A Simple, Fast, and Effective Reparameterization of IBM Model 2

Chris Dyer **Victor Chahuneau** **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{cdyer,vchahune,nasmith}@cs.cmu.edu

Abstract

We present a simple log-linear reparameterization of IBM Model 2 that overcomes problems arising from Model 1's strong assumptions and Model 2's overparameterization. Efficient inference, likelihood evaluation, and parameter estimation algorithms are provided. Training the model is consistently ten times faster than Model 4. On three large-scale translation tasks, systems built using our alignment model outperform IBM Model 4.

An open-source implementation of the alignment model described in this paper is available from **http://github.com/clab/fast_align** .

1 Introduction

Word alignment is a fundamental problem in statistical machine translation. While the search for more sophisticated models that provide more nuanced explanations of parallel corpora is a key research activity, simple and effective models that scale well are also important. These play a crucial role in many scenarios such as parallel data mining and rapid large scale experimentation, and as subcomponents of other models or training and inference algorithms. For these reasons, IBM Models 1 and 2, which support exact inference in time $\Theta(|\mathbf{f}| \cdot |\mathbf{e}|)$, continue to be widely used.

This paper argues that both of these models are suboptimal, even in the space of models that permit such computationally cheap inference. Model 1 assumes all alignment structures are uniformly likely (a problematic assumption, particularly for frequent word types), and Model 2 is vastly overparameterized, making it prone to degenerate behavior on account of overfitting.[1] We present a simple log-linear reparameterization of Model 2 that avoids both problems (§2). While inference in log-linear models is generally computationally more expensive than in their multinomial counterparts, we show how the quantities needed for alignment inference, likelihood evaluation, and parameter estimation using EM and related methods can be computed using two simple algebraic identities (§3), thereby defusing this objection. We provide results showing our model is an order of magnitude faster to train than Model 4, that it requires no staged initialization, and that it produces alignments that lead to significantly better translation quality on downstream translation tasks (§4).

2 Model

Our model is a variation of the lexical translation models proposed by Brown et al. (1993). Lexical translation works as follows. Given a source sentence $\mathbf{f}$ with length n, first generate the length of the target sentence, m. Next, generate an *alignment*, $\mathbf{a} = \langle a_1, a_2, \ldots, a_m \rangle$, that indicates which source word (or null token) each target word will be a translation of. Last, generate the m output words, where each e_i depends only on f_{a_i}.

The model of alignment configurations we propose is a log-linear reparameterization of Model 2.

[1] Model 2 has independent parameters for every alignment position, conditioned on the source length, target length, and current target index.

$$\text{Given : } \mathbf{f},\ n = |\mathbf{f}|,\ m = |\mathbf{e}|,\ p_0,\ \lambda,\ \boldsymbol{\theta}$$

$$h(i, j, m, n) = -\left|\frac{i}{m} - \frac{j}{n}\right|$$

$$\delta(a_i = j \mid i, m, n) = \begin{cases} p_0 & j = 0 \\ (1 - p_0) \times \frac{e^{\lambda h(i,j,m,n)}}{Z_\lambda(i,m,n)} & 0 < j \leq n \\ 0 & \text{otherwise} \end{cases}$$

$$a_i \mid i, m, n \sim \delta(\cdot \mid i, m, n) \qquad 1 \leq i \leq m$$
$$e_i \mid a_i, f_{a_i} \sim \theta(\cdot \mid f_{a_i}) \qquad 1 \leq i \leq m$$

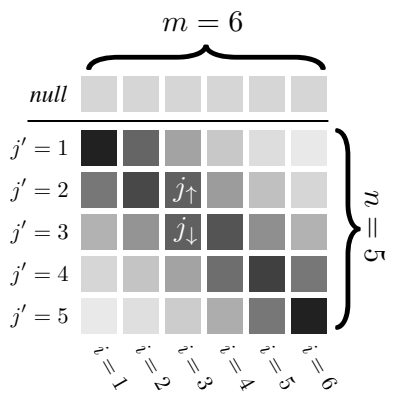

Figure 1: Our proposed generative process yielding a translation **e** and its alignment **a** to a source sentence **f**, given the source sentence **f**, alignment parameters p_0 and λ, and lexical translation probabilities $\boldsymbol{\theta}$ (left); an example visualization of the distribution of alignment probability mass under this model (right).

