# Parameter Agreement Training with and without Parallel Data

## Abstract

## 1 Introduction

Alignment is a crucial subtask for many larger NLP problems: Word alignments are the backbone for extracting translation rules used in statistical machine translation; phoneme alignments are used to transliterate Japanese katakana to English and vice-versa. The goal is to align entities, which could be words or phonemes, between two sequences, which could be pairs of source and target sentences or pairs of phoneme sequences. The popular models used in these tasks are both *generative* and *directed*, that is, they define a directed sequence of events by which one side of the pair produces the other side. For example, the IBM models for word alignment Brown et al. (1993) describe a stochastic process by which the target sentence is generated from the source sentence via word alignments. These processes can sometimes produce different alignments, depending on the direction of generation. However, the directionality may come with a price, as it impose restriction on the inference space - in the word alignment models (IBM 1 or 2 and HMM), only one-to-many word alignments (target to source) are possible, which may not fit well for some language pairs (for example, German compound target words usually arise from multiple English source words). To remedy this, we resort to ad-hoc alignment symmetrization approaches , such as grow-diag-final-and (Koehn et al., 2003), *after* training. However, while post-processing resolves some of the problems, independent training can still lead the two models to divergent local maxima; ones whose disagreeing inferences cannot be fixed by post-processing techniques. A second problem for training alignment models is the need for parallel data. For example, in Knight and Graehl (1998), the authors use (some number)

of parallel English-Katana sequences to training phoneme alignment models. While large amounts of parallel data needed for word alignment is readily available for many language pairs, acquiring parallel phoneme sequences for transliteration can be very challenging.

There has been previous work on solving the problem of model disagreement in word alignment, such as alignment by agreement (Liang et al., 2006) (ABA). They suggest to jointly train the two directional models, each maximizing its own data likelihood (as would be done independently) as well as a model coupling term that encodes some measure of agreement between the models. Their suggested agreement measure further exploits the observed parallel training data by rewarding similarity between the two alignment posteriors (one for each direction) computed on each training sentence pair. Indeed, their experimental results show that training with posterior agreement leads to better performing models (reduced AER). However, in Liang et al. (2006), the authors do not optimize a clear objective function and their approach cannot be applied to training alignment models without parallel data Ravi and Knight (2009) alleviate the reliance on parallel data and improve monolingual phoneme alignment over the baseline, but their approach does not encourage model agreement, which can improve alignment accuracy.

In our paper, we present a *single* approach that achieves both model agreement during training, and can be used in non-parallel settings. Our approach, Parameter Agreement Training (PAT), operates with a different measure of agreement in mind (Section **??**)). The general idea can be easily illustrated in the word alignment setting: Consider the two directional word alignment models' translation tables. For any bilingual pair of words $e$ and $f$, it seems reasonable that $t(f \mid e)$ is high/low whenever $t(e \mid f)$ is high/low. This

insert number

intuition follows simply since word translation is generally invertible [1]. With the above in mind, we designe an agreement measure that encourages agreement in the magnitudes of the parameters of the models in each direction. Our resulting objective function can be optimized using EM, where the E-step remains unchanged and the M-step reduces to an instance of convex programming. Furthermore, since our approach encourages agreement between the model parameters, it can be applied in non-parallel settings such as unsupervised phoneme alignment with monolingual data. Using our approach, we are able to significantly improve over Ravi and Knight (2009), achieving a $50\%$ reduction in error. For word alignment, our results compare favorably with results from previous approaches for model agreement.

## 2 Background

### 2.1 Generative Models

We are concerned with learning generative models that describe transformations of a source-language entity $e$ to a target-language entity $f$. We consider two different settings for these models.

In the parallel data setting, both source and target entities are observed and the generative story describes how $f$ can arise from $e$. Denoting $x = (e, f)$ the model takes the following form:

$$P_\Theta(x) = p(f \mid e) = \sum_a p(a, f \mid e), \quad (1)$$

where $a$ denotes a hidden variable that corresponds to unknown choices in the generative story (in our case, $a$ is an alignment, see next subsection) and $\Theta$ denotes the parameters of the model.

In the decipherment setting, only the target entity $f$ is observed and the source entity is considered hidden. Letting $x = (f)$ the decipherment model takes the form:

$$P_\Theta(x) = p(f) = \sum_e p(e) \sum_a p(a, f \mid e) \quad (2)$$

According to the generative story that follows from this equation, the observed entity $f$ can arise from any source entity $e$ by first selecting $e \sim p(e)$ and then proceeding according to the parallel data generative story.

