Accurate Word Alignment Induction from Neural Machine Translation

Yun Chen[*1], **Yang Liu**[2], **Guanhua Chen**[3], **Xin Jiang**[4], **Qun Liu**[4]
[1]Shanghai University of Finance and Economics, Shanghai, China
[2]Tsinghua University, Beijing, China
[3]The University of Hong Kong, Hong Kong, China
[4]Huawei Noah's Ark Lab, Hong Kong, China
yunchen@sufe.edu.cn, liuyang2011@tsinghua.edu.cn,
ghchen@eee.hku.hk, {jiang.xin, qun.liu}@huawei.com

Abstract

Despite its original goal to jointly learn to align and translate, prior researches suggest that the state-of-the-art neural machine translation model Transformer captures poor word alignment through its attention mechanism. In this paper, we show that attention weights do capture accurate word alignment, which could only be revealed if we choose the correct decoding step and layer to induce word alignment. We propose to induce alignment with the to-be-aligned target token as the decoder input and present two simple but effective interpretation methods for word alignment induction, either through the attention weights or the leave-one-out measures. In contrast to previous studies, we find that attention weights capture better word alignment than the leave-one-out measures under our setting. Using the proposed method with attention weights, we greatly improve over fast-align on word alignment induction. Finally, we present a multi-task learning framework to train the Transformer model and show that by incorporating GIZA++ alignments into our multi-task training, we can induce significantly better alignments than GIZA++.

1 Introduction

The task of word alignment is to find lexicon translation equivalents from parallel corpus. It is one of the fundamental tasks in natural language processing and is widely studied by the community (Dyer et al., 2013; Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003; Liu and Sun, 2015). Word alignments are useful in many scenarios, such as error analysis (Ding et al., 2017; Li et al., 2019), the introduction of coverage and fertility models (Tu et al., 2016), inserting external constraints in interactive machine translation (Hasler et al., 2018) and providing guidance for human translators in computer-aided translation (Dagan et al., 1993).

Word alignment is part of the pipeline in statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005), but is not necessarily needed for neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). The attention mechanism (Bahdanau et al., 2015) in NMT does not functionally play the role of word alignment between the source and the target, at least not in the same way as its analog in SMT. It is hard to interpret the attention activations and extract meaningful word alignments, especially from Transformer which uses multiple attention components for each of the stacked decoder layers. As a result, the most widely used word alignment tools are still external statistical models such as fast-align (Dyer et al., 2013) and GIZA++ (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003).

Recently, there is a resurgence of interest in the community to study word alignment for the Transformer. One simple solution is to induce word alignments from the attention weights between the encoder and decoder. The attention weights are averaged across all heads from the penultimate decoder layer. Then the next target word is aligned with the source word that has the maximum attention weights. However, such schedule only captures noisy word alignment (Alkhouli et al., 2018; Li et al., 2019; Ding et al., 2019; Garg et al., 2019). One of the major problems is that it induces alignment before observing the to-be-aligned target token (Peter et al., 2017; Ding et al., 2019). Suppose for the same source sentence, there are two alternative translations that diverge at decoding step i, generating y_i and y_i' which respectively correspond to different source words. Presumably, the source word that is aligned to y_i and y_i' should change correspondingly. However, this is not possible under

* Corresponding author. Part of the work was done when Yun was in Huawei Noahs Ark Lab.

the above method, because the alignment scores are computed before prediction of y_i or y'_i.

To alleviate this problem, some researchers modify the transformer architecture by adding alignment modules that predicts the to-be-aligned target token (Zenkel et al., 2019) or the training loss by adding an additional alignment loss which is computed with full target sentence in a multi-task learning framework (Garg et al., 2019). Others argue that using only attention weights is **insufficient** for generating clean word alignment and propose to induce word alignment with feature importance measures, such as leave-one-out (LOO) measures (Li et al., 2019) and gradient-based measures (Ding et al., 2019; Garg et al., 2019). However, all previous work induce alignment at the decoding step when the to-be-aligned target token is the decoder output.

In this work, we propose to induce word alignment at the decoding step when the to-be-aligned target token is the decoder input instead of the output. In this way, we can incorporate the information of the to-be-aligned target token easily. Specifically, we present two novel methods for alignment induction, one with attention weights and one with LOO measures. These methods are pure interpretation methods and do not require any parameter update or architecture change. We demonstrate that if the correct decoding step and layer is chosen, attention weights in vanilla Transformer is much more effective than previous known (Alkhouli et al., 2018; Li et al., 2019; Ding et al., 2019; Garg et al., 2019) and **sufficient** for generating clean word alignment interpretation. It is able to reduce the Alignment Error Rate (AER) by 6.9-9.8 points over the naive attention weights baseline and 3.4-6.7 points over fast-align under three evaluation sets. Our method is also complementary to the multi-task learning framework proposed by Garg et al. (2019). We demonstrate a combination of our method and the multi-task learning can further improve alignment performance, reducing AER by 0.4-2.7 points over GIZA++.

2 Background

2.1 Neural Machine Translation

Let $\mathbf{x} = \{x_1, ..., x_{|\mathbf{x}|}\}$ and $\mathbf{y} = \{y_1, ..., y_{|\mathbf{y}|}\}$ be source and target sentences. Neural machine translation models the target sentence given the source sentence as $p(\mathbf{y}|\mathbf{x};\theta)$, where θ is a set of model parameters to be learned. The negative log-likelihood (NLL) training loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{NLL}} &= -\log p(\mathbf{y}|\mathbf{x};\theta) \\ &= -\sum_{t=1}^{|\mathbf{y}|+1} \log p(y_t|y_{0:t-1}, \mathbf{x};\theta), \quad (1)\end{aligned}$$

where $y_0 = \langle\text{bos}\rangle$ and $y_{|\mathbf{y}|+1} = \langle\text{eos}\rangle$ represent the beginning and end of target sentences respectively.

The NMT model can be implemented with different architectures. In this paper, we use Transformer (Vaswani et al., 2017). Transformer is an encoder-decoder model that only relies on attention. Each decoder layer attends to the encoder output with multi-head attention, which consists of N attention heads running in parallel.

