Non-linear n-best List Reranking with Few Features

Artem Sokolov*
Dept. of Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
sokolov@cl.uni-heidelberg.de

Guillaume Wisniewski and **François Yvon**
LIMSI-CNRS & Univ. Paris Sud
BP-133
91403 Orsay, France
{firstname.lastname}@limsi.fr

Abstract

In Machine Translation, it is customary to compute the model score of a predicted hypothesis as a linear combination of multiple features, where each feature assesses a particular facet of the hypothesis. The choice of a linear combination is usually justified by the possibility of efficient inference (decoding); yet, the appropriateness of this simple combination scheme to the task at hand is rarely questioned. In this paper, we propose an approach that replaces the linear scoring function with a *non-linear* scoring function. To investigate the applicability of this approach, we rescore n-best lists generated with a conventional machine translation engine (using a linear scoring function for generating its hypotheses) with a non-linear scoring function learned using the learning-to-rank framework. Moderate, though consistent, gains in BLEU are demonstrated on the WMT'10, WMT'11 and WMT'12 test sets.

1 Introduction

In modern statistical machine translation (SMT), the dominating approach to model the probability that sentence $\mathbf{e}$ is a translation of source sentence $\mathbf{f}$ is to use linear models (Och and Ney, 2002): $p(\mathbf{e}, \mathbf{a}|\mathbf{f}) \sim \exp(\bar{\lambda} \cdot \bar{g}(\mathbf{a}, \mathbf{e}, \mathbf{f}))$, where $\mathbf{a}$ is an alignment between $\mathbf{e}$ and $\mathbf{f}$, $\bar{g}(\mathbf{a}, \mathbf{e}, \mathbf{f})$ is the *feature vector* representing various compatibility measures between $\mathbf{a}$, $\mathbf{e}$ and $\mathbf{f}$, and $\bar{\lambda}$ is a parameter vector. Using this model, searching for the most probable translation boils down to finding $(\mathbf{a}, \mathbf{e})$ that maximizes the *scoring function* $\bar{\lambda} \cdot \bar{g}(\mathbf{a}, \mathbf{e}, \mathbf{f})$.

Several papers have recently pointed out that the scoring function is the main bottleneck of today's SMT systems: the search space of decoders contains hypotheses of very high quality that are discarded because of their model score (Wisniewski et al., 2010; Sokolov et al., 2012; Turchi et al., 2012). The choice of a linear model seems mainly motivated by the simplicity of integrating the scoring function during decoding and of optimizing the model during training. It may also be motivated by the acknowledged success of linear models in many NLP tasks. The situation in SMT is quite different: while several SMT systems have been proposed that use thousands (Chiang et al., 2009) or millions (Lavergne et al., 2011) of features, in practice, however, the majority of available systems are based on linear scoring functions defined over a very small number of features (between 10 and 20). In the same time, most NLP systems routinely use million of features to achieve state-of-the-art performance.

The small number of features and the simplicity of the scoring model contrast with automatic evaluation metrics that hinge on complex quality measures, specially hand-crafted to mimic the human notion of translation quality. Because of the difficulty or even impossibility to adequately define the latter, quality measures generally depend on multiple inter-constrained characteristics describing the source sentence and the translation hypotheses. For instance, popular evaluation metrics, like BLEU (Papineni et al., 2002) or METEOR (Banerjee and Lavie, 2005), consider quantity and (fuzzy) alignments of

*This work was done while the first author was at LIMSI.

common n-grams in a reference and a hypothesis.

To sum up, approximating such complex quality measures with a linear combination of a few loosely related probabilistic features appears like a daunting task and there is little chance that current scoring functions can actually sort good translations from the remaining lot of hypotheses in the low-dimensional feature space. Indirect confirmation of the difficulty of this task comes from the inability of MERT's advanced variants to come nearer to oracle BLEU scores or to substantially increase performance (Kumar et al., 2009), even when an almost exact optimization method is used (Galley and Quirk, 2011). Modeling inadequacy, and, in particular, the use of over-simplistic linear scoring functions in low-dimensional space can be held responsible for this disappointing performance.

This paper can be seen as an attempt to verify whether the mere replacement of a linear with a non-linear scoring function in a conventional phrase-based SMT system that uses only a few dozens features can actually improve performance, by capturing more precisely the complex boundaries between good and bad translations.

The rest of the paper is organized as follows. In the next section, we review related work. In Section 3, we describe a ranking approach to tuning SMT systems, together with our method of learning a non-linear scoring function in the learning-to-rank paradigm. Next, we explain feature transformations (Section 4) used in the experiments reported in Section 5. Discussions in Section 6 close the paper.

2 Related Work

Recently, new approaches to tuning SMT systems have received attention, namely the ranking methods (Hopkins and May, 2011; Haddow et al., 2011). The motivation for these is as follows. Although BLEU is defined for a pair of corpora, one can use the same formula to calculate a sentence-level approximation of BLEU that evaluates the similarity between a single hypothesis $\mathbf{e}$ and its reference $\mathbf{r}$, and to order hypotheses according to it. This natural ordering is used by ranking approaches in SMT to learn system parameters, taking advantage of the fact that one can deduce information about parameters even from the comparison between mediocre or bad hypotheses.

