NeurAlign: Combining Word Alignments Using Neural Networks

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 better word alignments from the individual alignment systems. We show that our alignment combination approach yields a significant relative improvement of 8.8% over the best of the individual alignment systems and a significant relative improvement of 5.5% over the best known alignment system in terms of f-measure.

1 Introduction

Parallel texts are a valuable resource in natural language processing and an essential resource for approaches that project knowledge from one language onto another, like machine translation (MT). Alignment of words—finding corresponding words in two translationally equivalent sentences—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), and projection of language resources (Yarowsky et al., 2001).

Word-level alignment is also an important intermediate step in statistical machine translation, yet the problem of finding accurate alignments is complicated by a number of factors. Although word-level aligners tend to perform well when there is *enough* training data, the quality of word alignment decreases as the size of training data decreases. Even with large amounts of training data, statistical systems have been shown to be susceptible to misaligning phrasal constructions (Dorr et al., 2002) due to the existence of 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 attempts to merge their outputs. We view word alignment as a pattern classification problem and we treat alignment combination as a *classifier ensemble* (Hansen and Salamon, 1990; Wolpert, 1992). The ensemble-based approach was developed to facilitate the selection of 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). Specifically, 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 also been shown to be effective with (1) high-dimensional input vectors, (2) relatively sparse data, and (3) noisy data that have high within-class variability. The word alignment problem, when transformed into a classification problem, has these characteristics.

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. We demonstrate that NeurAlign induces more accurate word alignment than each of the individual alignment systems and yields significant improvements over the current best system.

2 Related Work

Previous algorithms for improving word alignments have attempted to incorporate additional knowledge into their modeling. For example, Och (2002) uses a maximum entropy model while Liu (2005) uses a log-linear combination of different linguistic features. Additional linguistic knowledge can be in the form of part-of-speech tags on both sides (Toutanova et al., 2002) or dependency relations in one language (Cherry and Lin, 2003).

Other approaches to improving alignment have involved combining alignment models, e.g., using a log-linear combination (Och and Ney, 2003) or a combination of mutually independent association clues (Tiedemann, 2003).

A simpler approach was developed by Ayan et al. (2004), where different word alignment outputs are combined using a linear combination of feature-based weights associated with the individual aligners. Our method is more general than this one in that it employs a neural network model that is capable of learning nonlinear functions.

Classifier ensembles are used in several NLP applications. The most common strategy is to generate different variations of the same algorithm by using bagging or boosting techniques and then combining them using different voting methods. 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 statistical 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:

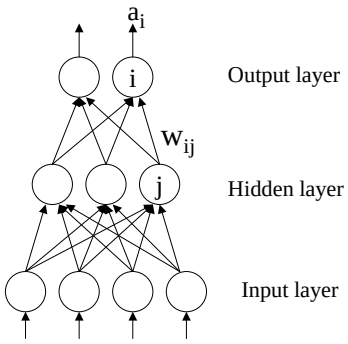


Figure 1: Multilayer Perceptron Overview

$$net_i = \sum_j w_{ij} a_j$$

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 solving binary classification problems.

The critical question is the computation of weights associated with the links connecting the neurons. The typical technique for such a computation is the backpropagation algorithm, which uses the principle of gradient descent (Rumelhart et al., 1986). In this paper, we use the resilient backpropagation (RPROP) algorithm (Riedmiller and Braun, 1993), which is an adaptive modification of the gradient descent method. This algorithm has been shown to converge faster and generalize better than the standard gradient descent method.

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 $E = e_1, \ldots, e_s$ and $F = f_1, \ldots, f_t$ be two sentences in two different languages. An alignment link (e_i, f_j) between words e_i and f_j corresponds to a translational equivalence between e_i and f_j in the E-to-F direction. Let A_k be an alignment between sentences E and F, where each element $a \in A_k$ is a pair of words, (e_i, f_j), that are aligned to each other. 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 (e_i, f_j). A *neighborhood* of an alignment link (e_i, f_j)—denoted by $N(e_i, f_j)$—consists of 8 word pairs in a 3×3 window with (e_i, f_j) in the center in the alignment matrix. Each element of $N(e_i, f_j)$ is called a *neighboring link* of (e_i, f_j).

Our goal is to combine the information in $A_1, \ldots, A_l$ such that the resulting alignment is closer to T. Some straightforward solutions are taking the intersection, or union of the individual alignments, or performing a majority voting for each possible alignment link (e_i, f_j). In this paper, 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 (e_i, f_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.

