Tuning as Linear Regression

Marzieh Bazrafshan, **Tagyoung Chung** and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We propose a tuning method for statistical machine translation, based on the pairwise ranking approach. Hopkins and May (2011) presented a method that uses a binary classifier. In this work, we use linear regression and show that our approach is as effective as using a binary classifier and converges faster.

1 Introduction

Since its introduction, the minimum error rate training (MERT) (Och, 2003) method has been the most popular method used for parameter tuning in machine translation. Although MERT has nice properties such as simplicity, effectiveness and speed, it is known to not scale well for systems with large numbers of features. One alternative that has been used for large numbers of features is the Margin Infused Relaxed Algorithm (MIRA) (Chiang et al., 2008). MIRA works well with a large number of features, but the optimization problem is much more complicated than MERT. MIRA also involves some modifications to the decoder itself to produce hypotheses with high scores against gold translations.

Hopkins and May (2011) introduced the method of pairwise ranking optimization (PRO), which casts the problem of tuning as a ranking problem between pairs of translation candidates. The problem is solved by doing a binary classification between "correctly ordered" and "incorrectly ordered" pairs. Hopkins and May (2011) use the maximum entropy classifier MegaM (Daumé III, 2004) to do the binary classification. Their method compares well to the results of MERT, scales better for high dimensional feature spaces, and is simpler than MIRA.

In this paper, we use the same idea for tuning, but, instead of using a classifier, we use linear regression. Linear regression is simpler than maximum entropy based methods. The most complex computation that it needs is a matrix inversion, whereas maximum entropy based classifiers use iterative numerical optimization methods.

We implemented a parameter tuning program with linear regression and compared the results to PRO's results. The results of our experiments are comparable to PRO, and in many cases (also on average) we get a better maximum BLEU score. We also observed that on average, our method reaches the maximum BLEU score in a smaller number of iterations.

The contributions of this paper include: First, we show that linear regression tuning is an effective method for tuning, and it is comparable to tuning with a binary maximum entropy classifier. Second, we show linear regression is faster in terms of the number of iterations it needs to reach the best results.

2 Tuning as Ranking

The parameter tuning problem in machine translation is finding the feature weights of a linear translation model that maximize the scores of the candidate translations measured against reference translations. Hopkins and May (2011) introduce a tuning method based on ranking the candidate translation pairs, where the goal is to learn how to rank pairs of candidate translations using a gold scoring function.

PRO casts the tuning problem as the problem of ranking pairs of sentences. This method iteratively generates lists of "k-best" candidate translations for each sentence, and tunes the weight vector for those candidates. MERT finds the weight vector that maximizes the score for the highest scored candidate translations. In contrast, PRO finds the weight vector which classifies pairs of candidate translations into "correctly ordered" and "incorrectly ordered," based on the gold scoring function. While MERT only considers the highest scored candidate to tune the weights, PRO uses the entire k-best list to learn the ranking between the pairs, which can help prevent overfitting.

Let $g(e)$ be a scoring function that maps each translation candidate e to a number (score) using a set of reference translations. The most commonly used gold scoring function in machine translation is the BLEU score, which is calculated for the entire corpus, rather than for individual sentences. To use BLEU as our gold scoring function, we need to modify it to make it decomposable for single sentences. One way to do this is to use a variation of BLEU called BLEU+1 (Lin and Och, 2004), which is a smoothed version of the BLEU score.

We assume that our machine translation system scores translations by using a scoring function which is a linear combination of the features:

$$h(e) = w^{\mathrm{T}} x(e) \tag{1}$$

where w is the weight vector and x is the feature vector. The goal of tuning as ranking is learning weights such that for every two candidate translations e_1 and e_2, the following inequality holds:

$$g(e_1) > g(e_2) \Leftrightarrow h(e_1) > h(e_2) \tag{2}$$

Using Equation 1, we can rewrite Equation 2:

$$g(e_1) > g(e_2) \Leftrightarrow w^{\mathrm{T}}(x(e_1) - x(e_2)) > 0 \tag{3}$$

This problem can be viewed as a binary classification problem for learning w, where each data point is the difference vector between the feature vectors of a pair of translation candidates, and the target of the point is the sign of the difference between their gold scores (BLEU+1). PRO uses the MegaM classifier to solve this problem. MegaM is a binary maximum entropy classifier which returns the weight vector w as a linear classifier. Using this method, Hopkins and May (2011) tuned the weight vectors for various translation systems. The results were close to MERT's and MIRA's results in terms of BLEU score, and the method was shown to scale well to high dimensional feature spaces.

