NeurAlign: Combining Word Alignments Using Neural Networks

Necip Fazil Ayan, Bonnie J. Dorr and **Christof Monz**
Department of Computer Science
University of Maryland
College Park, MD 20742
{nfa,bonnie,christof}@umiacs.umd.edu

Abstract

This paper presents a novel approach to combining different word alignments. We view word alignment as a pattern classification problem, where alignment combination is treated as a classifier ensemble, and alignment links are adorned with linguistic features. A neural network model is used to learn word alignments from the individual alignment systems. We show that our alignment combination approach yields a significant 20-34% relative error reduction over the best-known alignment combination technique on English-Spanish and English-Chinese data.

1 Introduction

Parallel texts are a valuable resource in natural language processing and essential for projecting knowledge from one language onto another. Word-level alignment is a critical component of a wide range of NLP applications, such as construction of bilingual lexicons (Melamed, 2000), word sense disambiguation (Diab and Resnik, 2002), projection of language resources (Yarowsky et al., 2001), and statistical machine translation. Although word-level aligners tend to perform well when there is *sufficient* training data, the quality decreases as the size of training data decreases. Even with large amounts of training data, statistical aligners have been shown to be susceptible to mis-aligning phrasal constructions (Dorr et al., 2002) due to many-to-many correspondences, morphological language distinctions, paraphrased and free translations, and a high percentage of function words (about 50% of the tokens in most texts).

This paper presents a novel approach to alignment combination, *NeurAlign*, that treats each alignment system as a black box and merges their outputs. We view word alignment as a pattern classification problem and treat alignment combination as a *classifier ensemble* (Hansen and Salamon, 1990; Wolpert, 1992). The ensemble-based approach was developed to select the best features of different learning algorithms, including those that may not produce a globally optimal solution (Minsky, 1991).

We use neural networks to implement the classifier-ensemble approach, as these have previously been shown to be effective for combining classifiers (Hansen and Salamon, 1990). Neural nets with 2 or more layers and non-linear activation functions are capable of learning any function of the feature space with arbitrarily small error. Neural nets have been shown to be effective with (1) high-dimensional input vectors, (2) relatively sparse data, and (3) noisy data with high within-class variability, all of which apply to the word alignment problem.

The rest of the paper is organized as follows: In Section 2, we describe previous work on improving word alignments and use of classifier ensembles in NLP. Section 3 gives a brief overview of neural networks. In Section 4, we present a new approach, *NeurAlign*, that learns how to combine individual word alignment systems. Section 5 describes our experimental design and the results on English-Spanish and English-Chinese. We demonstrate that NeurAlign yields significant improvements over the best-known alignment combination technique.

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 65–72, Vancouver, October 2005.

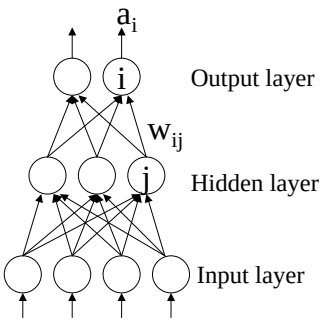


Figure 1: Multilayer Perceptron Overview

2 Related Work

Previous algorithms for improving word alignments have attempted to incorporate additional knowledge into their modeling. For example, Liu (2005) uses a log-linear combination of linguistic features. Additional linguistic knowledge can be in the form of part-of-speech tags. (Toutanova et al., 2002) or dependency relations (Cherry and Lin, 2003). Other approaches to improving alignment have combined alignment models, e.g., using a log-linear combination (Och and Ney, 2003) or mutually independent association clues (Tiedemann, 2003).

A simpler approach was developed by Ayan et al. (2004), where word alignment outputs are combined using a linear combination of feature weights assigned to the individual aligners. Our method is more general in that it uses a neural network model that is capable of learning nonlinear functions.

Classifier ensembles are used in several NLP applications. Some NLP applications for classifier ensembles are POS tagging (Brill and Wu, 1998; Abney et al., 1999), PP attachment (Abney et al., 1999), word sense disambiguation (Florian and Yarowsky, 2002), and parsing (Henderson and Brill, 2000).

The work reported in this paper is the first application of classifier ensembles to the word-alignment problem. We use a different methodology to combine classifiers that is based on *stacked generalization* (Wolpert, 1992), i.e., learning an additional model on the outputs of individual classifiers.

