Top-Rank Enhanced Listwise Optimization for Statistical Machine Translation

Huadong Chen,[†] **Shujian Huang,**[†*] **David Chiang,**[‡] **Xinyu Dai,**[†] **Jiajun Chen**[†]
[†]State Key Laboratory for Novel Software Technology, Nanjing University
{chenhd,huangsj,daixinyu,chenjj}@nlp.nju.edu.cn
[‡]Department of Computer Science and Engineering, University of Notre Dame
dchiang@nd.edu

Abstract

Pairwise ranking methods are the basis of many widely used discriminative training approaches for structure prediction problems in natural language processing (NLP). Decomposing the problem of ranking hypotheses into pairwise comparisons enables simple and efficient solutions. However, neglecting the global ordering of the hypothesis list may hinder learning. We propose a listwise learning framework for structure prediction problems such as machine translation. Our framework directly models the entire translation list's ordering to learn parameters which may better fit the given listwise samples. Furthermore, we propose *top-rank enhanced* loss functions, which are more sensitive to ranking errors at higher positions. Experiments on a large-scale Chinese-English translation task show that both our listwise learning framework and top-rank enhanced listwise losses lead to significant improvements in translation quality.

1 Introduction

Discriminative training methods for structured prediction in natural language processing (NLP) aim to estimate the parameters of a model that assigns a score to each hypothesis in the (possibly very large) search space. For example, in statistical machine translation (SMT), the model assigns a score to each possible translation, and in syntactic parsing, the function assigns a score to each possible syntactic tree. Ideally, the model should assign scores that rank hypotheses according to their true quality. In this paper, we consider the problem of discriminative training for SMT.

Traditional SMT systems use log-linear models with only about a dozen features, such as translation probabilities and language model probabilities (Yamada and Knight, 2001; Koehn et al., 2003; Chiang, 2005; Liu et al., 2006). These models can be tuned by minimum error rate training (MERT) (Och, 2003), which directly optimizes BLEU using coordinate ascent combined with a global line search.

To enable training of modern SMT systems, which can have thousands of features or more, many research efforts have been made towards scalable discriminative training methods (Chiang et al., 2008; Hopkins and May, 2011; Bazrafshan et al., 2012). Most of these methods either define loss functions that push the model to correctly compare pairs of hypotheses, or use approximate optimization methods that effectively do the same. For practical reasons, only a subset of the pairs are considered; these pairs are selected by either sampling (Hopkins and May, 2011) or heuristic methods (Watanabe et al., 2007; Chiang et al., 2008).

But this pairwise approach neglects the global ordering of the list of hypotheses, which may lead to problems trying to learn good parameter values. Inspired by research in information retrieval (IR) (Cao et al., 2007; Xia et al., 2008), we propose to directly model the ordering of the whole translation list, instead of decomposing it into translation pairs.

Previous research has tried to integrate listwise methods into SMT, but almost all of them focus on the reranking task, which aims to rescore the fixed translation lists generated by a baseline system. They try to either use listwise approaches to training the reranking model (Li et al., 2013; Niehues et al., 2015) or replace the pointwise ranking function, i.e. the log-linear model, with a listwise ranking function by introducing listwise features (Zhang et al., 2016). In this paper, we

*Corresponding author.

Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 90–99,
Vancouver, Canada, August 3 - August 4, 2017.

focus on listwise approaches that can learn better discriminative models for SMT. We present a listwise learning framework for tuning translation systems that uses two listwise ranking objectives originally developed for IR, ListNet (Cao et al., 2007) and ListMLE (Xia et al., 2008). But unlike standard IR problems, structured prediction problems usually have a huge search space, and at each training iteration, the list of search results can vary. The usual strategy is to form the union of all lists of search results, but this can lead to a "patchy" list that doesn't represent the full search space well. The listwise approaches always based on the permutation probability distribution over the list. Modeling the distribution over a "patchy" list, whose elements were generated by different parameters will affect listwise approaches' performance. To address this issue, we design an *instance-aggregating* method: Instead of treating the data as a fixed-size set of lists that each grow over time as new translations are added at each iteration, we treat the data as a growing set of lists; each time a sentence is translated, the k-best list of translations is added as a new list.

We also extend standard listwise training by considering the importance of different instances in the list. Based on the intuition that instances at the top may be more important for ranking, we propose *top-rank enhanced* loss functions, which incorporate a position-dependent cost that penalizes errors occurring at the top of the list more strongly.

We conduct large-scale Chinese-to-English translation experiments showing that our top-rank enhanced listwise learning methods significantly outperform other tuning methods with high dimensional feature sets. Additionally, even with a small basic feature set, our methods still obtain better results than MERT.

