Unsupervised Token-wise Alignment to Improve Interpretation of Encoder-Decoder Models

Shun Kiyono [1*] **Sho Takase** [2] **Jun Suzuki** [1,2,4]
Naoaki Okazaki [3] **Kentaro Inui** [1,4] **Masaaki Nagata** [2]
[1] Tohoku University [2] NTT Communication Science Laboratories
[3] Tokyo Institute of Technology [4] RIKEN Center for Advanced Intelligence Project
{kiyono,jun.suzuki,inui}@ecei.tohoku.ac.jp,
{takase.sho, nagata.masaaki}@lab.ntt.co.jp,
okazaki@c.titech.ac.jp

Abstract

Developing a method for understanding the inner workings of black-box neural methods is an important research endeavor. Conventionally, many studies have used an attention matrix to interpret how Encoder-Decoder-based models translate a given source sentence to the corresponding target sentence. However, recent studies have empirically revealed that an attention matrix is not optimal for token-wise translation analyses. We propose a method that explicitly models the token-wise alignment between the source and target sequences to provide a better analysis. Experiments show that our method can acquire token-wise alignments that are superior to those of an attention mechanism[1].

1 Introduction

The Encoder-Decoder model with an attention mechanism (EncDec) (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) has been an epoch-making development that has led to great progress in many natural language generation tasks, such as machine translation (Bahdanau et al., 2015), dialog generation (Shang et al., 2015), and headline generation (Rush et al., 2015). An enormous number of studies have attempted to enhance the ability of EncDec.

Furthermore, several intensive studies have also attempted to analyze and interpret the inside of black-box EncDec models, especially how they translate a given source sentence to the corresponding target sentence (Ding et al., 2017). One typical approach to this is to visualize an attention matrix, which is a collection of attention vectors (Bahdanau et al., 2015; Luong et al., 2015; Tu et al., 2016). The assumption behind this interpretation is that the attention matrix has a "soft" token-wise alignment between the source and target sequences, and thus we can use EncDec models to skim which tokens in the source are converted to which tokens in the target.

However, recent studies have empirically revealed that an attention model can operate not only for token-wise alignment but also for other functionalities, such as reordering (Ghader and Monz, 2017; Liu et al., 2016). In addition, Luong et al. (2015) reported that the quality of attention matrix-based alignment is quantitatively inferior to that of the Berkeley aligner (Liang et al., 2006). Koehn and Knowles (2017) also reported that attention matrix-based alignment is significantly different from that acquired from an off-the-shelf aligner for English-German language pairs. From these recent findings, the goal of this paper is to provide a method that can offer a better interpretation of how EncDec models translate a given source sentence to the corresponding target sentence.

In this paper, we focus exclusively on the headline generation task, which is categorized as a *lossy-compression* generation (*lossy-gen*) task (Nallapati et al., 2016). Compared with a machine translation task, which is categorized as a *loss-less* generation (*lossless-gen*) task, the headline generation task additionally requires EncDec models to appropriately select salient ideas in given source sentences (Suzuki and Nagata, 2017). Therefore, the *lossy-gen* task seems to make modeling by EncDec much harder. In fact, our preliminary experiments revealed that the attention mechanism in EncDec models largely fails to capture token-wise alignments, e.g., less than 10 percent accuracy, even if we use one of the current state-of-the-art EncDec models (Table 3).

To obtain a better analysis of how EncDec models translate a given source sentence to the corre-

[*] This work is a product of collaborative research program of Tohoku University and NTT Communication Science Laboratories.

[1] Our code for reproducing the experiments is available at https://github.com/butsugiri/UAM

Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 74–81
Brussels, Belgium, November 1, 2018.

sponding target sentence in the headline generation task, this paper introduces the Unsupervised token-wise Alignment Module (UAM), a novel component that can be plugged into EncDec models. Unlike a conventional attention model, the proposed UAM explicitly captures token-wise alignments between the source and target sequences on the final hidden layer. One can plug the UAM into a EncDec model during a training phase and easily understand the EncDec model's behavior by analyzing the UAM's token-wise alignments. Moreover, the UAM does not require any gold alignment data.

To demonstrate the effectiveness of the UAM, we evaluate EncDec models with the UAM in the headline generation task (Rush et al., 2015), a widely used benchmark for EncDec models. Our experiments show that (i) EncDec models with a UAM achieve comparable (or even superior) performance to the current state-of-the-art headline generation model, and (ii) the produced token-wise alignment is practical regardless of the absence of gold alignment during its training phase.

