Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability

Jonathan H. Clark Chris Dyer Alon Lavie Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jhclark,cdyer,alavie,nasmith}@cs.cmu.edu

Abstract

In statistical machine translation, a researcher seeks to determine whether some innovation (e.g., a new feature, model, or inference algorithm) improves translation quality in comparison to a baseline system. To answer this question, he runs an experiment to evaluate the behavior of the two systems on held-out data. In this paper, we consider how to make such experiments more statistically reliable. We provide a systematic analysis of the effects of optimizer instability—an extraneous variable that is seldom controlled for—on experimental outcomes, and make recommendations for reporting results more accurately.

1 Introduction

The need for statistical hypothesis testing for machine translation (MT) has been acknowledged since at least Och (2003). In that work, the proposed method was based on bootstrap resampling and was designed to improve the statistical reliability of results by controlling for randomness across test sets. However, there is no consistently used strategy that controls for the effects of unstable estimates of model parameters.[1] While the existence of optimizer instability is an acknowledged problem, it is only infrequently discussed in relation to the reliability of experimental results, and, to our knowledge, there has yet to be a systematic study of its effects on hypothesis testing. In this paper, we present a series of experiments demonstrating that optimizer instability can account for substantial amount of variation in translation quality,[2] which, if not controlled for, could lead to incorrect conclusions. We then show that it is possible to control for this variable with a high degree of confidence with only a few replications of the experiment and conclude by suggesting new best practices for significance testing for machine translation.

2 Nondeterminism and Other Optimization Pitfalls

Statistical machine translation systems consist of a model whose parameters are estimated to maximize some objective function on a set of development data. Because the standard objectives (e.g., 1-best BLEU, expected BLEU, marginal likelihood) are not convex, only approximate solutions to the optimization problem are available, and the parameters learned are typically only locally optimal and may strongly depend on parameter initialization and search hyperparameters. Additionally, stochastic optimization and search techniques, such as minimum error rate training (Och, 2003) and Markov chain Monte Carlo methods (Arun et al., 2010),[3] constitute a second, more obvious source of noise in the optimization procedure.

This variation in the parameter vector affects the quality of the model measured on both development

[1] We hypothesize that the convention of "trusting" BLEU score improvements of, e.g., > 1, is not merely due to an appreciation of what qualitative difference a particular quantitative improvement will have, but also an implicit awareness that current methodology leads to results that are not consistently reproducible.

[2] This variation directly affects the output translations, and so it will propagate to both automated metrics as well as human evaluators.

[3] Online subgradient techniques such as MIRA (Crammer et al., 2006; Chiang et al., 2008) have an implicit stochastic component as well based on the order of the training examples.

data and held-out test data, independently of any experimental manipulation. Thus, when trying to determine whether the difference between two measurements is significant, it is necessary to control for variance due to noisy parameter estimates. This can be done by replication of the optimization procedure with different starting conditions (e.g., by running MERT many times).

Unfortunately, common practice in reporting machine translation results is to run the optimizer once per system configuration and to draw conclusions about the experimental manipulation from this single sample. However, it could be that a particular sample is on the "low" side of the distribution over optimizer outcomes (i.e., it results in relatively poorer scores on the test set) or on the "high" side. The danger here is obvious: a high baseline result paired with a low experimental result could lead to a useful experimental manipulation being incorrectly identified as useless. We now turn to the question of how to reduce the probability falling into this trap.

3 Related Work

The use of statistical hypothesis testing has grown apace with the adoption of empirical methods in natural language processing. Bootstrap techniques (Efron, 1979; Wasserman, 2003) are widespread in many problem areas, including for confidence estimation in speech recognition (Bisani and Ney, 2004), and to determine the significance of MT results (Och, 2003; Koehn, 2004; Zhang et al., 2004; Zhang and Vogel, 2010). Approximate randomization (AR) has been proposed as a more reliable technique for MT significance testing, and evidence suggests that it yields fewer type I errors (i.e., claiming a significant difference where none exists; Riezler and Maxwell, 2005). Other uses in NLP include the MUC-6 evaluation (Chinchor, 1993) and parsing (Cahill et al., 2008). However, these previous methods assume model parameters are elements of the system rather than extraneous variables.