2 Background

2.1 Log-linear models

In this paper, we assume a log-linear model, which defines a scoring function on target translation hypotheses $\mathbf{e}$, given a source sentence $\mathbf{f}$:

$$Pr(\mathbf{e} \mid \mathbf{f}) = \frac{\exp s(\mathbf{e}, \mathbf{f})}{\sum_{\mathbf{e}'} \exp s(\mathbf{e}', \mathbf{f})} \quad (1)$$

$$s(\mathbf{e}, \mathbf{f}) = \mathbf{w} \cdot \mathbf{h}(\mathbf{e} \mid \mathbf{f}) \quad (2)$$

where $\mathbf{h}(\mathbf{e} \mid \mathbf{f})$ is the feature vector and $\mathbf{w}$ is the feature weight vector.

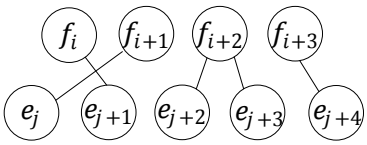

Figure 1: An example of word-phrase features for a phrase translation. The f_i and e_j represent the i-th in the source phrase and j-th word in the target phrase, respectively.

The process of training a SMT system includes both learning the sub-models, which are included in the feature vector $\mathbf{h}$, and learning the weight vector $\mathbf{w}$.

Then the decoding of SMT systems can be formulated as a search for the translation $\hat{\mathbf{e}}$ with the highest model score:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e} \in \mathcal{E}} s(\mathbf{e}, \mathbf{f}) \quad (3)$$

where $\mathcal{E}$ is the set of all reachable hypotheses.

2.2 SMT Features

In this paper, we use a hierarchical phrase based translation system (Chiang, 2005). For convenient comparison, we divide features of SMT into the following three sets.

Basic Features: The basic features are those commonly used in hierarchical phrase based translation systems, including a language model, four translation model features, word, phrase and rule penalties, and penalties for unknown words, the glue rule and null translations.

Extended Features: Inspired by Chen et al. (2013), we manually group the parallel training data into 15 sets, according to their genre and origin. The translation models trained on each set are used as separate features. We also add an indicator feature for each individual training set to mark where the translation rule comes from. The extended features provide additional 60 translation model features and 16 indicator features, which is too many to be tuned with MERT.

Sparse Features: We use word-phrase pair features as our sparse features, which reflect the word-phrase correspondence in a hierarchical phrase (Watanabe et al., 2007). Figure 1 illustrates an example of word-phrase pair features for a phrase translation pair $f_i, ..., f_{i+3}$ and $e_j, ..., e_{j+4}$. Word-phrase pair features (f_i, e_{j+1}), (f_{i+1}, e_j), $(f_{i+2}, e_{j+2}e_{j+3})$, (f_{i+3}, e_{j+4}) will be fired for the translation rule with the given word alignment. In

practice, these feature only fire when all the source and target words in the feature are both in the top 100 most frequent words.

2.3 Tuning via Pairwise Ranking

The beam search strategy for SMT decoding process makes it convenient to get a k-best translation list for each source sentence. Given a set of source sentences and their corresponding translation lists, the tuning problem could be regarded as a ranking task. Many recently proposed SMT tuning methods are based on the pairwise ranking framework (Chiang et al., 2008; Hopkins and May, 2011; Bazrafshan et al., 2012).

Pairwise ranking optimization (PRO) (Hopkins and May, 2011) is a commonly used tuning method. The idea of PRO is to sample pairs $(\mathbf{e}, \mathbf{e}')$ from the k-best list, and train a linear binary classifier to predict whether $\mathit{eval}(\mathbf{e}) > \mathit{eval}(\mathbf{e}')$ or $\mathit{eval}(\mathbf{e}) < \mathit{eval}(\mathbf{e}')$, where $\mathit{eval}(\cdot)$ is an extrinsic metric like BLEU. In this paper, we use sentence-level BLEU with add-one smoothing (Lin and Och, 2004).

The method gets a comparable BLEU score to MERT and MIRA (Chiang et al., 2008), and scales well on large feature sets. Other pairwise ranking methods employ similar procedures.

3 Listwise Learning Framework

Although ranking methods have shown their effectiveness in tuning for SMT systems (Hopkins and May, 2011; Watanabe, 2012; Dreyer and Dong, 2015), most proposed ranking approaches view tuning as pairwise ranking. These approaches decompose the ranking of the hypothesis list into pairs, which might limit the training method's ability to learn better parameters. To preserve the ranking information, we first formulate training as an instance of the listwise ranking problem. Then we propose a learning method based on the iterative learning framework of SMT tuning and further investigate the top-rank enhanced losses.