Ranking approaches, however, were until now used only from the perspective of redefining the target loss in optimization. Scoring functions remained simple linear combinations, and the demonstrated performance remains basically the same as for classical MERT (Hopkins and May, 2011). As we will see, while our gains are still modest, they are higher than those obtained with previous ranking approaches based on linear scoring functions.

One method for deriving flexible scoring functions is boosting. Being an attractive learning algorithm, it was applied several times in the context of SMT. However, to the best of our knowledge all attempts concentrated on boosting for classification (like AdaBoost) and boosting from the ranking perspective was never applied to machine translation. Duh and Kirchhoff (2008) and Xiao et al. (2010) use the whole MERT procedure as a weak learner and maintain a distribution over n-best-lists to allow concentrating on the ones where, under current model, a winning hypothesis is too far from this n-best-lists' oracle. The definition of BLEU needs to be changed to allow running MERT on weighted n-best-lists. The final model is a voting scheme of the linear models found on each invocation of weak MERT. Although in the end, a non-linear scoring function can be obtained, this non-linearity is a byproduct of the voting selection process and, contrary to our approach, is not constructed directly. Lagarda and Casacuberta (2008) apply AdaBoost by reweighting on each boosting iteration a separate "translation model" introduced into the linear model.

3 Non-Linear Hypotheses Reranking

Motivated by the inability of linear models to improve standard MERT training, we propose to learn the scoring function of an SMT system in a richer class of functions, namely the family of linear combinations of simple *non-linear* functions defined on individual features.

The well-known RankBoost algorithm (Freund et al., 2003) is able to efficiently select the best predictor in this class of functions by optimizing a weighted pair-wise ranking loss. This loss, closely related to the one introduced by Hopkins and May (2011), penalizes incorrect (with respect to a

sentence-level evaluation of translation quality) orderings of pairs of hypotheses.

We restrict ourselves to n-best list reranking (as do most ranking approaches) to avoid, at this preliminary stage, tighter integration with decoder, which is still a subject of ongoing work. We thus concentrate, in this paper, on *post-factum* reranking of the n-best-list produced by the last iteration of MERT.

3.1 Finding Rescoring Function with Boosting

In this section, we first briefly recollect the principle of RankBoost and explain how it can be used to train the scoring function of an SMT system.

RankBoost learns a scoring function H, defined on feature vectors, which is a linear combination of T simple, non-linear functions h_t defined on feature vectors $\bar{g}(\mathbf{e}, \mathbf{f})$[1] and called *weak learners*:

$$H(\bar{g}(\mathbf{e}, \mathbf{f}); \bar{\lambda}) = \sum_{t=1}^{T} \alpha_t h_t(\bar{g}(\mathbf{e}, \mathbf{f})), \qquad (1)$$

where each α_t is the weight assigned to the weak function h_t and $\bar{\lambda}$ denotes the set of parameters of the scoring function, namely the coefficients $(\alpha)_{t=1}^{T}$ and the parameters of the weak learners.

The function H is expected to score hypotheses: if hypothesis $\mathbf{e}_1$ is preferred to $\mathbf{e}_2$, then $H(\bar{g}(\mathbf{e}_1, \mathbf{f}); \bar{\lambda})$ should be greater than $H(\bar{g}(\mathbf{e}_2, \mathbf{f}); \bar{\lambda})$. RankBoost achieves this by optimizing a convex upper bound of the following loss function (Freund et al., 2003):

$$\mathcal{L} = \sum_{\mathbf{f}} \sum_{\substack{\mathbf{e}_i, \mathbf{e}_j \in n\text{-best}(\mathbf{f}) \\ b(\mathbf{e}_i, \mathbf{r_f}) < b(\mathbf{e}_j, \mathbf{r_f})}} D(i,j) [\![H(\bar{g}(\mathbf{e}_i, \mathbf{f})) \geq H(\bar{g}(\mathbf{e}_j, \mathbf{f}))]\!]$$

where $b(\mathbf{e}, \mathbf{r_f})$ is the value of the sentence-level quality measure (e.g., BLEU) between hypothesis $\mathbf{e}$ and reference $\mathbf{r_f}$. Operator $[\![A]\!] = 1$ if the A is true and 0 otherwise and the positively-valued preference values D weight hypothesis pairs from the training set. The higher $D(i,j)$, the more important it is to preserve the relative ordering of two hypotheses $\mathbf{e}_i$ and $\mathbf{e}_j$ (for example, a very good $\mathbf{e}_j$ against very poor $\mathbf{e}_i$). Thus, for each source sentence the loss $\mathcal{L}$ adds up a penalty equals to $D(i,j)$ each time a low quality (according to b) hypothesis $\mathbf{e}_i$ is scored higher than a high quality hypothesis $\mathbf{e}_j$. The pairs over which the loss is defined, are drawn from n-best lists that correspond to each source sentence $\mathbf{f}$.

Optimizing the loss function $\mathcal{L}$ to find the scoring function H is done in a step-wise fashion outlined in Algorithm 1 (for details see (Freund et al., 2003)). Having built H with t weak learners, the next in turn $(t+1)$'s weak function is selected, optimized and weighted with the corresponding coefficient α_{t+1}. In addition to the training of the t-th weak learner, on each step t of the learning process, values of importance weights D are updated to concentrate on hypotheses pairs that have not been correctly ranked so far (Freund et al., 2003).

