Consensus Training for Consensus Decoding in Machine Translation

Adam Pauls, John DeNero and **Dan Klein**
Computer Science Division
University of California at Berkeley
{adpauls,denero,klein}@cs.berkeley.edu

Abstract

We propose a novel objective function for discriminatively tuning log-linear machine translation models. Our objective explicitly optimizes the BLEU score of *expected* n-gram counts, the same quantities that arise in forest-based consensus and minimum Bayes risk decoding methods. Our continuous objective can be optimized using simple gradient ascent. However, computing critical quantities in the gradient necessitates a novel dynamic program, which we also present here. Assuming BLEU as an evaluation measure, our objective function has two principle advantages over standard max BLEU tuning. First, it specifically optimizes model weights for downstream consensus decoding procedures. An unexpected second benefit is that it reduces overfitting, which can improve test set BLEU scores when using standard Viterbi decoding.

1 Introduction

Increasing evidence suggests that machine translation decoders should not search for a single top scoring Viterbi derivation, but should instead choose a translation that is sensitive to the model's entire predictive distribution. Several recent consensus decoding methods leverage compact representations of this distribution by choosing translations according to n-gram posteriors and expected counts (Tromble et al., 2008; DeNero et al., 2009; Li et al., 2009; Kumar et al., 2009). This change in decoding objective suggests a complementary change in *tuning* objective, to one that optimizes expected n-gram counts directly. The ubiquitous minimum error rate training (MERT) approach optimizes Viterbi predictions, but does not explicitly boost the aggregated posterior probability of desirable n-grams (Och, 2003).

We therefore propose an alternative objective function for parameter tuning, which we call *consensus* BLEU or CoBLEU, that is designed to maximize the *expected counts* of the n-grams that appear in reference translations. To maintain consistency across the translation pipeline, we formulate CoBLEU to share the functional form of BLEU used for evaluation. As a result, CoBLEU optimizes exactly the quantities that drive efficient consensus decoding techniques and precisely mirrors the objective used for fast consensus decoding in DeNero et al. (2009).

CoBLEU is a continuous and (mostly) differentiable function that we optimize using gradient ascent. We show that this function and its gradient are efficiently computable over packed forests of translations generated by machine translation systems. The gradient includes expectations of *products* of features and n-gram counts, a quantity that has not appeared in previous work. We present a new dynamic program which allows the efficient computation of these quantities over translation forests. The resulting gradient ascent procedure does not require any k-best approximations. Optimizing over translation forests gives similar stability benefits to recent work on lattice-based minimum error rate training (Macherey et al., 2008) and large-margin training (Chiang et al., 2008).

We developed CoBLEU primarily to complement consensus decoding, which it does; it produces higher BLEU scores than coupling MERT with consensus decoding. However, we found an additional empirical benefit: CoBLEU is less prone to overfitting than MERT, even when using Viterbi decoding. In experiments, models trained to maximize tuning set BLEU using MERT consistently degraded in performance from tuning to test set, while CoBLEU-trained models generalized more robustly. As a result, we found that optimizing CoBLEU improved test set performance reliably using consensus decoding and occasionally using Viterbi decoding.

(a) **Hypotheses ranked by $\theta_{TM} = \theta_{LM} = 1$**

Sentence *f*: Il était une rime
Reference *r*: Once upon a rhyme

	TM	LM	Pr
H1) Once on a rhyme	-3	-7	0.67
H2) Once upon a rhyme	-5	-6	0.24
H3) Once upon a time	-9	-3	0.09

(b) **Computing Consensus Bigram Precision**

$$\begin{aligned}
\mathbb{E}_\theta[c(\text{"Once upon"}, d)|f] &= 0.24 + 0.09 = 0.33 \\
\mathbb{E}_\theta[c(\text{"upon a"}, d)|f] &= 0.24 + 0.09 = 0.33 \\
\mathbb{E}_\theta[c(\text{"a rhyme"}, d)|f] &= 0.67 + 0.24 = 0.91 \\
\sum_g \mathbb{E}_\theta[c(g, d)|f] &= 3[0.67 + 0.24 + 0.09] \\
\frac{\sum_g \min\{\mathbb{E}_\theta[c(g, d)|f], c(g, r)\}}{\sum_g \mathbb{E}_\theta[c(g, d)|f]} &= \frac{0.33 + 0.33 + 0.91}{3}
\end{aligned}$$

Figure 1: (a) A simple hypothesis space of translations for a single sentence containing three alternatives, each with two features. The hypotheses are scored under a log-linear model with parameters θ equal to the identity vector. (b) The expected counts of all bigrams that appear in the computation of consensus bigram precision.

2 Consensus Objective Functions

Our proposed objective function maximizes n-gram precision by adapting the BLEU evaluation metric as a tuning objective (Papineni et al., 2002). To simplify exposition, we begin by adapting a simpler metric: bigram precision.