---

[1]The word $e$ should be in the set of translations of the translation of $e$

Training these models entails the same objective - maximizing the observed data log-likelihood $L(\Theta \mid X)$:

$$\max_\Theta \ L(\Theta \mid X) = \max_\Theta \ \log \prod_n P_\Theta(x_n)$$
$$= \max_\Theta \ \sum_n \log P_\Theta(x_n)$$

Where $X = \{e_n, f_n\}_{n=1}^N$ in the parallel data setting and $X = \{f_n\}_{n=1}^N$ in the decipherment setting.

Next, we make these generative stories concrete by looking at $p(a, f \mid e)$ and $p(e)$ (if necessary) in two sequence alignment tasks - word alignment and back-transliteration decipherment.

### 2.2 Parallel Data: Word Alignment

Word alignment is one of the major building blocks in the conventional machine translation pipeline. In word alignment, the entity $e = (e_1, \ldots, e_I)$ represents a sentence in the source language and $f = (f_1, \ldots, f_J)$ represents a sentence in the target language. The generative story for the transformation of $e$ to $f$ follows according to Eq. 1 where the hidden variable $a = (a_1, \ldots, a_J)$ corresponds to a sequence of $J$ indices and encodes an asymmetric alignment under which the $j$th target word is aligned to the $a_j$th source word (that is, under the generative story, $f_j$ is thought to be generated from $e_{a_j}$).

The HMM word alignment model is considered both computationally efficient and effective. Since the HMM formulation is not convex, it is common practice to initialized word alignment using the simpler IBM 1 word alignment model. Both these models assign the following conditional probability for a fixed sentence pair $(e, f)$ and a fixed alignment $a$:

$$p(a, f \mid e) = \prod_{j=1}^J t(f_j \mid e_{a_j}) \cdot a(a_j \mid a_{j'}) \quad (3)$$

That is, both models are parameterized using two tables of conditional probability distributions $\Theta = (t, a)$. The translation table $t(f_j \mid e_i)$ encodes the probability of translating $e_{a_j}$ to $f_j$ and the distortion table $a$ is a model dependent probability table that quantifies the probability of aligning the $j$th target position with the $a_j$th source position, possibly given a previous alignment decision.

Specifically, IBM model 1 uses a degenerate distortion table that assigns a uniform probability to all possible source positions and the HMM model encodes distortions using a first-order dependency on the previously non null-aligned position $a_{j'}$. The exact details behind the HMM distortion parameters are implementation specific and therefore beyond the scope of this paper. See * for details of the GIZA++ implementation and ** for those of the ABA implementation.

### 2.3 Decipherment: Back-Transliteration

Transliteration is a mapping of terms between writing systems of different languages. Usually, the mapping tries to preserve the sound of a term as it is uttered in the original language. For example, the word "computer" in English is transliterated to Japanese as "ko n pyu u ta a" (in Romaji). The process restoring transliterated foreign words to their original script is called *back-transliteration*.

Romaji or Katakana?

Knight and Graehl (1998) suggest a generative story for transliteration of English terms $\mathbf{w}$ into Japanese terms $\mathbf{k}$ (see top of Figure 1):

1. First, a word $\mathbf{w}$ is generated according to an English language model $p(\mathbf{w})$.

2. $\mathbf{w}$ is then mapped to an English phonemes sequence $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_I)$ according to a pronunciation model.

3. The phoneme sequence $\mathbf{e}$ is mapped to a sequence of Japanese phonemes $\mathbf{j} = (\mathbf{j}_1, \ldots, \mathbf{j}_J)$ according to a phoneme mapping model $p(\mathbf{j} \mid \mathbf{e})$.

4. Finally, the Japanese phoneme sequence $\mathbf{j}$ is mapped to a Japanese word $\mathbf{k}$ in Katakana.

Each model corresponds to an FST that can be constructed and trained independently. In particular, models 1,3,4 are estimated using monolingual data only, whereas the phoneme mapping model $p(\mathbf{j} \mid \mathbf{e})$ is trained over a parallel phoneme corpus $\{(\mathbf{e}_n, \mathbf{j}_n)\}$ in which each English phoneme is restricted to map to one or two Japanese phonemes.