Algorithm 1: RankBoost optimization cycle

Input: max number of weak learners T, initial distribution $D_1(i,j)$ on hypotheses pairs
Output: set of weak learners $\{h_t, \alpha_t\}, t = 1, \ldots, T$
for $t = 1, \ldots, T$ **do**

1. Find weak learner h_t by minimizing Z_t over h

$$Z_t(\alpha, h) = \sum_{\mathbf{f}} \sum_{i,j} D_t(i,j) e^{\alpha(h(\bar{g}(\mathbf{e}_i, \mathbf{f})) - h(\bar{g}(\mathbf{e}_j, \mathbf{f})))}$$

2. Find optimal α_t by minimizing $Z_t(\alpha, h)$ over α
3. Update

$$D_{t+1}(i,j) = \frac{1}{Z_t} D_t(i,j) e^{\alpha_t(h_t(\bar{g}(\mathbf{e}_i, \mathbf{f})) - h_t(\bar{g}(\mathbf{e}_j, \mathbf{f})))},$$

where $= Z_t(\alpha_t, h_t)$ is a normalization factor

end

3.2 Weak Learners

The success of RankBoost relies on the definition a good family of weak learners, powerful enough to capture information relevant to ranking, and in the same time robust to over-fitting. We tested three families of weak learners that only depend on one single feature:

- decision stumps

$$h(\bar{g}(\mathbf{e}, \mathbf{f}); \theta, k) = [\![g^k > \theta]\!],$$

- linear weak learners:

$$h(\bar{g}(\mathbf{e}, \mathbf{f}); k) = g^k, \text{ and}$$

[1] In the remainder, we will omit alignment $\mathbf{a}$ from the arguments of feature vectors $\bar{g}(\mathbf{e}, \mathbf{a}, \mathbf{f})$ to simplify notation.

- piece-wise linear decision learners:

$$h(\bar{g}(\mathbf{e}, \mathbf{f}); \theta, k) = g^k \cdot [\![g^k > \theta]\!],$$

where k is the selected feature index and θ is a learned threshold. The first family is arguably the simplest possible family of 1-dimensional non-linear functions, with which it is however possible to approximate complex curves. Linear weak learners are interesting, as they allow us to discriminate between the effects of adopting ranking loss and of introducing non-linearity in scoring function. Finally, piece-wise linear functions are capable, on the one hand, of modeling linear dependencies in regions where decision stumps would require many more weak learners to approximate a line. On the other hand, they are more flexible than simple linear functions. Being capable of modeling functions that are more generic than linear functions remains important for the task of n-best list reranking as the decoder uses beam-search pruning based on linear scoring functions.

Training stump learners can be done using the approximate "3-rd method" described in (Freund et al., 2003), the other two learners using a straightforward generalization of the same method.

4 Features

4.1 Baseline Configurations

We test our proposal on two decoder configurations that differ by the number of features considered. First, the *basic configuration* uses only the 11 features routinely found in any SMT decoder;[2] the *extended configuration* contains an enriched set of 23 features and corresponds to the state-of-the-art translation system – the best system for the French-English pair in the recent WMT'12 evaluation (Le et al., 2012b). The additional features considered are mainly based on neural network language models and translation models (Le et al., 2011; Le et al., 2012a). These features are integrated within a reranking step optimized with MERT.[3] A summary of the number of features used in each system is given in Table 1.

	configuration	feature sets	#features
linear	basic		11
	extended		23
non-linear	basic	scale	32
		scale & rank	44
	extended	scale	68
		scale & rank	92

Table 1: Feature numbers for the baseline linear and two tested non-linear configurations, with two variants of feature sets.

4.2 Feature Transformations for Reranking

In a linear model, the order induced by the scoring function is kept unchanged when the features are rescaled. This is no longer the case for non-linear models the output of which depends on the actual feature value. Thus features should now be regularized so that a learner (applied in a same fashion to each hypothesis) could learn a consistent model.

In this work, we consider three different ways to regularize features. First, to make them comparable, we normalize features by dividing them by the number of words and (separately) by the number of phrases contained in a particular output hypothesis. In this way we obtained about twice more features then we initially had.

To further regularize feature values, we rescale them into the interval $[0, 1]$: for each separate source sentence $\mathbf{f}$, and for each hypothesis $\mathbf{e}_j \in n\text{-best}(\mathbf{f})$, feature g_j^k was mapped to:

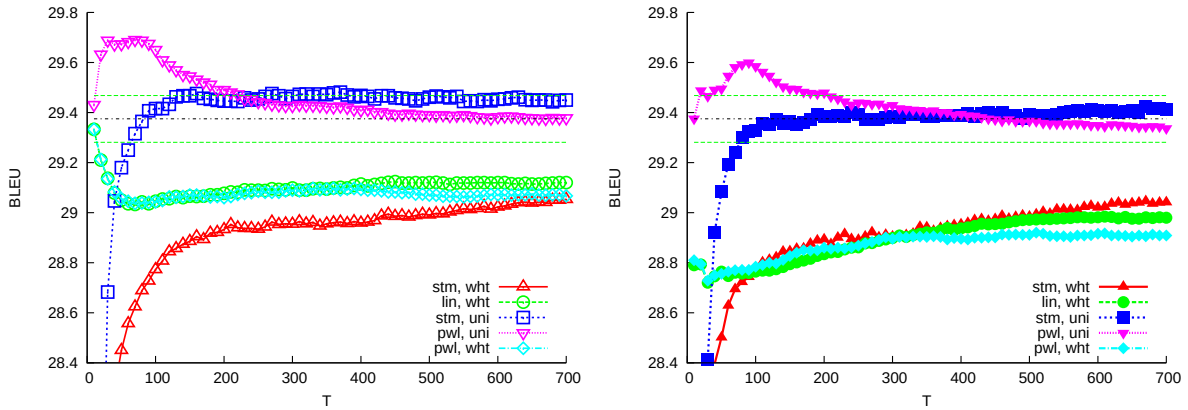
$$\tilde{g}_j^k = \frac{g_j^k - \min_{i \in n\text{-best}(\mathbf{f})} g_i^k}{\max_{i \in n\text{-best}(\mathbf{f})} g_i^k - \min_{i \in n\text{-best}(\mathbf{f})} g_i^k}.$$

We call these renormalized features *scale-features*.

Finally, we also consider additional *rank-features*: for each hypothesis $\mathbf{e}_j$ and feature g_j^k, the rank feature is equal to the rank of this hypothesis if the n-best$(\mathbf{f})$ were sorted by the value of g^k.

[2] Target language model, translation model, 2 CFB lexicalized reordering models, 4 lexical translation weights, distortion and 2 penalties for words and phrases: 11 features in total.

[3] To construct the extended feature set, the basic feature set was first augmented with two supplementary translation models on bilingual tuples and four lexicalized reordering features, for a total of 17 features. After tuning with MERT and decoding to obtain 300-best lists, we further augmented them with the 5 following features: the optimized linear score (over previous 17 features) for each hypotheses, a feature for a continuous space *monolingual* neural-network target language model (Le et al., 2011) and four supplementary *bilingual* neural-network translation models (Le et al., 2012a). In total – 23 features, on which MERT was relaunched to obtain the extended baseline.



(a) scale-features (b) scale- & rank-features

Figure 1: Comparison of weak learners, weighting schemes and feature sets on the WMT'10 test set. Weighting: uniform (`uni`), weighted with sentence-BLEU (`wht`). Learners: stumps (`stm`), linear (`lin`), piece-wise linear (`pwl`). Mean baseline MERT score is shown with 1σ interval.

In our experiments, we only consider scale-features (after normalization) and rank-features. Before feature rescaling an additional `score` feature is added that is equal to the score of the linear model found by MERT. Original features are discarded to avoid interference with learning. In the following, we make a distinction between configurations with only scale-features and configurations where scale-features are combined with rank-features (Table 1).

5 Experiments

We now describe the experiments conducted to validate our approach. We will first describe our experimental setting, then evaluate the impact of the various hyper-parameters of RankBoost and finally evaluate the results achieved in terms of translation quality.

5.1 Experimental Setting

We evaluate our approach for the French-English language pair with the N-code[4] decoder. Similar results were obtained with Moses but are not reported here because of space reasons.

Basic and extended systems are trained on the data provided for the WMT'12 Evaluation task[5], tuned with MERT on the WMT'09 test data and evaluated on WMT'10, WMT'11 and the WMT'12 test sets.[6] For each baseline (basic and extended) configuration, we run 8 independent MERT pipelines on the `newstest2009` set, each with 20 restart points and 30 random independent directions supplemental to the default axis-aligned direction. For the extended configuration, n-best lists of the last MERT iteration are augmented with 5 neural-network models (Section 4.1) and reoptimized with MERT before applying features transformations.

RankBoost training is performed on the WMT'09 evaluation set on 100-best and 300-best lists, respectively, for the basic and extended configurations, using the final n-best lists after the complete MERT optimization, separately for each MERT rerun.

In all our experiments, we consider the sentence level BLEU+1 approximation (Lin and Och, 2004) to evaluate translation quality. Similarly to (Hopkins and May, 2011), to reduce the number of pairs and to speed up learning we sampled the n-best lists leaving only $2,000$ randomly selected pairs with quality difference superior to 0.05 BLEU points. Tests with different number of sampled pairs showed little sensitivity to this parameter in the range between 50 and $5,000$ pairs (outside of this interval the performance considerably decreases). Conversion to a bipartite ranking problem (that enables a more efficient and simpler algorithm (Freund et al., 2003)) with gaps did marginally help on for 100-best lists; as the size of the n-best list goes up the effect vanishes.

[4] http://ncode.limsi.fr/

[5] http://www.statmt.org/wmt12

[6] All BLEU scores are reported using the multi-bleu.pl script on the scale from 0 to 100.

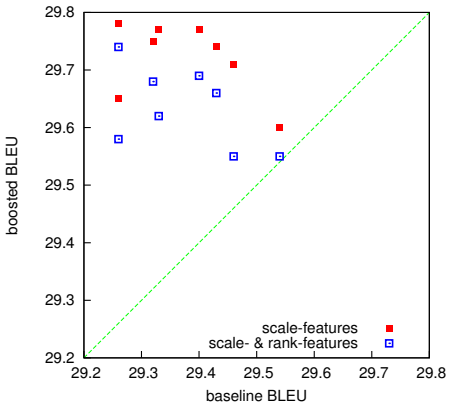


Figure 2: Maximum relative gains in BLEU on the WMT'10 test set. Each point on the x-axis corresponds to an outcome of one of 8 reruns of MERT; while the y-axis shows the peak BLEU (over T) for the same system after reranking with RankBoost.