2.1 Bigram Precision Tuning

Let the tuning corpus consist of source sentences $F = f_1 \dots f_m$ and human-generated references $R = r_1 \dots r_m$, one reference for each source sentence. Let e_i be a translation of f_i, and let $E = e_1 \dots e_m$ be a corpus of translations, one for each source sentence. A simple evaluation score for E is its bigram precision BP(R, E):

$$\text{BP}(R, E) = \frac{\sum_{i=1}^m \sum_{g_2} min\{c(g_2, e_i), c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} c(g_2, e_i)}$$

where g_2 iterates over the set of bigrams in the target language, and $c(g_2, e)$ is the count of bigram g_2 in translation e. As in BLEU, we "clip" the bigram counts of e in the numerator using counts of bigrams in the reference sentence.

Modern machine translation systems are typically tuned to maximize the evaluation score of Viterbi derivations[1] under a log-linear model with parameters θ. Let $d^*_\theta(f_i) = \arg\max_d P_\theta(d|f_i)$ be the highest scoring derivation d of f_i. For a system employing Viterbi decoding and evaluated by bigram precision, we would want to select θ to maximize MaxBP(R, F, θ):

$$\frac{\sum_{i=1}^m \sum_{g_2} min\{c(g_2, d^*_\theta(f_i)), c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} c(g_2, d^*_\theta(f_i))}$$

On the other hand, for a system that uses expected bigram counts for decoding, we would prefer to choose θ such that expected bigram counts match bigrams in the reference sentence. To this end, we can evaluate an entire posterior distribution over derivations by computing the same clipped precision for expected bigram counts using CoBP(R, F, θ):

$$\frac{\sum_{i=1}^m \sum_{g_2} min\{\mathbb{E}_\theta[c(g_2, d)|f_i], c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} \mathbb{E}_\theta[c(g_2, d)|f_i]} \quad (1)$$

where

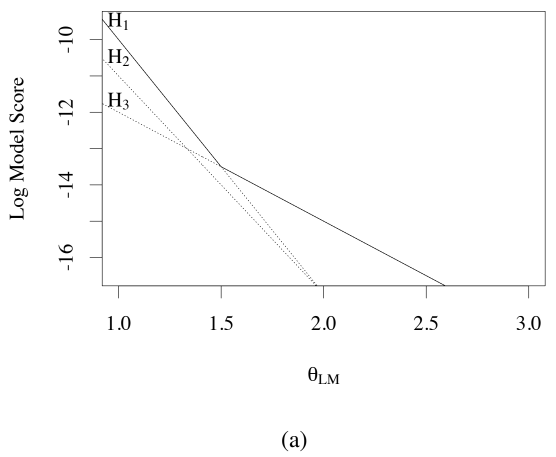
$$\mathbb{E}_\theta[c(g_2, d)|f_i] = \sum_d P_\theta(d|f_i) c(g_2, d)$$

is the expected count of bigram g_2 in all derivations d of f_i. We define the precise parametric form of $P_\theta(d|f_i)$ in Section 3. Figure 1 shows proposed translations for a single sentence along with the bigram expectations needed to compute CoBP.

Equation 1 constitutes an objective function for tuning the parameters of a machine translation model. Figure 2 contrasts the properties of CoBP and MaxBP as tuning objectives, using the simple example from Figure 1.

Consensus bigram precision is an instance of a general recipe for converting n-gram based evaluation metrics into consensus objective functions for model tuning. For the remainder of this paper, we focus on consensus BLEU. However, the techniques herein, including the optimization approach of Section 3, are applicable to many differentiable functions of expected n-gram counts.

[1]By *derivation*, we mean a translation of a foreign sentence along with any latent structure assumed by the model. Each derivation corresponds to a particular English translation, but many derivations may yield the same translation.



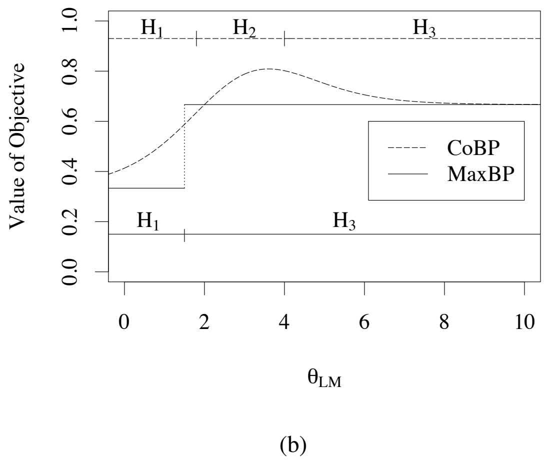


Figure 2: These plots illustrate two properties of the objectives *max bigram precision* (MaxBP) and *consensus bigram precision* (CoBP) on the simple example from Figure 1. (a) MaxBP is only sensitive to the convex hull (the solid line) of model scores. When varying the single parameter θ_{LM}, it entirely disregards the correct translation H_2 because H_2 never attains a maximal model score. (b) A plot of both objectives shows their differing characteristics. The horizontal segmented line at the top of the plot indicates the range over which consensus decoding would select each hypothesis, while the segmented line at the bottom indicates the same for Viterbi decoding. MaxBP is only sensitive to the single point of discontinuity between H_1 and H_3, and disregards H_2 entirely. CoBP peaks when the distribution most heavily favors H_2 while suppressing H_1. Though H_2 never has a maximal model score, if θ_{LM} is in the indicated range, consensus decoding would select H_2, the desired translation.