3.1 Training Objectives

3.1.1 The Permutation Probability Model

In order to directly model the translation list, we first introduce a probabilistic model proposed by Guiver and Snelson (2009). A ranking of a list of k translations can be thought of as a function π from $[1, k]$ to translations, where each $\pi(t)$ is the t-th translation candidate in the ranking. A scoring function z (which could be either the model score, s, or the BLEU score, *eval*) induces a probability distribution over rankings:

$$P_z(\pi) = \prod_{j=1}^{k} \frac{\exp z(\pi(j))}{\sum_{t=j}^{k} \exp z(\pi(t))}. \quad (4)$$

3.1.2 Loss Functions

Based on the probabilistic model above, the loss function can be defined as the difference between the distribution over the ranking according to $eval(\cdot)$ and $s(\cdot)$. Thus, we introduce the following two standard listwise losses.

ListNet: The ListNet loss is the cross entropy between the distributions calculated from $eval(\cdot)$ and $s(\cdot)$, respectively, over all permutations.

Due to the exponential number of permutations, Cao et al. (2007) propose a top-one loss instead. Given the function $eval(\cdot)$ and $s(\cdot)$, the top-one loss is defined as:

$$L_{\text{Net-T}} = -\sum_{j=1}^{k} P'_{eval}(\mathbf{e}_j) \log P'_s(\mathbf{e}_j)$$

$$P'_z(\mathbf{e}_j) = \frac{\exp z(\mathbf{e}_j)}{\sum_{i=1}^{k} \exp z(\mathbf{e}_i)}$$

where $\mathbf{e}_j$ is the j-th element in the k-best list, and $P'_z(\mathbf{e}_j)$ is the probability that translation $\mathbf{e}_j$ is ranked at the top by the function z.

ListMLE: The ListMLE loss is the negative log-likelihood of the permutation probability of the correct ranking π_{eval}, calculated according to $s(\cdot)$ (Xia et al., 2008):

$$\begin{aligned} L_{\text{MLE}} &= -\log P_s(\pi_{eval}) \\ &= -\sum_{j=1}^{k} \log \frac{\exp s(\pi_{eval}(j))}{\sum_{t=j}^{k} \exp s(\pi_{eval}(t))}. \end{aligned} \quad (5)$$

The training objective, which we want to minimize, is simply the total loss over all the lists in the tuning set.

3.2 Training with Instance Aggregating

Because there can be exponentially many possible translations of a sentence, it's only feasible to rank the k best translations rather than all of them; because the feature weights change at each iteration, we have a different k-best list to rank at each iteration. This is different from standard ranking problems in which the training instances stay the same each iteration.

Algorithm 1 MERT-like tuning algorithm

Require: Training sentences $\{\mathbf{f}\}$, maximum number of iterations I, randomly initialized model parameters $\mathbf{w}^0$.

for $i = 0$ to I **do**
  for source sentences $\mathbf{f}$ **do**
    Decode $\mathbf{f}$: $\mathcal{E}_{\mathbf{f}}^i = \text{KbestDecoder}(\mathbf{f}, \mathbf{w}^i)$
    $T \leftarrow T \cup \{\mathcal{E}_{\mathbf{f}}^i\}$
  end for
  Training: $\mathbf{w}^{i+1} = \text{Optimization}(T, \mathbf{w}^i)$
end for

Many previous tuning methods address this problem by merging the k-best list at the current iteration with the k-best lists at all previous iterations into a single list (Hopkins and May, 2011). We call this *k-best merging*. More formally, if $\mathcal{E}_{\mathbf{f}}^i$ is the k-best list of source sentence $\mathbf{f}$ at iteration i, then at each iteration, the model is trained on the set of lists:

$$\mathcal{E}_{\mathbf{f}} = \bigcup_{j=0}^{i} \mathcal{E}_{\mathbf{f}}^j$$
$$T = \{\mathcal{E}_{\mathbf{f}} \mid \forall \mathbf{f}\}$$

For each source sentence $\mathbf{f}$, T has only one training sample, which is a better and better approximation to the full hypothesis set of $\mathbf{f}$ as more iterations pass.

Unlike previous tuning methods, our tuning method focuses on the distribution over permutation of the whole list. Moreover, unlike with listwise optimization methods used in IR, the k-best list produced for a source sentence at one iteration can differ dramatically from the k-best list produced at the next iteration. Merging k-best lists across iterations, each of which represents only a tiny fraction of the full search space, will lead to a "patchy" list that may hurt the learning performance of the listwise optimization algorithms.

