End-to-End Neural Word Alignment Outperforms GIZA++

Thomas Zenkel, Joern Wuebker, John DeNero
Lilt, Inc.
first_name@lilt.com

Abstract

Word alignment was once a core unsupervised learning task in natural language processing because of its essential role in training statistical machine translation (MT) models. Although unnecessary for training neural MT models, word alignment still plays an important role in interactive applications of neural machine translation, such as annotation transfer and lexicon injection. While statistical MT methods have been replaced by neural approaches with superior performance, the twenty-year-old GIZA++ toolkit remains a key component of state-of-the-art word alignment systems. Prior work on neural word alignment has only been able to outperform GIZA++ by using its output during training. We present the first end-to-end neural word alignment method that consistently outperforms GIZA++ on three data sets. Our approach repurposes a Transformer model trained for supervised translation to also serve as an unsupervised word alignment model in a manner that is tightly integrated and does not affect translation quality.

1 Introduction

Although word alignments are no longer necessary to train machine translation (MT) systems, they still play an important role in applications of neural MT. For example, they enable injection of an external lexicon into the inference process to enforce the use of domain-specific terminology or improve the translations of low-frequency content words (Arthur et al., 2016). The most important application today for word alignments is to transfer text annotations from source to target (Müller, 2017; Tezcan and Vandeghinste, 2011; Joanis et al., 2013; Escartın and Arcedillo, 2015). For example, if part of a source sentence is underlined, the corresponding part of its translation should be underlined as well. HTML tags and other markup must be transferred for published documents. Although annotations could in principle be generated directly as part of the output sequence, they are instead typically transferred via word alignments because example annotations typically do not exist in MT training data.

The Transformer architecture provides state-of-the-art performance for neural machine translation (Vaswani et al., 2017). The decoder has multiple layers, each with several attention heads, which makes it difficult to interpret attention activations as word alignments. As a result, the most widely used tools to infer word alignments, namely GIZA++ (Och and Ney, 2003) and FastAlign (Dyer et al., 2013), are still based on the statistical IBM word alignment models developed nearly thirty years ago (Brown et al., 1993). No previous unsupervised neural approach has matched their performance. Recent work on alignment components that are integrated into neural translation models either underperform the IBM models or must use the output of IBM models during training to outperform them (Zenkel et al., 2019; Garg et al., 2019).

This work combines key components from Zenkel et al. (2019) and Garg et al. (2019) and presents two novel extensions. Statistical alignment methods contain an explicit bias towards contiguous word alignments in which adjacent source words are aligned to adjacent target words. This bias is expressed in statistical systems using a hidden Markov model (HMM) (Vogel et al., 1996), as well as symmetrization heuristics such as the grow-diag-final algorithm (Och and Ney, 2000b; Koehn et al., 2005). We design an auxiliary loss function that can be added to any attention-based network to encourage contiguous attention matrices.

The second extension replaces heuristic symmetrization of word alignments with an activation optimization technique. After training two alignment models that translate in opposite direc-

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1605–1617
July 5 - 10, 2020.

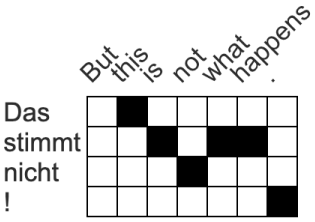


Figure 1: Word alignment generated by a human annotator.

tions, we infer a symmetrized attention matrix that jointly optimizes the likelihood of the correct output words under both models in both languages. Ablation experiments highlight the effectiveness of this novel extension, which is reminiscent of agreement-based methods for statistical models (Liang et al., 2006; Graça et al., 2008; DeNero and Macherey, 2011).

End-to-end experiments show that our system is the first to consistently yield higher alignment quality than GIZA++ using a fully unsupervised neural model that does not use the output of a statistical alignment model in any way.

2 Related Work

2.1 Statistical Models

Statistical alignment models directly build on the lexical translation models of Brown et al. (1993), known as the IBM models. The most popular statistical alignment tool is GIZA++ (Och and Ney, 2000b, 2003; Gao and Vogel, 2008). For optimal performance, the training pipeline of GIZA++ relies on multiple iterations of IBM Model 1, Model 3, Model 4 and the HMM alignment model (Vogel et al., 1996). Initialized with parameters from previous models, each subsequent model adds more assumptions about word alignments. Model 2 introduces non-uniform distortion, and Model 3 introduces fertility. Model 4 and the HMM alignment model introduce relative distortion, where the likelihood of the position of each alignment link is conditioned on the position of the previous alignment link. While simpler and faster tools exist such as FastAlign (Dyer et al., 2013), which is based on a reparametrization of IBM Model 2, the GIZA++ implementation of Model 4 is still used today in applications where alignment quality is important.