3 Neural Networks

A multi-layer perceptron (MLP) is a feed-forward neural network that consists of several units (neurons) that are connected to each other by weighted links. As illustrated in Figure 1, an MLP consists of one input layer, one or more hidden layers, and one output layer. The external input is presented to the input layer, propagated forward through the hidden layers and creates the output vector in the output layer. Each unit i in the network computes its output with respect to its net input $net_i = \sum_j w_{ij} a_j$, where j represents all units in the previous layer that are connected to the unit i. The output of unit i is computed by passing the net input through a non-linear activation function f, i.e. $a_i = f(net_i)$.

The most commonly used non-linear activation functions are the log sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ or hyperbolic tangent sigmoid function $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$. The latter has been shown to be more suitable for binary classification problems.

The critical question is the computation of weights associated with the links connecting the neurons. In this paper, we use the resilient backpropagation (RPROP) algorithm (Riedmiller and Braun, 1993), which is based on the gradient descent method, but converges faster and generalizes better.

4 NeurAlign Approach

We propose a new approach, *NeurAlign*, that learns how to combine individual word alignment systems. We treat each alignment system as a classifier and transform the combination problem into a classifier ensemble problem. Before describing the NeurAlign approach, we first introduce some terminology used in the description below.

Let $E = e_1, \ldots, e_t$ and $F = f_1, \ldots, f_s$ be two sentences in two different languages. An alignment link (i, j) corresponds to a translational equivalence between words e_i and f_j. Let A_k be an alignment between sentences E and F, where each element $a \in A_k$ is an alignment link (i, j). Let $\mathcal{A} = \{A_1, \ldots, A_l\}$ be a set of alignments between E and F. We refer to the true alignment as T, where each $a \in T$ is of the form (i, j). A *neighborhood* of an alignment link (i, j)—denoted by $N(i, j)$—consists of 8 possible alignment links in a 3×3 window with (i, j) in the center of the window. Each element of $N(i, j)$ is called a *neighboring link* of (i, j).

Our goal is to combine the information in $A_1, \ldots, A_l$ such that the resulting alignment is closer to T. A straightforward solution is to take the intersection or union of the individual alignments, or

perform a majority voting for each possible alignment link (i, j). Here, we use an additional model to learn how to combine outputs of $A_1, \ldots, A_l$.

We decompose the task of combining word alignments into two steps: (1) Extract features; and (2) Learn a classifier from the transformed data. We describe each of these two steps in turn.

4.1 Extracting Features

Given sentences E and F, we create a (potential) alignment instance (i, j) for all possible word combinations. A crucial component of building a classifier is the selection of features to represent the data. The simplest approach is to treat each alignment-system output as a separate feature upon which we build a classifier. However, when only a few alignment systems are combined, this feature space is not sufficient to distinguish between instances. One of the strategies in the classification literature is to supply the input data to the set of features as well.

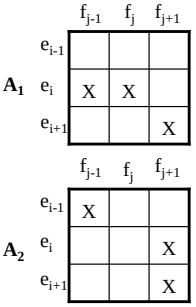
While combining word alignments, we use two types of features to describe each instance (i, j): (1) linguistic features and (2) alignment features. Linguistic features include POS tags of both words (e_i and f_j) and a dependency relation for one of the words (e_i). We generate POS tags using the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish. Dependency relations are produced using a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies.

Alignment features consist of features that are extracted from the outputs of individual alignment systems. For each alignment $A_k \in \mathcal{A}$, the following are some of the alignment features that can be used to describe an instance (i, j):

1. Whether (i, j) is an element of A_k or not
2. Translation probability $p(f_j|e_i)$ computed over A_k[1]
3. Fertility of (i.e., number of words in F that are aligned to) e_i in A_k
4. Fertility of (i.e., number of words in E that are aligned to) f_j in A_k
5. For each neighbor $(x, y) \in N(i, j)$, whether $(x, y) \in A_k$ or not (8 features in total)
6. For each neighbor $(x, y) \in N(i, j)$, translation probability $p(f_y|e_x)$ computed over A_k (8 features in total)

It is also possible to use variants, or combinations, of these features to reduce feature space.

Figure 2 shows an example of how we transform the outputs of 2 alignment systems, A_1 and A_2, for an alignment link (i, j) into data with some of the features above. We use -1 and 1 to represent the absence and existence of a link, respectively. The neighboring links are presented in row-by-row order.