5.2 Selecting RankBoost Parameters

Reranking n-best lists with RankBoost requires us to choose a family of weak learners and a weighting scheme. In addition to the three weak learner families introduced in Section 3.2, we test two weighting schemes: *uniform*, in which $D(i,j) = 1$ for all i, j and *weighted*, in which $D(i,j) = b(\mathbf{e}_j, \mathbf{r}) - b(\mathbf{e}_i, \mathbf{r}) > 0$. The weighted scheme gives more importance to pairs with large difference in their scores, which, intuitively should help learning. Such a scheme has already been proposed in (Hopkins and May, 2011).

The comparison of weighting schemes and weak families is depicted in Figure 1. As can be seen in the plot, the piece-wise linear weak learners outperform all others configurations by up to 1 BLEU point whatever features set is used.

Contrary to our anticipations regarding weighted loss optimization, the uniform scheme performs much better than the other one. This result may be explained by the noise contained in the training set, especially in the tail of the n-best list, which makes efforts in modeling the complete order useless.

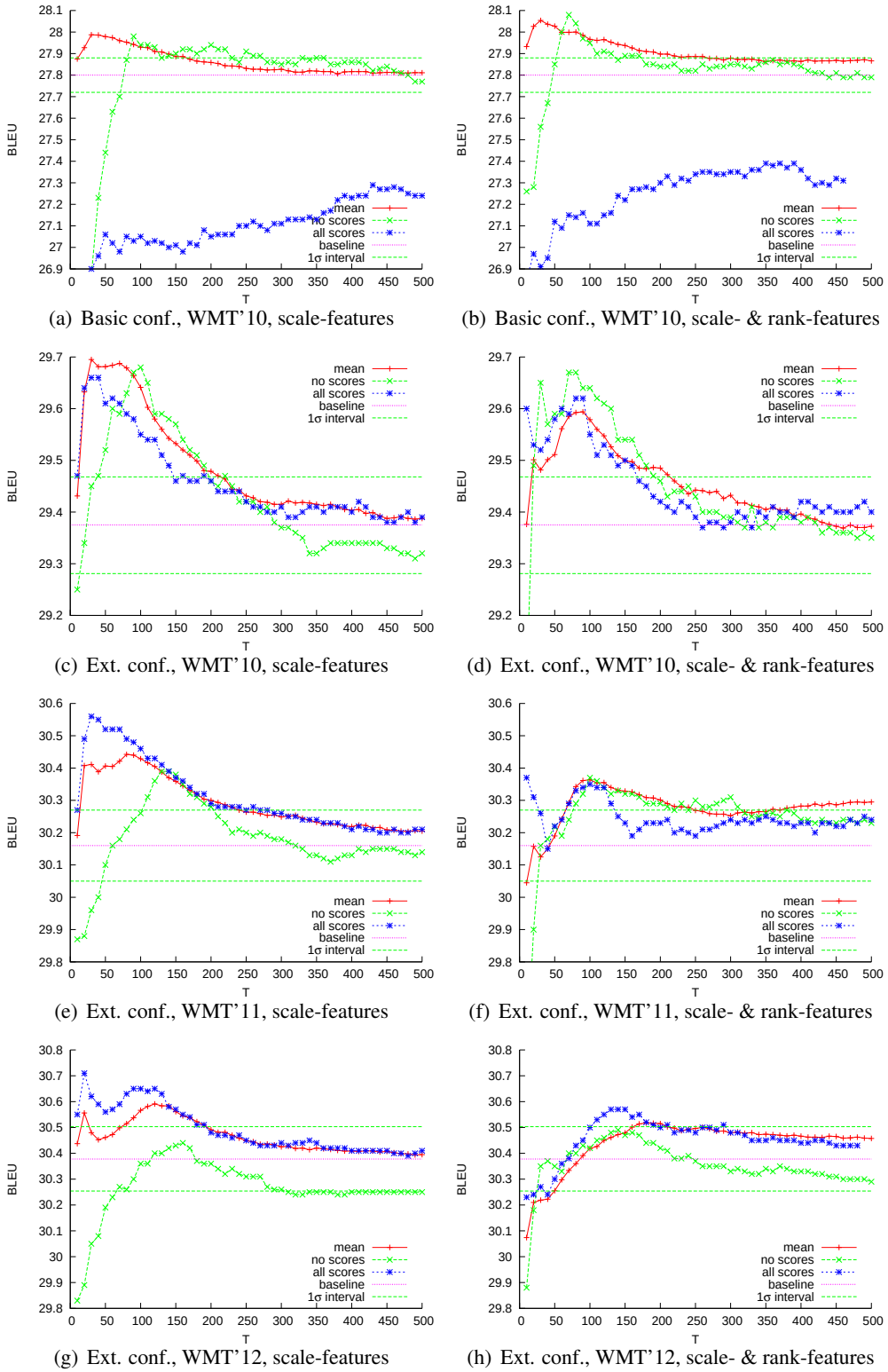
In view of these two findings, in the remainder, we only report the performance for the uniform weighting and piece-wise linear weak functions.