In contrast to GIZA++, our neural approach is easy to integrate on top of an attention-based translation network, has a training pipeline with fewer steps, and leads to superior alignment quality. Moreover, our fully neural approach that shares most parameters with a neural translation model can potentially take advantage of improvements to the underlying translation model, for example from domain adaptation via fine-tuning.

2.2 Neural Models

Most neural alignment approaches in the literature, such as Tamura et al. (2014) and Alkhouli et al. (2018), rely on alignments generated by statistical systems that are used as supervision for training the neural systems. These approaches tend to learn to copy the alignment errors from the supervising statistical models.

Zenkel et al. (2019) use attention to extract alignments from a dedicated alignment layer of a neural model without using any output from a statistical aligner, but fail to match the quality of GIZA++.

Garg et al. (2019) represents the current state of the art in word alignment, outperforming GIZA++ by training a single model that is able to both translate and align. This model is supervised with a guided alignment loss, and existing word alignments must be provided to the model during training. Garg et al. (2019) can produce alignments using an end-to-end neural training pipeline guided by attention activations, but this approach underperforms GIZA++. The performance of GIZA++ is only surpassed by training the guided alignment loss using GIZA++ output. Our method also uses guided alignment training, but our work is the first to surpass the alignment quality of GIZA++ without relying on GIZA++ output for supervision.

Stengel-Eskin et al. (2019) introduce a discriminative neural alignment model that uses a dot-product-based distance measure between learned source and target representation to predict if a given source-target pair should be aligned. Alignment decisions condition on the neighboring decisions using convolution. The model is trained using gold alignments. In contrast, our approach is fully unsupervised; it does not require gold alignments generated by human annotators during training. Instead, our system implicitly learns reasonable alignments by predicting future target words as part of the translation task, but selects attention activations using an auxiliary loss function to find contiguous alignment links that explain the data.

3 Background

3.1 The Alignment Task

Given a source-language sentence $x = x_1, \ldots, x_n$ of length n and its target-language translation $y = y_1, \ldots, y_m$ of length m, an alignment $\mathcal{A}$ is a set of pairs of source and target positions:

$$\mathcal{A} \subseteq \{(s,t) : s \in \{1,\ldots,n\}, t \in \{1,\ldots,m\}\}$$

Aligned words are assumed to correspond to each other, i.e. the source and the target word are translations of each other within the context of the sentence. Gold alignments are commonly generated by multiple annotators based on the Blinker guidelines (Melamed, 1998). The most commonly used metric to compare automatically generated alignments to gold alignments is alignment error rate (AER) (Och and Ney, 2000b).

3.2 Attention-Based Translation Models

Bahdanau et al. (2015) introduced attention-based neural networks for machine translation. These models typically consist of an encoder for the source sentence and a decoder that has access to the previously generated target tokens and generates the target sequence from left to right. Before predicting a token, the decoder "attends" to the position-wise source representations generated by the encoder, and it produces a context vector that is a weighted sum of the contextualized source embeddings.

The Transformer (Vaswani et al., 2017) attention mechanism uses a query Q and a set of k key-value pairs K, V with $Q \in \mathbb{R}^d$ and $V, K \in \mathbb{R}^{k \times d}$. Attention logits A_L computed by a scaled dot product are converted into a probability distribution A using the softmax function. The attention A serves as mixture weights for the values V to form a context vector c:

$$A_L = \mathsf{calcAttLogits}(Q,K) = \frac{Q \cdot K^T}{\sqrt{d}}$$
$$A = \mathsf{calcAtt}(Q,K) = \mathsf{softmax}(A_L)$$
$$c = \mathsf{applyAtt}(A,V) = A \cdot V$$

A state-of-the-art Transformer includes multiple attention heads whose context vectors are stacked to form the context activation for a layer, and the encoder and decoder have multiple layers. For all experiments, we use a downscaled Transformer model trained for translation with a 6-layer encoder, a 3-layer decoder, and 256-dimensional hidden states and embedding vectors.

For the purpose of word alignment, this translation Transformer is used as-is to extract representations of the source and the target sequences, and our alignment technique does not change the parameters of the Transformer. Therefore, improvements to the translation system can be expected to directly carry over to alignment quality, and the alignment component does not affect translation output in any way.

3.3 Alignment Layer

To improve the alignment quality achieved by interpreting attention activations, Zenkel et al. (2019) designed an additional *alignment layer* on top of the Transformer architecture. In the alignment layer, the context vector is computed as $\mathsf{applyAtt}(A,V)$, just as in other decoder layers, but this context vector is the only input to predicting the target word via a linear layer and a softmax that gives a probability distribution over the target vocabulary. This design forces attention onto the source positions that are most useful in predicting the target word. Figure 2 depicts its architecture.

This alignment layer uses the learned representations of the underlying translation model. Alignments can be extracted from the activations of this model by running a forward pass to obtain the attention weights A from the alignment layer and subsequently selecting the maximum probability source position for each target position as an alignment link: $\{(\mathsf{argmax}_i\,(A_{i,j}), j) : j \in [1,m]\}$.