Features for the alignment link (*i* , *j*)	
pos(e_i) , pos(f_j)	Noun, Prep
rel(e_i)	Modifier
outputs of aligners	1 (for A_1), -1 (for A_2)
neighbors (A_1)	-1, -1, -1, **1**, -1, -1, -1, **1**
neighbors (A_2)	**1**, -1, -1, -1, **1**, -1, -1, **1**
neighbors ($A_1 \cup A_2$)	**1**, -1, -1, **1**, **1**, -1, -1, **1**
total neighbors	2 (for A_1), 3 (for A_2)
fertility(e_i)	2 (for A_1), 1 (for A_2)
fertility(f_j)	1 (for A_1), 0 (for A_2)

Figure 2: An Example of Transforming Alignments into Classification Data

For each sentence pair $E = e_1, \ldots, e_t$ and $F = f_1, \ldots, f_s$, we generate $s \times t$ instances to represent the sentence pair in the classification data.

Supervised learning requires the correct output, which here is the true alignment T. If an alignment link (i, j) is an element of T, then we set the correct output to 1, and to -1, otherwise.

4.2 Learning A Classifier

Once we transform the alignments into a set of instances with several features, the remaining task is to learn a classifier from this data. In the case of word alignment combination, there are important issues to consider for choosing an appropriate classifier. First, there is a very limited amount of manually annotated data. This may give rise to poor generalizations because it is very likely that unseen data include lots of cases that are not observed in the training data.

Second, the distribution of the data according to the classes is skewed. In a preliminary study on an English-Spanish data set, we found out that only 4% of the all word pairs are aligned to each other by humans, among a possible 158K word pairs. Moreover, only 60% of those aligned word pairs were

[1]The translation probabilities can be borrowed from the existing systems, if available. Otherwise, they can be generated from the outputs of individual alignment systems using likelihood estimates.

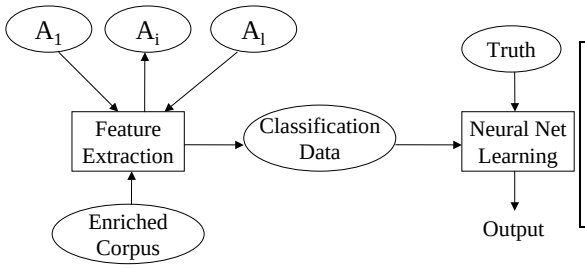


Figure 3: NeurAlign$_1$—Alignment Combination Using All Data At Once

		SPANISH						
		Adj	Adv	Comp	Det	Noun	Prep	Verb
E	Adj	18	-	-	82	40	96	66
N	Adv	-	8	-	-	50	67	75
G	Comp	-	-	12	-	46	37	96
L	Det	-	-	-	10	60	100	-
I	Noun	42	77	100	94	23	98	84
S	Prep	-	-	-	93	70	22	100
H	Verb	42	-	-	100	66	78	43

Table 1: Error Rates according to POS Tags for GIZA++ (E-to-S) (in percentages)

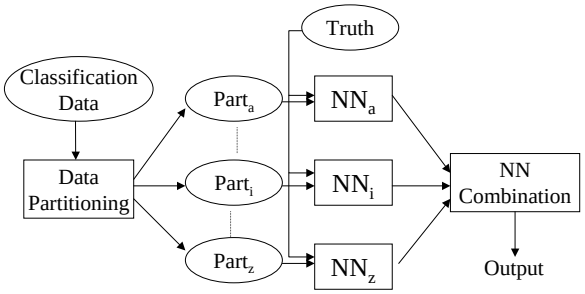


Figure 4: NeurAlign$_2$—Alignment Combination with Partitioning

also aligned by the individual alignment systems that were tested.

Finally, given the distribution of the data, it is difficult to find the right features to distinguish between instances. Thus, it is prudent to use as many features as possible and let the learning algorithm filter out the redundant features.

Below, we describe how neural nets are used at different levels to build a good classifier.

