Beyond Log-Linear Models:
Boosted Minimum Error Rate Training for N-best Re-ranking

Kevin Duh[*]
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
kevinduh@u.washington.edu

Katrin Kirchhoff
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
katrin@ee.washington.edu

Abstract

Current re-ranking algorithms for machine translation rely on log-linear models, which have the potential problem of underfitting the training data. We present **BoostedMERT**, a novel boosting algorithm that uses Minimum Error Rate Training (MERT) as a weak learner and builds a re-ranker far more expressive than log-linear models. BoostedMERT is easy to implement, inherits the efficient optimization properties of MERT, and can quickly boost the BLEU score on N-best re-ranking tasks. In this paper, we describe the general algorithm and present preliminary results on the IWSLT 2007 Arabic-English task.

1 Introduction

N-best list re-ranking is an important component in many complex natural language processing applications (e.g. machine translation, speech recognition, parsing). Re-ranking the N-best lists generated from a 1st-pass decoder can be an effective approach because (a) additional knowledge (features) can be incorporated, and (b) the search space is smaller (i.e. choose 1 out of N hypotheses).

Despite these theoretical advantages, we have often observed little gains in re-ranking machine translation (MT) N-best lists in practice. It has often been observed that N-best list rescoring only yields a moderate improvement over the first-pass output although the potential improvement as measured by the oracle-best hypothesis for each sentence is much higher. This shows that hypothesis features are either not discriminative enough, or that the reranking model is too weak

This performance gap can be mainly attributed to two problems: optimization error and modeling error (see Figure 1).[1] Much work has focused on developing better algorithms to tackle the optimization problem (e.g. MERT (Och, 2003)), since MT evaluation metrics such as BLEU and PER are riddled with local minima and are difficult to differentiate with respect to re-ranker parameters. These optimization algorithms are based on the popular log-linear model, which chooses the English translation e of a foreign sentence f by the rule:

$$\arg\max_e p(e|f) \equiv \arg\max_e \sum_{k=1}^{K} \lambda_k \phi_k(e,f)$$

where $\phi_k(e,f)$ and λ_k are the K features and weights, respectively, and the argmax is over all hypotheses in the N-best list.

We believe that standard algorithms such as MERT already achieve low optimization error (this is based on experience where many random re-starts of MERT give little gains); instead the score gap is mainly due to modeling errors. Standard MT systems use a small set of features (i.e. $K \approx 10$) based on language/translation models.[2] Log-linear models on such few features are simply not expressive enough to achieve the oracle score, regardless of how well the weights $\{\lambda_k\}$ are optimized.

[*] Work supported by an NSF Graduate Research Fellowship.

[1] Note that we are focusing on closing the gap to the oracle score on the training set (or the development set); if we were focusing on the test set, there would be an additional term, the generalization error.

[2] In this work, we do not consider systems which utilize a large smorgasbord of features, e.g. (Och and others, 2004).

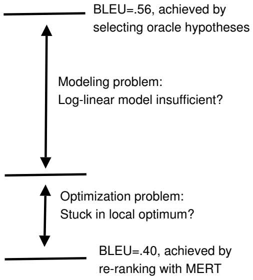

Figure 1: Both modeling and optimization problems increase the (training set) BLEU score gap between MERT re-ranking and oracle hypotheses. We believe that the modeling problem is more serious for log-linear models of around 10 features and focus on it in this work.

To truly achieve the benefits of re-ranking in MT, one must go beyond the log-linear model. The re-ranker should not be a mere dot product operation, but a more dynamic and complex decision maker that exploits the structure of the N-best re-ranking problem.

We present **BoostedMERT**, a general framework for learning such complex re-rankers using standard MERT as a building block. BoostedMERT is easy to implement, inherits MERT's efficient optimization procedure, and more effectively boosts the training score. We describe the algorithm in Section 2, report experiment results in Section 3, and end with related work and future directions (Sections 4, 5).

2 BoostedMERT

The idea for BoostedMERT follows the boosting philosophy of combining several weak classifiers to create a strong overall classifier (Schapire and Singer, 1999). In the classification case, boosting maintains a distribution over each training sample: the distribution is increased for samples that are incorrectly classified and decreased otherwise. In each boosting iteration, a weak learner is trained to optimize on the weighted sample distribution, attempting to correct the mistakes made in the previous iteration. The final classifier is a weighted combination of weak learners. This simple procedure is very effective in reducing training and generalization error.

In BoostedMERT, we maintain a sample distribution $d_i, i = 1 \dots M$ over the M N-best lists.[3] In each boosting iteration t, MERT is called as as sub-procedure to find the best feature weights λ^t on d_i.[4] The sample weight for an N-best list is increased if the currently selected hypothesis is far from the oracle score, and decreased otherwise. Here, the oracle hypothesis for each N-best list is defined as the hypothesis with the best sentence-level BLEU. The final ranker is a combination of (weak) MERT ranker outputs.