2.2 CoBLEU

The logarithm of the single-reference[2] BLEU metric (Papineni et al., 2002) has the following form:

$$\ln \text{BLEU}(R, E) = \left(1 - \frac{|R|}{\sum_{i=1}^{m}\sum_{g_1} c(g_1, e_i)}\right)_{-} + \frac{1}{4}\sum_{n=1}^{4} \ln \frac{\sum_{i=1}^{m}\sum_{g_n} \min\{c(g_n, e_i), c(g_n, r_i)\}}{\sum_{i=1}^{m}\sum_{g_n} c(g_n, e_i)}$$

Above, $|R|$ denotes the number of words in the reference corpus. The notation $(\cdot)_{-}$ is shorthand for $min(\cdot, 0)$. In the inner sums, g_n iterates over all n-grams of order n. In order to adapt BLEU to be a consensus tuning objective, we follow the recipe of Section 2.1: we replace n-gram counts from a candidate translation with expected n-gram counts under the model.

$$\text{CoBLEU}(R, F, \theta) = \left(1 - \frac{|R|}{\sum_{i=1}^{m}\sum_{g_1} \mathbb{E}_\theta[c(g_1, d)|f_i]}\right)_{-} + \frac{1}{4}\sum_{n=1}^{4} \ln \frac{\sum_{i=1}^{m}\sum_{g_n} \min\{\mathbb{E}_\theta[c(g_n, d)|f_i], c(g_n, r_i)\}}{\sum_{i=1}^{m}\sum_{g_n} \mathbb{E}_\theta[c(g_n, d)|f_i]}$$

The brevity penalty term in BLEU is calculated using the expected length of the corpus, which equals the sum of all expected unigram counts. We call this objective function *consensus* BLEU, or CoBLEU for short.

[2]Throughout this paper, we use only a single reference, but our objective readily extends to multiple references.

3 Optimizing CoBLEU

Unlike the more common MaxBLEU tuning objective optimized by MERT, CoBLEU is continuous. For distributions $P_\theta(d|f_i)$ that factor over synchronous grammar rules and n-grams, we show below that it is also analytically differentiable, permitting a straightforward gradient ascent optimization procedure.[3] In order to perform gradient ascent, we require methods for efficiently computing the gradient of the objective function for a given parameter setting θ. Once we have the gradient, we can perform an update at iteration t of the form

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta_t \nabla_\theta \text{CoBLEU}(R, F, \theta^{(t)})$$

where η_t is an adaptive step size.[4]

[3]Technically, CoBLEU is non-differentiable at some points because of *clipping*. At these points, we must compute a sub-gradient, and so our optimization is formally sub-gradient ascent. See the Appendix for details.

[4]After each successful step, we grow the step size by a constant factor. Whenever the objective does not decrease after a step, we shrink the step size by a constant factor and try again until a decrease is attained.

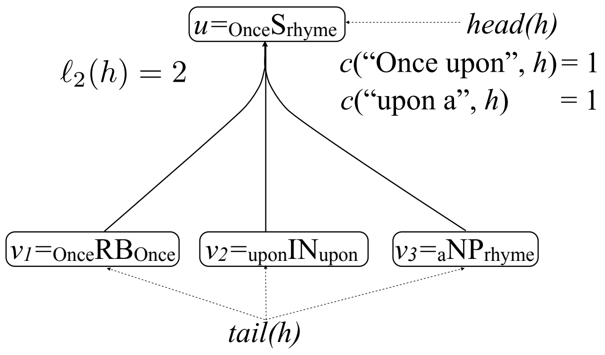


Figure 3: A hyperedge h represents a "rule" used in syntactic machine translation. *tail(h)* refers to the "children" of the rule, while $head(h)$ refers to the "head" or "parent". A forest of translations is built by combining the nodes v_i using h to form a new node $u = head(h)$. Each forest node consists of a grammar symbol and target language boundary words used to track n-grams. In the above, we keep one boundary word for each node, which allows us to track bigrams.

In this section, we develop an analytical expression for the gradient of CoBLEU, then discuss how to efficiently compute the value of the objective function and gradient.

3.1 Translation Model Form

We first assume the general hypergraph setting of Huang and Chiang (2007), namely, that derivations under our translation model form a hypergraph. This framework allows us to speak about both phrase-based and syntax-based translation in a unified framework.

We define a probability distribution over derivations d via θ as:

$$P_\theta(d|f_i) = \frac{w(d)}{Z(f_i)}$$

with

$$Z(f_i) = \sum_{d'} w(d')$$

where $w(d) = \exp(\theta^\top \Phi(d, f_i))$ is the weight of a derivation and $\Phi(d, f_i)$ is a featurized representation of the derivation d of f_i. We further assume that these features decompose over *hyperedges* in the hypergraph, like the one in Figure 3. That is, $\Phi(d, f_i) = \sum_{h \in d} \Phi(h, f_i)$.