2 Headline Generation Task

We address the headline generation task introduced in Rush et al. (2015). The source (input) is the first sentence of a news article, and the target (output) is the article's headline. We say I and J represent the numbers of tokens in the source and target, respectively. An important assumption in headline generation is that the target must be shorter than the source ($I > J$).

Here, we denote a source sequence as sequence $\boldsymbol{X}$ of one-hot vectors. Let $\boldsymbol{x}_i \in \{0,1\}^{V_s}$ represent the one-hot vector of the i-th token in $\boldsymbol{X}$, where V_s represents the number of tokens in the source-side vocabulary $\mathcal{V}_s$. We use $\boldsymbol{x}_{1:I}$ to represent $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$; namely, $\boldsymbol{X} = \boldsymbol{x}_{1:I}$. Similarly, let $\boldsymbol{y}_j \in \{0,1\}^{V_t}$ represent the one-hot vector of the j-th token in target sequence $\boldsymbol{Y}$, where V_t is the number of tokens in the target-side vocabulary $\mathcal{V}_t$. Here, we define $\boldsymbol{Y}$ as always containing two additional one-hot vectors of special tokens $\langle bos \rangle$ for $\boldsymbol{y}_0$ and $\langle eos \rangle$ for $\boldsymbol{y}_{J+1}$, namely, $\boldsymbol{Y} = \boldsymbol{y}_{0:J+1}$.

3 Encoder-Decoder Model with Attention Mechanism (EncDec)

This section briefly describes EncDec as the base model of our method[2].

[2]Our EncDec configuration follows the model described in Luong et al. (2015).

3.1 Model Definition

EncDec models the following conditional probability:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(\boldsymbol{y}_j|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}). \quad (1)$$

Encoder EncDec encodes the source one-hot vector sequence $\boldsymbol{x}_{1:I}$ and generates the hidden state sequence $\boldsymbol{h}_{1:I}$, where $\boldsymbol{h}_i \in \mathbb{R}^H$ for all i and H is the size of the hidden state.

We employ bidirectional RNN (BiRNN) as the encoder of the base model. BiRNN is composed of two separate RNNs for the forward ($\overrightarrow{\mathrm{RNN}}_{\mathrm{src}}$) and backward ($\overleftarrow{\mathrm{RNN}}_{\mathrm{src}}$) directions. The forward RNN reads the source sequence $\boldsymbol{X}$ from left to right and constructs hidden states ($\vec{\boldsymbol{h}}_{1:I}$). Similarly, the backward RNN reads the input in the reverse order to obtain another sequence of hidden states ($\overleftarrow{\boldsymbol{h}}_{1:I}$). Finally, we take the summation of hidden states in each direction to construct the final representation of the source sequence ($\boldsymbol{h}_{1:I}$).

Concretely, for a given time step i, the representation $\boldsymbol{h}_i$ is constructed as follows:

$$\vec{\boldsymbol{h}}_i = \overrightarrow{\mathrm{RNN}}_{\mathrm{src}}(\boldsymbol{E}_s \boldsymbol{x}_i, \vec{\boldsymbol{h}}_{i-1}), \quad (2)$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\mathrm{RNN}}_{\mathrm{src}}(\boldsymbol{E}_s \boldsymbol{x}_i, \overleftarrow{\boldsymbol{h}}_{i+1}), \quad (3)$$

$$\boldsymbol{h}_i = \vec{\boldsymbol{h}}_i + \overleftarrow{\boldsymbol{h}}_i \quad (4)$$

where $\boldsymbol{E}_s \in \mathbb{R}^{D \times V_s}$ denotes the word embedding matrix of the source-side and D denotes the size of word embedding.

Decoder The decoder is the unidirectional RNN in the input-feeding approach (Luong et al., 2015). Concretely, decoder RNN takes the output of the previous time step $\boldsymbol{y}_{j-1}$, decoder hidden state $\vec{\boldsymbol{z}}_{j-1}$ and final hidden state $\boldsymbol{z}_{j-1}$ to derive the hidden state of current time step $\boldsymbol{z}_j$:

$$\vec{\boldsymbol{z}}_j = \overrightarrow{\mathrm{RNN}}_{\mathrm{trg}}(\boldsymbol{E}_t \boldsymbol{y}_{j-1}, \boldsymbol{z}_{j-1}, \vec{\boldsymbol{z}}_{j-1}), \quad (5)$$

$$\vec{\boldsymbol{z}}_0 = \vec{\boldsymbol{h}}_I + \overleftarrow{\boldsymbol{h}}_1 \quad (6)$$

where $\boldsymbol{E}_t \in \mathbb{R}^{D \times V_t}$ denotes the word embedding matrix of the decoder. Here, $\boldsymbol{z}_0$ is defined as a zero vector.