However, collecting parallel data is a time consuming and laborious process. To combat the need for parallel data, Ravi and Knight (2009) suggest a to estimate model 2 in the decipherment setting, for which only Japanese words $K = \{\mathbf{k}_n\}_{n=1}^{|K|}$

need be collected. Assuming one-to-one transformations for models 2 and 4, the generative process for transliteration is identical to that of Eq. 2:

$$
\begin{aligned}
p(\mathbf{k}) = p(\mathbf{j}) &= \sum_{\mathbf{w}} p(\mathbf{j}, \mathbf{w}) \\
&= \sum_{\mathbf{e}} p(\mathbf{e}) p(\mathbf{j} \mid \mathbf{e}) \\
&= \sum_{\mathbf{e}} p(\mathbf{e}) \sum_{\mathbf{a}} p(\mathbf{a}, \mathbf{j} \mid \mathbf{e})
\end{aligned}
$$

Where $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_J)$ is a *monotone* alignment sequence (i.e., $\mathbf{a}_j \leq \mathbf{a}_{j+1}$ for all $1 \leq j < |J|$).

## 3 Parameter Agreement Training

We now describe our method, which we call Parameter Agreement Training. In translation, we believe that if a word $e$ can be translated to a word $f$, then $f$ can be translated to $e$. This intuition holds for other tasks such as word or phoneme alignment. Thus, a desirable form of model agreement is parameter sparsity agreement:

$$
P_0(f \mid e) = 0 \text{ iff } P_1(e \mid f) = 0.
$$

### 3.1 Regularization

To encourage this form of parameter agreement we add a regularizer that couples the two models together:

$$
\max_{P_0, P_1} L(P_0|X) + L(P_1|X) + \lambda R(P_0, P_1) \quad (4)
$$

where $\lambda \geq 0$ is a regularization coefficient to be tuned and

$$
R(P_0, P_1) = \sum_{e,f} \sqrt{P_0(f \mid e) P_1(e \mid f)}. \quad (5)
$$

This regularizer has two attractive properties:

- It is concave on the probability simplex which lends to an efficiently solvable convex optimization program when the log-likelihood terms are themselves concave. (concavity proof: each term in the summation is concave over the region $[0, 1]^2$ and the sum of concave functions is concave)

- It encourages parameter sparsity agreement - If $P_0(e \mid f) = 0$ but $P_1(f \mid e) > 0$, shifting weight away from $P_1(f \mid e)$ could only increase the regularizer value.
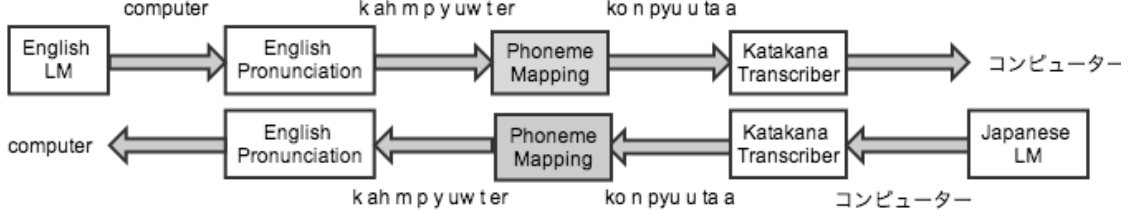
Figure 1: The transliteration generative story as a cascade of FSTs . **Top:** transliteration of the word "computer" to Japanese. **Bottom:** the reverse process. Each box represents a transducer. In PAT for deciphering transliterations, the two cascades are jointly trained to maximize both the data log-likelihood and the parameter agreement of the two phoneme mapping models (shaded).

## 3.2 Optimization Procedure

We take an EM approach for optimizing the objective function. In the E step, expected counts for each event $P_0(f \mid e)$ or $P_1(e \mid f)$ are collected separately. In the M step, the regularized complete data log-likelihood function is constructed from the expected counts, and is then maximized via projected gradient ascent: At each time step, the models $P_0, P_1$ are updated according to the gradient of the regularized objective function and projected back onto the probability simplex. We use simple stopping conditions based on objective function convergence or a fixed number of iterations.

Inspecting the regularizer's gradient we see that each update brings the two models closer:

$$\frac{\partial R}{\partial P_0} = \frac{1}{2}\sqrt{\frac{P_1}{P_0}} \qquad \frac{\partial R}{\partial P_1} = \frac{1}{2}\sqrt{\frac{P_1}{P_0}}.$$

# 4 Transliteration

## 4.1 Parameter Agreement Training

In PAT, the model corresponding to the inverse generative story is also used. This inverse process is illustrated at the bottom of Figure 1.