To address this challenge, we propose *instance aggregating*: instead of merging k-best lists across different iterations, we view the translation lists from different iterations as individual training instances:

$$T = \{\mathcal{E}_{\mathbf{f}}^j \mid \forall \mathbf{f}, 0 \leq j \leq i\}.$$

With this method, each source sentence $\mathbf{f}$ has i training instances at the i-th training iteration. In this way, we avoid "patchy" lists and obtain a better set of instances for tuning.

Algorithm 2 Listwise Optimization Algorithm

Require: Training instances T, model parameters $\mathbf{w}$, maximum number of epochs J, batch size b, number of batches B

for $j = 0$ to J **do**
  for $i = 0$ to B **do**
    Sample a minibatch of b lists from T without replacement
    Calculate loss function L
    Calculate gradient ∇L
    $\mathbf{w}_{t+1} = \text{AdaDelta}(\mathbf{w}_t, L, \Delta\mathbf{w})$
  end for
end for
$\mathbf{w} = \text{BestBLEU}([\mathcal{E}]_1^m)$

The above instance aggregating method can be used in a MERT-like iterative tuning algorithm as shown in Algorithm 1, which can be easily integrated into current open source systems. The two standard listwise losses can be easily optimized using gradient-based methods (Algorithm 2); both losses are convex, so convergence to a global optimum is guaranteed. The gradients of ListNET and ListMLE with respect to the parameters $\mathbf{w}$ for a single sentence are:

$$\frac{\partial L_{\text{Net-T}}}{\partial \mathbf{w}} = -\sum_{j=1}^{k} P'_{eval}(\mathbf{e}_j) \left(\frac{\partial s(\mathbf{e}_j)}{\partial \mathbf{w}} - \sum_{j=1}^{k} \frac{\exp s(\mathbf{e}_j)}{\sum_{j'=1}^{k} \exp s(\mathbf{e}_{j'})} \frac{\partial s(\mathbf{e}_j)}{\partial \mathbf{w}} \right)$$

$$\frac{\partial L_{\text{MLE}}}{\partial \mathbf{w}} = -\sum_{j=1}^{k} \left(\frac{\partial s(\pi_{eval}(j))}{\partial \mathbf{w}} - \sum_{t=j}^{k} \frac{\exp s(\pi_{eval}(t))}{\sum_{t'=j}^{k} \exp s(\pi_{eval}(t'))} \frac{\partial s(\pi_{eval}(t))}{\partial \mathbf{w}} \right)$$

For optimization, we use a mini-batch stochastic gradient descent (SGD) algorithm together with AdaDelta (Zeiler, 2012) algorithm to adaptively set the learning rate.

4 Top-Rank Enhanced Losses

In evaluating an SMT system, one naturally cares much more about the top-ranked results than the lower-ranked results. Therefore, we think that getting the ranking right at the top of a list is more relevant for tuning. Therefore, we should pay more

attention to the top-ranked translations instead of forcing the model to rank the entire list correctly.

Position-dependent Attention: To do this, we assign a higher cost to ranking errors that occur at the top and a lower cost to errors at the bottom. To make the cost sensitive to position, we define it as:

$$c(j) = \frac{k - j + 1}{\sum_{t=1}^{k} t} \tag{6}$$

where j is the position in the ranking and k is the size of the list.

Based on this cost function, we propose simple top-rank enhanced listwise losses as extensions of both the ListNet loss and the ListMLE loss. The loss functions are defined as follows:

$$L_{\text{MLE-TE}} = -\sum_{j=1}^{k} c(j) \log \frac{\exp s(\pi_{eval}(j))}{\sum_{t=j}^{k} \exp s(\pi_{eval}(t))}$$

$$L_{\text{Net-TE}} = -\sum_{\forall \pi \in \Omega_k} P''_{eval}(\pi) \sum_{j=1}^{k} c(j) \log q_j(\pi)$$

$$q_j(\pi) = \frac{\exp z(\pi(j))}{\sum_{t=j}^{k} \exp z(\pi(t))}.$$

Along similar lines, Xia et al. (2008) also proposed a top-n ranking method, which assumes that only the correct ranking of top-n hypotheses is useful. Compared to our top-rank enhanced losses, it may be too harsh to discard information about the rest of the ordering altogether; our method retains the whole ordering but weights it by position.

5 Experiments and Results

5.1 Data and Preparation

We conduct experiments on a large scale Chinese-English translation task. The parallel data comes from LDC corpora[1], which consists of 8.2 million of sentence pairs. Monolingual data includes Xinhua portion of Gigaword corpus. We use NIST MT03 evaluation test data as the development set, MT02, MT04 and MT05 as the test set.