Attention Mechanism The attention architecture of the base model is the same as that of the *Global Attention* model proposed by Luong et al. (2015). Attention is responsible for constructing

the final hidden state $\boldsymbol{z}_j$ from the decoder hidden state $\vec{\boldsymbol{z}}_j$ and encoder hidden states $(\boldsymbol{h}_{1:I})$.

First, the model computes the attention vector $\boldsymbol{\alpha}_j \in \mathbb{R}^I$ from the decoder hidden state $\vec{\boldsymbol{z}}_j$ and encoder hidden states $(\boldsymbol{h}_{1:I})$. From among three attention scoring functions proposed in Luong et al. (2015), we employ a *general* function. This function calculates the attention score in bilinear form. Specifically, the attention score between the i-th source hidden state and the j-th decoder hidden state is computed by the following equation:

$$\boldsymbol{\alpha}_j[i] = \frac{\exp(\boldsymbol{h}_i^\top \boldsymbol{W}_\alpha \vec{\boldsymbol{z}}_j)}{\sum_{i=1}^{I} \exp(\boldsymbol{h}_i^\top \boldsymbol{W}_\alpha \vec{\boldsymbol{z}}_j)} \tag{7}$$

where $\boldsymbol{W}_\alpha \in \mathbb{R}^{H \times H}$ is a parameter matrix, and $\boldsymbol{\alpha}_j[i]$ denotes the i-th element of $\boldsymbol{\alpha}_j$.

$\boldsymbol{\alpha}_j$ is then used for collecting the source-side information that is relevant for predicting the target token. This is done by taking the weighted sum on the encoder hidden states:

$$\boldsymbol{c}_j = \sum_{i=1}^{I} \boldsymbol{\alpha}_j[i] \boldsymbol{h}_i \tag{8}$$

Next, the source-side information is mixed with the decoder hidden state to derive final hidden state $\boldsymbol{z}_j$. Concretely, the context vector $\boldsymbol{c}_j$ is concatenated with $\vec{\boldsymbol{z}}_j$ to form vector $\boldsymbol{u}_j \in \mathbb{R}^{2H}$. $\boldsymbol{u}_j$ is then fed into a single fully-connected layer with $\tanh$ nonlinearity:

$$\boldsymbol{z}_j = \tanh(\boldsymbol{W}_s \boldsymbol{u}_j) \tag{9}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{H \times 2H}$ is a parameter matrix.

Finally, $\boldsymbol{z}_j$ is fed into the softmax layer. The model generates a target-side token based on the probability distribution $\boldsymbol{o}_j \in \mathbb{R}^{V_t}$ as

$$\boldsymbol{o}_j = \mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{z}_j + \boldsymbol{b}_o), \tag{10}$$

where $\boldsymbol{W}_o \in \mathbb{R}^{V_t \times H}$ is a parameter matrix and $\boldsymbol{b}_o \in \mathbb{R}^{V_t}$ is a bias term.

3.2 Training of EncDec

To train EncDec, let $\mathcal{D}$ be training data for headline generation, which consists of source-headline sentence pairs. Let θ represent all parameters in EncDec. Our goal is to find the optimal parameter set $\hat{\theta}$ that minimizes the following objective function $G_0(\theta)$ for the given training data $\mathcal{D}$:

$$G_0(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y}) \in \mathcal{D}} \ell_{\mathrm{trg}}(\boldsymbol{Y}, \boldsymbol{X}, \theta),$$

$$\ell_{\mathrm{trg}}(\boldsymbol{Y}, \boldsymbol{X}, \theta) = -\log\Big(p(\boldsymbol{Y}|\boldsymbol{X}, \theta)\Big). \tag{11}$$