The training data for the reverse direction consists of set of English pronunciations $E = \{e_n\}_{n=1}^{|E|}$ and independent training maximizes:

$$L(E) = \sum_{e \in E} \log \sum_{j} P_1(e \mid j) \cdot P_1(j) \quad (6)$$

To apply PAT, we maximize the joint regularized objective function:

$$L(J) + L(E) + \lambda R(P_0(j|e), P_1(e|j)) \quad (7)$$

Once the two models are trained, Japanese words can be decoded using either model. In practice we followed Ravi and Knight and used the $P_0$.

## 4.2 Decipherment Experiments

Ravi and Knight (Ravi and Knight, 2009) compare back-transliteration whole-name error rates (WNER) on a list of 100 US senator names (In WNER, a decoding is correct if both the first and last name are decoded correctly). They report 40% error rate in the parallel setting, and a 73% error rate when using their best decipherment setting.

In order to compare PAT against independent training, we reproduced the decipherment setting (Ravi and Knight, 2009). For the FST machinery, we prepared:

- An English pronunciation FST from the CMU pronunciation dictionary. A Japanese pronunciation FST that was hand built.

- Pronounceable word unigram LMs, constructed over the top 40K most frequent capitalized words in the gigaword corpus and the top 25K most frequent Katakana terms in the Japanese news 2005-2008 Leipzig corpora

- for $P_0$, We implemented an FST similar to the best setting reported by (Ravi and Knight, 2009). We note that $P_0$ is restricted to either 1-to-1 or 1-to-2 phoneme mappings. $P_1$ was taken to be the reverse FST.

For training data we take $E$ and $J$ to be the pronunciations of the top 50% most frequent terms in their language models. Using the entire set lead to poor baseline results, possibly since uncommon English terms do not get transliterated, and uncommon Japanese may be borrowed from a language other than English. It note that $E$ and $J$ are far from being parallel, since they were collected over non parallel corpora.
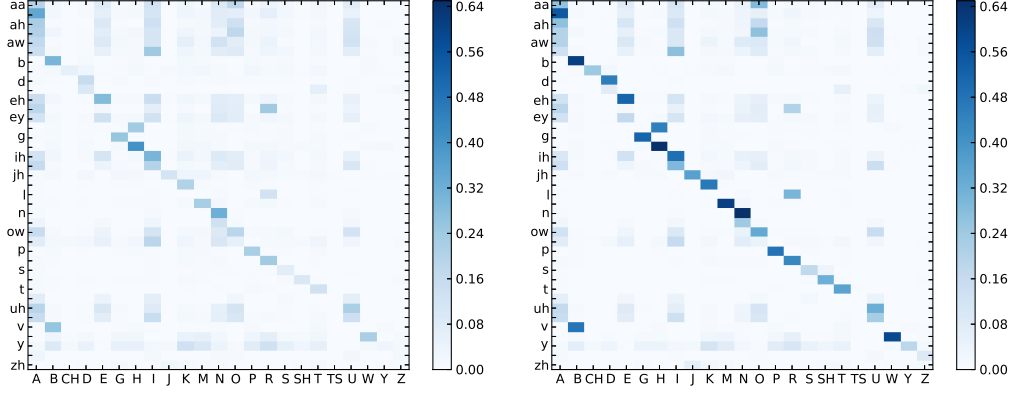
Figure 2: Compared to independent training. PAT (right) learns sparser, peaked models.

In training time we maximized the objective (Equation 7) with respect to the phoneme mapping models $P_0, P_1$ and over $\lambda \in \{1, 2, 3, 4\}$. Training was stopped after 15 EM iterations.

The development set consistent of 50 frequent Japanese words and their English origin. We selected the NER minimizing model over the 15 iterations, $\lambda$ and a weight $\alpha \in \{1, 2, 3\}$ that exponentiated the $P_0$ parameters before decoding.

We compiled our own list of 100 US statesmen and used it as a test set. Table 4.2 reports WNER, average normalized edit distance (NED) and the number of $P_0$ parameters with value greater than 0.01 ($P_0 > 0.01$) as an indication of model sparsity.

|              | WNER | NED   | $P_0 > 0.01$ |
|--------------|------|-------|--------------|
| Independent  | 67%  | 23.18 | 649          |
| PAT (our)    | 59%  | 17.35 | 421          |
| parallel data| 43%  | 10.8  | 152          |

Table 1: PAT reduces error rates and learns sparser models.