The Chinese side of the corpora is word segmented using ICTCLAS[2]. Word alignments of the parallel data are learned by running GIZA++ (Och and Ney, 2003) in both directions and refined under the "grow-diag-final-and" method. We train a 5-gram language model on the monolingual data with Modified Kneser-Ney smoothing(Chen and Goodman, 1999). Throughout the experiments, our translation system is an in-house implementation of the hierarchical phrase-based translation system (Chiang, 2005). The translation quality is evaluated by 4-gram case-insensitive BLEU (Papineni et al., 2002). Statistical significance testing between systems is conducted by bootstrap resampling implemented by Clark et al. (2011).

Data	Usage	Sents.
LDC	TM train	8,260,093
Gigaword	LM train	14,684,074
MT03	train	919
MT02	test	878
MT04	test	1,788
MT05	test	1,082

Table 1: Experimental data and statistics.

5.2 Tuning Settings

We build baselines for extended and sparse feature sets with two different tuning methods. First, we tune with PRO (Hopkins and May, 2011). As reported by Cherry and Foster (2012), it's hard to find the setting that performs well in general. We use MegaM version (Daumé III, 2004) with 30 iterations for basic feature set and 100 iterations for extended and sparse feature sets. Second, we run the k-best batch MIRA (KB-MIRA) which shows comparable results with online version of MIRA (Cherry and Foster, 2012; Green et al., 2013). In our experiments, we run KB-MIRA with standard settings in Moses[3]. For the basic feature set, the baseline is tuned with MERT (Och, 2003).

For all our listwise tuning methods, we set batch size to 10. In our experiments, we can't find a epoch size perform well in general, so we set epoch size to 100 for ListMLE with basic features, 200 for ListMLE with extended and sparse features, and 300 for ListNet. These values are set to achieve the best performance on the development set.

We set beam size to 20 throughout our experiments unless otherwise noted. Following Clark et al. (2011), we run the same training procedure 3 times and present the average results for stability. All tuning methods are executed for 40 iter-

[1]The corpora include LDC2002E18, LDC2003E14, LDC2004E12, LDC2004T08, LDC2005T10 and LDC2007T09

[2]http://ictclas.nlpir.org/

[3]http://www.statmt.org/moses/

Methods	MT02	MT04	MT05	AVG
Net$_m$	40.36	38.30	37.93	38.86(+0.00)
ListNet	40.75	38.69	38.31	39.25(+0.39)
MLE$_m$	39.82	37.88	37.65	38.45(+0.00)
ListMLE	40.40	38.21	38.04	38.88(+0.43)

Table 2: The comparison of instances aggregating and k-best merging on the extended feature set.(Net$_m$ and MLE$_m$ denote ListNet and ListMLE with k-best merging respectively.)

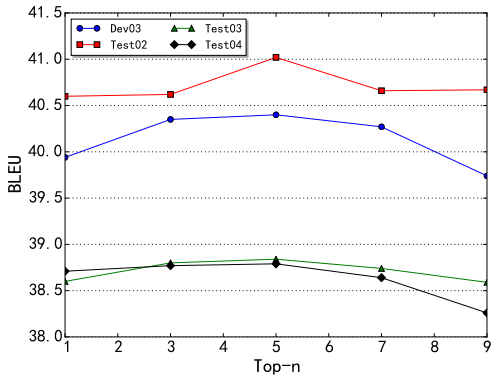

Figure 2: Effect of different n for Top-n ListMLE. We investigate the effect on the extended feature set.

ations of the outer loop and returned the weights that achieve the best development BLEU scores. For all tuning methods on sparse feature set, we use the weight vector tuned by PRO on the extended feature set as initial weights.

5.3 Experiments of Listwise Learning Framework

We first investigate the effectiveness of our instance aggregating training procedure. The results are presented in Table 2. The table compare training with instance aggregating and k-best merging. As the result suggested, with the instance aggregating method, the performance improves on both listwise tuning approaches. For the rest of this paper, we use the instance aggregating as standard setting for listwise tuning approaches.

To verify the performance of our proposed listwise learning framework, we first compare systems with standard listwise losses to the baseline systems. The first four rows in Table 3 show the results. ListNet can outperform PRO by 0.55 BLEU score and 0.26 BLEU score on extended feature set and sparse feature set, respectively. Its main reason is that our listwise methods can obtain structured order information when we take complete translation list as instance.

We also observe that ListMLE can only get a modest performance compare to ListNet. We think the objective function of standard ListMLE which forces the whole list ranking in a correct order is too hard. ListNet mainly benefits from its top one permutation probability which only concerns the permutation with the best object ranked first.