Our formulation, which we write as $\delta(a_i = j \mid i, m, n)$, is shown in Fig. 1.[2] The distribution over alignments is parameterized by a null alignment probability p_0 and a precision $\lambda \geq 0$ which controls how strongly the model favors alignment points close to the diagonal. In the limiting case as $\lambda \to 0$, the distribution approaches that of Model 1, and, as it gets larger, the model is less and less likely to deviate from a perfectly diagonal alignment. The right side of Fig. 1 shows a graphical illustration of the alignment distribution in which darker squares indicate higher probability.

3 Inference

We now discuss two inference problems and give efficient techniques for solving them. First, given a sentence pair and parameters, compute the marginal likelihood and the marginal alignment probabilities. Second, given a corpus of training data, estimate likelihood maximizing model parameters using EM.

3.1 Marginals

Under our model, the marginal likelihood of a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ can be computed exactly in time $\Theta(|\mathbf{f}| \cdot |\mathbf{e}|)$. This can be seen as follows. For each position in the sentence being generated, $i \in [1, 2, \ldots, m]$, the alignment to the source and its translation is independent of all other translation and alignment decisions. Thus, the probability that the ith word of **e** is e_i can be computed as:

$$p(e_i, a_i \mid \mathbf{f}, m, n) = \delta(a_i \mid i, m, n) \times \theta(e_i \mid f_{a_i})$$
$$p(e_i \mid \mathbf{f}, m, n) = \sum_{j=0}^{n} p(e_i, a_i = j \mid \mathbf{f}, m, n).$$

We can also compute the posterior probability over alignments using the above probabilities,

$$p(a_i \mid e_i, \mathbf{f}, m, n) = \frac{p(e_i, a_i \mid \mathbf{f}, m, n)}{p(e_i \mid \mathbf{f}, m, n)}. \quad (1)$$

Finally, since all words in **e** (and their alignments) are conditionally independent,[3]

$$p(\mathbf{e} \mid \mathbf{f}) = \prod_{i=1}^{m} p(e_i \mid \mathbf{f}, m, n)$$
$$= \prod_{i=1}^{m} \sum_{j=0}^{n} \delta(a_i \mid i, m, n) \times \theta(e_i \mid f_{a_i}).$$

[2] Vogel et al. (1996) hint at a similar reparameterization of Model 2; however, its likelihood and its gradient are not efficient to evaluate, making it impractical to train and use. Och and Ney (2003) likewise remark on the overparameterization issue, removing a single variable of the original conditioning context, which only slightly improves matters.

[3] We note here that Brown et al. (1993) derive their variant of this expression by starting with the joint probability of an alignment and translation, marginalizing, and then reorganizing common terms. While identical in implication, we find the direct probabilistic argument far more intuitive.

3.2 Efficient Partition Function Evaluation

Evaluating and maximizing the data likelihood under log-linear models can be computationally expensive since this requires evaluation of normalizing partition functions. In our case,

$$Z_\lambda(i, m, n) = \sum_{j'=1}^{n} \exp \lambda h(i, j', m, n).$$

While computing this sum is obviously possible in $\Theta(|\mathbf{f}|)$ operations, our formulation permits exact computation in $\Theta(1)$, meaning our model can be applied even in applications where computational efficiency is paramount (e.g., MCMC simulations). The key insight is that the partition function is the (partial) sum of two geometric series of unnormalized probabilities that extend up and down from the probability-maximizing diagonal. The closest point on or above the diagonal $j_\uparrow$, and the next point down $j_\downarrow$ (see the right side of Fig. 1 for an illustration), is computed as follows:

$$j_\uparrow = \left\lfloor \frac{i \times n}{m} \right\rfloor, \qquad j_\downarrow = j_\uparrow + 1.$$

Starting at $j_\uparrow$ and moving up the alignment column, as well as starting at $j_\downarrow$ and moving down, the unnormalized probabilities decrease by a factor of $r = \exp \frac{-\lambda}{n}$ per step.