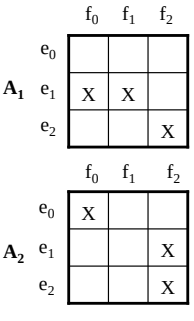
While combining word alignments, we use two types of features to describe each instance (e_i, f_j): (1) linguistic features and (2) alignment features. Linguistic features include POS tags of both words (e_i and f_j) and a dependency relation for one of the words (e_i). We generate POS tags using the MXPOST tagger (Ratnaparkhi, 1996) for English, and Connexor for Spanish. Dependency relations are produced using a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies. Such features were shown to be effective in improving different alignment models (Toutanova et al., 2002; Cherry and Lin, 2003; Liu et al., 2005).

Alignment features also include the neighbors of the given instance. Specifically, for each alignment $A_k \in \mathcal{A}$, the following are some of the alignment features that can be used to describe an instance (e_i, f_j):

1. Whether (e_i, f_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 $n \in N(e_i, f_j)$, whether $n \in A_k$ or not (8 features in total)
6. For each neighbor $n = (e_x, f_y) \in N(e_i, f_j)$, 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 a pair of words e_1 and f_1 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 pair (e_1, f_1)	
pos(e_1, f_1)	Noun, Prep
rel(e_1)	Mod
Outputs	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_1)	2 (for A_1), 1 (for A_2)
Fertility(f_1)	1 (for A_1), 0 (for A_2)

Figure 2: An Example of Transforming Alignments into Classification Data

[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.

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

For supervised learning requires the correct output to be specified. We obtain this information from the true alignment T. Specifically, if an alignment link (e_i, f_j) is an element of T, then we set the correct output to 1. Otherwise, we set it to -1.

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. We did a preliminary analysis on 199 English-Spanish sentences from a mixed corpus (UN + Bible + FBIS) to compare the outputs of two alignment systems (GIZA++ alignments in either direction (Och and Ney, 2000)) to a set of manually annotated data. As shown in Table 1, only 6K of nearly 158K instances are assigned to a class of 1. Moreover, only 60% of those 6K instances that are assigned to class 1 are also assigned to class 1 by the individual alignment systems.

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.

GIZA++		**Manual**		
D1	**D2**	**-1**	**1**	**Total**
-1	-1	150,797	1,546	152,343
-1	1	497	491	988
1	-1	545	456	1,001
1	1	69	3,672	3,741
Total		151,908	6,165	158,073

Table 1: Distribution of Instances According to Outputs of GIZA++ Alignments

In the remainder of this section, we describe how neural nets are used at different levels to build a good classifier.

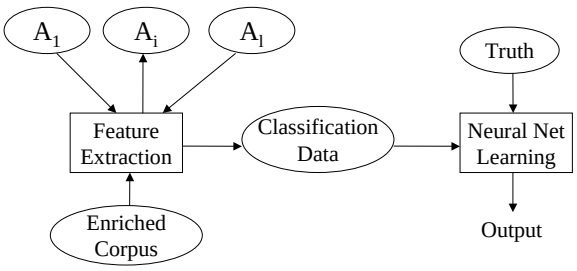


Figure 3: Overview of Alignment Combination Using All Data At Once

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 combination 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). This new training data is passed to a neural net learning module, which outputs whether an alignment link exists or not for all training instances.

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, in order to reduce the complexity of the task assigned to each individual classifier.

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

[2]For this analysis, we ignored the cases where both systems

		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 2: Error Rates (in Percentage) According to POS Tags for GIZA++ (E-to-S)

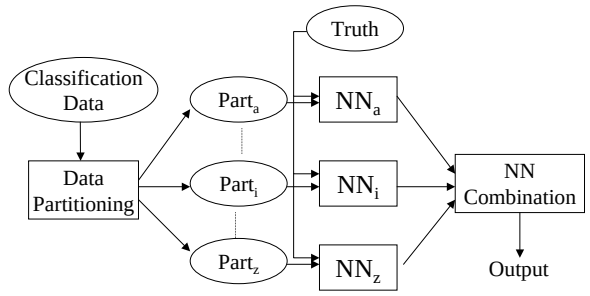


Figure 4: Alignment Combination with Data Partitioning

use a standard measure of error, i.e., the percentage of misclassified instances out of the total number of instances. Table 2 shows error rates (by percentage) according to POS tags for GIZA++ (E-to-S).[3]

Table 2 reveals 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 unusually 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 entire training data into different POS subsets, and training neural networks on each subset separately might be a better approach than training on the entire data at once.

Figure 4 illustrates the combination approach with neural nets after partitioning the data into disjoint subsets. 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 merge the outputs of individual neural nets to obtain the output for the entire data. The major advantage of learning with partitions is that it results in different generalizations for each partition. Moreover, this approach enables the use of different subsets of the feature space for each neural net.

produced an output of -1 (i.e., the words are not aligned).

[3]Only those 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.