5.4 Effect of Top-rank Enhanced Losses

To verify our assumption that the correct rank in the top portion of a list is more informative, we conduct this set of experiments. Figure 2 shows the results of top-n ListMLE with different n. Compared to ListMLE in Table 2, we find top-n ListMLE can make significant improvements, which means that the top rank is more important. We can observe an improvement in all test sets when we set n from 1 to 5, but when we further increase n, the results dropped. This situation indicates that the correct ranking at the top of the list is more informative and forcing the model to rank the bottom correctly as important as the top will sacrifice the ability to guide better search.

In Table 3, top-5 ListMLE which only aims to rank the top five translations correctly can outperform the baseline and standard ListMLE. With our position-dependent attention, the top-rank enhanced ListMLE can make further improvement over the baseline system(+1.07 and +0.73 on extended and sparse feature sets, respectively.) and achieves the best performance.

The top-n loss might be too loose as an approximation of the measure of BLEU. Compared to top-n ListMLE, our top-rank enhanced ListMLE can further utilize the different portions of the list by different weights. To verify the claim, we further examined the learning processes of the two losses. For simplicity, the experiment is conducted on a translation list generated by random parameters. The results are shown in Figure 3. We can see that our top-rank enhanced loss almost completely inversely correlates with BLEU after iteration 70. In contrast, after iteration 150, although top-5 loss is still decreasing, BLEU starts to drop.

Due to the high computation cost of ListNet, we only perform the top-rank enhanced ListMLE in this paper. Our preliminary experiments indicate that the performance of ListNet can be further improved with a top-2 loss. We think our top-rank

Method	Extended Features				Sparse Features			
	MT02	MT04	MT05	AVG	MT02	MT04	MT05	AVG
PRO	40.30	38.12	37.69	38.70(+0.00)	40.63	38.46	38.24	39.11(+0.00)
KB-MIRA	40.48	37.71	37.37	38.52(-0.18)	40.67	38.48	38.21	39.12(+0.01)
ListNet	**40.75***	**38.69**$^{+}$	**38.31***	**39.25(+0.55)**	**40.91***	**38.77***	**38.42**	**39.37(+0.26)**
ListMLE	40.40	38.21	38.04	38.88(+0.18)	40.63	38.68	38.24	39.18(+0.07)
ListMLE-T5	41.02*	38.84^{+}	38.79^{+}	39.55(+0.85)	41.12*	38.91*	38.89*	39.64(+0.53)
ListMLE-TE	**41.15**$^{+}$	**39.01**$^{+}$	**39.16**$^{+}$	**39.77(+1.07)**	**41.25**$^{+}$	**39.00**$^{+}$	**39.27**$^{+}$	**39.84(+0.73)**

Table 3: BLEU4 in percentage for comparing of baseline systems and systems with listwise losses. $^{+}$, * marks results that are significant better than the baseline system with $p < 0.01$ and $p < 0.05$. (ListMLE-T5 and ListMLE-TE refer to top-5 LisMLE and our top-rank enhanced ListMLE, respectively.)

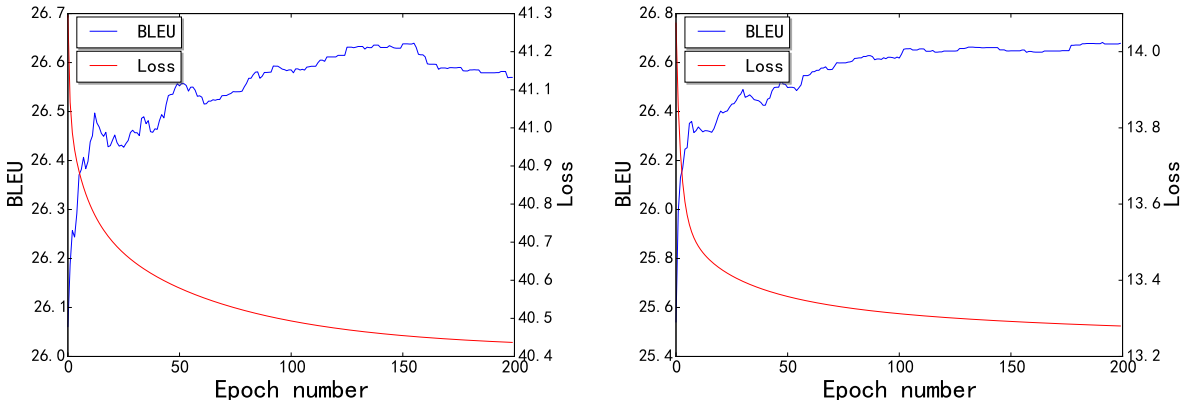

Figure 3: Listwise losses v.s. BLEU in (a) top-5 ListMLE and (b) top-rank enhanced ListMLE