The alignment layer predicts the next target token y_i based on the source representations x extracted from the encoder of the Transformer and all past target representations $y_{<i}$ extracted from the decoder. Thus the probability is conditioned as $p(y_i|x, y_{<i})$. The encoder representation used as key and value for the attention component is the sum of the input embeddings and the encoder output. This ensures that lexical and context information are both salient in the input to the attention component.

3.4 Attention Optimization

Extracting alignments with attention-based models works well when used in combination with greedy translation inference (Li et al., 2019). However, the alignment task involves predicting an alignment between a sentence and an observed translation, which requires forced decoding. When a token in the target sentence is unexpected given the preceding target prefix, attention activations computed

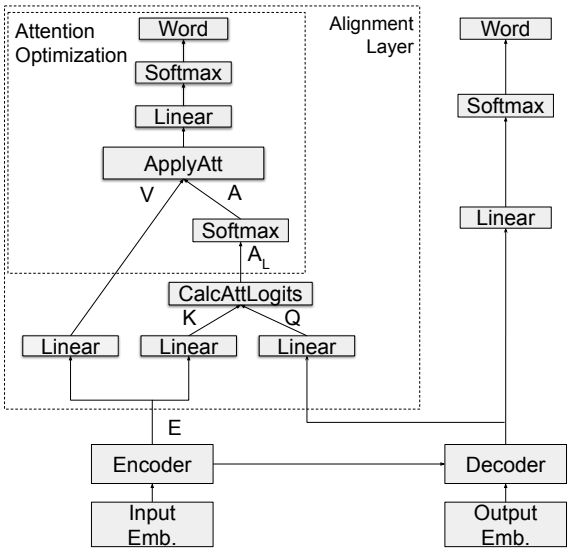


Figure 2: Architecture of the alignment layer. During inference the attention logits A_L of the sub-network *Attention Optimization* are optimized towards predicting the next word correctly.

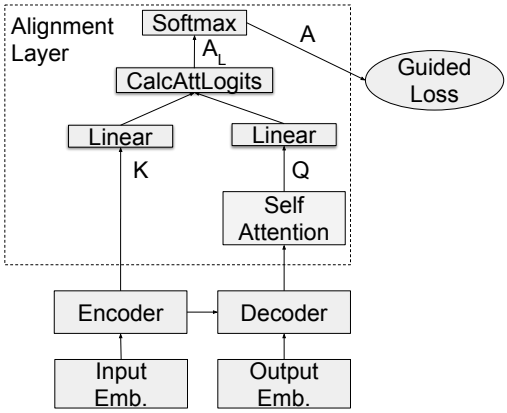


Figure 3: Alignment layer with additional unmasked self attention sublayer to use the full decoder context.

during forced decoding are not reliable because they do not explicitly condition on the target word being aligned.

Zenkel et al. (2019) introduce a method called *attention optimization*, which searches for attention activations that maximize the probability of the output sequence by directly optimizing the attention activations A in the alignment layer using gradient descent for the given sentence pair (x, y) to maximize the probability of each observed target token y_i while keeping all other parameters of the neural network M fixed:

$$\text{argmax}_A \; p(y_i|y_{<i}, x, A; M)$$

Attention optimization yields superior alignments when used during forced decoding when gradient descent is initialized with the activations from a forward pass through the alignment layer.

3.5 Full Context Model with Guided Alignment Loss

The models described so far are based on autoregressive translation models, so they are limited to only attend to the left context of the target sequence. However, for the word alignment task the current and future target context is also available and should be considered at inference time. Garg et al. (2019) train a single model to both predict the target sentence and the alignments using guided alignment training. When the model is trained to predict alignments, the full target context can be used to obtain improved alignment quality.

The alignment loss requires supervision by a set of alignment links for each sentence pair in the training data. These alignments can be generated by the current model or can be provided by an external alignment system or human annotators. Assuming one alignment link per target token, we denote the alignment source position for the target token at position t as a_t.[1] The guided alignment loss L_a, given attention probabilities $A_{a_t,t}$ for each source position a_t and target position t for a target sequence of length m, is defined as:

$$L_a(A) = -\frac{1}{m}\sum_{i=1}^{m}\log(A_{a_t,t})$$

As depicted in Figure 3, we insert an additional self-attention component into the original alignment layer, and leave the encoder and decoder of the Transformer unchanged. In contrast to Garg et al. (2019), this design does not require updating any translation model parameters; we only optimize the alignment layer parameters with the guided alignment loss. Adding an alignment layer for guided alignment training has a small parameter overhead as it only adds a single decoder layer, resulting in an increase in parameters of less than 5%.[2]

Unlike the standard decoder-side self-attention layers in the Transformer architecture, the current and future target context are not masked in the

[1] For the purpose of the guided alignment loss we assume target tokens that do not have an alignment link to be aligned to the end-of-sentence (EOS) token of the source sequence.