In this setting, we can analytically compute the gradient of CoBLEU. We provide a sketch of the derivation of this gradient in the Appendix. In computing this gradient, we must calculate the following expectations:

$$\mathbb{E}_\theta\left[c(\phi_k, d)|f_i\right] \tag{2}$$

$$\mathbb{E}_\theta\left[\ell_n(d)|f_i\right] \tag{3}$$

$$\mathbb{E}_\theta\left[c(\phi_k, d) \cdot \ell_n(d)|f_i\right] \tag{4}$$

where $\ell_n(d) = \sum_{g_n} c(g_n, d)$ is the sum of all n-grams on derivation d (its "length"). The first expectation is an expected count of the kth feature ϕ_k over all derivations of f_i. The second is an expected length, the total expected count of all n-grams in derivations of f_i. We call the final expectation an expected *product of counts*. We now present the computation of each of these expectations in turn.

3.2 Computing Feature Expectations

The expected feature counts $\mathbb{E}_\theta[c(\phi_k, d)|f_i]$ can be written as

$$\begin{aligned}\mathbb{E}_\theta[c(\phi_k, d)|f_i] &= \sum_d P_\theta(d|f_i) c(\phi_k, d) \\ &= \sum_h P_\theta(h|f_i) c(\phi_k, h)\end{aligned}$$

We can justify the second step since feature counts are local to hyperedges, i.e. $c(\phi_k, d) = \sum_{h \in d} c(\phi_k, h)$. The posterior probability $P_\theta(h|f_i)$ can be efficiently computed with inside-outside scores. Let $\mathrm{I}(u)$ and $\mathrm{O}(u)$ be the standard inside and outside scores for a node u in the forest.[5]

$$P_\theta(h|f_i) = \frac{1}{Z(f)}\, w(h)\, \mathrm{O}(head(h)) \prod_{v \in tail(h)} \mathrm{I}(v)$$

where $w(h)$ is the weight of hyperedge h, given by $exp(\theta^\top \Phi(h))$, and $Z(f) = \mathrm{I}(root)$ is the inside score of the root of the forest. Computing these inside-outside quantities takes time linear in the number of hyperedges in the forest.

3.3 Computing n-gram Expectations

We can compute the expectations of any specific n-grams, or of total n-gram counts ℓ, in the same way as feature expectations, provided that target-side n-grams are also localized to hyperedges (e.g. consider ℓ to be a feature of a hyperedge whose value is the number of n-grams on h). If the nodes in our forests are annotated with target-side

[5] Appendix Figure 7 gives recursions for $\mathrm{I}(u)$ and $\mathrm{O}(u)$.

boundary words as in Figure 3, then this will be the case. Note that this is the same approach used by decoders which integrate a target language model (e.g. Chiang (2007)). Other work has computed n-gram expectations in the same way (DeNero et al., 2009; Li et al., 2009).

3.4 Computing Expectations of Products of Counts

While the previous two expectations can be computed using techniques known in the literature, the expected product of counts $\mathbb{E}_\theta[c(\phi_k, d) \cdot \ell_n(d)|f_i]$ is a novel quantity. Fortunately, an efficient dynamic program exists for computing this expectation as well. We present this dynamic program here as one of the contributions of this paper, though we omit a full derivation due to space restrictions.

To see why this expectation cannot be computed in the same way as the expected feature or n-gram counts, we expand the definition of the expectation above to get

$$\sum_d P_\theta(d|f_i)\,[c(\phi_k, d)\ell_n(d)]$$

Unlike feature and n-gram counts, the product of counts in brackets above does not decompose over hyperedges, at least not in an obvious way. We can, however, still decompose the feature counts $c(\phi_k, d)$ over hyperedges. After this decomposition and a little re-arranging, we get

$$\begin{aligned} &= \sum_h c(\phi_k, h) \sum_{d:h\in d} P_\theta(d|f_i)\ell_n(d) \\ &= \frac{1}{Z(f_i)} \sum_h c(\phi_k, h) \left[\sum_{d:h\in d} w(d)\ell_n(d) \right] \\ &= \frac{1}{Z(f_i)} \sum_h c(\phi_k, h) \hat{\mathrm{D}}^n_\theta(h|f_i) \end{aligned}$$

The quantity $\hat{\mathrm{D}}^n_\theta(h|f_i) = \sum_{d:h\in d} w(d)\ell_n(d)$ is the sum of the weight-length products of all derivations d containing hyperedge h. In the same way that $P_\theta(h|f_i)$ can be efficiently computed from inside and outside probabilities, this quantity $\hat{\mathrm{D}}^n_\theta(h|f_i)$ can be efficiently computed with two new inside and outside quantities, which we call $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$. We provide recursions for these quantities in Figure 4. Like the standard inside and outside computations, these recursions run in time linear in the number of hyperedges in the forest.

While a full exposition of the algorithm is not possible in the available space, we give some brief intuition behind this dynamic program. We first define $\hat{\mathrm{I}}_n(u)$:

$$\hat{\mathrm{I}}_n(u) = \sum_{d_u} w(d_u)\ell_n(d)$$

where d_u is a derivation rooted at node u. This is a sum of weight-length products similar to $\hat{\mathrm{D}}$. To give a recurrence for $\hat{\mathrm{I}}$, we rewrite it:

$$\hat{\mathrm{I}}_n(u) = \sum_{d_u} \sum_{h\in d_u} [w(d_u)\ell_n(h)]$$

Here, we have broken up the total value of $\ell_n(d)$ across hyperedges in d. The bracketed quantity is a score of a *marked derivation* pair (d, h) where the edge h is some specific element of d. The score of a marked derivation includes the weight of the derivation and the factor $\ell_n(h)$ for the marked hyperedge.