2.2 Alignment by Attention

The encoder output from the last encoder layer is denoted as $\mathbf{h} = \{h_1, ..., h_{|\mathbf{x}|}\}$, and the hidden states at decoder layer l as $\mathbf{z} = \{z_1^l, ..., z_{|\mathbf{y}|+1}^l\}$. For decoder layer l, we define the layer averaged target-to-source attention weights as $\mathbf{W}^l$:

$$\begin{aligned}\mathbf{W}^l &= \frac{1}{N}\sum_n \mathbf{W}_n^l \\ \mathbf{W}_n^l &= \text{softmax}(\frac{\mathbf{Q}_n^l \mathbf{U}_n^{l\top}}{\sqrt{d_u}}), \quad (2)\end{aligned}$$

where $\mathbf{Q}_n^l$, $\mathbf{U}_n^l$ is the query and key matrix for head n of the target-to-source attention at decoder layer l, and the element $\mathbf{W}_{i,j}^l$ in $\mathbf{W}^l$ measures the relevance between decoder hidden state z_i^l and encoder output h_j. For simplicity, below we use the term "attention weights" to denote the layer averaged target-to-source attention weights.

Given a trained Transformer model, word alignment can be extracted from the attention weights (Alkhouli et al., 2018; Li et al., 2019; Ding et al., 2019; Garg et al., 2019). More specifically, word alignment $\mathbf{A}$ is extracted from attention weights $\mathbf{W}^l$ according to the style of maximum a posterior strategy (MAP) as follows:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } j = \text{argmax}_{j'}\, \mathbf{W}_{i,j'}^l \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $\mathbf{A}_{ij} = 1$ indicates y_i is aligned to x_j. Garg et al. (2019) show that attention weights from the penultimate layer, i.e., $l = L - 1$, can induce the best alignments.

Although simple to implement, this method fails to obtain satisfactory word alignment (Alkhouli

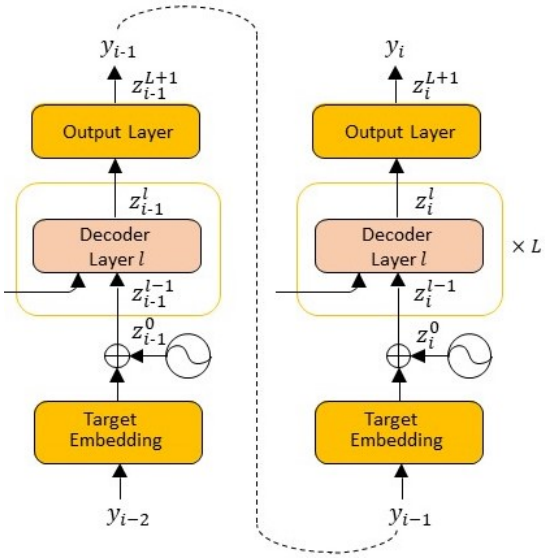


Figure 1: The forced decoding process illustration with Transformer at decoding time step $i-1$ and i.

et al., 2018; Li et al., 2019; Ding et al., 2019; Garg et al., 2019). First of all, $\mathbf{W}_{i,j}^{l}$ does not naturally measure the relevance between y_i and x_j. It measures the relevance between decoder hidden state z_i^l and encoder output h_j. However, at decoding step i, the decoder input is y_{i-1} and the output is y_i. Decoder representation z_i^l may better represent y_{i-1} instead of y_i, especially for the bottom layers. Second, since the attention weight $\mathbf{W}_{i,j}^{l}$ is computed before observing y_i, it becomes difficult for it to learn the target token y_i's alignment to the source tokens, as discussed in Section 1.

As a result, it is necessary to develop novel methods for alignment induction. This method should be able to (1) take into account the relationship of z_i^l, y_i and y_{i-1}, and (2) adapt the alignment induction with the to-be-aligned target token.

3 Alignment from Vanilla Transformer

We propose a novel framework to induce alignment from vanilla Transformer model. We first discuss how to represent the to-be-aligned target token at each decoder layer, then present the alignment induction method including: 1) alignment induction with attention weights or LOO measure and 2) layer selection criterion that determines the layer to induce alignment. We denote the method that induces alignment with attention weights as *align-att* and LOO measure as *align-loo*.

3.1 Target Token Representation

As shown in Fig. 1, we denote the input to the first decoder layer at decoding step i as z_i^0, the decoder hidden state as z_i^l ($1 \leq l \leq L$) and the probability distribution $p(y|y_{0:i-1}, \mathbf{x}; \theta)$ at the output layer as z_i^{L+1}. Note that $z_i^l \in \mathbb{R}^d$ for $0 \leq l \leq L$, while $z_i^{L+1} \in \mathbb{R}^V$. Regardless of the value of l, previous work (Li et al., 2019; Ding et al., 2019; Garg et al., 2019) compute the alignment scores for target token y_i using the attention weights or feature importance measures associated with decoder representation z_i^l. This is not convincing, especially for small values of l.

At decoding step i, the input to the decoder is y_{i-1}, while at the output layer, it predicts the probability of y_i. Therefore, we assume that z_i^0 is more relevant to y_{i-1}, while z_i^{L+1} is more relevant to y_i. With l increasing from 0 to $L+1$, the decoder gathers more information from the context, and may gradually change its representation from that more relevant to y_{i-1} to that more relevant to y_i. Therefore, for small values of l, z_i^l may better represent the input token y_{i-1} than the output token y_i.

In order to compute the alignment score of the target token after knowing its identity, we propose to induce alignment from decoder representations with the to-be-aligned target token as the input. Specifically, we represent y_i as z_{i+1}^l and induce word alignment from some layer l[b]. Since knowing the target token y_i is important for accurate alignment induction (Section 1), we believe our method can induce better word alignment.