[2] The translation model contains 15 million parameters, while the additional alignment layer has 700 thousand parameters.

alignment layer self-attention component in order to provide the full target sentence as context. Alignment layer parameters are trained using the guided alignment loss.

4 Contiguity Loss

Contiguous alignment connections are very common in word alignments, especially for pairs of Indo-European languages. That is, if a target word at position t is aligned to a source word at position s, the next target word at position $t+1$ is often aligned to $s-1, s$ or $s+1$ (Vogel et al., 1996).

Our goal is to design a loss function that encourages alignments with contiguous clusters of links.

The attention activations form a 2-dimensional matrix $A \in \mathbb{R}^{n \times m}$, where n is the number of source tokens and m the number of target tokens: each entry represents a probability that specifies how much attention weight the network puts on each source word to predict the next target word. By using a convolution with a static kernel K over these attention scores, we can measure how much attention is focused on each rectangle within the two dimensional attention matrix:

$$\bar{A} = \mathsf{conv}(A, K)$$

$$L_C = -\sum_{t=1}^{m} \log(\max_{s \in \{1,\ldots,n\}} (\bar{A}_{s,t}))$$

We use a 2×2 kernel $K \in \mathbb{R}^{2 \times 2}$ with each element set to 0.5. Therefore, $\bar{A} \in \mathbf{R}^{n \times m}$ will contain the normalized attention mass of each 2×2 square of the attention matrix A. The resulting values after the convolution will be in the interval $[0.0, 1.0]$. For each target word we select the square with the highest attention mass, encouraging a sparse distribution over source positions in $\bar{A}$ and thus effectively training the model towards strong attention values on neighboring positions. We mask the contiguity loss such that the end of sentence symbol is not considered during this procedure. We apply a position-wise dropout of 0.1 on the attention logits before using the softmax function to obtain A, which turned out to be important to avoid getting stuck in trivial solutions during training.[3]

Optimizing the alignment loss especially encourages diagonal and horizontal patterns[4] as visualized in Figure 4. These correspond well to a large portion of patterns appearing in human alignment annotations as shown in Figure 1.

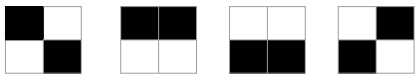


Figure 4: Example of alignment patterns that lead to a minimal contiguity loss.

5 Bidirectional Attention Optimization

A common way to extract word alignments is to train two models, one for the forward direction (source to target) and one for the backward direction (target to source). For each model, one can extract separate word alignments and symmetrize these using heuristics like grow-diagonal (Och and Ney, 2000b; Koehn et al., 2005).

However, this approach uses the hard word alignments of both directions as an input, and does not consider any other information of the forward and backward model. For attention-based neural networks it is possible to adapt attention optimization as described in Section 3.4 to consider two models at the same time. The goal of attention optimization is to find attention activations that lead to the correct prediction of the target sequence for a single neural network. We extend this procedure to optimize the likelihood of the sentence pair jointly under both the forward and the backward model, with the additional bias to favor contiguous alignments. Figure 5 depicts this procedure.

5.1 Initialization

Since attention optimization uses gradient descent to find good attention activations, it is important to start with a reasonable initialization. We extract the attention logits (attention before applying the softmax) from the forward $(A_L)_F$ and the backward model $(A_L)_B$ and average these to get a starting point for gradient descent: $(A_L)_{init} = \frac{1}{2}((A_L)_F + (A_L)_B^T)$.

5.2 Optimization

Our goal is to find attention logits A_L that lead to the correct prediction for both the forward M_F

[3] A trivial solution the network converged to when adding the contiguity loss without dropout was to align each target token to the same source token.

[4] Vertical patterns are not encouraged, as it is not possible to have an attention probability above 0.5 for two source words and the same target word, because we use the softmax function over the source dimension.

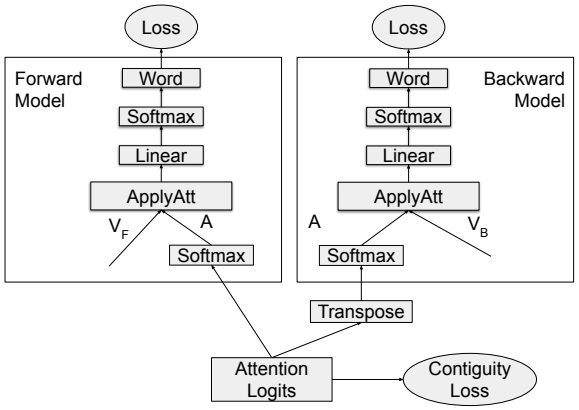


Figure 5: Bidirectional Attention Optimization. We optimize the attention logits towards the correct prediction of the next token when used for both the forward and backward model. The attention values V_F and V_B extracted from the forward and backward model remain static. Additionally, the attention logits are biased towards producing contiguous alignments.