4.2.1 NeurAlign$_1$: Learning All At Once

Figure 3 illustrates how we combine alignments using all the training data at the same time (NeurAlign$_1$). First, the outputs of individual alignments systems and the original corpus (enriched with additional linguistic features) are passed to the feature extraction module. This module transforms the alignment problem into a classification problem by generating a training instance for every pair of words between the sentences in the original corpus. Each instance is represented by a set of features (described in Section 4.1). The new training data is passed to a neural net learner, which outputs whether an alignment link exists for each training instance.

4.2.2 NeurAlign$_2$: Multiple Neural Networks

The use of multiple neural networks (NeurAlign$_2$) enables the decomposition of a complex problem into smaller problems. *Local experts* are learned for each smaller problem and these are then merged. Following Tumer and Ghosh (1996), we apply spatial partitioning of training instances using proximity of patterns in the input space to reduce the complexity of the tasks assigned to individual classifiers.

We conducted a preliminary analysis on 100 randomly selected English-Spanish sentence pairs from a mixed corpus (UN + Bible + FBIS) to observe the distribution of errors according to POS tags in both languages. We examined the cases in which the individual alignment and the manual annotation were different—a total of 3,348 instances, where 1,320 of those are misclassified by GIZA++ (E-to-S).[2] We use a standard measure of error, i.e., the percentage of misclassified instances out of the total number of instances. Table 1 shows error rates (by percentage) according to POS tags for GIZA++ (E-to-S).[3]

Table 1 shows that the error rate is relatively low in cases where both words have the same POS tag. Except for verbs, the lowest error rate is obtained when both words have the same POS tag (the error rates on the diagonal). On the other hand, the error rates are high in several other cases, as much as 100%, e.g., when the Spanish word is a determiner or a preposition.[4] This suggests that dividing the training data according to POS tag, and training neural networks on each subset separately might be better than training on the entire data at once.

Figure 4 illustrates the combination approach with neural nets after partitioning the data into dis-

[2]For this analysis, we ignored the cases where both systems produced an output of -1 (i.e., the words are not aligned).

[3]Only POS pairs that occurred at least 10 times are shown.

[4]The same analysis was done for the other direction and resulted in similar distribution of error rates.

joint subsets (NeurAlign$_2$). Similar to NeurAlign$_1$, the outputs of individual alignment systems, as well as the original corpus, are passed to the feature extraction module. Then the training data is split into disjoint subsets using a subset of the available features for partitioning. We learn different neural nets for each partition, and then merge the outputs of the individual nets. The advantage of this is that it results in different generalizations for each partition and that it uses different subsets of the feature space for each net.

5 Experiments and Results

This section describes our experimental design, including evaluation metrics, data, and settings.

5.1 Evaluation Metrics

Let A be the set of alignment links for a set of sentences. We take S to be the set of sure alignment links and P be the set of probable alignment links (in the gold standard) for the same set of sentences. Precision (Pr), recall (Rc) and alignment error rate (AER) are defined as follows:

$$Pr = \frac{|A \cap P|}{|A|} \quad Rc = \frac{|A \cap S|}{|S|}$$

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

A manually aligned corpus is used as our gold standard. For English-Spanish data, the manual annotation is done by a bilingual English-Spanish speaker. Every link in the English-Spanish gold standard is considered a sure alignment link (i.e., $P = S$).

For English-Chinese, we used 2002 NIST MT evaluation test set. Each sentence pair was aligned by two native Chinese speakers, who are fluent in English. Each alignment link appearing in both annotations was considered a sure link, and links appearing in only one set were judged as probable. The annotators were not aware of the specifics of our approach.

5.2 Evaluation Data and Settings

We evaluated NeurAlign$_1$ and NeurAlign$_2$, using 5-fold cross validation on two data sets:

1. A set of 199 English-Spanish sentence pairs (nearly 5K words on each side) from a mixed corpus (UN + Bible + FBIS).
2. A set of 491 English-Chinese sentence pairs (nearly 13K words on each side) from 2002 NIST MT evaluation test set.

We computed precision, recall and error rate on the entire set of sentence pairs for each data set.[5]

To evaluate NeurAlign, we used GIZA++ in both directions (E-to-F and F-to-E, where F is either Chinese (C) or Spanish (S)) as input and a *refined alignment* approach (Och and Ney, 2000) that uses a heuristic combination method called *grow-diag-final* (Koehn et al., 2003) for comparison. (We henceforth refer to the refined-alignment approach as "RA.")