This sum over marked derivations gives the inside recurrence in Figure 4 by the following decomposition. For $\hat{\mathrm{I}}_n(u)$ to sum over all marked derivation pairs rooted at u, we must consider two cases. First, the marked hyperedge could be at the root, in which case we must choose child derivations from regular inside scores and multiply in the local ℓ_n, giving the first summand of $\hat{\mathrm{I}}_n(u)$. Alternatively, the marked hyperedge is in exactly one of the children; for each possibility we recursively choose a marked derivation for one child, while the other children choose regular derivations. The second summand of $\hat{\mathrm{I}}_n(u)$ compactly expresses a sum over instances of this case. $\hat{\mathrm{O}}_n(u)$ decomposes similarly: the marked hyperedge could be local (first summand), under a sibling (second summand), or higher in the tree (third summand).

Once we have these new inside-outside quantities, we can compute $\hat{\mathrm{D}}$ as in Figure 5. This combination states that marked derivations containing h are either marked at h, below h, or above h.

As a final detail, computing the gradient $\nabla C_n^{\text{clip}}(\theta)$ (see the Appendix) involves a clipped version of the expected product of counts, for which a clipped $\hat{D}$ is required. This quantity can be computed with the same dynamic program with a slight modification. In Figure 4, we show the difference as a choice point when computing $\ell_n(h)$.

3.5 Implementation Details

As stated, the runtime of computing the required expectations for the objective and gradient is linear in the number of hyperedges in the forest. The

$$\hat{\mathrm{I}}_n(u) = \sum_{h\in\mathrm{IN}(u)} w(h) \left[\ell_n(h) \prod_{v\in tail(h)} \mathrm{I}(v) + \sum_{v\in tail(h)} \hat{\mathrm{I}}_n(v) \prod_{w\neq v} \mathrm{I}(w) \right]$$

$$\hat{\mathrm{O}}_n(u) = \sum_{h\in\mathrm{OUT}(u)} w(h) \left[\ell_n(h)\, \mathrm{O}(head(h)) \prod_{\substack{v\in tail(h)\\ v\neq u}} \mathrm{I}(v) \;+\; \mathrm{O}(head(h)) \sum_{\substack{v\in tail(h)\\ v\neq u}} \hat{\mathrm{I}}_n(v) \prod_{\substack{w\in tail(h)\\ w\neq v\\ w\neq u}} \mathrm{I}(w) \;+\; \hat{\mathrm{O}}_n(head(h)) \prod_{\substack{w\in tail(h)\\ w\neq u}} \mathrm{I}(w) \right]$$

$$\ell_n(h) = \begin{cases} \sum_{g_n} c(g_n, h) & \text{computing unclipped counts} \\ \sum_{g_n} c(g_n, h) \mathbb{1}\left[\mathbb{E}_\theta[c(g_n, d)] \leq c(g_n, r_i)\right] & \text{computing clipped counts} \end{cases}$$

Figure 4: Inside and Outside recursions for $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$. $\mathrm{IN}(u)$ and $\mathrm{OUT}(u)$ refer to the incoming and outgoing hyperedges of u, respectively. $\mathrm{I}(\cdot)$ and $\mathrm{O}(\cdot)$ refer to standard inside and outside quantities, defined in Appendix Figure 7. We initialize with $\hat{\mathrm{I}}_n(u) = 0$ for all terminal forest nodes u and $\hat{\mathrm{O}}_n(root) = 0$ for the root node. $\ell_n(h)$ computes the sum of all n-grams of order n on a hyperedge h.

$$\hat{\mathrm{D}}^n_\theta(h|f_i) =$$
$$w(h) \left[\ell_n(h) \mathrm{O}(head(h)) \prod_{v\in tail(h)} \mathrm{I}(v) + \mathrm{O}(head(h)) \sum_{v\in tail(h)} \hat{\mathrm{I}}_n(v) \prod_{\substack{v\in tail(h)\\ w\neq v}} \mathrm{I}(w) + \hat{\mathrm{O}}_n(head(h)) \prod_{w\in tail(h)} \mathrm{I}(w) \right]$$

Figure 5: Calculation of $\hat{\mathrm{D}}^n_\theta(h|f_i)$ after $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$ have been computed.

number of hyperedges is very large, however, because we must track n-gram contexts in the nodes, just as we would in an integrated language model decoder. These contexts are required both to correctly compute the model score of derivations and to compute clipped n-gram counts. To speed our computations, we use the cube pruning method of Huang and Chiang (2007) with a fixed beam size.

For regularization, we added an L_2 penalty on the size of θ to the CoBLEU objective, a simple addition for gradient ascent. We did not find that our performance varied very much for moderate levels of regularization.