3.2 Alignment Induction

Given a source sentence $\mathbf{x} = \{x_1, ..., x_{|\mathbf{x}|}\}$ and a target sentence $\mathbf{y} = \{y_1, ..., y_{|\mathbf{y}|}\}$, we first define the matrix of alignment scores as $\mathbf{S}^l \in \mathbb{R}^{|\mathbf{y}| \times |\mathbf{x}|}$, in which an element $\mathbf{S}_{i,j}^l$ is the alignment score of source word x_j to target word y_i, computed according to the relevance of x_j and z_{i+1}^l, the representation of y_i at the l-th layer:

$$\mathbf{S}_{i,j}^l = r\big(z_{i+1}^l, x_j\big). \tag{4}$$

With this alignment scoring matrix $\mathbf{S}^l$, we can generate word alignment $\mathbf{A}$ following (Alkhouli et al., 2018; Li et al., 2019; Ding et al., 2019; Garg et al., 2019):

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } j = \text{argmax}_{j'}\, \mathbf{S}_{i,j'}^l \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{A}_{ij} = 1$ indicates y_i is aligned to x_j. The relevance score in Eq. 4 can be defined either with attention weights or LOO measures.

Attention Weights In Transformer, the attention weights $\mathbf{W}^l_{i+1,j}$ measures the relevance between h_j and z^l_{i+1}. Since h_j represents the source word x_j, we can use the attention weights to define the relevance score:

$$r(z^l_{i+1}, x_j) = \mathbf{W}^l_{i+1,j}, \tag{6}$$

where $1 \leq l \leq L$ because attention weights are computed at these layers.

LOO Measure The intuition to this method is that when the decoder representation z^l_{i+1} is aligned to the source word x_j, the influence of masking x_j should be much higher for z^l_{i+1} than other representations at the same layer. This approach is called Leave-One-Out (LOO). More specifically, we define the alignment score of source word x_j to decoder representation z^l_{i+1} as:

$$r(z^l_{i+1}, x_j) = \text{Dis}(u, v) \tag{7}$$

where $u = z^l_{i+1}(\mathbf{y}_{<i+1}, \mathbf{x})$, $v = z^l_{i+1}(\mathbf{y}_{<i+1}, \mathbf{x}_{(j,\mathbf{0})})$ and $\mathbf{x}_{(j,\mathbf{0})}$ denotes the source sentence by replacing x_j with a word whose embedding is a zero vector. We use European distance to define the distance for layer $1 \leq l \leq L$ and KL divergence for layer $l = L + 1$:

$$\text{Dis}(u, v) = \begin{cases} ||u - v||_2 & \text{if } 1 \leq l \leq L \\ D_{\text{KL}}(u|v) & \text{otherwise} \end{cases} \tag{8}$$

In summary, we induce alignment from layer l with

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } j = \text{argmax}_{j'}\, r(z^l_{i+1}, x_j) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $r(z^l_{i+1}, x_j)$ is defined in Eq. 6 for the method *align-att* and Eq. 7 and 8 for the method *align-loo*.

3.3 Layer Selection Criterion

In this part, a surrogate layer selection criterion is proposed to select the best layer to induce alignment without manually labelled word alignments. Experiments show that this criterion correlates well with AER metric.

Given parallel sentence pairs $\langle \mathbf{x}, \mathbf{y} \rangle$, we train a source-to-target model $\theta_{\mathbf{x}\to\mathbf{y}}$ and a target-to-source model $\theta_{\mathbf{y}\to\mathbf{x}}$. We assume that the word alignment induced from these two models should agree with each other (Cheng et al., 2016). Therefore, we evaluate the quality of the alignment by computing the AER score on the validation set with the source-to-target alignment as the hypothesis and the target-to-source alignment as the reference. For each model, we can obtain K word alignments from K different layers ($K = L$ for *align-att* and $K = L + 1$ for *align-loo*). In total, we obtain $K \times K$ AER scores. We select the one with the lowest AER score, and its corresponding layers of the source-to-target and target-to-source models are the layers we will use to induce word alignment at test time:

$$l^{\text{b}}_{\mathbf{x}\to\mathbf{y}}, l^{\text{b}}_{\mathbf{y}\to\mathbf{x}} = \text{argmin}_{i,j}\, \text{AER}(\mathbf{A}^i_{\mathbf{x}\to\mathbf{y}}, \mathbf{A}^j_{\mathbf{y}\to\mathbf{x}}) \tag{10}$$

4 Alignment with Multi-task Learning

Inspired by Garg et al. (2019), we present a multi-task learning framework to train a novel Transformer model that can incorporate external alignments from GIZA++. With the guidance of GIZA++ alignments, this method can further improve word alignment induction. Note that this method is not an interpretation method for vanilla Transformer as it modifies the training process.

Specifically, we add an alignment loss to the regular loss function, supervising one attention head at the decoding step when the to-be-alignment target token is the decoder input and layer l^{b}, the best layer selected with Eq. 10, to learn GIZA++ alignments. The alignment loss is defined as:

$$\mathcal{L}_a = -\frac{1}{|\mathbf{y}|} \sum_{i=1}^{|\mathbf{y}|} \sum_{j=1}^{|\mathbf{x}|} \left(\mathbf{G}^p \odot \log \mathbf{W}^{l^{\text{b}}}_{n_0}\right)_{i,j}, \tag{11}$$

where $\mathbf{W}^{l^{\text{b}}}_{n_0}$ is the attention matrix computed by an arbitrary alignment head n_0 from the layer l^{b}, and $\mathbf{G}^p$ denotes the normalized reference alignment extracted with GIZA++ [1]. Combing with the regular NLL loss, we train the Transformer with:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda \mathcal{L}_a, \tag{12}$$

where λ is a hyper-parameter to balance these two losses. At test time, we extract word alignment from the same head n_0 and the same layer l^{b} with:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } j = \text{argmax}_{j'}(\mathbf{W}^{l^{\text{b}}}_{n_0})_{i+1,j'} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

[1] We simply normalize rows corresponding to target tokens that are aligned to at least one source token of the alignment matrix $\mathbf{G}$ extracted with GIZA++.

Similar with Garg et al. (2019), the alignment loss can be computed at the same forward pass as computing the regular NLL loss or with another forward pass that removes the future mask at the decoder side (full-context). Full-context improves alignment extraction under their setting, because it considers the future context especially the to-be-aligned target token when computing alignment scores. However, it also brings two problems. First, two forward training introduces additional computation cost at training stage. Second, with future context, the alignment is run as a separate post-processing step after the full sentence translation is completed. Therefore, methods with full-context are not suitable for translation error analysis and cases where alignment is computed during the decoding process, e.g. constrained decoding with attention (Hasler et al., 2018). Since our method already computes the alignment scores after observing the to-be-aligned target token, we believe future context is not that necessary under our setting. We compare our method with (*align-att-mtl*) or without full-context (*align-att-mtl-fullc*) in the experiments.