5.3 RankBoost Performance

The achieved maximum (over T) final BLEU-gains exhibit an unexpected dependence on the performance of the MERT-optimized system it is trained from. The BLEUs for 8 different runs of MERT and the best scores achieved by reranking are depicted in Figure 2. It clearly appears that poor-performing runs of MERT, possibly stuck in a local minima, are regularly amenable to a much larger relative improvement with non-linear reranking, surpassing the results of the best MERT run. While it is expected that re-ranking would make worse systems go relatively farther, provided the improvement were up to roughly the same BLEU score on a given test set, in practice the *absolute* values of BLEU for a non-linearly rescored "poor" MERT-optimized system always surpass these values for a "good" system. Indeed, the best final rescoring results were obtained for the runs that performed the worst with a linear function and the runs with best performance after MERT show little improvement after non-linear rescoring and the worst final absolute BLEUs. Further investigation is needed to understand and exploit this behavior. On Figure 3, the performance on two test sets is represented as a function of the number of weak learners T. Means and variances of different MERT runs are also reported. The results show that it is possible to gain up to 0.4 in BLEU over standard MERT training by using non-linear scoring functions. It also appears that the impact of the weak learner numbers is always the same: at the beginning, performance quickly improves with T, reaches a peak and, then, slowly decreases, which means the reranker is overfitting the training set. Consequently, for the application of our method, it is necessary to use a validation set to find the optimal value for T.

The results of applying this methods are presented in Table 2 for the extended configuration with scale-features. For these experiments the value of T is chosen by maximizing the BLEU score on WMT'10 (resp., WMT'11, WMT'12) test set and the performance is then evaluated on WMT'11 and WMT'12 (resp., WMT'10 and WMT'12 or WMT'10 and WMT'11) test set. Although, because of heterogeneous nature of these two particular test sets the improvements are not as good as the potential observed in Figure 3, the method always improves over the mean MERT performance and often exceeds the maximum values in the interval of MERT results.

Table 2 also includes, as a baseline, the BLEU scores obtained with a large-margin linear reranker,



(a) Basic conf., WMT'10, scale-features

(b) Basic conf., WMT'10, scale- & rank-features

(c) Ext. conf., WMT'10, scale-features

(d) Ext. conf., WMT'10, scale- & rank-features

(e) Ext. conf., WMT'11, scale-features

(f) Ext. conf., WMT'11, scale- & rank-features

(g) Ext. conf., WMT'12, scale-features

(h) Ext. conf., WMT'12, scale- & rank-features

Figure 3: RankBoost performance with piece-wise linear weak learners and uniform weighting as a function of number of weak learners. Left plots are obtained with only scale-features, the right ones with additional rank-features included. Each plot contains 3 curves: averaged BLEU over rerankings of 8 runs tuned with MERT, with the linear score included as a feature (`mean`), BLEU of a reranked system without linear score included (`no scores`) and BLEU with all 8 linear scores included as separate features (`all scores`). Mean baseline MERT score is shown with its 1σ interval.

SVMrank (Joachims, 2006), trained on the same set of transformed features.[7] On these experiments, the gain achieved on this feature set is higher than for linear-based reranking on untransformed features (Hopkins and May, 2011). The performances of SVMrank and RankBoost are only marginally different and, in both cases, the gains remain modest compared to oracle performance. As a consequence, these results do not allow one to decide in favor of linear or non-linear modeling under the particular tested conditions (linear-based pruning, no integration with decoder and heterogeneous data).

To check whether the test data's heterogeneity is one of the causes for less than optimal results in Table 2, we randomly split WMT'10 test set into 10 folds of equal-size validation and test subsets. For each fold, parameter T found on the validation subset was used to run the reranker on the corresponding test subset. We found that in almost all cases the peaks on the validation and test subsets perfectly coincide and relative improvements for every split are similar (Table 3). This confirms that test sets' heterogeneity can noteworthy decrease performance, and homogeneous dataset can consistently benefit from the non-linear reranking. It is important, though, to be sure that the BLEU differences are statistically significant: the probability that the observed difference could be obtained by chance should be less than a commonly used threshold of 0.95. We used the Approximate Randomization (AR) method (Noreen, 1989), that is known to have less Type I errors (falsely claiming a significant difference) in SMT applications (Riezler and Maxwell, 2005) than another popular technique of bootstrapping (Koehn, 2004). The AR method repeatedly randomly exchanges hypotheses from baseline and tested system's translation outputs and calculates BLEU score of the shuffled sets. The ratio of cases when the difference of BLEU score exceeds the observed difference on the original outputs is called p-value; if it is larger than $1 - 0.95 = 0.05$, the difference is claimed to be not significant. For the 8 independent runs of MERT[8] and 10 random splits for each of them, in 86% of cases the difference is significant (Table 3). Most insignificant differences were detected for the case of a strong MERT baseline, which was found, earlier in this section, hard to improve upon.