For the English-Spanish experiments, GIZA++ was trained on 48K sentence pairs from a mixed corpus (UN + Bible + FBIS), with nearly 1.2M of words on each side, using 10 iterations of Model 1, 5 iterations of HMM, and 5 iterations of Model 4. For the English-Chinese experiments, we used 107K sentence pairs from FBIS corpus (nearly 4.1M English and 3.3M Chinese words) to train GIZA++, using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

5.3 Neural Network Settings

In our experiments, we used a multi-layer perceptron (MLP) consisting of 1 input layer, 1 hidden layer, and 1 output layer. The hidden layer consists of 10 units, and the output layer consists of 1 unit. All units in the hidden layer are fully connected to the units in the input layer, and the output unit is fully connected to all the units in the hidden layer. We used hyperbolic tangent sigmoid function as the activation function for both layers.

One of the potential pitfalls is overfitting as the number of iterations increases. To address this, we used the *early stopping with validation set* method. In our experiments, we held out (randomly selected) 1/4 of the training set as the validation set.

Neural nets are sensitive to the initial weights. To overcome this, we performed 5 runs of learning for each training set. The final output for each training is obtained by a majority voting over 5 runs.

[5] The number of alignment links varies over each fold. Therefore, we chose to evaluate all data at once instead of evaluating on each fold and then averaging.

5.4 Results

This section describes the experiments on English-Spanish and English-Chinese data for testing the effects of feature selection, training on the entire data (NeurAlign$_1$) or on the partitioned data (NeurAlign$_2$), using two input alignments: GIZA++ (E-to-F) and GIZA++ (F-to-E). We used the following additional features, as well as the outputs of individual aligners, for an instance (i, j) (set of features 2–7 below are generated separately for each input alignment A_k):

1. $posE_i, posF_j, relE_i$: POS tags and dependency relation for e_i and f_j.
2. $neigh(i, j)$: 8 features indicating whether a neighboring link exists in A_k.
3. $fertE_i, fertF_j$: 2 features indicating the fertility of e_i and f_j in A_k.
4. $NC(i, j)$: Total number of existing links in $N(i, j)$ in A_k.
5. $TP(i, j)$: Translation probability $p(f_j|e_i)$ in A_k.
6. $NghTP(i, j)$: 8 features indicating the translation probability $p(f_y|e_x)$ for each $(x, y) \in N(i, j)$ in A_k.
7. $AvTP(i, j)$: Average translation probability of the neighbors of (i, j) in A_k.

We performed statistical significance tests using two-tailed paired t-tests. Unless otherwise indicated, the differences between NeurAlign and other alignment systems, as well as the differences among NeurAlign variations themselves, were statistically significant within the 95% confidence interval.

5.4.1 Results for English-Spanish

Table 2 summarizes the precision, recall and alignment error rate values for each of our two alignment system inputs plus the three alternative alignment-combination approaches. Note that the best performing aligner among these is the RA method, with an AER of 21.2%. (We include this in subsequent tables for ease of comparison.)

Feature Selection for Training All Data At Once: NeurAlign$_1$ Table 3 presents the results of training neural nets using the entire data (NeurAlign$_1$) with different subsets of the feature space. When we used POS tags and the dependency relation as features, NeurAlign$_1$ performs worse than RA. Using the neighboring links as the feature set gave slightly (not significantly) better results than RA. Using POS tags, dependency relations, and neighboring links also resulted in better performance than RA but the difference was not statistically significant.

Alignments	Pr	Rc	AER
E-to-S	87.0	67.0	24.3
S-to-E	88.0	67.5	23.6
Intersection	**98.2**	59.6	25.9
Union	80.6	**74.9**	22.3
RA	83.8	74.4	**21.2**

Table 2: Results for GIZA++ Alignments and Their Simple Combinations

When we used fertilities along with the POS tags and dependency relations, the AER was 20.0%—a significant relative error reduction of 5.7% over RA. Adding the neighboring links to the previous feature set resulted in an AER of 17.6%—a significant relative error reduction of 17% over RA.

Interestingly, when we removed POS tags and dependency relations from this feature set, there was no significant change in the AER, which indicates that the improvement is mainly due to the neighboring links. This supports our initial claim about the clustering of alignment links, i.e., when there is an alignment link, usually there is another link in its neighborhood. Finally, we tested the effects of using translation probabilities as part of the feature set, and found out that using translation probabilities did no better than the case where they were not used. We believe this happens because the translation probability $p(f_j|e_i)$ has a unique value for each pair of e_i and f_j; therefore it is not useful to distinguish between alignment links with the same words.