Algorithm 1 presents more detailed pseudocode. We use the following notation: Let $\{\mathbf{x_i}\}$ represent the set of M training N-best lists, $i = 1 \dots M$. Each N-best list $\mathbf{x_i}$ contains N feature vectors (for N hypotheses). Each feature vector is of dimension K, which is the same dimension as the number of feature weights λ obtained by MERT. Let $\{\mathbf{b_i}\}$ be the set of BLEU statistics for each hypothesis in $\{\mathbf{x_i}\}$, which is used to train MERT or to compute BLEU scores for each hypothesis or oracle.

Algorithm 1 BoostedMERT

```
Input: N-best lists $\{\mathbf{x_i}\}$, BLEU scores $\{\mathbf{b_i}\}$
Input: Initialize sample distribution $d_i$ uniformly
Input: Initialize $y^0 = [0]$, a constant zero vector
Output: Overall Ranker: $f^T$
for $t = 1$ to $T$ do
    Weak ranker: $\lambda^t$ = MERT($\{\mathbf{x_i}\}$,$\{\mathbf{b_i}\}$,$d_i$)

    if ($t \geq 2$): $\{y^{t-1}\}$ = PRED($f^{t-1}, \{\mathbf{x_i}\}$)
    $\{y^t\}$ = PRED($\lambda^t, \{\mathbf{x_i}\}$)
    $\alpha^t$ = MERT($[y^{t-1}; y^t]$,$\{\mathbf{b_i}\}$)
    Overall ranker: $f^t = y^{t-1} + \alpha^t y^t$

    for $i = 1$ to $M$ do
        $a_i$ = [BLEU of hypothesis selected by $f^t$]
        divided by [BLEU of oracle hypothesis]
        $d_i$ = $\exp(-a_i)$/normalizer
    end for
end for
```

[3] As such, it differs from RankBoost, a boosting-based ranking algorithm in information retrieval (Freund et al., 2003). If applied on MT, RankBoost would maintain a weight for each pair of hypotheses and would optimize a pairwise ranking metric, which is quite dissimilar to BLEU.

[4] This is done by scaling each BLEU statistic, e.g. n-gram precision, reference length, by the appropriate sample weights before computing corpus-level BLEU. Alternatively, one could sample (with replacement) the N-best lists using the distribution and use the resulting stochastic sample as input to an unmodified MERT procedure.

The pseudocode can be divided into 3 sections:

1. Line 2 finds the best log-linear feature weights on distribution d_i. MERT is invoked as a weak learner, so this step is computationally efficient for optimizing MT-specific metrics.

2. Lines 4-7 create an overall ranker by combining the outputs of the previous overall ranker f^{t-1} and current weak ranker λ^t. PRED is a general function that takes a ranker and a M N-best lists and generates a set of M N-dim output vector y representing the predicted reciprocal rank. Specifically, suppose a 3-best list and a ranker predicts ranks (1,3,2) for the 1st, 2nd, and 3rd hypotheses, respectively. Then y = (1/1,1/3,1/2) = (1,0.3,0.5).[5]

   Finally, using a 1-dimensional MERT, the scalar parameter α^t is optimized by maximizing the BLEU of the hypothesis chosen by $y^{t-1}+\alpha^t y^t$. This is analogous to the line search step in boosting for classification (Mason et al., 2000).

3. Lines 9-11 update the sample distribution d_i such that N-best lists with low accuracies a_i are given higher emphasis in the next iteration. The per-list accuracy a_i is defined as the ratio of selected vs. oracle BLEU, but other measures are possible: e.g. ratio of ranks, difference of BLEU.

The final classifier f^T can be seen as a voting procedure among multiple log-linear models generated by MERT. The weighted vote for hypotheses in an N-best list $\mathbf{x_i}$ is represented by the N-dimensional vector: $\hat{y} = \sum_{t=1}^{T} \alpha^t y^t = \sum_{t=1}^{T} \alpha^t$ PRED$(\lambda^t, \mathbf{x_i})$. We choose the hypothesis with the maximum value in $\hat{y}$

Finally, we stress that the above algorithm is an novel extension of boosting to re-ranking problems. There are many open questions and one can not always find a direct analog between boosting for classification and boosting for ranking. For instance, the distribution update scheme of Lines 9-11 is recursive in the classification case (i.e. $d_i = d_i * \exp$(LossOfWeakLearner)), but due to the non-decompositional properties of $\arg\max$ in re-ranking, we have a non-recursive equation based on the overall learner ($d_i = \exp$(LossOfOverallLearner)). This has deep implications on the dynamics of boosting, e.g. the distribution may stay constant in the non-recursive equation, if the new weak ranker gets a small α.

3 Experiments

The experiments are done on the IWSLT 2007 Arabic-to-English task (clean text condition). We used a standard phrase-based statistical MT system (Kirchhoff and Yang, 2007) to generated N-best lists (N=2000) on `Development4`, `Development5`, and `Evaluation` sub-sets. `Development4` is used as the Train set; N-best lists that have the same sentence-level BLEU statistics for all hypotheses are filtered since they are not important in impacting training. `Development5` is used as Dev set (in particular, for selecting the number of iterations in boosting), and `Evaluation` (Eval) is the blind dataset for final ranker comparison. Nine features are used in re-ranking.