Prior work on optimizer noise in MT has focused primarily on *reducing* optimizer instability (whereas our concern is how to deal with optimizer noise, when it exists). Foster and Kuhn (2009) measured the instability of held-out BLEU scores across 10 MERT runs to improve tune/test set correlation. However, they only briefly mention the implications of the instability on significance. Cer et al. (2008) explored regularization of MERT to improve generalization on test sets. Moore and Quirk (2008) explored strategies for selecting better random "restart points" in optimization. Cer et al. (2010) analyzed the standard deviation over 5 MERT runs when each of several metrics was used as the objective function.

4 Experiments

In our experiments, we ran the MERT optimizer to optimize BLEU on a held-out development set many times to obtain a set of optimizer samples on two different pairs of systems (4 configurations total). Each pair consists of a baseline system (System A) and an "experimental" system (System B), which previous research has suggested will perform better.

The first system pair contrasts a baseline phrase-based system (Moses) and experimental hierarchical phrase-based system (Hiero), which were constructed from the Chinese-English BTEC corpus (0.7M words), the later of which was decoded with the cdec decoder (Koehn et al., 2007; Chiang, 2007; Dyer et al., 2010). The second system pair contrasts two German-English Hiero/cdec systems constructed from the WMT11 parallel training data (98M words).[4] The baseline system was trained on unsegmented words, and the experimental system was constructed using the most probable segmentation of the German text according to the CRF word segmentation model of Dyer (2009). The Chinese-English systems were optimized 300 times, and the German-English systems were optimized 50 times.

Our experiments used the default implementation of MERT that accompanies each of the two decoders. The Moses MERT implementation uses 20 random restart points per iteration, drawn uniformly from the default ranges for each feature, and, at each iteration, 200-best lists were extracted with the current weight vector (Bertoldi et al., 2009). The cdec MERT implementation performs inference over the decoder search space which is structured as a hypergraph (Kumar et al., 2009). Rather than using restart points, in addition to optimizing each feature independently, it optimizes in 5 random directions per iteration by constructing a search vector by uniformly sampling each element of the vector from $(-1, 1)$ and then renormalizing so it has length 1. For all systems, the initial weight vector was manually initialized so as to yield reasonable translations.

[4] http://statmt.org/wmt11/

Metric	System	Avg	$\overline{s_{\text{sel}}}$	s_{dev}	s_{test}
BTEC Chinese-English ($n = 300$)					
BLEU ↑	System A	48.4	1.6	0.2	0.5
	System B	49.9	1.5	0.1	0.4
MET ↑	System A	63.3	0.9	-	0.4
	System B	63.8	0.9	-	0.5
TER ↓	System A	30.2	1.1	-	0.6
	System B	28.7	1.0	-	0.2
WMT German-English ($n = 50$)					
BLEU ↑	System A	18.5	0.3	0.0	0.1
	System B	18.7	0.3	0.0	0.2
MET ↑	System A	49.0	0.2	-	0.2
	System B	50.0	0.2	-	0.1
TER ↓	System A	65.5	0.4	-	0.3
	System B	64.9	0.4	-	0.4

Table 1: Measured standard deviations of different automatic metrics due to test-set and optimizer variability. s_{dev} is reported only for the tuning objective function BLEU.

Results are reported using BLEU (Papineni et al., 2002), METEOR[5] (Banerjee and Lavie, 2005; Denkowski and Lavie, 2010), and TER (Snover et al., 2006).

4.1 Extraneous variables in one system

In this section, we describe and measure (on the example systems just described) three extraneous variables that should be considered when evaluating a translation system. We quantify these variables in terms of standard deviation s, since it is expressed in the same units as the original metric. Refer to Table 1 for the statistics.