5 Experiments

5.1 Settings

Dataset For fair comparison, we follow Zenkel et al. (2019) in data setup. We evaluate all approaches on German-English, Romanian-English and French-English language pairs by measuring AER against the human alignment reference. Since no validation set is provided in their setup, we follow Ding et al. (2019) to set the last 1,000 sentences of the training data before preprocessing as validation set, which is used for training and best alignment layer selection. All models are trained in both translation directions and symmetrized with *grow-diag-final* (Koehn et al., 2005). See appendix for details.

NMT Systems We follow Ding et al. (2019) for training the NMT model. Specifically, we use fairseq-py[2] to train the Transformer model, following the setting in *transformer_iwslt_de_en*.

Baselines We have proposed methods to induce word alignment from the Transformer, two (*align-att* and *align-loo*) from vanilla Transformer trained with regular NLL loss, and two (*align-att-mtl* and *align-att-mtl-fullc*) from Transformer trained with multi-task learning framework. We compare our methods with 12 baselines:

• *fast-align offline*: the fast-align method (Dyer et al., 2013) under offline setting [3].

• *fast-align online*: the fast-align method (Dyer et al., 2013) under online setting. Since all the neural methods including ours are run in an online setting at test time, we believe this is a slightly better baseline to compared with.

• *GIZA++*: GIZA++ method (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003). It is an offline method [4].

• *Attention*: the Attention method reported in Ding et al. (2019), extracting alignment from the last decoder layer.

• *SmoothGrad*: the SmoothGrad method reported in Li et al. (2016).

• *SD-SmoothGrad*: the saliency-driven SmoothGrad method reported in Ding et al. (2019).

• *PD*: the prediction difference (PD) model reported in Li et al. (2019).

• *naive-loo*: we follow the same process as *align-loo*, except that we use z_i^l to represent y_i. We find that $l^{\text{b}} = L + 1$ for all translation tasks.

• *naive-att*: we follow the same process as *align-att*, except that we use z_i^l to represent y_i. We find that $l^{\text{b}} = L - 1$ for all translation models.

• *naive-att-mtl*: the multi-task learning method without full-context reported in Garg et al. (2019).

• *naive-att-mtl-fullc*: the multi-task learning method with full-context reported in Garg et al. (2019).

• *AddSGD*: the method that explicitly adds an alignment module to Transformer reported in Zenkel et al. (2019).

For each sentence pair, *naive-loo*, *align-loo* and *PD* forward once with $|\mathbf{x}| + 1$ masked sentence pairs as the input, while *SmoothGrad* and *SD-SmoothGrad* forward and backward once with m ($m = 30$ in Ding et al. (2019)) noisy sentence pairs as the input. All the other methods forward once with one sentence pair as the input. Therefore, the computation cost for methods with attention weights is much lower compared to methods with feature importance measures.

[2] https://github.com/pytorch/fairseq

[3] https://github.com/clab/fast_align

[4] https://github.com/moses-smt/giza-pp

Method	Modify	Fullc	de-en			fr-en			ro-en		
			de→en	en→de	bidir	fr→en	en→fr	bidir	ro→en	en→ro	bidir
Existing Statistical Methods											
fast-align offline	-	Y	28.5	30.4	25.7	16.3	17.1	12.1	33.6	36.8	31.8
fast-align online	-	Y	30.4	32.3	27.4	16.8	19.5	13.2	36.2	40.9	33.4
GIZA++	-	Y	18.8	19.6	17.8	7.1	7.2	6.1	27.4	28.7	26.0
Existing Neural Methods											
Attention (2019)	-	N	53.4	58.6	42.3	48.7	48.1	33.8	51.6	51.1	43.3
SmoothGrad (2016)	-	N	36.4	45.8	30.3	25.5	27.0	15.6	41.3	39.9	33.7
SD-SmoothGrad (2019)	-	N	36.4	43.0	29.0	25.9	29.7	15.3	41.2	41.4	32.7
PD (2019)	-	N	38.1	44.8	34.4	32.4	31.1	23.1	40.2	40.8	35.6
naive-loo	-	N	36.6	42.3	33.0	29.5	29.2	20.4	36.5	36.9	32.7
naive-att (2019)	-	N	33.3	36.5	29.3	30.7	24.6	18.5	34.2	36.8	32.0
naive-att-mtl (2019)	loss	N	23.5	25.3	21.3	16.3	15.2	11.1	31.0	28.3	28.3
naive-att-mtl-fullc (2019)	loss	Y	16.2	19.1	**15.9**	8.4	9.8	**5.6**	25.6	24.1	**23.6**
AddSGD (2019)	loss +arch	N	26.6	30.4	21.2	20.5	23.8	10.0	32.3	34.8	27.6
Our Neural Methods											
align-loo	-	N	31.5	34.5	24.4	25.4	27.3	13.9	34.2	34.6	28.7
align-att	-	N	20.9	25.7	20.4	14.1	17.0	8.7	29.1	26.7	25.1
align-att-mtl	loss	N	16.7	19.7	**16.0**	9.6	10.6	**5.7**	24.5	24.3	**23.3**
align-att-mtl-fullc	loss	Y	16.6	19.5	**16.0**	8.8	10.2	**5.6**	28.8	25.3	24.3

Table 1: AER on the test set with different alignment methods. *bidir* are symmetrized alignment results. Since we use the same dataset and preprocessing as Ding et al. (2019), for *Attention*, *SmoothGrad*, *SD-SmoothGrad* and *AddSGD*, the results are quoted from Ding et al. (2019). Since Garg et al. (2019) report that BPE is beneficial for statistical alignment models, which is consistent with our own experiments, we report the BPE-based fast-align and GIZA++ results. The column Modify represents whether the method modifies the training loss (loss) or the Transformer architecture (arch). The column Fullc denotes whether full-context is used to induce alignment at test time. Best bidir neural results that induce alignment without modification of loss or architecture are marked with underlines, and comparable best bidir results among all methods are marked with boldface.