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, and G be the set of alignment links in the gold standard for the same set of sentences. Precision (P), recall (R) and f-measure (F) are defined as follows:

$$P = \frac{|A \cap G|}{|A|}, \;\; R = \frac{|A \cap G|}{|G|}, \;\; F = \frac{2 \times P \times R}{P + R}$$

A manually aligned corpus is used as our gold standard. The manual annotation is done by a bilingual English-Spanish speaker who has no knowledge about the specifics of our approach.[5]

5.2 Evaluation Data and Settings

We evaluated our two approaches, NeurAlign$_1$ and NeurAlign$_2$, using 5-fold cross validation on our mixed corpus of 199 English-Spanish sentences. We computed precision, recall and f-measure on all 199 sentences at once rather than computing these for each test set and averaging the results.[6]

To evaluate NeurAlign, we use two GIZA++ alignments (E-to-S and S-to-E) as input and we compare our performance to that of three other alignment-combination techniques: (1) intersection of E-to-S and S-to-E; (2) union of E-to-S and S-to-E; and (3) a *refined alignment* approach (Och and Ney, 2000) that uses a heuristic combination method called *grow-diag-final* (Koehn et al., 2003). (We

[5]Since our gold standard has only one alignment annotation and there is no distinction between sure and probable alignments, we cannot use alignment error rate as an evaluation metric.

[6]Because the number of alignment links varies over each fold, we chose to evaluate on the entire data at once instead of evaluating on each fold and then averaging.

henceforth refer to the refined-alignment approach as "RA.")

GIZA++ was trained on 48K sentence pairs from a mixed corpus (UN + Bible + FBIS) in English and Spanish (nearly 1.2 millions of words on each side). The average sentence length in English sentences was 25 words while the average sentence length in Spanish was 27 words. We used 10 iterations of Model 1, 5 iterations of HMM, 5 iterations of Model 3, and 5 iterations of Model 4 to train GIZA++.

Table 3 summarizes the precision, recall and f-measure values for each of our two alignment system inputs plus the three alternative alignment-combination approaches. Note that the best performing aligner (thus far) is the RA method, with an f-score of 78.8. (We include this in subsequent tables for ease of comparison.)

Alignments	P	R	F
E-to-S	87.0	67.0	75.7
S-to-E	88.0	67.5	76.4
Intersection	**98.2**	59.6	74.1
Union	80.6	**74.9**	77.7
RA	83.8	74.4	**78.8**

Table 3: Results for GIZA++ Alignments and Their Simple Combinations

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 of the neural-net approach is its overfitting as the number of iterations increases. To decrease the overfitting potential, we used the *early stopping with validation set* method (Nelson and Illingworth, 1991). 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 reduce the effects of weight initialization, 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.

5.4 Results

This section describes the experiments for testing the effects of feature selection, training on the entire data (NeurAlign$_1$) or on the partitioned data (NeurAlign$_2$), using two input alignment systems: GIZA++ (E-to-S) and GIZA++ (S-to-E). We used the following additional features, as well as the outputs of individual aligners, for an instance (e_i, f_j):

1. $pos(e_i), pos(f_j), rel(e_i)$: POS tags and dependency relation for the words in the pair (e_i, f_j).
2. $neigh(e_i, f_j)$: 8 features indicating whether a neighbor exists in any of the input alignments.
3. $NC(e_i, f_j)$: 2 features indicating the total number of existing links in the neighborhood of (e_i, f_j) (one for each input alignment).
4. $fert(e_i), fert(f_j)$: the fertility of (the number of words that are aligned to) e_i and f_j, respectively (one for each input alignment).
5. $TP(e_i, f_j)$: 2 features indicating the translation probability $p(f_j|e_i)$ (one for each input alignment).
6. $AvTP(e_i, f_j)$: 2 features indicating the average translation probability $p(f_j|e_i)$ for the neighbors of (e_i, f_j) (one for each input alignment).
7. $NghTP(e_i, f_j)$: 8 features indicating the translation probability $p(f_j|e_i)$ for each neighbor of (e_i, f_j).

We performed statistical significance tests on all f-score values reported below 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 Feature Selection for Training All Data At Once: NeurAlign$_1$

Table 4 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, the f-score was significantly lower for the neural networks than for RA (77.5 vs. 78.8). Using the neighboring links as the feature set gave slightly (not significantly) better results than RA (78.9 vs. 78.8). Using POS tags, dependency relations, and neighbor-

ing links also resulted in better performance than RA (79.5) but the difference was not statistically significant.

When we used fertilities along with the POS tags and dependency relations, the f-score was 80.0—a significant relative improvement of 1.5% over RA. Adding the neighboring links to the previous feature set resulted in an f-score of 82.4—a significant relative improvement of 4.6% over RA. Interestingly, when we removed POS tags and dependency relations from this feature set, we got slightly lower results but there was no significant change in the f-score (82.1), which indicates that NeurAlign$_1$ performs significantly better than the best existing method even without POS tags and dependency relations. Finally, we tested the effects of using translation probabilities as part of the feature set. Interestingly, using translation probabilities did no better than the case where translation probabilities were not used.