Local optima effects s_{dev} The first extraneous variable we discuss is the stochasticity of the optimizer. As discussed above, different optimization runs find different local maxima. The noise due to this variable can depend on many number of factors, including the number of random restarts used (in MERT), the number of features in a model, the number of references, the language pair, the portion of the search space visible to the optimizer (e.g. 10-best, 100-best, a lattice, a hypergraph), and the size of the tuning set. Unfortunately, there is no proxy to estimate this effect as with bootstrap resampling. To control for this variable, we must run the optimizer multiple times to estimate the spread it induces on the *development set*. Using the n optimizer samples, with m_i as the translation quality measurement of the development set for the ith optimization run, and $\overline{m}$ is the average of all m_is, we report the standard deviation over the tuning set as s_{dev}:

$$s_{\text{dev}} = \sqrt{\sum_{i=1}^{n} \frac{(m_i - \overline{m})^2}{n-1}}$$

A high s_{dev} value may indicate that the optimizer is struggling with local optima and changing hyperparameters (e.g. more random restarts in MERT) could improve system performance.

Overfitting effects s_{test} As with any optimizer, there is a danger that the optimal weights for a tuning set may not generalize well to unseen data (i.e., we overfit). For a randomized optimizer, this means that parameters can generalize to different degrees over multiple optimizer runs. We measure the spread induced by optimizer randomness on the test set metric score s_{test}, as opposed to the overfitting effect in isolation. The computation of s_{test} is identical to s_{dev} except that the m_is are the translation metrics calculated on the *test set*. In Table 1, we observe that $s_{\text{test}} > s_{\text{dev}}$, indicating that optimized parameters are likely not generalizing well.

Test set selection $\overline{s_{\text{sel}}}$ The final extraneous variable we consider is the selection of the test set itself. A good test set should be representative of the domain or language for which experimental evidence is being considered. However, with only a single test corpus, we may have unreliable results because of idiosyncrasies in the test set. This can be mitigated in two ways. First, replication of experiments by testing on multiple, non-overlapping test sets can eliminate it directly. Since this is not always practical (more test data may not be availabile), the widely-used bootstrap resampling method (§3) also controls for test set effects by resampling multiple "virtual" test sets from a single set, making it possible to infer distributional parameters such as the standard deviation of the translation metric over (very similar) test sets.[6] Furthermore, this can be done for each of our optimizer samples. By averaging the bootstrap-estimated standard deviations over

[5] METEOR version 1.2 with English ranking parameters and all modules.

[6] Unlike actually using multiple test sets, bootstrap resampling does not help to re-estimate the mean metric score due to test set spread (unlike actually using multiple test sets) since the mean over bootstrap replicates is approximately the aggregate metric score.

optimizer samples, we have a statistic that jointly quantifies the impact of test set effects and optimizer instability on a test set. We call this statistic $\overline{s_{\text{sel}}}$. Different values of this statistic can suggest methodological improvements. For example, a large $\overline{s_{\text{sel}}}$ indicates that more replications will be necessary to draw reliable inferences from experiments on this test set, so a larger test set may be helpful.

To compute $\overline{s_{\text{sel}}}$, assume we have n independent optimization runs which produced weight vectors that were used to translate a test set n times. The test set has ℓ segments with references $\mathbf{R} = \langle R_1, R_2, \ldots, R_\ell \rangle$. Let $\mathcal{X} = \langle \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n \rangle$ where each $\mathbf{X}_i = \langle X_{i1}, X_{i2}, \ldots, X_{i\ell} \rangle$ is the list of translated segments from the ith optimization run list of the ℓ translated segments of the test set. For each hypothesis output $\mathbf{X}_i$, we construct k bootstrap replicates by drawing ℓ segments uniformly, *with replacement*, from $\mathbf{X}_i$, together with its corresponding reference. This produces k virtual test sets for each optimization run i. We designate the score of the jth virtual test set of the ith optimization run with m_{ij}. If $\overline{m_i} = \frac{1}{k}\sum_{j=1}^{k} m_{ij}$, then we have:

$$s_i = \sqrt{\sum_{j=1}^{k} \frac{(m_{ij} - \overline{m_i})^2}{k-1}}$$

$$\overline{s_{\text{sel}}} = \frac{1}{n}\sum_{i=1}^{n} s_i$$

4.2 Comparing Two Systems

In the previous section, we gave statistics about the distribution of evaluation metrics across a large number of experimental samples (Table 1). Because of the large number of trials we carried out, we can be extremely confident in concluding that for both pairs of systems, the experimental manipulation accounts for the observed metric improvements, and furthermore, that we have a good estimate of the magnitude of that improvement. However, it is not generally feasible to perform as many replications as we did, so here we turn to the question of how to compare two systems, accounting for optimizer noise, but without running 300 replications.