5.2 Results

Comparison with Baselines Table 1 compares our methods with all the baselines. It shows that our method *align-att* significantly outperforms all neural baselines that extract alignment from vanilla Transformer (Modify: -). For example, it outperforms *SD-SmoothGrad*(2016), the state-of-the-art method to extract alignment from vanilla Transformer, by 6.6-8.6 AER scores across different language pairs, not to mention that the induction computation cost of *align-att* is much less than *SD-SmoothGrad*. Our method *align-att* also outperforms *align-loo* in terms of both alignment quality and induction computation cost. The success of *align-att* shows that vanilla Transformer has captured alignment information in an implicit way, which could be revealed if the correct decoding step and layer are chosen to induce alignment. we also compare our methods with statistical alignment methods. *align-att* significantly outperforms *fast-align*, both under online and offline settings. It achieves slightly worse performance compared with GIZA++. Since *align-att* is better than *align-loo* across all datasets, we focus on experiments with *align-att* in the remaining sections.

We also compare our multi-task learning methods with Garg et al. (2019). Our methods with or without full-context obtain similar results, which is different from Garg et al. (2019). Their method with full context (*naive-attn-mtl-fullc*) obtains significant better results than that without full context (*naive-attn-mtl*). In *naive-attn-mtl*, alignment scores for target token y_i is computed before observing y_i. The rest target tokens, especially y_i, are very important for better alignment. However, in our method *align-attn-mtl*, the alignment scores are already computed after observing y_i, deemphasizing the rest target tokens. Our method without full-context (*align-attn-mtl*) obtains comparable result with the baseline method with full-context (*naive-attn-mtl-fullc*) with less training cost and less reliance on full-context. Finally, *align-attn-mtl* also significantly outperforms all statistical methods, improving over GIZA++ with 0.4-2.7 AER scores. In summary, our method *align-attn-mtl* achieves the best alignment performance, with less computation cost and broader application scenarios.

(a) Validation AER for Layer Selection

de→en / en→de	1	2	3	4	5	6
1	42.2	35.4	35.7	67.5	89.2	88.8
2	45.1	39.5	39.1	67.1	87.8	88.2
3	42.5	34.6	**34.2**	65.2	87.4	87.6
4	74.4	73.0	72.3	80.6	89.5	89.7
5	84.8	86.7	86.1	87.3	88.7	88.9
6	87.1	88.2	87.6	88.1	88.7	88.6

(b) Test AER for Verification

layer	1	2	3	4	5	6
de→en	31.5	22.7	**20.9**	55.7	80.5	81.5
en→de	27.4	31.3	**25.7**	68.5	83.4	85.1

Table 2: Layer selection criterion verification with *align-att* on de-en alignment. (a) For each cell, we induce hypothesis alignment from de→en translation and reference alignment from en→de translation. $l^{\text{b}} = 3$ for both translation directions in this table. (b) Test AER when inducing word alignment from different layers. Layer 3 induces the best alignment for both translation directions, which verifies the value of l^{b} selected in (a).

Layer Selection Criterion To test whether the layer selection criterion can select the right layer to induce alignment, we first determine the best layer $l^{\text{b}}_{\mathbf{x}\to\mathbf{y}}$ and $l^{\text{b}}_{\mathbf{y}\to\mathbf{x}}$ based on the layer selection criterion. Then we evaluate the AER scores of alignment induced from different layers on the test set, and check whether the layers with the lowest AER score are consistent with $l^{\text{b}}_{\mathbf{x}\to\mathbf{y}}$ and $l^{\text{b}}_{\mathbf{y}\to\mathbf{x}}$. Table 2 demonstrate that with *align-att*, the layers selected by the layer selection criterion are indeed the best layer to induce alignment. We also verify the layer selection criterion for the method *align-loo*. See the appendix for the details.

Relevance Measure Verification To investigate the relevance between decoder hidden states and the corresponding input/output token, we design an experiment to prob whether the decoder hidden states contain the identity information of the input and output token, following Brunner et al. (2019). Formally, for decoder hidden state z^l_i, the input token is identifiable if there exists a function g such that $y_{i-1} = g(z^l_i)$. We cannot prove the existence of g analytically. Instead, for each layer l we learn a projection function $\hat{g}_l$ to project from the hidden state space to the input token embedding space $\hat{y}^l_i = \hat{g}_l(z^l_i)$ and then search for the nearest neighbour y_k within the same sentence. We say that the decoder hidden state can identify the input token if $k = i - 1$. Similarly, we follow the same process to project $\hat{z}^l_i$ into the output token embedding space and say that the decoder hidden state can identify the output token if $k = i$. We report the identifiability rate defined as the percentage of correctly identified tokens.

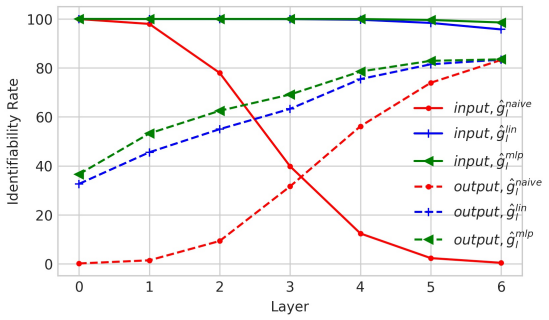
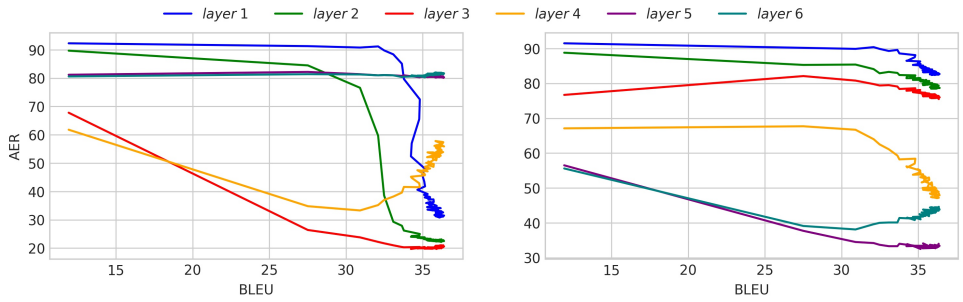


Figure 2: Identifiability rate of the input and output tokens for decoder hidden states at different layers.