5.4 Weak Learner Analysis

As the piece-wise linear functions include linear functions as a special class, it is possible that RankBoost still selects linear models over the interval $[0, 1]$ of scale-features. An analysis of the models learned shows that the learning process has roughly 3 phases: (1) during $\sim$ 10 iterations (on average) the most selected feature is the final linear `score` feature, promoting the same hypotheses as MERT by effectively reusing the same linear model; (2) between $\sim$ 10-th and $\sim$ 50-th iterations, other features are selected, but still in their linear form over the interval $[0, 1]$, resulting in a linear scoring function, with a better test BLEU than the one found by MERT; (3) above 50 iterations, piece-wise linear models start to appear in the sum (1). As reranking is based on n-best lists (formed with a linear model) it is not surprising that RankBoost sticks to the already optimized score during the first phase, starting during the second stage to take advantage of the ordering information in pair-wise loss and finally resorting to non-linear functions to further improve quality.

As can be seen in Figure 3, the phase when the model gives maximum test performance depends on a particular test set. In our case for WMT'10 it was mostly thanks to the "ranking" second phase, while for WMT'12 it is the use of non-linear functions that helps achieve gains over baselines.

Inclusion or exclusion of the linear `score` feature was found to have large effect. To separate its influence from the rest of features we performed experiments with this feature excluded (`no scores`) and with all such `score` features from all 8 runs included (`all scores`). Not being able to profit from the `score` features, it often takes longer for the former variant (`no scores`) to reach a maximum and vice versa for the configuration with `all scores` (except in the basic configuration).

6 Discussion and Conclusion

We have presented a non-linear approach to reranking n-best lists of phrase-based SMT system. In our experiments the approach was shown to potentially

[7] The regularization parameter C has been set to 1. SVMrank on original features gives considerably lower performance.

[8] The same data used to plot Figure 2.

test \ valid		WMT'10	WMT'11	WMT'12	MERT mean	MERT interval	SVMrank	300-best oracle
	mean	-	29.68±0.07	29.58±0.09			29.74±0.03	
WMT'10	all scores	-	29.66	29.55	29.38±0.09	[29.26,29.54]	29.68	39.72
	no scores	-	29.58	29.54			29.70	
	mean	30.42±0.07	-	30.41±0.05			30.50±0.05	
WMT'11	all scores	30.55	-	30.46	30.16±0.11	[29.97,30.34]	30.48	41.11
	no scores	30.26	-	30.35			30.38	
	mean	30.50±0.08	30.52±0.06	-			30.64±0.02	
WMT'12	all scores	30.59	30.62	-	30.38±0.12	[30.19,30.62]	30.70	40.64
	no scores	30.36	30.42	-			30.55	

Table 2: BLEU scores for the extended system, scale-feature set and three test sets.

MERT	BLEU											
run #	baseline	reranked	Δ	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
1	29.43	29.74	0.31	$0.25_{p=0.02}$	$0.32_{p=0.00}$	$0.29_{p=0.01}$	$0.38_{p=0.00}$	$0.30_{p=0.01}$	$0.24_{p=0.02}$	$0.27_{p=0.02}$	$0.25_{p=0.01}$	$0.34_{p=0.00}$	$0.38_{p=0.00}$
2	29.54	29.60	0.06	$0.00_{p=0.90}$	$0.02_{p=0.81}$	$0.03_{p=0.74}$	$0.03_{p=0.76}$	$-0.02_{p=0.77}$	$0.04_{p=0.68}$	$-0.08_{p=0.42}$	$-0.01_{p=0.86}$	$0.00_{p=0.00}$	$0.09_{p=0.42}$
3	29.26	29.78	0.52	$0.45_{p=0.00}$	$0.49_{p=0.00}$	$0.47_{p=0.00}$	$0.32_{p=0.00}$	$0.39_{p=0.00}$	$0.48_{p=0.00}$	$0.50_{p=0.00}$	$0.47_{p=0.00}$	$0.48_{p=0.00}$	$0.60_{p=0.00}$
4	29.32	29.75	0.43	$0.35_{p=0.00}$	$0.37_{p=0.00}$	$0.40_{p=0.00}$	$0.41_{p=0.00}$	$0.41_{p=0.00}$	$0.43_{p=0.00}$	$0.39_{p=0.00}$	$0.41_{p=0.00}$	$0.33_{p=0.00}$	$0.39_{p=0.00}$
5	29.26	29.65	0.39	$0.33_{p=0.00}$	$0.39_{p=0.00}$	$0.36_{p=0.00}$	$0.28_{p=0.01}$	$0.32_{p=0.00}$	$0.29_{p=0.00}$	$0.36_{p=0.00}$	$0.46_{p=0.00}$	$0.36_{p=0.00}$	$0.40_{p=0.00}$
6	29.33	29.77	0.44	$0.39_{p=0.00}$	$0.38_{p=0.00}$	$0.46_{p=0.00}$	$0.30_{p=0.00}$	$0.32_{p=0.00}$	$0.40_{p=0.00}$	$0.43_{p=0.00}$	$0.40_{p=0.00}$	$0.38_{p=0.00}$	$0.46_{p=0.00}$
7	29.40	29.77	0.37	$0.34_{p=0.00}$	$0.23_{p=0.03}$	$0.30_{p=0.00}$	$0.13_{p=0.18}$	$0.26_{p=0.02}$	$0.30_{p=0.00}$	$0.31_{p=0.01}$	$0.20_{p=0.02}$	$0.33_{p=0.00}$	$0.36_{p=0.00}$
8	29.46	29.71	0.25	$0.21_{p=0.02}$	$0.28_{p=0.00}$	$0.26_{p=0.00}$	$0.13_{p=0.14}$	$0.20_{p=0.03}$	$0.18_{p=0.06}$	$0.18_{p=0.04}$	$0.22_{p=0.02}$	$0.29_{p=0.00}$	$0.27_{p=0.00}$