Feature Selection for Training on Partitioned Data: NeurAlign$_2$ In order to train on partitioned data (NeurAlign$_2$), we needed to establish appropriate features for partitioning the training data. Table 4 presents the evaluation results for NeurAlign$_1$ (i.e., no partitioning) and NeurAlign$_2$ with different features for partitioning (English POS tag, Spanish POS tag, and POS tags on both sides). For training on each partition, the feature space included POS tags (e.g., Spanish POS tag in the case where partitioning is based on English POS tag only), dependency relations, neighborhood features, and fertilities. We observed that partitioning based on POS tags on one side reduced the AER to 17.4% and

Features	Pr	Rc	AER
$posE_i, posF_j, relE_i$	90.6	67.7	22.5
$neigh(i,j)$	91.3	69.5	21.1
$posE_i, posF_j, relE_i,$ $neigh(i,j)$	**91.7**	70.2	20.5
$posE_i, posF_j, relE_i,$ $fertE_i, fertF_j$	91.4	71.1	20.0
$posE_i, posF_j, relE_i,$ $neigh(i,j), NC(i,j)$ $fertE_i, fertF_j$	89.5	**76.3**	**17.6**
$neigh(i,j), NC(i,j)$ $fertE_i, fertF_j$	89.7	75.7	17.9
$posE_i, posF_j, relE_i,$ $fertE_i, fertF_j,$ $neigh(i,j), NC(i,j),$ $TP(i,j), AvTP(i,j)$	90.0	75.7	17.9
RA	83.8	74.4	21.2

Table 3: Combination with Neural Networks: NeurAlign$_1$ (All-Data-At-Once)

17.1%, respectively. Using POS tags on *both* sides reduced the error rate to 16.9%—a significant relative error reduction of 5.6% over no partitioning. All four methods yielded statistically significant error reductions over RA—we will examine the fourth method in more detail below.

Alignment	Pr	Rc	AER
NeurAlign$_1$	89.7	75.7	17.9
NeurAlign$_2[posE_i]$	91.1	75.4	17.4
NeurAlign$_2[posF_j]$	91.2	**76.0**	17.1
NeurAlign$_2[posE_i, posF_j]$	**91.6**	**76.0**	**16.9**
RA	83.8	74.4	21.2

Table 4: Effects of Feature Selection for Partitioning

Once we determined that partitioning by POS tags on both sides brought about the biggest gain, we ran NeurAlign$_2$ using this partitioning, but with different feature sets. Table 5 shows the results of this experiment. Using dependency relations, word fertilities and translation probabilities (both for the link in question and the neighboring links) yielded a significantly lower AER (18.6%)—a relative error reduction of 12.3% over RA. When the feature set consisted of dependency relations, word fertilities, and neighborhood links, the AER was reduced to 16.9%—a 20.3% relative error reduction over RA. We also tested the effects of adding translation probabilities to this feature set, but as in the case of NeurAlign$_1$, this did not improve the alignments.

In the best case, NeurAlign$_2$ achieved substantial and significant reductions in AER over the input alignment systems: a 28.4% relative error reduction over S-to-E and a 30.5% relative error reduction over E-to-S. Compared to RA, NeurAlign$_2$ also achieved significantly better results over RA: relative improvements of 9.3% in precision, 2.2% in recall, and 20.3% in AER.

Features	Pr	Rc	AER
$relE_i, fertE_i, fertF_j,$ $TP(i,j), AvTP(i,j),$ $NghTP(i,j)$	**91.9**	73.0	18.6
$neigh(i,j)$	90.3	74.0	18.7
$relE_i, fertE_i, fertF_j,$ $neigh(i,j), NC(i,j)$	91.6	76.0	**16.9**
$relE_i, fertE_i, fertF_j,$ $neigh(i,j), NC(i,j),$ $TP(i,j), AvTP(i,j)$	91.4	**76.1**	**16.9**
RA	83.8	74.4	21.2

Table 5: Combination with Neural Networks: NeurAlign$_2$ (Partitioned According to POS tags)

5.4.2 Results for English-Chinese

The results of the input alignments to NeurAlign, i.e., GIZA++ alignments in two different directions, NeurAlign$_1$ (i.e., no partitioning) and variations of NeurAlign$_2$ with different features for partitioning (English POS tag, Chinese POS tag, and POS tags on both sides) are shown in Table 6. For comparsion, we also include the results for RA in the table. For brevity, we include only the features resulting in the best configurations from the English-Spanish experiments, i.e., POS tags, dependency relations, word fertilities, and neighborhood links (the features in the third row of Table 5). The ground truth used during the training phase consisted of all the alignment links with equal weight.

