Parameter Agreement Training with and without Parallel Data

Abstract

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 in transliteration from Japanese katakana to English and vice-versa. 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 sequence pair produces the other side. For example, the IBM models for word alignment **?**), describe a stochastic process by which the target sentence is generated from the source sentence via word alignments. For word alignment, the generative processes allow a target word to align to *at most* one source word which causes the generative processes to produce different alignments in the source-target and target-source directions. For both word alignment and phoneme alignment, the models in each direction are trained independently of each other. While this is not so much a problem for phoneme alignment, word alignment models end up producing different alignments in each direction and to correct for the errors that arise in each direction, we typically resort to adhoc alignment symmetrization approaches (**?**). A second problem for training alignment models is the need for parallel data. For example, in **?**), the authors use of parallel English-Katana sequences to train the 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 (**?**) and training phoneme alignment models from large amounts of monolingual data (**?**). However, in **?**), the authors do not optimize a clear objective function and their approach cannot be applied to training alignment models without parallel data. **?**) 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*, encourages model agreement by adding a regularization term that penalizes disagreement between the (magnitudes of the ?) model parameters in each direction. Our regularizer is convex, allowing us to efficiently optimize the objective function with the EM algorithm. We apply our techniques to word alignment and phoneme alignment. Using our approach, we are able to significantly improve over **?**). In word alignment, our results compare favorably with results from previous approaches for model agreement.

insert number

1 Introduction

Many tasks in machine translation (MT) are first approached by modeling the transfer of information from one language to another (say, English to French). The resulting directional models often follow a simple generative story that explains how source language entities could have generate corresponding target language entities. For example, the IBM word alignment models (**?**) explain how strings composing an English sentence can be translated and rearranged (distorted) into strings composing a French sentence and pronunciation Finite State Transducer (FST) can be used to explain how names in English undergo phonetic changes as they are uttered in French.

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). For this reason, it is common practice to produce word alignment inferences using models trained in both directions and then apply a post-processing symmetrization technique (such as grow-diag-final-and). 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.

In Alignment by Agreement (ABA), Liang et al. (2006) 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). At the same time however, the ABA formulation cements the dependency on parallel data thus limiting the applicability of their method to other settings.

In this paper, we present Parameter Agreement Training (PAT); a method that operates with a different measure of agreement in mind (Section **??**)) Remaining in context of word alignment (as we have so far), the general idea is as follows: 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 intuition follows simply since word translation is generally invertible [1]. With the above in mind, we designed an agreement measure that encourages agreement in the two directional models parameters magnitudes. 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, our new agreement measure is free of the parallel data dependency which in turn broadens the range of settings our method can be applied to beyond conventional word alignment. In particular, our method can be applied in the challenging decipherment setting, under which no parallel data is provided.

We conduct two sets of experiments (Section **??**): (1) Our word alignment experiments (with parallel data) show comparable F- and BLEU score improvements to those of ABA on various language pairs. Combining the phrase-tables of both ABA and PAT leads to further increase in BLEU score. (2) In the decipherment settings (without parallel data) we repeat the Japanese to English back-transliteration experiments as described by Ravi and Knight (2009?). PAT nearly splits the difference between the baseline (Ravi and Knight's formulation) and a method trained using parallel data. Finally, ... ?

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:

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

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 Θ 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:

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

According to the generative story that follows from this equation, the observed entity f can arise

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

from any source entity e by first selecting $e \sim p(e)$ and then proceeding according to the parallel data generative story.

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 jth target word is aligned to the a_jth 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:

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

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 jth target position with the a_jth 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 (**?**) 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 (**?**) suggest a to estimate model 2 in the decipherment setting, for

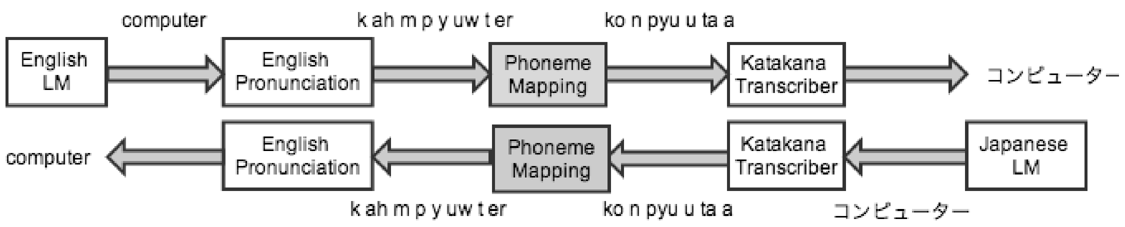

Figure 1: The transliteration generative story as a cascade of FSTs . **Top:** transliteration of the word "computer" to Japanese. **Bottom:** the reverse process. Each box represents a transducer. In PAT for deciphering transliterations, the two cascades are jointly trained to maximize both the data log-likelihood and the parameter agreement of the two phoneme mapping models (shaded).

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.

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 $E = \{\mathbf{e_n}\}_{n=1}^{|E|}$ and independent training maximizes:

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

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

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

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 (**?**) 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 (**?**). 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 (**?**). 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.

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, λ 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:

$$P(\mathbf{f_1^J}, \mathbf{a_1^J} \mid \mathbf{e_1^I}) = \prod_{j=1}^{J} P_d(a_j \mid a_{j-1}, j) P_t(f_j \mid e_{a_j}) \quad (8)$$

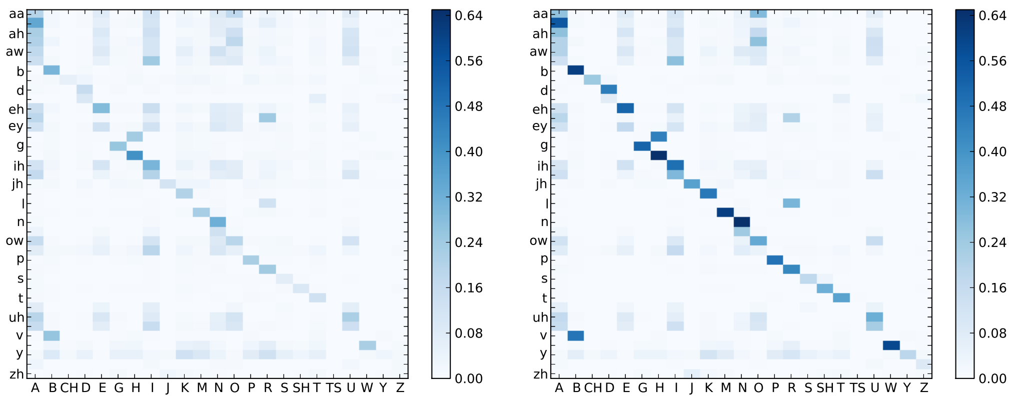

Figure 2: Compared to independent training. PAT (right) learns sparser, peaked models.

During training, we typically estimate the parameters that maximize

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

.

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

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

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* (**?**). Symmetrization remedies some of the mistakes that the independently trained models make, garbage collection in particular (**?**). 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 **?**) and **?**), the authors show that training the alignment models jointly can improve alignment quality. In **?**), 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 **?**), 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	$P_0 > 0.01$
?)	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.