Further evidence that PAT learns sparser models can be seen in Figure 2 which compares the 1-1 phoneme mapping of the selected $P_0$ decipherment models.

## 5 Word Alignment

Given a *target* French sentence, $\mathbf{f_1^J} = f_1, f_2, \ldots, f_j, \ldots, f_J$, and a *source* English sentence, $\mathbf{e_1^I} = e_1, e_2, \ldots, e_i, \ldots, e_I$, word alignment models describe the generative process by which the source sentence creates

the target ($E \rightarrow F$) ausing latent alignments $\mathbf{a_1^J} = a_1, a_2, \ldots, a_j, \ldots, a_J$. The alignment variable $a_j$ specifies the English word $e_{a_j}$ that the French word $f_j$ is aligned to.

The IBM Models 1–2 and the HMM alignment models have two sets of parameters, the translation probabilities $P_t(f_j \mid e_{a_j})$ and distortion probabilities, $P_d(a_j \mid a_{j-1}, j)$. These models differ in their implementation and estimation of the distortion probabilities, but share the same translation probabilities. The general form of the joint probability of a target sentence and alignment given the source sentence is:

$$P(\mathbf{f_1^J}, \mathbf{a_1^J} \mid \mathbf{e_1^I}) = \prod_{j=1}^{J} P_d(a_j \mid a_{j-1}, j) P_t(f_j \mid e_{a_j})$$

(8)

During training, we typically estimate the parameters that maximize

$$\log \sum_{a_1^J} P(\mathbf{f_1^J}, \mathbf{a_1^J} \mid \mathbf{e_1^I}) = \log P(\mathbf{f_1^J} \mid \mathbf{e_1^I}) \quad (9)$$

.

For details on parameter estimation of these models, and how to deal with empty words, the reader can refer to Och and Ney (2003). For translation, we use the *Viterbi* alignments,

$$\hat{\mathbf{a_1^J}} = \arg\max_{\mathbf{a_1^J}} P(\mathbf{f_1^J}, \mathbf{a_1^J} \mid \mathbf{e_1^I}) \quad (10)$$

The generative story allows each target word $f_j$ to align to only one source word $e_{a_j}$. This can be problematic when the target language has compound words that must align to two or more source

language words. The standard solution is to train another model in the *reverse* direction ($F \rightarrow E$), $P(\mathbf{e_1^L}, \mathbf{a_1^L} \mid \mathbf{f_1^J})$ and then symmetrize the Viterbi alignments in both directions using heuristics like *grow-diag-final* (Koehn et al., 2003). Symmetrization remedies some of the mistakes that the independently trained models make, garbage collection in particular (Liang et al., 2006). However, the $E \rightarrow F$ and $F \rightarrow E$ models do not communicate during training, which could guide the parameters in the wrong direction. In Liang et al. (2006) and Ganchev et al. (2010), the authors show that training the alignment models jointly can improve alignment quality. In Liang et al. (2006), the authors encourage the probabilities over alignments in each direction to agree, per sentence pair. However, this renders the inference intractable and the authors have to resort to an approximation, without specifying the objective that the approximate procedure ends up optimizing. In contrast, we optimize a clear objective which improves in every iteration. In Ganchev et al. (2010), the authors optimize a clear objective which encourages agreement between the alignments. However, their optimization procedure is expensive and scaling to large datasets is a challenge. .

need to make sure that it doesn't scale. I suspect it doesn't

| | de-en | cz-en | $P_0 > 0.01$ |
|---|---|---|---|
| Liang et al. (2006) | 71% | 70% | - |
| PAT (our) | 71% | 69% | - |

Table 2: Alignment F-score improves.

## 6 Conclusion

We have presented Parameter Agreement Training (PAT) - an approach for jointly training two conditional models that encourages agreement on parameter values. We derived an optimization procedure based on the EM framework, and argued that it is efficiently solvable due to the concavity of our agreement regularizer. In contrast to previous work on agreement training, we have demonstrated that our approach can be successfully applied even without parallel data.

## References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Comput. Linguist.*, 24(4):599–612, December.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*.

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

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 37–45, Stroudsburg, PA, USA. Association for Computational Linguistics.