and the backward model M_B, while also representing contiguous alignments. We will use the cross entropy loss CE for a whole target sequence y of length m to define the loss, given probabilities for each target token $p(y_t|A_t; M)$ under model parameters M and a given attention activation vector A_t:

$$\mathsf{CE}(p(y|A; M)) = \sum_{t=1}^{m} -\log(p(y_t|A_t; M))$$

Let x, y be the source and target sequence, so that we can define a loss function for each component with the interpolation parameter λ for the contiguity loss L_C as follows:

$$L_F = \mathsf{CE}(p(y|\mathsf{softmax}(A_L); M_F))$$
$$L_B = \mathsf{CE}(p(x|\mathsf{softmax}(A_L^T); M_B))$$
$$L = L_F + L_B + \lambda L_C$$

We apply gradient descent to optimize all losses simultaneously, thus approximating a solution of $\mathsf{argmin}_{A_L} L(x, y|A_L, M_F, M_B)$.

5.3 Alignment Extraction

After optimizing the attention logits, we still have to decide which alignment links to extract, i.e. how to convert the soft attentions into hard alignments. For neural models using a single direction a common method is to extract the alignment with the highest attention score for each target token. For our bidirectional method we use the following approach:

We merge the attention probabilities extracted from both directions using element-wise multiplication, where $\otimes$ denotes a Hadamard product:

$$A_F = \mathsf{softmax}(A_L)$$
$$A_B = \mathsf{softmax}(A_L^T)^T$$
$$A_M = A_F \otimes A_M$$

This favors alignments that effectively predict observed words in both the source and target sentences.

Given the number of source tokens n and target tokens m in the sentence, we select $\min(n, m)$ alignments that have the highest values in the merged attention scores A_M. In contrast to selecting one alignment per target token, this allows unaligned tokens, one-to-many, many-to-one and many-to-many alignment patterns.

6 Experiments

6.1 Data

We use the same experimental setup[5] as described by Zenkel et al. (2019) and used by Garg et al. (2019). It contains three language pairs: German→English, Romanian→English and English→French (Och and Ney, 2000a; Mihalcea and Pedersen, 2003). We learn a joint byte pair encoding (BPE) for the source and the target language with 40k merge operation (Sennrich et al., 2016). To convert from alignments between word pieces to alignments between words, we align a source word to a target word if an alignment link exists between any of its word pieces.

Using BPE units instead of words also improved results for GIZA++ (e.g., 20.9% vs. 18.9% for German→English in a single direction). Therefore, we use the exact same input data for GIZA++ and all our neural approaches. For training GIZA++ we use five iterations each for Model 1, the HMM model, Model 3 and Model 4.

6.2 Training

Most of the language pairs do not contain an adequately sized development set for word alignment experiments. Therefore, rather than early stopping, we used a fixed number of updates for each training stage across all languages pairs: 90k for training the translation model, 10k for the alignment layer and 10k for guided alignment training (batch-size:

[5] https://github.com/lilt/alignment-scripts

36k words). Training longer did not improve or degrade test-set AER on German→English; the AER only fluctuated by less than 1% when training the alignment layer for up to 20k updates while evaluating it every 2k updates.

We also trained a base transformer with an alignment layer for German→English, but achieved similar results in terms of AER, so we used the smaller model described in sub-section 3.2 for other language pairs. We adopted most hyperparameters from Zenkel et al. (2019), see the Supplemental Material for a summary. We tuned the interpolation factor for the contiguity loss on German→English.

6.3 Contiguity Loss

Results of ablation experiments for the contiguity loss can be found in Table 1. Our first experiment uses the contiguity loss during training and we extract the alignments from the forward pass using a single direction without application of attention optimization. We observe an absolute improvement of 6.4% AER (34.2% to 27.8%) after adding the contiguity loss during training.

Afterwards, we use the model trained with contiguity loss and use attention optimization to extract alignments. Adding the contiguity loss during attention optimization further improves the AER scores by 1.2%. Both during training and attention optimization we used an interpolation coefficient of $\lambda = 1.0$ for the contiguity loss.

By visualizing the attention activations in Figure 7 we see that the contiguity loss leads to sparse activations. Additionally, by favoring contiguous alignments it disambiguates correctly the alignment between the words "we" and "wir", which appear twice in the sentence pair. In the remaining experiments we use the contiguous loss for both training and attention optimization.

While we used a kernel of size 2x2 in our experiments, we also looked at different sizes. Using a 1x1 kernel[6] during attention optimization leads to an AER of 22.8%, while a 3x3 kernel achieves the best result with an AER of 21.2%, compared to 21.5% of the 2x2 kernel. Larger kernel sizes lead to slightly worse results: 21.4% for a 4x4 kernel and 21.5% for a 5x5 kernel.