3 Linear Regression Tuning

In this paper, we use the same idea as PRO for tuning, but instead of using a maximum entropy classifier, we use a simple linear regression to estimate the vector w in Equation 3. We use the least squares method to estimate the linear regression. For a matrix of data points $\mathbf{X}$, and a target vector $\mathbf{g}$, the weight vector can be calculated as:

$$w = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{g} \tag{4}$$

Adding L_2 regularization with parameter λ has the following closed form solution:

$$w = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{g} \tag{5}$$

Following the sampling method used in PRO, the matrices $\mathbf{X}$ and vector $\mathbf{g}$ are prepared as follows:

For each sentence,

1. Generate a list containing the k best translations of the sentence, with each translation e scored by the decoder using a function of the form $h(e) = w^{\mathrm{T}}x(e)$.

2. Use the uniform distribution to sample n random pairs from the set of candidate translations.

3. Calculate the gold scores g for the candidates in each pair using BLEU+1. Keep a pair of candidates as a potential pair if the difference between their g scores is bigger than a threshold t.

4. From the potential pairs kept in the previous step, keep the s pairs that have the highest differences in g and discard the rest.

5. For each pair e_1 and e_2 kept in step 4, make two data points $(x(e_1) - x(e_2), g(e_1) - g(e_2))$ and $(x(e_2) - x(e_1), g(e_2) - g(e_1))$.

The rows of $\mathbf{X}$ consist of the inputs of the data points created in step 5, i.e., the difference vectors $x(e_1) - x(e_2)$. Similarly, the corresponding rows in $\mathbf{g}$ are the outputs of the data points, i.e., the gold score differences $g(e_1) - g(e_2)$.

One important difference between the linear regression method and PRO is that rather than using the signs of the gold score differences and doing a binary classification, we use the differences of the gold scores directly, which allows us to use the information about the magnitude of the differences.

4 Experiments

4.1 Setup

We used a Chinese-English parallel corpus with the English side parsed for our experiments. The corpus consists of 250K sentence pairs, which is 6.3M words on the English side. The corpus derives from newswire texts available from LDC.[1] We used a 392-sentence development set with four references for parameter tuning, and a 428-sentence test set with four references for testing. They are drawn from the newswire portion of NIST evaluations (2004, 2005, 2006). The development set and the test set only had sentences with less than 30 words for decoding speed.

We extracted a general SCFG (GHKM) grammar using standard methods (Galley et al., 2004; Wang et al., 2010) from the parallel corpus with a modification to preclude any unary rules (Chung et al., 2011). All rules over scope 3 are pruned (Hopkins and Langmead, 2010). A set of nine standard features was used for the experiments, which includes globally normalized count of rules, lexical weighting (Koehn et al., 2003), and length penalty. Our in-house decoder was used for experiments with a trigram language model. The decoder is capable of both CNF parsing and Earley-style parsing with cube-pruning (Chiang, 2007).

We implemented linear regression tuning using the method explained in Section 3. Following Hopkins and May (2011), we used the following parameters for the sampling task: For each sentence, the decoder generates the 1500 best candidate translations ($k = 1500$), and the sampler samples 5000 pairs ($n = 5000$). Each pair is kept as a potential data point if their BLEU+1 score difference is bigger than 0.05 ($t = 0.05$). Finally, for each sentence, the sampler keeps the 50 pairs with the highest difference in BLEU+1 ($s = 50$) and generates two data points for each pair.

	Average of max BLEU		Max BLEU	
	dev	test	dev	test
Regression	27.7 (0.91)	26.4 (0.82)	29.0	27.6
PRO	26.9 (1.05)	25.6 (0.84)	28.0	27.2

Table 1: Average of maximum BLEU scores of the experiments and the maximum BLEU score from the experiments. Numbers in the parentheses indicate standard of deviations of maximum BLEU scores.

4.2 Results

We ran eight experiments with random initial weight vectors and ran each experiment for 25 iterations. Similar to what PRO does, in each iteration, we linearly interpolate the weight vector learned by the regression (w) with the weight vector of the previous iteration (w_{t-1}) using a factor of 0.1:

$$w_t = 0.1 \cdot w + 0.9 \cdot w_{t-1} \tag{6}$$

For the sake of comparison, we also implemented PRO with exactly the same parameters, and ran it with the same initial weight vectors.

