dev	test	Δt
20.47	20.97		20.47	20.97		20.47	20.98	
23.71	24.19	01:29	20.74	21.17	02:15	20.66	21.03	04:15
25.26	24.59	01:20	25.35	25.84	02:12	25.72	26.29	04:01
25.29	25.92	01:48	25.67	26.26	03:40	25.97	26.18	04:24
			25.72	26.41	04:00	26.00	26.24	05:21
						26.01	26.24	07:09

Table 3: Lattice MERT's dev-, test- and time-performance on the larger task.

6 Future work

In this paper, we have provided a sound formalization for the lattice MERT algorithm, resulting in an efficient implementation based on the OpenFst toolkit, and small improvements on our control test set. Further experiments with richer feature sets are needed to confirm the improvements brought by this new tuning module.

[6] www.statmt.org/wmt10

[7] See details in (Allauzen et al., 2010).

[8] www.statmt.org/moses

[9] All experiments were run on a server with 64G of memory and two Xeon processors with 4 cores at 2.27 Ghz. Lattice MERT is multi-threaded.

We believe that this procedure can still be optimized in many ways. For instance, we found that 1/3 of the total time of the optimization procedure is spent performing $\oplus$ operation. In comparison, $\otimes$ takes about 2.5 times less time. On average, each invocation of $\oplus$ eliminated 55% of lines of the union of its operand. This suggests that speed can be gained from optimizing the computations of envelopes. Another possible direction for future work is to investigate means to speed up the shortest distance calculation with heuristic search techniques. While during the first iteration this may reproduce the work of the decoder, this is especially desirable to be applied to the merged lattices used in the following iterations. This, together with the use of better stopping criteria, may prevent the uncontrolled growth of lattices.

Acknowledgments

This work has been partially funded by OSEO under the Quaero program.

References

Allauzen, Cyril, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. of the Int. Conf. on Implementation and Application of Automata*, pages 11–23.

Allauzen, Alexandre, Josep M. Crego, Ilknur Durgar El-Kahlout, and François Yvon. 2010. LIMSI's statistical translation systems for WMT'10. In *Proc. of the Joint Workshop on SMT and MetricsMATR*, pages 54–59.

Cer, Daniel, Dan Jurafsky, and Christopher D. Manning. 2008. Regularization and search for minimum error rate training. In *Proc. of the Workshop on SMT*, pages 26–34.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of the ACL*, pages 7–12.

Foster, George and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *Proc. of the Workshop on SMT*, pages 242–249.

Kumar, Shankar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of the Joint Conf. of the Annual Meeting of the ACL and the Conf. on NLP of the AFNLP*, volume 1, pages 163–171.

Lambert, Patrik and Rafael E. Banchs. 2006. Tuning Machine Translation Parameters with SPSA. In *Proc. of the Int. Workshop on Spoken Language Translation*, pages 190–196, Kyoto, Japan.

Larkin, Samuel, Boxing Chen, George Foster, Ulrich Germann, Eric Joanis, Howard Johnson, and Roland Kuhn. 2010. Lessons from NRC's Portage system at WMT 2010. In *Proc. of the Joint Workshop on SMT and MetricsMATR*, pages 127–132.

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

Mariño, José B., Rafael E. Banchs R, Josep M. Crego, Adrià de Gispert, Patrick Lambert, José A.R. Fonollosa, and Marta R. Costa-Jussà. 2006. N-gram-based machine translation. *Comput. Ling.*, 32(4):527–549.

Mohri, Mehryar. 2002. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7:321–350.

Mohri, Mehryar. 2009. Weighted automata algorithms. In Droste, Manfred, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 6, pages 213–254.

Moore, Robert C. and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proc. of the COLING*, pages 585–592, Manchester, UK.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the Annual Meeting of the ACL*, volume 1, pages 160–167.

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

Powell, M.J.D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer J.*, 7:152–162.

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

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

Zaidan, Omar F. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Ling.*, 91:79–88.

Zens, Richard, Sasa Hasan, and Hermann Ney. 2007. A systematic comparison of training criteria for statistical machine translation. In *Proc. of the Joint Conf. on EMNLP-CoNLL*, pages 524–532.