6.4 Bidirectional Attention Optimization

The most commonly used methods to merge alignments from models trained in opposite directions are variants of grow-diagonal. We extract hard alignments for both German→English and English→German with (monolingual) attention optimization, which leads to an AER of 21.5% and 25.6%, respectively. Merging these alignments with grow-diagonal leads to an AER of 19.6%, while grow-diagonal-final yields an AER of 19.7%.

We tuned the interpolation factor λ for the contiguity loss during bidirectional optimization. A parameter of 1.0 leads to an AER of 18.2%, 2.0 leads to 18.0% while 5.0 leads to 17.9%. Compared to unidirectional attention optimization it makes sense to pick a higher interpolation factor for the contiguity loss, as it is applied with the loss of the forward *and* backward model.

For the remaining experiments we use 5.0 as the interpolation factor. Bidirectional attention optimization improves the resulting alignment error rate compared to the grow-diagonal heuristic by up to 1.8% for German→English. These results are summarized in Table 2.

Variants of grow-diagonal have to rely on the hard alignments generated by the forward and the backward model. They only choose from these alignment links and therefore do not have the ability to generate new alignment links.

In contrast, bidirectional attention optimization takes the parameters of the underlying models into account and optimizes the underlying attention logits simultaneously for both models to fit the sentence pair. In the example in Figure 8 bidirectional attention optimization is able to correctly predict

Method	No Contiguity	Contiguity
Forward	34.2%	27.8%
Att. Opt	22.7%	21.5%

Table 1: AER results with and without using the contiguity loss when extracting alignments from the forward pass or when using attention optimization for the language pair German→English.

	AER
DeEn	21.5%
EnDe	25.6%
Grow-diag	19.6%
Grow-diag-final	19.7%
Bidir. Att. Opt	17.9%

Table 2: Comparison of AER scores between bidirectional attention optimization and methods to merge hard alignments.

[6] A 1x1 only encourages sparse alignments, and does not encourage contiguous alignments.

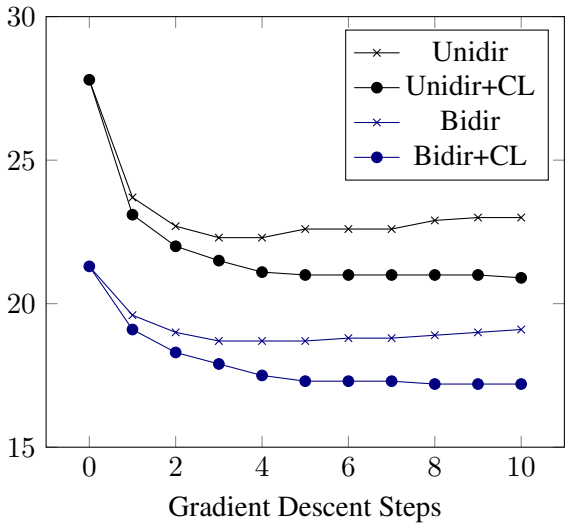


Figure 6: AER with respect to gradient descent steps during attention optimization for German→English. Both unidirectional (Unidir) and bidirectional (Bidir) optimization benefit from the contiguity loss (CL). Without the contiguity loss AER slightly degrades after more than three optimization steps.

an alignment link between "übereinstimmend" and "proven" that did not appear at all in the individual alignments of the forward and backward model.

We plot the behavior of attention optimization with a varying number of gradient descent steps in Figure 6. For both unidirectional and bidirectional models attention optimization leads to steadily improving results. Without using the additional contiguity loss, the lowest AER appears after three gradient descent steps and slightly increases afterwards. When using the contiguity loss AER results continue to decrease with additional steps. The contiguity loss seems to stabilize optimization and avoids overfitting of the optimized attention activations when tuning them for a single sentence pair.

6.5 Guided Alignment Training

We now use the alignment layer with the full decoder context by adding an additional self-attention layer that does not mask out the future target context. We extract alignments from the previous models with bidirectional attention optimization and use those alignments for guided alignment training.

This works surprisingly well. While the alignments used for training yielded an AER of 17.9% after bidirectional attention optimization (Table 4), the full context model trained with these alignments further improved the AER to 16.0% while using a single model for German→English (Table 3). After guided alignment training is complete, we do not apply attention optimization, since that would require a distribution over target words, which is not available in this model.

Method	DeEn	EnFr	RoEn
Att. Opt.	21.5%	15.0%	29.2%
+Guided	16.0%	6.6%	23.4%
Zenkel et al. (2019)	26.6%	23.8%	32.3%
GIZA++	18.9%	7.9%	27.3%

Table 3: Comparison of unidirectional models with GIZA++.