Features	P	R	F
$pos(e_i), pos(f_j), rel(e_i)$	90.6	67.7	77.5
$neigh(e_i, f_j)$	91.3	69.5	78.9
$pos(e_i), pos(f_j), rel(e_i)$, $neigh(e_i, f_j)$	**91.7**	70.2	79.5
$pos(e_i), pos(f_j), rel(e_i)$, $fert(e_i), fert(f_j)$	91.4	71.1	80.0
$pos(e_i), pos(f_j), rel(e_i)$, $neigh(e_i, f_j), NC(e_i, f_j)$ $fert(e_i), fert(f_j)$	89.5	**76.3**	**82.4**
$neigh(e_i, f_j), NC(e_i, f_j)$ $fert(e_i), fert(f_j)$	89.7	75.7	82.1
$pos(e_i), pos(f_j), rel(e_i)$, $fert(e_i), fert(f_j)$, $neigh(e_i, f_j), NC(e_i, f_j)$, $TP(e_i, f_j), AvTP(e_i, f_j)$	90.0	75.7	82.1
RA	83.8	74.4	78.8

Table 4: Combination with Neural Networks: NeurAlign$_1$ (All-Data-At-Once)

5.4.2 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 5 presents the evaluation results for different feature-based partitionings (no 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 increased the f-score to 82.6 and 82.9, respectively—relative improvements of 0.6% (insignificant) and 1.0%, respectively, over no partitioning. Using POS tags on *both* sides increased the f-score to 83.1— a significant relative improvement of 1.2% over no partitioning. All four methods (including no partitioning) yielded statistically significant improvements over RA—we will examine the fourth method in more detail below.

Partition on	P	R	F
No partitioning	89.7	75.7	82.1
$pos(e_i)$	91.1	75.4	82.6
$pos(f_j)$	91.2	**76.0**	82.9
$pos(e_i), pos(f_j)$	**91.6**	**76.0**	**83.1**
RA	83.8	74.4	78.8

Table 5: 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 6 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 better f-score (81.4)—a relative improvement of 3.3% over RA. Using only the neighborhood links resulted in a slightly (not significantly) lower f-score (81.3). When the feature set consisted of dependency relations, word fertilities, and neighborhood links, the f-score increased to 83.1—a 5.5% relative improvement over RA. We also tested the effects of adding translation probabilities to this feature set. As in the case of NeurAlign$_1$, using translation probabilities did not improve the alignments.

Features	P	R	F
$rel(e_i), fert(e_i), fert(f_j)$, $TP(e_i, f_j), AvTP(e_i, f_j)$, $NghTP(e_i, f_j)$	**91.9**	73.0	81.4
$neigh(e_i, f_j)$	90.3	74.0	81.3
$rel(e_i), fert(e_i), fert(f_j)$, $neigh(e_i, f_j), NC(e_i, f_j)$	91.6	76.0	**83.1**
$rel(e_i), fert(e_i), fert(f_j)$, $neigh(e_i, f_j), NC(e_i, f_j)$, $TP(e_i, f_j), AvTP(e_i, f_j)$	91.4	**76.1**	**83.1**
RA	83.8	74.4	78.8

Table 6: Combination with Neural Networks: NeurAlign$_2$ (Partitioned According to Pairs of POS tags)

In the best case, NeurAlign$_2$ achieved very large

(significant) f-score increases over the input alignment systems: an 8.8% relative improvement over S-to-E and a 9.8% relative improvement over E-to-S. Precision improved 4.1% and 5.3%, respectively, and recall improved 12.6% and 13.4%, respectively. When compared to RA, NeurAlign$_2$ also achieved significantly better results over RA: relative improvements of 9.3% in precision, 2.2% in recall, and 5.5% in f-score.

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 into disjoint subsets and learning a different classifier for each subset is more effective than using the entire data at once. Our experimental results indicate that NeurAlign yields a significant 8.8% relative improvement in f-measure over the best of the individual alignment systems and a significant 5.5% relative improvement in f-measure over the best known alignment system.

We are planning to extend our combination approach to combine word alignment systems based on different models, and investigate the effectiveness of our technique on different language pairs. We also intend to evaluate the effectiveness of our improved alignment approach in a larger context, to measure its impact on machine translation and projection of resources from one language to another.

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 *Proceedings 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.

Marilyn M. Nelson and W. T. Illingworth. 1991. *A Practical Guide to Neural Nets*. Addison-Wesley, Reading, MA.

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

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL'2002*, pages 295–302.

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.

D. E. Rumelhart, G. Hinton, and R.Williams. 1986. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol. I Foundations*, pages 318–362. MIT Press, Cambridge, MA.

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*.