For each initial weight vector, we selected the iteration at which the BLEU score on the development set is highest, and then decoded using this weight vector on the test set. The results of our experiments are presented in Table 1. In the first column, we show the average over the eight initial weight vectors of the BLEU score achieved, while in the second column we show the results from the initial weight vector with the highest BLEU score on the development set. Thus, while the second column corresponds to a tuning process where the single best result is retained, the first column shows the expected behavior of the procedure on a single initial weight vector. The linear regression method has

[1] We randomly sampled our data from various different sources (LDC2006E86, LDC2006E93, LDC2002E18, LDC2002L27, LDC2003E07, LDC2003E14, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E26, LDC2005E83, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006E24, LDC2006E92, LDC2006E24) The language model is trained on the English side of entire data (1.65M sentences, which is 39.3M words.)

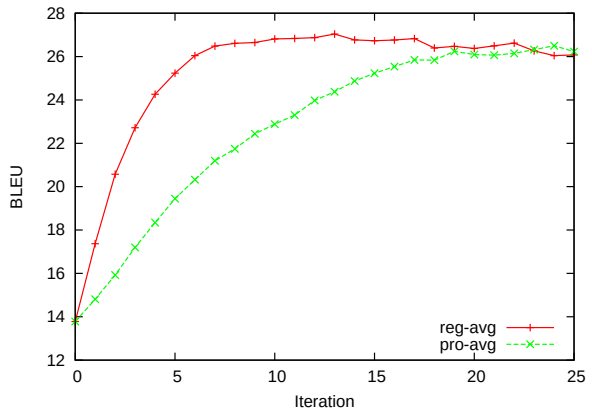

Figure 1: Average of eight runs of regression and PRO.

higher BLEU scores on both development and test data for both the average over initial weights and the maximum over initial weights.

Figure 1 shows the average of the BLEU scores on the development set of eight runs of the experiments. We observe that on average, the linear regression experiments reach the maximum BLEU score in a smaller number of iterations. On average, linear regression reached the maximum BLEU score after 14 iterations and PRO reached the maximum BLEU score after 20 iterations. One iteration took several minutes for both of the algorithms. The largest portion of this time is spent on decoding the development set and reading in the k-best list. The sampling phase, which includes performing linear regression or running MegaM, takes a negligible amount of time compared to the rest of the operations.

We experimented with adding L_2 regularization to linear regression. As expected, the experiments with regularization produced lower variance among the different experiments in terms of the BLEU score, and the resulting set of the parameters had a smaller norm. However, because of the small number of features used in our experiments, regularization was not necessary to control overfitting.

5 Discussion

We applied the idea of tuning as ranking and modified it to use linear regression instead of binary classification. The results of our experiments show that tuning as linear regression is as effective as PRO, and on average it reaches a better BLEU score in a fewer number of iterations.

In comparison with MERT, PRO and linear regression are different in the sense that the latter two approaches take into account rankings of the k-best list, whereas MERT is only concerned with separating the top 1-best sentence from the rest of the k-best list. PRO and linear regression are similar in the sense that both are concerned with ranking the k-best list. Their difference lies in the fact that PRO only uses the information on the relative rankings and uses binary classification to rank the points; on the contrary, linear regression directly uses the information on the magnitude of the differences. This difference between PRO and linear regression explains why linear regression converges faster and also may explain the fact that linear regression achieves a somewhat higher BLEU score. In this sense, linear regression is also similar to MIRA since MIRA's loss function also uses the information on the magnitude of score difference. However, the optimization problem for linear regression is simpler, does not require any changes to the decoder, and therefore the familiar MERT framework can be kept.

Acknowledgments We thank the anonymous reviewers for their helpful comments. This work was supported by NSF grant IIS-0910611.

References

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Tagyoung Chung, Licheng Fang, and Daniel Gildea. 2011. Issues concerning decoding with synchronous context-free grammar. In *Proceedings of the ACL 2011 Conference Short Papers*, Portland, Oregon. Association for Computational Linguistics.

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

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*, pages 273–280, Boston.

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010*

Conference on Empirical Methods in Natural Language Processing, pages 646–655, Cambridge, MA, October. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, Edmonton, Alberta.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of Coling 2004*, pages 501–507, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36:247–277, June.