Method	DeEn	EnFr	RoEn
Bidir. Att. Opt.	17.9%	8.4%	24.1%
+Guided	16.3%	5.0%	23.4%
Zenkel et al. (2019)	21.2%	10.0%	27.6%
Garg et al. (2019)	20.2%	7.7%	26.0%
GIZA++	18.7%	5.5%	26.5%

Table 4: Comparison of neural alignment approaches with GIZA++ after using symmetrization of the forward and backward model.

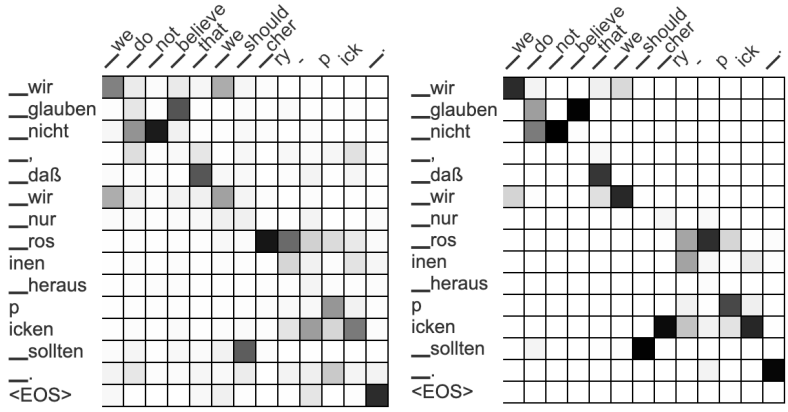
6.6 End-to-End Results

We now report AER results across all three language pairs. Precision and recall scores are included in the Supplemental Material. We first extract alignments from a unidirectional model, a common use case where translations and alignments need to be extracted simultaneously. Table 3 compares our results to GIZA++ and Zenkel et al. (2019).[7] We observe that guided alignment training leads to gains across all language pairs. In a single direction our approach consistently outperforms GIZA++ by an absolute AER difference between 1.3% (EnFr) and 3.9% (RoEn).

Table 4 compares bidirectional results after symmetrization. We compare to purely neural and purely statistical systems.[8] For symmetrizing alignments of the guided model and GIZA++, we use grow-diagonal. Bidirectional attention optimization is already able to outperform GIZA++ and Garg et al. (2019) on all language pairs except English→French. Using guided alignment training further improves results across all language pairs

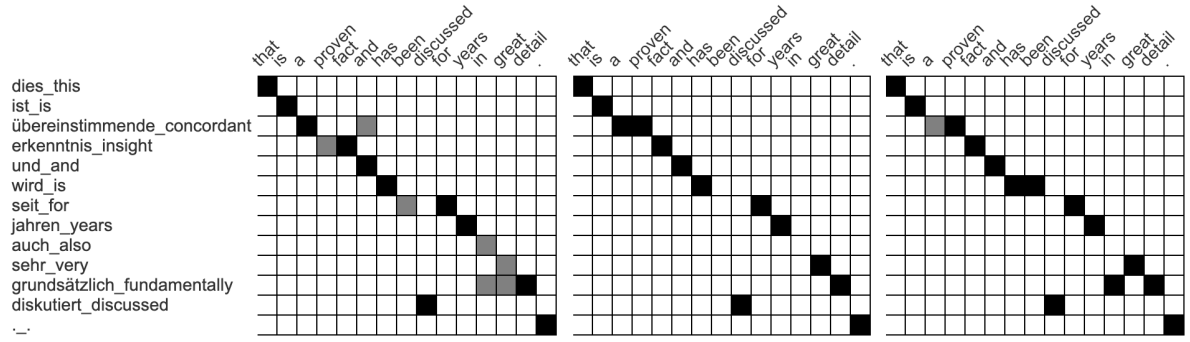
[7] Garg et al. (2019) only report bidirectional results after symmetrization.

[8] For additional comparisons including neural models bootstrapped with GIZA++ alignments, see the Supplemental Material.



(a) Without Contiguity Loss (b) With Contiguity Loss

Figure 7: Attention activations of the alignment layer after attention optimization. Using the contiguity loss during training leads to sparse activations, the correct alignment of the two occurrences of "we"-"wir" and to correct alignment of the period.



(a) Intersection/Union (b) Bidir. Optimization (c) Gold Alignments

Figure 8: Example of symmetrization with bidirectional attention optimization. We show all alignments extracted from the forward and backward direction with unidirectional attention optimization in Subfigure 8a (alignments that are only present in one direction are grey). Bidirectional attention optimization is able to extract the correct alignment between "übereinstimmend" and "proven" which did neither appear as an alignment link in the forward nor in the backward direction.

and leads to a consistent AER improvement compared to GIZA++ and neural results reported by Garg et al. (2019).

These results show that it is possible to outperform GIZA++ both in a single direction and after symmetrization without using any alignments generated from statistical alignment systems to bootstrap training.