Fig. 2 presents the result on the validation set of de→en translation. We try a naive baseline $\hat{g}^{\text{naive}}_l(z^l_i) = z^l_i$ and two projection functions: a linear perceptron $\hat{g}^{\text{lin}}_l$ and a non-linear multi-layer perceptron $\hat{g}^{\text{mlp}}_l$. The result shows that with trainable projection function $\hat{g}^{\text{lin}}_l$ and $\hat{g}^{\text{mlp}}_l$, all layers can identify the input tokens, although more hidden states cannot be mapped back to their input tokens anymore in higher layers. Besides, at the bottom layers, the input tokens remain identifiable and the output tokens are hard to identify, regardless of the projection function we use. This verifies that for bottom layers, the hidden states are more relevant to the input token than the output token. Finally, it is much easier to identify the input token. For example, when projecting with mlp, all layers can identify more than 98% of the input tokens. However, even for the best layer, we can only identify 83.5% of the output tokens. This observation verifies that computing alignment scores of target word y_i according to hidden state z^l_{i+1} is better than with z^l_i since z^l_i even may not be able to identify y_i.

AER v.s. BLEU During training, vanilla Transformer gradually learns to align and translate. To analyze how the alignment behavior changes at different layers and for checkpoints with different translation quality, we plot AER on the test set v.s. BLEU on the validation set for different layers and checkpoints on the de→en translation. We compare the baseline *naive-att* and our method *align-att* and induce alignments from all layers. *naive-att* aligns source tokens to the output token based on current decoder hidden state (align output token), while



(a) *align-att* (b) *naive-att*

Figure 3: AER on the test set v.s. BLEU on the validation set on the de→en translation, evaluated with different checkpoints.

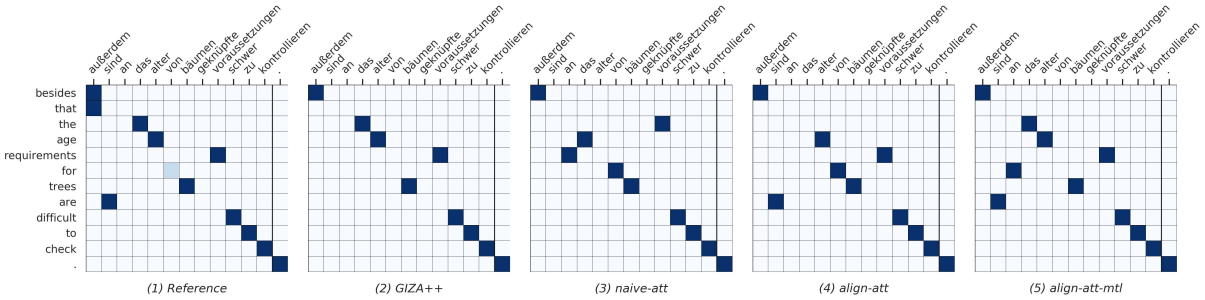


Figure 4: One example from de-en alignment test set. Golden alignments is shown in (1), blue squares and light blue squares represent *sure* and *possible* alignment separately.

align-att aligns source tokens to the input token (align input token).

The experiment results are show in Fig. 3. We observe that at the beginning of training, layer 3 and 4 learn to align the input token, while layer 5 and 6 learn to align the output token. However, with increasing of BLEU scores, layer 4 tend to change from aligning input token to aligning output token, and layer 1 and 2 begin to align input token. This indicates that the vanilla Transformer gradually learns to align the input token from middle layers to bottom layers. Besides, the ability of layer 6 to align output token decreases. We hypothesize that layer 5 already has the ability to attend to the source tokens that are aligned to the output token, therefore attention weights in layer 6 may capture other information needed for translation. Finally, for models with highest BLEU scores, layer 5 aligns the output token best and layer 3 aligns the input token best.

5.3 Alignment Example

In Fig. 4, we present an alignment example from de-en alignment test set. Manual inspection of this example as well as others we find that our method *align-att* and *align-att-mtl* tend to align more tokens than GIZA++, and better align the beginning tokens compared to *naive-att*. Besides, with multi-task learning to incorporate GIZA++ alignment, we find that *align-att-mtl* sometimes successfully induces aligned pairs that captured by GIZA++, but not *align-att*.

6 Related Work

Inducing word alignment from RNNSearch (Bahdanau et al., 2015) has been explored by a number of works. Bahdanau et al. (2015) is the first to show word alignment example by using attention in RNNSearch. Ghader and Monz (2017) further demonstrate that the RNN-based NMT system achieves comparable alignment performance to that of GIZA++. Alignment has also been used to improve NMT performance, especially in low resource settings, by supervising the attention mechanisms of RNNSearch (Ghader and Monz, 2017; Chen et al., 2016; Liu et al., 2016; Alkhouli and Ney, 2017).

There is also a number of other studies that induce word alignment from Transformer. Li et al. (2019); Ding et al. (2019) claim that attention may not capture word alignment in Transformer, and propose to induce word alignment with prediction

difference (Li et al., 2019) or gradient-based measure (Ding et al., 2019). Zenkel et al. (2019) modify the Transformer architecture for better alignment induction by adding an extra alignment module that is restricted to attend solely on the encoder information to predict the next word. Garg et al. (2019) propose a multi-task learning framework to improve word alignment induction without decreasing translation quality, by supervising one attention head at penultimate layer with GIZA++ alignments. Although these methods are reported to improve over layer average baselines, they ignore that better alignment can be induced by computing alignment scores at the decoding step when the to-be-aligned target token is the input, thus fail to fully induce the word alignment implicitly learned in Transformer.

7 Conclusion

In this paper, we have presented a novel framework for alignment induction from a vanilla Transformer model. The basic idea is that it is better to induce alignment at the decoding step when the to-be-aligned target token is the input instead of the output. Our method with attention weights successfully induces satisfactory word alignments from standard Transformer model, demonstrating that Transformer indeed jointly learns to align and translate. We also combine the multi-task learning framework with our method and show that by incorporating GIZA++ alignments into the Transformer, we can achieve significantly better alignment results compared to GIZA++, even without the use of future context.