3.6 Related Work

The calculation of expected counts can be formulated using the expectation semiring framework of Eisner (2002), though that work does not show how to compute expected products of counts which are needed for our gradient calculations. Concurrently with this work, Li and Eisner (2009) have generalized Eisner (2002) to compute expected products of counts on translation forests. The training algorithm of Kakade et al. (2002) makes use of a dynamic program similar to ours, though specialized to the case of sequence models.

4 Consensus Decoding

Once model parameters θ are learned, we must select an appropriate decoding objective. Several new decoding approaches have been proposed recently that leverage some notion of consensus over the many weighted derivations in a translation forest. In this paper, we adopt the fast consensus decoding procedure of DeNero et al. (2009), which directly complements CoBLEU tuning. For a source sentence f, we first build a translation forest, then compute the expected count of each n-gram in the translation of f under the model. We extract a k-best list from the forest, then select the translation that yields the highest BLEU score relative to the forest's expected n-gram counts. Specifically, let $\mathrm{BLEU}(e; r)$ compute the similarity of a sentence e to a reference r based on the n-gram counts of each. When training with CoBLEU, we replace e with expected counts and maximize θ. In consensus decoding, we replace r with expected counts and maximize e.

Several other efficient consensus decoding procedures would similarly benefit from a tuning procedure that aggregates over derivations. For instance, Blunsom and Osborne (2008) select the translation sentence with highest posterior probability under the model, summing over derivations. Li et al. (2009) propose a variational approximation maximizing sentence probability that decomposes over n-grams. Tromble et al. (2008) minimize risk under a loss function based on the linear Taylor approximation to BLEU, which decomposes over n-gram posterior probabilities.

5 Experiments

We compared CoBLEU training with an implementation of minimum error rate training on two language pairs.

5.1 Model

Our optimization procedure is in principle tractable for any syntactic translation system. For simplicity, we evaluate the objective using an Inversion Transduction Grammar (ITG) (Wu, 1997) that emits phrases as terminal productions, as in (Cherry and Lin, 2007). Phrasal ITG models have been shown to perform comparably to the state-of-the art phrase-based system Moses (Koehn et al., 2007) when using the same phrase table (Petrov et al., 2008).

We extract a phrase table using the Moses pipeline, based on Model 4 word alignments generated from GIZA++ (Och and Ney, 2003). Our final ITG grammar includes the five standard Moses features, an n-gram language model, a length feature that counts the number of target words, a feature that counts the number of monotonic ITG rewrites, and a feature that counts the number of inverted ITG rewrites.

5.2 Data

We extracted phrase tables from the Spanish-English and French-English sections of the Europarl corpus, which include approximately 8.5 million words of bitext for each of the language pairs (Koehn, 2002). We used a trigram language model trained on the entire corpus of English parliamentary proceedings provided with the Europarl distribution and generated according to the ACL 2008 SMT shared task specifications.[6] For tuning, we used all sentences from the 2007 SMT shared task up to length 25 (880 sentences for Spanish and 923 for French), and we tested on the subset of the first 1000 development set sentences which had length at most 25 words (447 sentences for Spanish and 512 for French).

[6]See `http://www.statmt.org/wmt08` for details.

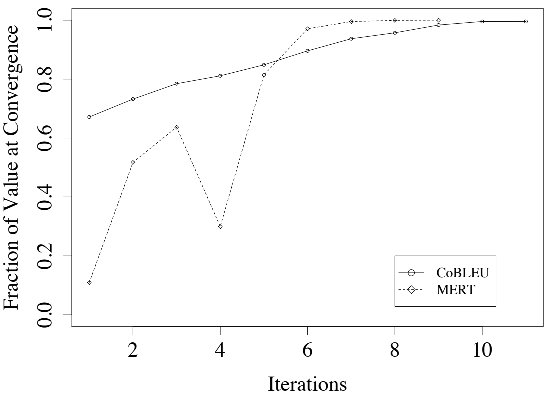


Figure 6: Trajectories of MERT and CoBLEU during optimization show that MERT is initially unstable, while CoBLEU training follows a smooth path to convergence. Because these two training procedures optimize different functions, we have normalized each trajectory by the final objective value at convergence. Therefore, the absolute values of this plot do *not* reflect the performance of either objective, but rather the smoothness with which the final objective is approached. The rates of convergence shown in this plot are not directly comparable. Each iteration for MERT above includes 10 iterations of coordinate ascent, followed by a decoding pass through the training set. Each iteration of CoBLEU training involves only one gradient step.

5.3 Tuning Optimization

We compared two techniques for tuning the nine log-linear model parameters of our ITG grammar. We maximized CoBLEU using gradient ascent, as described above. As a baseline, we maximized BLEU of the Viterbi translation derivations using minimum error rate training. To improve optimization stability, MERT used a cumulative k-best list that included all translations generated during the tuning process.