Table 3: Reranking results on random sub-datasets of WMT'10. The first column identifies MERT's run; second, third and forth contain, resp., baseline MERT performance, maximum achievable performance after reranking (on the full WMT'10) and difference between these numbers. The rest of columns contain differences between reranked results and baseline results on 10 random test halves of the full test WMT'10 test set. Reranked test results were obtained for T found on the validation half of the WMT'10 set. Confidence p-values are given for each difference in subscript.

boost performance by 0.4 BLEU-points (for optimal values of algorithms' hyper parameter T) on a modest size n-best list. In practice, however, mismatches between dev and test data diminish gains when T is optimized and tested on heterogeneous sets.

It should be emphasized that restricting non-linear reranking to n-best lists obtained by pruning with a linear function and using non-linear scoring functions defined on additive features is forced by the complexity of tighter integration of non-standard scoring functions into current decoders. It does probably not do full justice to the power of non-linear scoring function modeling as it does not interfere with pruning and scoring partial hypotheses; a stage during which many excellent translations are actually lost.

Consequently, the main conclusion of this work is the following: under the tested conditions that bypass tight integration with decoder, with pruning still based on linear scoring functions and with few standard features, we cannot claim that non-linearity makes a significant step towards reaching oracle performance. Non-linearity alone appears not to be sufficient, and should be exploited earlier in the search space construction and/or with an increased number of features. Despite these limitations, the gains obtained in our experiments can be attributed to the appropriateness of the ranking loss and to the flexibility of non-linear modeling, and surpass the gains demonstrated by solely resorting to ranking losses, without changing the scoring function family, and using original features.

Several works hint, that for linear models MERT on n-best lists already achieves almost optimal performance (Duh and Kirchhoff, 2008; Galley and Quirk, 2011), so in future work we plan to pursue extensions of this work to non-linear lattice rescoring. The main problem of the straight-forward application of the presented approach to lattice rescoring is the impossibility to combine general non-linear functions with dynamic programming, as this will lead to early discarding of hypotheses that could eventually receive greater scores than their retained competitors. This problem can be partially alleviated by restricting the weak function family used by RankBoost to the monotone functions (Freund et al.,

2003). Another step in this direction could be to model scoring functions that are non-linear with respect to features on arcs. This might avoid the problem with the applicability of approximate dynamic programming to pruning and search for the best hypotheses.

Acknowledgments

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

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proc. of the Conf. of the North American Chapter of the ACL : HLT*, NAACL '09, pages 218–226.

Kevin Duh and Katrin Kirchhoff. 2008. Beyond log-linear models: Boosted minimum error rate training for nbest re-ranking. In *Proc. of ACL, Short Papers*.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proc. of EMNLP*, pages 38–49.

Barry Haddow, Abhishek Arun, and Philipp Koehn. 2011. SampleRank training for phrase-based machine translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 261–271.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proc. of EMNLP*, pages 1352–1362.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proc. Int. Conf. on Knowledge Discovery and Data mining*, KDD '06, pages 217–226.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

Shankar Kumar, 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 ACL and the Int. Conf. on NLP of the AFNLP*, pages 163–171.

Antonio L. Lagarda and Francisco Casacuberta. 2008. Applying boosting to statistical machine translation. In *Proc of European Assoc. for Machine Translation*, pages 88–96.

Thomas Lavergne, Alexandre Allauzen, François Yvon, and Josep Maria Crego. 2011. From n-gram-based to CRF-based translation models. In *Proc. of the Workshop on Statistical Machine Translation (WMT)*, pages 542–553.

Hai Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *ICASSP*, pages 5524–5527.

Hai-Son Le, Alexander Allauzen, and François Yvon. 2012a. Continuous space translation models with neural networks. In *Conf. of the North American Chapter of the ACL: HLT*.

Hai-Son Le, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon. 2012b. LIMSI @ WMT12. In *Proc. of the Workshop on Statistical Machine Translation (WMT)*, pages 330–337, Montréal, Canada.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. of the Int. Conf. on Comp. Linguistics*, Geneva, Switzerland.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypothesis*. John Wiley & Sons.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *of the ACL*, pages 295–302.

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

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

Artem Sokolov, Guillaume Wisniewski, and François Yvon. 2012. Computing lattice BLEU oracle scores for machine translation. In *Proc. of the European ACL*, pages 120–129, Avignon, France.

Marco Turchi, Tiji De Bie, Cyril Goutte, and Nello Cristianini. 2012. Learning to Translate: a statistical and computational analysis. *Advances in Artificial Intelligence*, 2012.

Guillaume Wisniewski, Alexandre Allauzen, and François Yvon. 2010. Assessing phrase-based translation models with oracle decoding. In *Proc. of EMNLP*, pages 933–943.

Tong Xiao, Jingbo Zhu, Muhua Zhu, and Huizhen Wang. 2010. Boosting-based system combination for machine translation. In *Proc. of the ACL*, pages 739–748.