References

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 177–185, Belgium, Brussels. Association for Computational Linguistics.

Tamer Alkhouli and Hermann Ney. 2017. Biasing attention-based recurrent neural networks using external alignment information. In *Proceedings of the Second Conference on Machine Translation*, pages 108–117.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR*.

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

Gino Brunner, Yang Liu, Damián Pascual, Oliver Richter, Massimiliano Ciaramita, and Rogert Wattenhofer. 2019. On identifiability in transformers. *arXiv e-prints*.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *AMTA 2016, Vol.*, page 121.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based joint training for bidirectional attention-based neural machine translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2761–2767. AAAI Press.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Ido Dagan, Kenneth Church, and Willian Gale. 1993. Robust bilingual word alignment for machine aided translation. In *VERY LARGE CORPORA: ACADEMIC AND INDUSTRIAL PERSPECTIVES*.

Shuoyang Ding, Hainan Xu, and Philipp Koehn. 2019. Saliency-driven word alignment interpretation for neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 1–12, Florence, Italy. Association for Computational Linguistics.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada. 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, Atlanta, Georgia. 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.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

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*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

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.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

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

Jan-Thorsten Peter, Arne Nix, and Hermann Ney. 2017. Generating alignments using target foresight in attention-based neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108:27–36.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA. Association for 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 Appendices

A.1 Data Statistics

Dataset	Train	Validation	Test
de-en	1,905,696	994	508
fr-en	1,129,050	1,000	447
ro-en	446,688	999	248

Table 3: Number of sentences in each dataset.

A.2 Layer Selection Criterion with *align-loo*

(a) Validation AER for Layer Selection

de→en / en→de	1	2	3	4	5	6	7
1	92.6	92.6	92.6	92.6	92.6	92.6	92.5
2	92.6	48.8	**47.0**	48.9	60.4	76.2	84.3
3	92.6	51.2	49.7	51.0	60.9	75.8	84.5
4	92.6	53.9	52.5	53.1	61.6	75.5	84.0
5	92.6	64.3	63.7	63.8	68.4	78.2	84.3
6	92.5	77.8	78.0	77.8	79.1	82.6	85.6
7	92.6	83.8	83.1	82.2	83.8	85.7	86.8

(b) Test AER for Verification

layer	1	2	3	4	5	6	7
de→en	89.7	36.2	**31.5**	32.8	47.2	66.8	74.8
en→de	90	**34.5**	38.8	41.3	54.9	73.2	79.1

Table 4: Layer selection criterion verification with *align-aloo* on de-en alignment. (a) For each cell, we induce hypothesis alignment from de→en translation and reference alignment from en→de translation. $l^{\text{b}} = 3$ for the de→en translation and $l^{\text{b}} = 2$ for the en→de translation in this table. (b) Test AER when inducing word alignment from different layers. Layer 3 induces the best alignment for the de→en translation and layer 2 the best for the reverse direction, which verifies the value of l^{b} selected in (a).

We also check whether our proposed layer selection criterion can select the right layer to induce alignment with the method *align-loo*. We follow the same process as in Section 5.2. First, we determine the best layer $l^{\text{b}}_{\mathbf{x}\to\mathbf{y}}$ and $l^{\text{b}}_{\mathbf{y}\to\mathbf{x}}$ based on the layer selection criterion on the validation set, as shown in Table 4 (a). Then we evaluate the AER scores of alignment induced from different layers on the test set (shown in Table 4 (b)), and find that the layers with the lowest AER score are consistent with $l^{\text{b}}_{\mathbf{x}\to\mathbf{y}}$ and $l^{\text{b}}_{\mathbf{y}\to\mathbf{x}}$. This experiment result further verifies the effectiveness of the unsupervised layer selection criterion.

A.3 Best Layer to Induce Alignment

We also list the best layer to induce word alignment with different methods, all under the configure of *transformer_iwslt_de_en*. It shows that compared with *align-loo* and *align-att*, *naive-loo* and *naive-att* tend to induce alignment from higher layers, which is consistent with our intuition in Section 3.1. Besides, we also observe that with attention weights as the relevance measure, the best layer to obtain alignment is relatively consistent for different translation models under the same configure. With the method *align-att*, we have $l^{\text{b}} = 3$ for all translation models. With the method *naive-att*, we have $l^{\text{b}} = 5$ for all translation models.

	de-en		fr-en		ro-en	
	de→en	en→de	fr→en	en→fr	ro→en	en→ro
naive-loo	7	7	7	7	7	7
naive-att	5	5	5	5	5	5
align-loo	3	2	4	2	4	4
align-att	3	3	3	3	3	3

Table 5: The best layer to induce alignment with different methods, based on the layer selection criterion.

A.4 Alignment with Transformer-Big

		de→en	en→de	de-en bidir	en→fr
Transformer-small	BLEU	33.20	25.80	-	34.84
	AER	20.9	25.7	20.4	17.0
	best layer	3	3	-	3
Transformer-big	BLEU	32.48	28.11	-	20.94
	AER	21.0	25.0	19.8	17.2
	best layer	2	2	-	3

Table 6: Comparison of Transformer-small and Transformer-big. Note that Transformer-small is trained on the datatset discussed in Section 5.1, while Transformer-big is provided by fairseq-py and trained on WMT19 for de→ en translation, WMT16 for en→ de translation and WMT14 for en→ fr translation.[5]

In this experiment, we test whether good word alignment can be extracted from the Transformer model under other configurations. Instead of training the model from scratch by ourselves, we utilize the pretrained Transformer models provided by fairseq-py. All the models are based on the configuration of *transformer_wmt_en_de_big*, therefore we denote them as Transformer-big. We use the pretrained Transformer models on WMT15 en→fr, WMT16 en→de and WMT19 de→en[6] translations. We denote the small Transformer we train as Transformer-small, and compare these two configurations in Table 6. It shows that although the BLEU score obtained by Transformer-small and