One of the benefits of CoBLEU training is that we compute expectations efficiently over an entire forest of translations. This has substantial stability benefits over methods based on k-best lists. In Figure 6, we show the progress of CoBLEU as compared to MERT. Both models are initialized from 0 and use the same features. This plot exhibits a known issue with MERT training: because new k-best lists are generated at each iteration, the objective function can change drastically between iterations. In contrast, CoBLEU converges

Consensus Decoding

	Spanish			
	Tune	Test	Δ	Br.
MERT	32.5	30.2	-2.3	0.992
CoBLEU	31.4	30.4	-1.0	0.992
MERT→CoBLEU	31.7	**30.8**	-0.9	0.992
	French			
	Tune	Test	Δ	Br.
MERT	32.5	31.1*	-1.4	0.972
CoBLEU	31.9	30.9	-1.0	0.954
MERT→CoBLEU	32.4	**31.2***	-0.8	0.953

Table 1: Performance measured by BLEU using a consensus decoding method over translation forests shows an improvement over MERT when using CoBLEU training. The first two conditions were initialized by 0 vectors. The third condition was initialized by the final parameters of MERT training. Br. indicates the brevity penalty on the test set. The * indicates differences which are not statistically significant.

smoothly to its final objective because the forests do not change substantially between iterations, despite the pruning needed to track n-grams. Similar stability benefits have been observed for lattice-based MERT (Macherey et al., 2008).

5.4 Results

We performed experiments from both French and Spanish into English under three conditions. In the first two, we initialized both MERT and CoBLEU training uniformly with zero weights and trained until convergence. In the third condition, we initialized CoBLEU with the final parameters from MERT training, denoted MERT→CoBLEU in the results tables. We evaluated each of these conditions on both the tuning and test sets using the consensus decoding method of DeNero et al. (2009). The results appear in Table 1.

In Spanish-English, CoBLEU slightly outperformed MERT under the same initialization, while the opposite pattern appears for French-English. The best test set performance in both language pairs was the third condition, in which CoBLEU training was initialized with MERT. This condition also gave the highest CoBLEU objective value. This pattern indicates that CoBLEU is a useful objective for translation with consensus decoding, but that the gradient ascent optimization is getting stuck in local maxima during tuning. This issue can likely be addressed with annealing, as described in (Smith and Eisner, 2006).

Interestingly, the brevity penatly results in French indicate that, even though CoBLEU did not outperform MERT in a statistically significant way, CoBLEU tends to find shorter sentences with higher n-gram precision than MERT.

Viterbi Decoding

	Spanish		
	Tune	Test	Δ
MERT	32.5	30.2	-2.3
MERT→CoBLEU	30.5	30.9	**+0.4**
	French		
	Tune	Test	Δ
MERT	32.0	31.0	-1.0
MERT→CoBLEU	31.7	30.9	-0.8

Table 2: Performance measured by BLEU using *Viterbi* decoding indicates that CoBLEU is less prone to overfitting than MERT.

Table 1 displays a second benefit of CoBLEU training: compared to MERT training, CoBLEU performance degrades less from tuning to test set. In Spanish, initializing with MERT-trained weights and then training with CoBLEU actually decreases BLEU on the tuning set by 0.8 points. However, this drop in tuning performance comes with a corresponding increase of 0.6 on the test set, relative to MERT training. We see the same pattern in French, albeit to a smaller degree.

While CoBLEU ought to outperform MERT using consensus decoding, we expected that MERT would give better performance under Viterbi decoding. Surprisingly, we found that CoBLEU training actually outperformed MERT in Spanish-English and performed equally well in French-English. Table 2 shows the results. In these experiments, we again see that CoBLEU overfit the training set to a lesser degree than MERT, as evidenced by a smaller drop in performance from tuning to test set. In fact, test set performance actually improved for Spanish-English CoBLEU training while dropping by 2.3 BLEU for MERT.

6 Conclusion

CoBLEU takes a fundamental quantity used in consensus decoding, expected n-grams, and trains to optimize a function of those expectations. While CoBLEU can therefore be expected to increase test set BLEU under consensus decoding, it is more surprising that it seems to better regularize learning even for the Viterbi decoding condition. It is also worth emphasizing that the CoBLEU approach is applicable to functions of expected n-gram counts other than BLEU.

Appendix: The Gradient of CoBLEU

We would like to compute the gradient of

$$\left(1 - \frac{|R|}{\sum_{i=1}^m \sum_{g_1} \mathbb{E}_\theta[c(g_1, d)|f_i]}\right)_-$$

$$+\frac{1}{4}\sum_{n=1}^{4} \ln \frac{\sum_{i=1}^m \sum_{g_n} \min\{\mathbb{E}_\theta[c(g_n, d)|f_i], c(g_n, r_i)\}}{\sum_{i=1}^m \sum_{g_n} \mathbb{E}_\theta[c(g_n, d)|f_i]}$$

To simplify notation, we use $\mathbb{E}_\theta^i[\cdot]$ to mean $\mathbb{E}_\theta[\cdot|f_i]$. We also introduce the functions

$$C_n(\theta) = \sum_{i=1}^m \mathbb{E}_\theta^i\left[\ell_n(d)\right] = \sum_{i=1}^m \sum_{g_n} \mathbb{E}_\theta^i[c(g_n, e)]$$

$$C_n^{\text{clip}}(\theta) = \sum_{i=1}^m \sum_{g_n} \min\{\mathbb{E}_\theta^i[c(g_n, d)], c(r, g_n)\}$$

$C_n(\theta)$ represents the sum of the expected counts of all n-grams or order n in all translations of the source corpus F, while $C_n^{\text{clip}}(\theta)$ represents the sum of the same expected counts, but clipped with reference counts $c(g_n, r_i)$.