Methods	MT02	MT04	MT05	AVG
PRO	40.90	38.84	38.64	39.64(+0.00)
KB-MIRA	41.09	38.49	38.62	39.40(-0.06)
ListNet	41.49^{+}	39.25*	39.17*	39.97(+0.51)
ListMLE-T5	41.26*	39.63^{+}	39.32*	40.07(+0.61)
ListMLE-TE	**41.85**$^{+}$	**39.96**$^{+}$	**39.88**$^{+}$	**40.56(+1.10)**

Table 4: Comparison of baselines and listwise approaches with a larger k-best list on extended feature set.

enhanced method is also useful for ListNet, but due to its computational demands it needs to be further investigated.

5.5 Impact of the Size of Candidate Lists

Our listwise tuning methods directly model the order of the translation list, it is clear that the choice of the translation list size k has an impact on our methods. A larger candidate list size may result in the availability of more information during tuning. In order to verify our tuning methods' capability of handling the larger translation list, we increase k from 20 to 100. The comparison results are shown in Table 4. With a larger size k, our tuning methods also perform better than baselines. For ListNet and top-5 ListMLE, we observe that the improvements over baseline is smaller than size 20. This results show that the order information loss caused by directly drop the bottom is aggravated with larger list size. However, our top-rank enhanced method still get a slight better result than size 20 and significant improvement over baseline by 1.1 BLEU score. This indicate that our top-rank enhanced method is more stable and can still effectively exploit the larger size translation list.

5.6 Performance on Basic Feature Set

Since the effectiveness of high dimensional feature set, recent work pays more attention to this scenario. Although previous discriminative tuning methods can effectively handle high dimensional feature set, MERT is still the dominant tuning method for basic features. Here, we investigate our top-rank enhanced tuning methods' capability of handling basic feature set. Table 5 summarizes the comparison results. Firstly, we observe that ListNet and ListMLE can perform comparable with MERT. With our top-ranked enhanced method, we can get a better performance than

Methods	MT02	MT04	MT05	AVG
MERT	37.72	37.13	36.77	37.21(+0.00)
PRO	37.85	37.21	36.68	37.24(+0.03)
KB-MIRA	37.97	37.28	36.58	37.28(+0.07)
ListNet	37.71	37.47*	36.78	37.32(+0.11)
ListMLE	37.54	**37.54**	36.65	37.24(+0.03)
ListMLE-T5	37.90	37.32	36.84	37.35(+0.14)
ListMLE-TE	**38.03**	37.49*	**36.85**	**37.46(+0.25)**

Table 5: Comparison of baseline and liswise approaches on basic feature set.

MERT by 0.25 BLEU score. These results show that our top-ranked enhanced tuning method can learn more informations of translation list even with a basic feature set.

6 Related Work

The ranking problem is well studied in IR community. There are many methods been proposed, including pointwise (Nallapati, 2004), pairwise (Herbrich et al., 1999; Burges et al., 2005) and listwise (Cao et al., 2007; Xia et al., 2008) algorithms. Experiment results show that listwise methods deliver better performance than pointwise and pairwise methods in general (Liu, 2010).

Most NLP researches take ranking as an extra step after searching from its output space (Charniak and Johnson, 2005; Collins and Terry Koo, 2005; Duh, 2008). In SMT research, listwise approaches also have been employed for the reranking tasks. For example, Li et al. (2013) utilized two listwise approaches to rerank the translation outputs and achieved the best segment-level correlation with human judgments. Niehues et al. (2015) employed ListNet to rescore the k-best translations, which significantly outperforms MERT, KB-MIRA and PRO. Zhang et al. (2016) viewed the log-linear model as a pointwise ranking function and shifted it to listwise ranking function by introducing listwise features and outperformed the log-linear model. Compared to these efforts, our method takes a further step by integrating listwise ranking methods into the iterative training.

There are also some researches use ranking methods for tuning to guide better search. In SMT, previous attempts on using ranking as a tuning methods usually perform pairwise comparisons on a subset of translation pairs (Chiang et al., 2008; Hopkins and May, 2011; Watanabe, 2012; Bazrafshan et al., 2012; Guzmán et al., 2015). Dreyer and Dong (2015) even took all translation pairs of the k-best list as training instances, which only obtained a comparable result with PRO and the implementation is more complicate. In this paper, we model the entire list as a whole unit, and propose training objectives that are sensitive to different parts of the list.