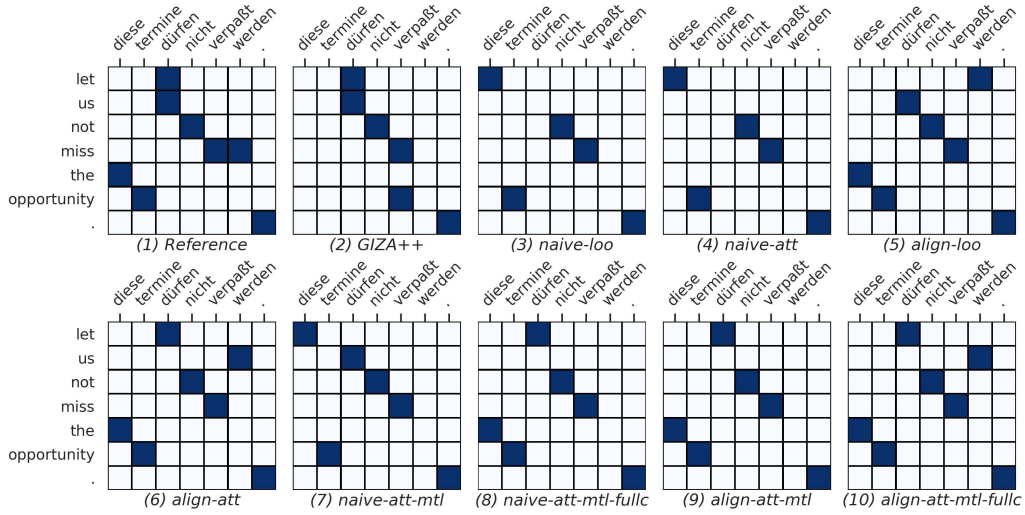
[5]We do not compare on other language pairs because fairseq does not provide translation models on those translation tasks.

[6]They provide an ensemble for WMT19 de→en translation. We simply use model4.pt to extract alignment.

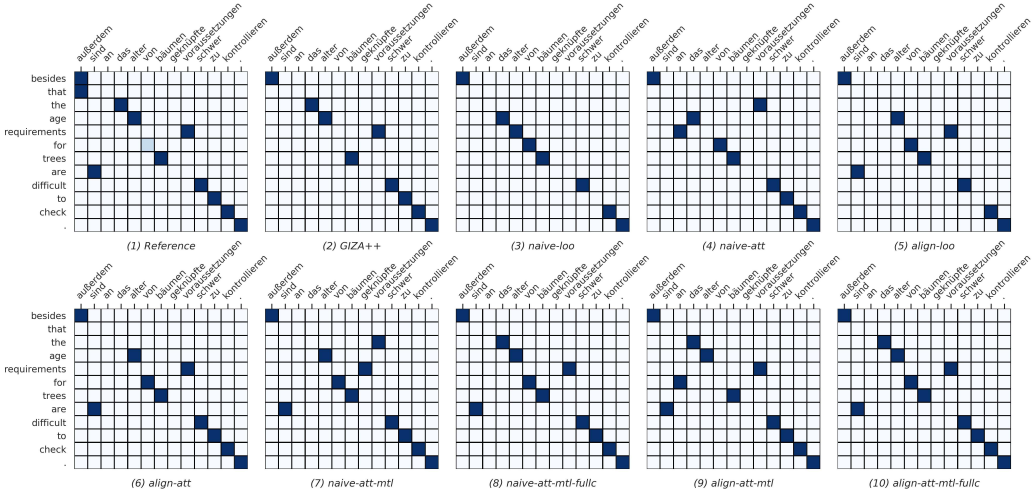
Transformer-big can be different, the AER scores of the two models with *align-att* are similar. We also list the layer from which *align-att* extracts alignment. Transformer-small and Transformer-big both have 6 decoder layers. However, the best layer for Transformer-small is always layer 3, while for Transformer-big, the best layer is layer 2 for de→en and en →de translations. This indicates that with more data and big model, the Transformer tends to acquire alignment information at lower layer.

A.5 More Alignment Examples

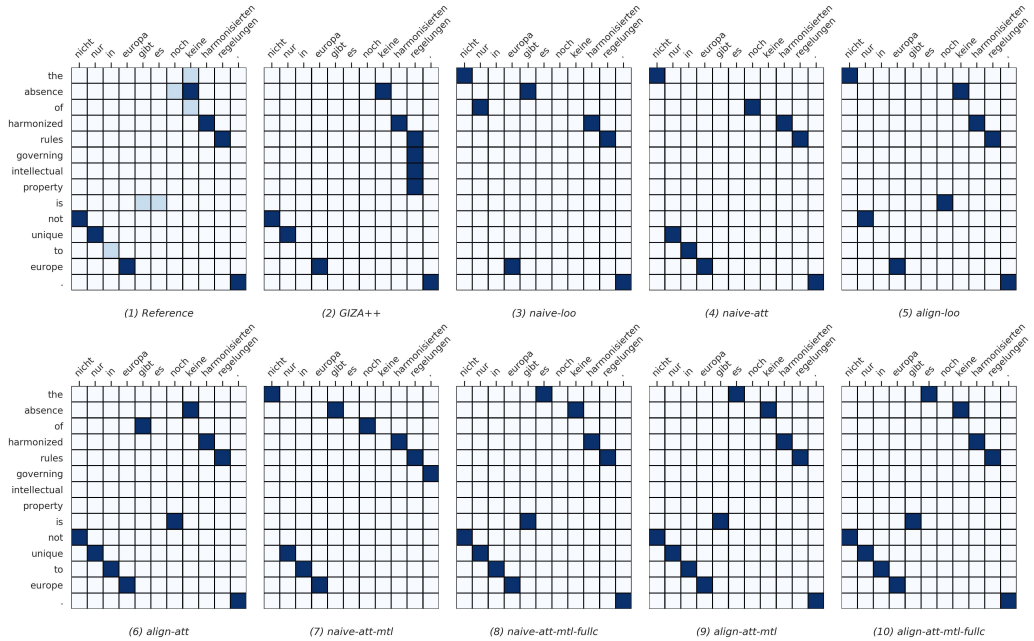
See next page.



(a) Alignment Example 1



(b) Alignment Example 2



(c) Alignment Example 3

Figure 5: Three alignment examples on the de-en alignment test set. All these are symmetrized alignment results.