We compare MERT vs. BoostedMERT. MERT is randomly re-started 30 times, and BoostedMERT is run for 30 iterations, which makes for a relatively fair comparison. MERT usually does not improve its Train BLEU score, even with many random restarts (again, this suggests that optimization error is low). Table 1 shows the results, with BoostedMERT outperforming MERT 42.0 vs. 41.2 BLEU on Eval. BoostedMERT has the potential to achieve 43.7 BLEU, if a better method for selecting optimal iterations can be devised.

It should be noted that the Train scores achieved by both MERT and BoostedMERT is still far from the oracle (around 56). We found empirically that BoostedMERT is somewhat sensitive to the size (M) of the Train set. For small Train sets, BoostedMERT can improve the training score quite drastically; for the current Train set as well as other larger ones, the improvement per iteration is much slower. We plan to investigate this in future work.

[5] There are other ways to define a ranking output that are worth exploring. For example, a hard argmax definition would be (1,0,0); a probabilistic definition derived from the dot product values can also be used. It is the definition of PRED that introduces non-linearities in BoostedMERT.

	MERT	BOOST	Δ
Train, Best BLEU	40.3	41.0	0.7
Dev, Best BLEU	24.0	25.0	1.0
Eval, Best BLEU	41.2	43.7	2.5
Eval, Selected BLEU	41.2	42.0	0.8

Table 1: The first three rows show the BLEU score for Train, Dev, and Eval from 30 iterations of BoostedMERT or 30 random re-restarts of MERT. The last row shows the actual BLEU on Eval when selecting the number of boosting iterations based on Dev. Last column indicates absolute improvements. BoostedMERT outperforms MERT by 0.8 points on Eval.

4 Related Work

Various methods are used to optimize log-linear models in re-ranking (Shen et al., 2004; Venugopal et al., 2005; Smith and Eisner, 2006). Although this line of work is worthwhile, we believe more gain is possible if we go beyond log-linear models. For example, Shen's method (2004) produces large-margins but observed little gains in performance.

Our BoostedMERT should not be confused with other boosting algorithms such as (Collins and Koo, 2005; Kudo et al., 2005). These algorithms are called boosting because they iteratively choose features (weak learners) and optimize the weights for the boost/exponential loss. They do not, however, maintain a distribution over N-best lists.

The idea of maintaining a distribution over N-best lists is novel. To the best of our knowledge, the most similar algorithm is AdaRank (Xu and Li, 2007), developed for document ranking in information retrieval. Our main difference lies in Lines 4-7 in Algorithm 1: AdaRank proposes a simple closed form solution for α and combines only weak features, not full learners (as in MERT). We have also implemented AdaRank but it gave inferior results.

It should be noted that the theoretical training bounds derived in the AdaRank paper is relevant to BoostedMERT. Similar to standard boosting, this bound shows that the training score can be improved exponentially in the number of iterations. However, we found that the conditions for which this bound is applicable is rarely satisfied in our experiments.[6]

5 Conclusions

We argue that log-linear models often underfit the training data in MT re-ranking, and that this is the reason we observe a large gap between re-ranker and oracle scores. Our solution, BoostedMERT, creates a highly-expressive ranker by voting among multiple MERT rankers.

Although BoostedMERT improves over MERT, more work at both the theoretical and algorithmic levels is needed to demonstrate even larger gains. For example, while standard boosting for classification can exponentially reduce training error in the number of iterations under mild assumptions, these assumptions are frequently not satisfied in the algorithm we described. We intend to further explore the idea of boosting on N-best lists, drawing inspirations from the large body of work on boosting for classification whenever possible.

References

M. Collins and T. Koo. 2005. Discriminative reranking for natural langauge parsing. *Computational Linguistics*, 31(1).

Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4.

K. Kirchhoff and M. Yang. 2007. The UW machine translation system for IWSLT 2007. In *IWSLT*.

T. Kudo, J. Suzuki, and H. Isozaki. 2005. Boosting-based parse reranking with subtree features. In *ACL*.

L. Mason, J. Baxter, P. Bartless, and M. Frean. 2000. Boosting as gradient descent. In *NIPS*.

F.J. Och et al. 2004. A smorgasbord of features for statistical machine translation. In *HLT/NAACL*.

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

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3).

L. Shen, A. Sarkar, and F.J. Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*.

D. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of COLING/ACL Companion Volume*.

A. Venugopal, A. Zollmann, and A. Waibel. 2005. Training and evaluating error minimization rules for SMT. In *ACL Workshop on Building/Using Parallel Texts*.

J. Xu and H. Li. 2007. AdaRank: A boosting algorithm for information retrieval. In *SIGIR*.

[6] The explanation for this is beyond the scope of this paper; the basic reason is that our weak rankers (MERT) are not weak in practice, so that successive iterations get diminishing returns.