With this notation, we can write our objective function CoBLEU(R, F, θ) in three terms:

$$\left(1 - \frac{|R|}{C_1(\theta)}\right)_-$$

$$+\frac{1}{4}\sum_{n=1}^{4} \ln C_n^{\text{clip}}(\theta) - \frac{1}{4}\sum_{n=1}^{4} \ln C_n(\theta)$$

We first state an identity:

$$\sum_{g_n} \frac{\partial}{\partial \theta_k} \mathbb{E}_\theta^i[c(g_n, d)] =$$

$$\mathbb{E}_\theta^i\left[c(\phi_k, d) \cdot \ell_n(d)\right] - \mathbb{E}_\theta^i\left[\ell_n(d)\right] \cdot \mathbb{E}_\theta^i[c(\phi_k, d)]$$

which can be derived by expanding the expectation on the left-hand side

$$\sum_{g_n} \sum_{d} \frac{\partial}{\partial \theta_k} P_\theta(d|f_i) c(g_n, d)$$

and substituting

$$\frac{\partial}{\partial \theta_k} P_\theta(d|f_i) =$$

$$P_\theta(d|f_i) c(\phi_k, d) - P_\theta(d|f_i) \sum_{d'} P_\theta(d'|f_i) c(\phi_k, d')$$

Using this identity and some basic calculus, the gradient $\nabla C_n(\theta)$ is

$$\sum_{i=1}^m \mathbb{E}_\theta^i\left[c(\phi_k, d) \cdot \ell_n(d)\right] - \mathbb{E}_\theta^i\left[\ell_n(d)\right] \mathbb{E}_\theta^i[c(\phi_k, d)]$$

$$\boxed{\begin{aligned} \mathrm{I}(u) &= \sum_{h \in \mathrm{IN}(u)} w(h) \left[\prod_{v \in tail(h)} \mathrm{I}(v)\right] \\ \mathrm{O}(u) &= \sum_{h \in OUT(u)} w(h) \left[\mathrm{O}(head(h)) \prod_{\substack{v \in tail(h) \\ v \neq u}} \mathrm{I}(v)\right] \end{aligned}}$$

Figure 7: Standard Inside-Outside recursions which compute $\mathrm{I}(u)$ and $\mathrm{O}(u)$. IN(u) and OUT(u) refer to the incoming and outgoing hyperedges of u, respectively. We initialize with $\mathrm{I}(u) = 1$ for all terminal forest nodes u and $\mathrm{O}(root) = 1$ for the root node. These quantities are referenced in Figure 4.

and the gradient $\nabla C_n^{\text{clip}}(\theta)$ is given by

$$\sum_{i=1}^m \sum_{g_n} \mathbb{E}_\theta^i\left[c(g_n, d) \cdot c(\phi_k, d)\right]$$

$$\cdot \mathbb{1}\left[\mathbb{E}_\theta^i[c(g_n, d)] \leq c(g_n, r_i)\right]$$

$$- \min\{\mathbb{E}_\theta^i[c(g_n, d)], c(r, g_n)\} \mathbb{E}_\theta^i[c(\phi_k, d)]$$

where $\mathbb{1}$ denotes an indicator function. At the top level, the gradient of the first term (the brevity penalty) is

$$\frac{|R| \nabla C_1(\theta)}{C_1(\theta)^2} \mathbb{1}\left[C_1(\theta) \leq |R|\right]$$

The gradient of the second term is

$$\frac{1}{4}\sum_{n=1}^{4} \frac{\nabla C_n^{\text{clip}}(\theta)}{C_n^{\text{clip}}(\theta)}$$

and the gradient of the third term is

$$-\frac{1}{4}\sum_{n=1}^{4} \frac{\nabla C_n(\theta)}{C_n(\theta)}$$

Note that, because of the indicator functions, CoBLEU is non-differentiable when $\mathbb{E}_\theta[c(g_n, d)|f_i] = c(g_n, r_i)$ or $C_n(\theta) = |R|$. Formally, we must compute a sub-gradient at these points. In practice, we can choose between the gradients calculated assuming the indicator function is 0 or 1; we always choose the latter.

References

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the Conference on Emprical Methods for Natural Language Processing*.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Syntax and Structure in Statistical Translation*.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *The Annual Conference of the Association for Computational Linguistics*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *The Annual Conference of the Association for Computational Linguistics*.

Sham Kakade, Yee Whye Teh, and Sam T. Roweis. 2002. An alternate objective function for markovian fields. In *Proceedings of ICML*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation.

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 *The Annual Conference of the Association for Computational Linguistics*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *In Proceedings of Empirical Methods in Natural Language Processing*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 108–116, Honolulu, Hawaii, October. Association for Computational Linguistics.

David Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *In Proceedings of the Association for Computational Linguistics*.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *The Conference on Empirical Methods in Natural Language Processing*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.