7 Conclusion

This work presents the first end-to-end neural approach to the word alignment task which consistently outperforms GIZA++ in terms of alignment error rate. Our approach extends a pre-trained state-of-the-art neural translation model with an additional alignment layer, which is trained in isolation without changing the parameters used for the translation task. We introduce a novel auxiliary loss function to encourage contiguity in the alignment matrix and a symmetrization algorithm that jointly optimizes the alignment matrix within two models which are trained in opposite directions. In a final step the model is re-trained to leverage full target context with a guided alignment loss. Our results on three language pairs are consistently superior to both GIZA++ and prior work on end-to-end neural alignment. As the resulting model repurposes a pre-trained translation model without changing its parameters, it can directly benefit from improvements in translation quality, e.g. by adaptation via fine-tuning.

References

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. *Proceedings of the Third Conference on Machine Translation*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 420–429. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Carla Parra Escartın and Manuel Arcedillo. 2015. Machine translation evaluation made fuzzier: A study on post-editing productivity and evaluation metrics in commercial settings. *Proceedings of MT Summit XV*, page 131.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software engineering, testing, and quality assurance for natural language processing*, pages 49–57. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4452–4461, Hong Kong, China. Association for Computational Linguistics.

João Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In *Advances in neural information processing systems*.

Eric Joanis, Darlene Stewart, Samuel Larkin, and Roland Kuhn. 2013. Transferring markup tags in statistical machine translation: A two-stream approach. *Machine Translation Summit XIV*, page 73.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *International Workshop on Spoken Language Translation (IWSLT) 2005*.

Xintong Li, Guanlin Li, Lemao Liu, Max Meng, and Shuming Shi. 2019. On the word alignment from neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1293–1303, Florence, Italy. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics.

I Dan Melamed. 1998. Annotation style guide for the blinker project. *arXiv preprint cmp-lg/9805004*.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts data driven machine translation and beyond*. Association for Computational Linguistics.

Mathias Müller. 2017. Treatment of markup in statistical machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 36–46.

Franz Josef Och and Hermann Ney. 2000a. A comparison of alignment models for statistical machine translation. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, volume 2.

Franz Josef Och and Hermann Ney. 2000b. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Elias Stengel-Eskin, Tzu-ray Su, Matt Post, and Benjamin Van Durme. 2019. A discriminative neural model for cross-lingual word alignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 910–920, Hong Kong, China. Association for Computational Linguistics.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Arda Tezcan and Vincent Vandeghinste. 2011. Smtcat integration in a technical domain. handling xml mark-up using pre and post-editing processing methods. In *Proceedings of the 15th International Conference of the European Association for Machine Translation (EAMT-2011)*, page 8. Centre for Computational Linguistics; Leuven.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

A Supplemental Material

Table 5 and Table 6 summarize the hyperparameters used for the translation model and the additional alignment layer. In Table 7 we report both AER results and precision and recall for all language pairs.

Hyperparameter	Value
Dropout Rate	0.1
Embedding Size	256
Hidden Units	512
Encoder Layers	6
Decoder Layers	3
Attention Heads Per Layer	8

Table 5: Hyperparameters of the translation model.

Hyperparameter	Value
Dropout Rate	0.1
Embedding Size	256
Hidden Units	256
Attention Heads	1

Table 6: Hyperparameters of the alignment layer.

Method	DeEn	EnDe	Bidir	EnFr	FrEn	Bidir	RoEn	EnRo	Bidir
Att. Opt.	21.5%	25.6%	17.9%	15.0%	14.3%	8.4%	29.2%	28.8%	24.1%
	76/81	73/76	85/79	81/92	82/93	90/95	74/68	74/69	85/69
Guided	16.0%	16.6%	16.3%	6.6%	6.3%	5.0%	23.4%	23.1%	23.4%
	88/80	89/78	93/76	92/95	93/95	96/94	88/68	90/67	93/65
GIZA++ word	20.9%	23.1%	21.4%	8.0%	9.8%	5.9%	28.7%	32.2%	27.9%
	86/72	87/69	94/67	91/93	92/88	98/90	83/63	80/59	94/59
GIZA++ subword	18.9%	20.4%	18.7%	7.9%	8.5%	5.5%	27.3%	29.4%	26.5%
	89/74	88/72	95/71	92/93	93/89	98/91	85/64	83/62	93/61
Zenkel et al. (2019)	26.6%	30.4%	21.2%	23.8%	20.5%	10.0%	32.3%	34.8%	27.6%
Garg et al. (2019)	n/a	n/a	20.2%	n/a	n/a	7.7%	n/a	n/a	26.0%
+ GIZA++	n/a	n/a	16.0%	n/a	n/a	4.6%	n/a	n/a	23.1%

Table 7: AER and—when available—precision/recall scores in percentage in the following row. The *Bidir* column reports results for the DeEn, EnFr and RoEn translation direction, respectively, and uses grow-diagonal for all columns except when attention optimization is used. For attention optimization we merge alignments with bidirectional attention optimization.