We begin with a visual illustration how optimizer instability affects test set scores when comparing two systems. Figure 1 plots the histogram of the 300 optimizer samples each from the two BTEC Chinese-English systems. The phrase-based system's distribution is centered at the sample mean 48.4, and the hierarchical system is centered at 49.9, a difference of 1.5 BLEU, corresponding to the widely replicated result that hierarchical phrase-based systems outperform conventional phrase-based systems in Chinese-English translation. Crucially, although the distributions are distinct, there is a non-trivial region of overlap, and experimental samples from the overlapping region could suggest the opposite conclusion!

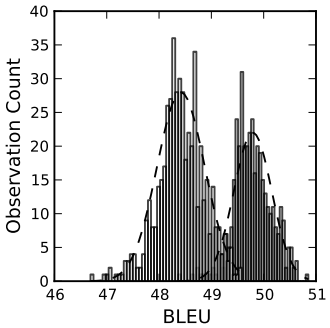

Figure 1: Histogram of test set BLEU scores for the BTEC phrase-based system (left) and BTEC hierarchical system (right). While the difference between the systems is 1.5 BLEU in expectation, there is a non-trivial region of overlap indicating that some random outcomes will result in little to no difference being observed.

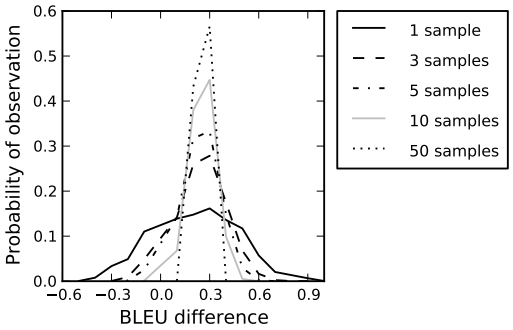

Figure 2: Relative frequencies of obtaining differences in BLEU scores on the WMT system as a function of the number of optimizer samples. The expected difference is 0.2 BLEU. While there is a reasonably high chance of observing a non-trivial improvement (or even a decline) for 1 sample, the distribution quickly peaks around the expected value given just a few more samples.

To further underscore the risks posed by this overlap, Figure 2 plots the relative frequencies with which different BLEU score deltas will occur, as a function of the number of optimizer samples used.

When is a difference significant? To determine whether an experimental manipulation results in a

statistically reliable difference for an evaluation metric, we use a stratified approximate randomization (AR) test. This is a nonparametric test that approximates a paired permutation test by sampling permutations (Noreen, 1989). AR estimates the probability (p-value) that a measured difference in metric scores arose by chance by randomly exchanging sentences between the two systems. If there is no significant difference between the systems (i.e., the null hypothesis is true), then this shuffling should not change the computed metric score. Crucially, this assumes that the samples being analyzed are representative of all extraneous variables that could affect the outcome of the experiment. Therefore, we must include multiple optimizer replications. Also, since metric scores (such as BLEU) are in general not comparable across test sets, we stratify, exchanging only hypotheses that correspond to the same sentence.

Table 2 shows the p-values computed by AR, testing the significance of the differences between the two systems in each pair. The first three rows illustrate "single sample" testing practice. Depending on luck with MERT, the results can vary widely from insignificant (at $p > .05$) to highly significant.

The last two lines summarize the results of the test when a small number of replications are performed, as ought to be reasonable in a research setting. In this simulation, we randomly selected n optimizer outputs from our large pool and ran the AR test to determine the significance; we repeated this procedure 250 times. The p-values reported are the p-values at the edges of the *95% confidence interval* (CI) according to AR seen in the 250 simulated comparison scenarios. These indicate that we are very likely to observe a significant difference for BTEC at $n = 5$, and a very significant difference by $n = 50$ (Table 2). Similarly, we see this trend in the WMT system: more replications leads to more significant results, which will be easier to reproduce. Based on the average performance of the systems reported in Table 1, we *expect* significance over a large enough number of independent trials.

			p-value	
n	System A	System B	BTEC	WMT
1	high	low	0.25	0.95
1	median	median	0.15	0.13
1	low	high	0.0003	0.003
			p-value (95% CI)	
5	**random**	**random**	0.001–0.034	0.001–0.38
50	**random**	**random**	0.001–0.001	0.001–0.33

Table 2: Two-system analysis: AR p-values for three different "single sample" scenarios that illustrate different pathological scenarios that can result when the sampled weight vectors are "low" or "high." For "random," we simulate an experiments with n optimization replications by drawing n optimized system outputs from our pool and performing AR; this simulation was repeated 250 times and the 95% CI of the AR p-values is reported.