To compute the value of the partition, we only need to evaluate the unnormalized probabilities at $j_\uparrow$ and $j_\downarrow$ and then use the following identity, which gives the sum of the first ℓ terms of a geometric series (Courant and Robbins, 1996):

$$s_\ell(g_1, r) = \sum_{k=1}^{\ell} g_1 r^{k-1} = g_1 \frac{1 - r^\ell}{1 - r}.$$

Using this identity, $Z_\lambda(i, m, n)$ can be computed as

$$s_{j_\uparrow}(e^{\lambda h(i, j_\uparrow, m, n)}, r) + s_{n - j_\downarrow}(e^{\lambda h(i, j_\downarrow, m, n)}, r).$$

3.3 Parameter Optimization

To optimize the likelihood of a sample of parallel data under our model, one can use EM. In the E-step, the posterior probabilities over alignments are computed using Eq. 1. In the M-step, the lexical translation probabilities are updated by aggregating these as counts and normalizing (Brown et al., 1993). In the experiments reported in this paper, we make the further assumption that $\theta_f \sim \text{Dirichlet}(\boldsymbol{\mu})$ where $\mu_i = 0.01$ and approximate the posterior distribution over the θ_f's using a mean-field approximation (Riley and Gildea, 2012).[4]

During the M-step, the λ parameter must also be updated to make the E-step posterior distribution over alignment points maximally probable under $\delta(\cdot \mid i, m, n)$. This maximizing value cannot be computed analytically, but a gradient-based optimization can be used, where the first derivative (here, for a single target word) is:

$$\begin{aligned} \nabla_\lambda \mathcal{L} = \mathbb{E}_{p(a_i \mid e_i, \mathbf{f}, m, n)} \left[h(i, a_i, m, n) \right] \\ - \mathbb{E}_{\delta(j' \mid i, m, n)} \left[h(i, j', m, n) \right] \end{aligned} \quad (2)$$

The first term in this expression (the expected value of h under the E-step posterior) is fixed for the duration of each M-step, but the second term's value (the derivative of the log-partition function) changes many times as λ is optimized.

3.4 Efficient Gradient Evaluation

Fortunately, like the partition function, the derivative of the log-partition function (i.e., the second term in Eq. 2) can be computed in constant time using an algebraic identity. To derive this, we observe that the values of $h(i, j', m, n)$ form an arithmetic sequence about the diagonal, with common difference $d = -1/n$. Thus, the quantity we seek is the sum of a series whose elements are the products of terms from an arithmetic sequence and those of the geometric sequence above, divided by the partition function value. This construction is referred to as an arithmetico-geometric series, and its sum may be computed as follows (Fernandez et al., 2006):

$$\begin{aligned} t_\ell(g_1, a_1, r, d) &= \sum_{k=1}^{\ell} \left[a_1 + d(k-1) \right] g_1 r^{k-1} \\ &= \frac{a_\ell g_{\ell+1} - a_1 g_1}{1 - r} + \frac{d \left(g_{\ell+1} - g_1 r \right)}{(1 - r)^2}. \end{aligned}$$

In this expression r, the g_1's and the ℓ's have the same values as above, $d = -1/n$ and the a_1's are

[4] The μ_i value was fixed at the beginning of experimentation by minimizing the AER on the 10k sentence French-English corpus discussed below.

equal to the value of h evaluated at the starting indices, $j_\uparrow$ and $j_\downarrow$; thus, the derivative we seek at each optimization iteration inside the M-step is:

$$\begin{aligned}\nabla_\lambda \mathcal{L} =& \mathbb{E}_{p(a_i|e_i,\mathbf{f},m,n)}\left[h(i,a_i,m,n)\right] \\ &- \frac{1}{Z_\lambda}(t_{j_\uparrow}(e^{\lambda h(i,j_\uparrow,m,n)}, h(i,j_\uparrow,m,n), r, d) \\ &+ t_{n-j_\downarrow}(e^{\lambda h(i,j_\downarrow,m,n)}, h(i,j_\uparrow,m,n), r, d)).\end{aligned}$$

4 Experiments

In this section we evaluate the performance of our proposed model empirically. Experiments are conducted on three datasets representing different language typologies and dataset sizes: the FBIS Chinese-English corpus (LDC2003E14); a French-English corpus consisting of version 7 of the Europarl and news-commentary corpora;[5] and a large Arabic-English corpus consisting of all parallel data made available for the NIST 2012 Open MT evaluation. Table 1 gives token counts.