7 Conclusion

In this paper, we propose a listwise learning framework for statistical machine translation. In order to adapt listwise approaches, we use an iterative training framework in which instances from different iterations are aggregated into the training set. To emphasize the top order of the list, we further propose top-rank enhanced listwise learning losses. Compared to previous efforts in SMT tuning, our method directly models the order information of the complete translation list. Experiments show our method could lead to significant improvements of translation quality in different feature sets and beam size.

Our current work focuses on the traditional SMT task. For future work, it will be interesting to integrate our methods to modern neural machine translation systems or other structure prediction problems. It may also be interesting to explore more methods on listwise tuning framework, such as investigating different methods to enhance top order of translation list directly w.r.t a given evaluation metric.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Science Foundation of China (No. 61672277, 61300158 and 61472183). Part of Huadong Chen's contribution was made while visiting University of Notre Dame. His visit was supported by the joint PhD program of China Scholarship Council.

References

Marzieh Bazrafshan, Tagyoung Chung, and Daniel Gildea. 2012. Tuning as linear regression. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 543–547. http://aclweb.org/anthology/N12-1062.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient

descent. In *Proceedings of the 22Nd International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '05, pages 89–96. https://doi.org/10.1145/1102351.1102363.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '07, pages 129–136. https://doi.org/10.1145/1273496.1273513.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, pages 173–180. http://aclweb.org/anthology/P05-1022.

Boxing Chen, Roland Kuhn, and George Foster. 2013. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1285–1293. http://aclweb.org/anthology/P13-1126.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13(4):359–394.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 427–436. http://aclweb.org/anthology/N12-1047.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, pages 263–270. http://aclweb.org/anthology/P05-1033.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 224–233. http://aclweb.org/anthology/D08-1024.

H. Jonathan Clark, Chris Dyer, Alon Lavie, and A. Noah Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 176–181. http://aclweb.org/anthology/P11-2031.

Collins and Michael Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics, Volume 31, Number 1, March 2005* http://aclweb.org/anthology/J05-1003.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression.

Markus Dreyer and Yuanzhe Dong. 2015. Apro: All-pairs ranking optimization for mt tuning. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1018–1023. https://doi.org/10.3115/v1/N15-1106.

Kevin Duh. 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, StatMT '08, pages 191–194. http://dl.acm.org/citation.cfm?id=1626394.1626425.

Spence Green, Sida Wang, Daniel Cer, and D. Christopher Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 311–321. http://aclweb.org/anthology/P13-1031.

John Guiver and Edward Snelson. 2009. Bayesian inference for plackett-luce ranking models. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '09, pages 377–384. https://doi.org/10.1145/1553374.1553423.

Francisco Guzmán, Preslav Nakov, and Stephan Vogel. 2015. Analyzing optimization for statistical machine translation: Mert learns verbosity, pro learns length. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 62–72. http://www.aclweb.org/anthology/K15-1007.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*. IET, volume 1, pages 97–102.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1352–1362. http://aclweb.org/anthology/D11-1125.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter*

of the Association for Computational Linguistics. http://aclweb.org/anthology/N03-1017.

Maoxi Li, Aiwen Jiang, and Mingwen Wang. 2013. Listwise approach to learning to rank for automatic evaluation of machine translation. *Proceedings of the XIV Machine Translation Summit* .

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: A method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Tie-Yan Liu. 2010. Learning to rank for information retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '10, pages 904–904. https://doi.org/10.1145/1835449.1835676.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 609–616. http://aclweb.org/anthology/P06-1077.

Ramesh Nallapati. 2004. Discriminative models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '04, pages 64–71. https://doi.org/10.1145/1008992.1009006.

Jan Niehues, Quoc-Khanh DO, Alexandre Allauzen, and Alex Waibel. 2015. Listnet-based mt rescoring. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 248–255. http://aclweb.org/anthology/W15-3030.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, pages 160–167. https://doi.org/http://dx.doi.org/10.3115/1075096.1075117.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.* 29(1):19–51. https://doi.org/http://dl.acm.org/citation.cfm?id=778824.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P02-1040.

Taro Watanabe. 2012. Optimized online rank learning for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 253–262. http://aclweb.org/anthology/N12-1026.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. http://aclweb.org/anthology/D07-1080.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '08, pages 1192–1199. https://doi.org/10.1145/1390156.1390306.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P01-1067.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

M. Zhang, Y. Liu, H. Luan, and M. Sun. 2016. Listwise ranking functions for statistical machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(8):1464–1472. https://doi.org/10.1109/TASLP.2016.2560527.