5 Discussion and Recommendations

No experiment can completely control for all possible confounding variables. Nor are metric scores (even if they are statistically reliable) a substitute for thorough human analysis. However, we believe that the impact of optimizer instability has been neglected by standard experimental methodology in MT research, where single-sample measurements are too often used to assess system differences. In this paper, we have provided evidence that optimizer instability can have a substantial impact on results. However, we have also shown that it is possible to control for it with very few replications (Table 2). We therefore suggest:

- Replication be adopted as standard practice in MT experimental methodology, especially in reporting results;[7]
- Replication of optimization (MERT) and test set evaluation be performed at least three times; more replications may be necessary for experimental manipulations with more subtle effects;
- Use of the median system according to a trusted metric when *manually* analyzing system output; preferably, the median should be determined based on one test set and a second test set should be manually analyzed.

Acknowledgments

We thank Michael Denkowski, Kevin Gimpel, Kenneth Heafield, Michael Heilman, and Brendan O'Connor for insightful feedback. This research was supported in part by the National Science Foundation through TeraGrid resources provided by Pittsburgh Supercomputing Center under TG-DBS110003; the National Science Foundation under IIS-0713402, IIS-0844507, IIS-0915187, and IIS-0915327; the DARPA GALE program, the U. S. Army Research Laboratory, and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533.

[7] Source code to carry out the AR test for multiple optimizer samples on the three metrics in this paper is available from `http://github.com/jhclark/multeval`.

References

A. Arun, B. Haddow, P. Koehn, A. Lopez, C. Dyer, and P. Blunsom. 2010. Monte Carlo techniques for phrase-based translation. *Machine Translation*, 24:103–121.

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proc. of ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

N. Bertoldi, B. Haddow, and J.-B. Fouet. 2009. Improved minimum error rate training in Moses. *Prague Bulletin of Mathematical Linguistics*, No. 91:7–16.

M. Bisani and H. Ney. 2004. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proc. of ICASSP*.

A. Cahill, M. Burke, R. O'Donovan, S. Riezler, J. van Genabith, and A. Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124.

D. Cer, D. Jurafsky, and C. D. Manning. 2008. Regularization and search for minimum error rate training. In *Proc. of WMT*.

D. Cer, C. D. Manning, and D. Jurafsky. 2010. The best lexical metric for phrase-based statistical MT system optimization. In *Proc. of NAACL*.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

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

N. Chinchor. 1993. The statistical significance of the MUC-5 results. *Proc. of MUC*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

M. Denkowski and A. Lavie. 2010. Extending the METEOR machine translation evaluation metric to the phrase level. In *Proc. of NAACL*.

C. Dyer, J. Weese, A. Lopez, V. Eidelman, P. Blunsom, and P. Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*.

C. Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proc. of NAACL*.

B. Efron. 1979. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

G. Foster and R. Kuhn. 2009. Stabilizing minimum error rate training. *Proc. of WMT*.

P. Koehn, A. Birch, C. Callison-burch, M. Federico, N. Bertoldi, B. Cowan, C. Moran, C. Dyer, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of ACL-IJCNLP*.

R. C. Moore and C. Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proc. of COLING*, Manchester, UK.

E. W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

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

K. Papineni, S. Roukos, T. Ward, and W.-j. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

S. Riezler and J. T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proc. of the Workshop on Intrinsic and Extrinsic Evaluation Methods for Machine Translation and Summarization*.

M. Snover, B. Dorr, C. Park, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*.

L. Wasserman. 2003. *All of Statistics: A Concise Course in Statistical Inference*. Springer.

Y. Zhang and S. Vogel. 2010. Significance tests of automatic machine translation metrics. *Machine Translation*, 24:51–65.

Y. Zhang, S. Vogel, and A. Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proc. of LREC*.