We begin with several preliminary results. First, we quantify the benefit of using the geometric series trick (§3.2) for computing the partition function relative to naïve summation. Our method requires only 0.62 seconds to compute all partition function values for $0 < i, m, n < 150$, whereas the naïve algorithm requires 6.49 seconds for the same.[6]

Second, using a 10k sample of the French-English data set (only 0.5% of the corpus), we determined 1) whether p_0 should be optimized; 2) what the optimal Dirichlet parameter α is; and 3) whether the commonly used "staged initialization" procedure (in which Model 1 parameters are used to initialize Model 2, etc.) is necessary for our model. First, like Och and Ney (2003) who explored this issue for training Model 3, we found that EM tended to find poor values for p_0, producing alignments that were overly sparse. By fixing the value at $p_0 = 0.08$, we obtained minimal AER. Second, like Riley and Gildea (2012), we found that small values of α improved the alignment error rate, although the impact was not particularly strong over large ranges of α. Finally, we (perhaps surprisingly) found that the standard staged initialization procedure was *less effective* in terms of AER than simply initializing our model with uniform translation probabilities and a small value of λ and running EM. Based on these observations, we fixed $p_0 = 0.08$, $\alpha = 0.01$, and set the initial value of λ to 4 for the remaining experiments.[7]

Table 1: CPU time (hours) required to train alignment models in one direction.

Language Pair	Tokens	Model 4	Log-linear
Chinese-English	17.6M	2.7	**0.2**
French-English	117M	17.2	**1.7**
Arabic-English	368M	63.2	**6.0**

Table 2: Alignment quality (AER) on the WMT 2012 French-English and FBIS Chinese-English data with different models and different estimators of θ_f.

Model	Estimator	FR-EN	ZH-EN
Model 1	EM	29.0	56.2
Model 1	∼Dir	26.6	53.6
Model 2	EM	21.4	53.3
Log-linear	EM	18.5	46.5
Log-linear	∼Dir	16.6	**44.1**
Model 4	EM	**10.4**	45.8

Table 3: Translation quality (BLEU) as a function of alignment type.

Language Pair	Model 4	Log-linear
Chinese-English	34.1	**34.7**
French-English	27.4	**27.7**
Arabic-English	54.5	**55.7**

We next compare the alignments produced by our model to the Giza++ implementation of the standard IBM models using the default training procedure and parameters reported in Och and Ney (2003). Our model is trained for 5 iterations using the procedure described above (§3.3). The algorithms are compared in terms of (1) time required for training;

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

[6] While this computational effort is a small relative to the total cost in EM training, in algorithms where λ changes more rapidly, for example in Bayesian posterior inference with Monte Carlo methods (Chahuneau et al., 2013), this savings can have substantial value.

[7] As an anonymous reviewer pointed out, it is a near certainty that tuning of these hyperparameters for each alignment task would improve results; however, optimizing hyperparameters of alignment models is quite expensive. Our intention is toshow that it is possible to obtain reasonable (if not optimal) results *without* careful tuning.

(2) alignment error rate (AER, lower is better);[8] and (3) translation quality (BLEU, higher is better) of hierarchical phrase-based translation system that used the alignments (Chiang, 2007). Table 1 shows the CPU time in hours required for training (one direction, English is generated). Our model is at least $10\times$ faster to train than Model 4. Table 3 reports the differences in BLEU on a held-out test set. Our model's alignments lead to consistently better scores than Model 4's do.[9]

5 Conclusion

We have presented a fast and effective reparameterization of IBM Model 2 that is a compelling replacement for for the standard Model 4. Although the alignment quality results measured in terms of AER are mixed, the alignments were shown to work exceptionally well in downstream translation systems on a variety of language pairs.

Acknowledgments

This work was sponsored by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533.

References

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

V. Chahuneau, N. A. Smith, and C. Dyer. 2013. Knowledge-rich morphological priors for Bayesian language models. In *Proc. NAACL*.

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

R. Courant and H. Robbins. 1996. The geometric progression. In *What Is Mathematics?: An Elementary Approach to Ideas and Methods*, pages 13–14. Oxford University Press.

P. A. Fernandez, T. Foregger, and J. Pahikkala. 2006. Arithmetic-geometric series. PlanetMath.org.

[8] Our Arabic training data was preprocessed using a segmentation scheme optimized for translation (Habash and Sadat, 2006). Unfortunately the existing Arabic manual alignments are preprocessed quite differently, so we did not evaluate AER.

[9] The alignments produced by our model were generally sparser than the corresponding Model 4 alignments; however, the extracted grammar sizes were sometimes smaller and sometimes larger, depending on the language pair.

N. Habash and F. Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proc. of NAACL*.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

D. Riley and D. Gildea. 2012. Improving the IBM alignment models using Variational Bayes. In *Proc. of ACL*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of COLING*.