Alignments	Pr	Rc	AER
E-to-C	70.4	68.3	30.7
C-to-E	66.0	69.8	32.2
NeurAlign$_1$	85.0	71.4	22.2
NeurAlign$_2[posE_i]$	85.7	74.6	20.0
NeurAlign$_2[posF_j]$	85.7	73.2	20.8
NeurAlign$_2[posE_i, posF_j]$	**86.3**	74.7	**19.7**
RA	61.9	**82.6**	29.7

Table 6: Results on English-Chinese Data

Without any partitioning, NeurAlign achieves an alignment error rate of 22.2%—a significant relative error reduction of 25.3% over RA. Partitioning the data according to POS tags results in significantly better results over no partitioning. When the data is partitioned according to both POS tags, NeurAlign reduces AER to 19.7%—a significant relative error reduction of 33.7% over RA. Compared to the input

alignments, the best version of NeurAlign achieves a relative error reduction of 35.8% and 38.8%, respectively.

6 Conclusions

We presented NeurAlign, a novel approach to combining the outputs of different word alignment systems. Our approach treats individual alignment systems as black boxes, and transforms the individual alignments into a set of data with features that are borrowed from their outputs and additional linguistic features (such as POS tags and dependency relations). We use neural nets to learn the true alignments from these transformed data.

We show that using POS tags to partition the transformed data, and learning a different classifier for each partition is more effective than using the entire data at once. Our results indicate that NeurAlign yields a significant 28-39% relative error reduction over the best of the input alignment systems and a significant 20-34% relative error reduction over the best known alignment combination technique on English-Spanish and English-Chinese data.

We should note that NeurAlign is not a standalone word alignment system but a supervised learning approach to improve already existing alignment systems. A drawback of our approach is that it requires annotated data. However, our experiments have shown that significant improvements can be obtained using a small set of annotated data. We will do additional experiments to observe the effects of varying the size of the annotated data while learning neural nets. We are also planning to investigate whether NeurAlign helps when the individual aligners are trained using more data.

We will extend our combination approach to combine word alignment systems based on different models, and investigate the effectiveness of our technique on other language pairs. We also intend to evaluate the effectiveness of our improved alignment approach in the context of machine translation and cross-language projection of resources.

Acknowledgments This work has been supported in part by ONR MURI Contract FCPO.810548265, Cooperative Agreement DAAD190320020, and NSF ITR Grant IIS-0326553.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and PP attachment. In *Proceedings of EMNLP'1999*, pages 38–45.

Necip F. Ayan, Bonnie J. Dorr, and Nizar Habash. 2004. Multi-Align: Combining linguistic and statistical techniques to improve alignments for adaptable MT. In *Proceedings of AMTA'2004*, pages 17–26.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proc. of ACL'1998*.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL'2003*.

Micheal Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of ACL'1997*.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL'2002*.

Bonnie J. Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. 2002. DUSTer: A method for unraveling cross-language divergences for statistical word–level alignment. In *Proceedings of AMTA'2002*.

Radu Florian and David Yarowsky. 2002. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of EMNLP'2002*, pages 25–32.

L. Hansen and P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

John C. Henderson and Eric Brill. 2000. Bagging and boosting a treebank parser. In *Proceedings of NAACL'2000*.

Philip Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL/HLT'2003*.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL'2005*.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

Marvin Minsky. 1999. Logical Versus Analogical or Symbolic Versus Connectionist or Neat Versus Scruffy. *AI Magazine*, 12:34–51.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL'2000*.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):9–51, March.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP'1996*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE Intl. Conf. on Neural Networks*, pages 586–591.

Jorg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of EACL'2003*, pages 339–346.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP'2002*.

Kagan Tumer and Joydeep Ghosh. 1996. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3–4):385–404, December.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT'2001*.