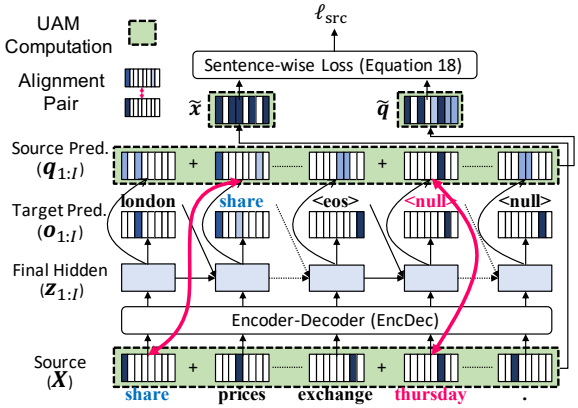


Figure 1: Overview of EncDec with UAM. UAM predicts the probability distribution over the source vocabulary $\boldsymbol{q}_j$ at each time step j. After predicting all the time steps, the SPM compares the sum of the predictions $\tilde{\boldsymbol{q}}$ with the sum of the source-side tokens $\tilde{\boldsymbol{x}}$ as an objective function ℓ_{src}.

Since $\boldsymbol{o}_j$ for each j is a vector representation of the probabilities of $p(\hat{\boldsymbol{y}}|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}, \theta)$ over the target vocabularies $\hat{\boldsymbol{y}} \in \mathcal{V}_t$, we can calculate ℓ_{trg} as

$$\ell_{\mathrm{trg}}(\boldsymbol{Y}, \boldsymbol{X}, \theta) = -\sum_{j=1}^{J+1} \boldsymbol{y}_j^\top \cdot \log\big(\boldsymbol{o}_j\big). \tag{12}$$

3.3 Inference of EncDec

In the inference step, we use the trained parameters to search for the best target sequence. We use beam search to find the target sequence that maximizes the product of the conditional probabilities as described in Equation 1. From among several stopping criteria for beam search (Huang et al., 2017), we adopt the widely used "shrinking beam" implemented in RNNsearch (Bahdanau et al., 2015)[3].

4 Proposed Method: Unsupervised Alignment Module (UAM)

Figure 1 illustrates the proposed method, UAM. UAM is the module attached on top of the decoder output layer of an EncDec model, and it explicitly represents a token-wise alignment by predicting source tokens simultaneously with target tokens.

Specifically, during the training of EncDec, the decoder estimates the probability distribution over the source-side vocabulary, $\boldsymbol{q}_j \in \mathbb{R}^{V_s}$, simultaneously with that of the target-side vocabulary,

[3] https://github.com/lisa-groundhog/GroundHog

$\boldsymbol{o}_j \in \mathbb{R}^{V_t}$, for each time step j. In Figure 1, when the decoder predicts the target-side token "share", we want to predict its corresponding source-side token "share." If we can correctly predict the corresponding source-side token for each target-side token, we can obtain token-wise alignment. As shown below, we can train the model for this prediction without gold token-wise alignment signals.

This way of representing alignments can be extended to ⟨*null*⟩ alignments. We first expand a given target sequence with a sequence of ⟨*null*⟩s so that its length is the same as that of the source sequence (in Figure 1, we extend "london share…⟨*eos*⟩" to "london share…⟨*eos*⟩⟨*null*⟩…⟨*null*⟩"). We then train the model so that it can predict a discarded source-side token (e.g., "thursday") when it predicts ⟨*null*⟩ for the target-side. Through the task of predicting source-side tokens corresponding to ⟨*null*⟩s, we expect the model to effectively learn to identify unimportant information in the source sequence.

4.1 Model Definition

EncDec with UAM jointly estimates the probability distributions over the source and target vocabularies. Specifically, UAM estimates a probability distribution over the source vocabulary $\boldsymbol{q}_j \in \mathbb{R}^{V_s}$ at each time step j in the decoding process by:

$$\boldsymbol{q}_j = \mathrm{softmax}(\boldsymbol{W}_q \boldsymbol{z}_j + \boldsymbol{b}_q), \tag{13}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{V_s \times H}$ is a parameter matrix, $\boldsymbol{z}_j \in \mathbb{R}^H$ is a decoder's final hidden layer, and $\boldsymbol{b}_q \in \mathbb{R}^{V_s}$ is a bias term. H is the size of the hidden state. EncDec also calculates a probability distribution over target vocabulary $\boldsymbol{o}_j$ from the same vector $\boldsymbol{z}_j$.

Next, we define $\boldsymbol{Y}'$ as a concatenation of the one-hot vectors of the target-side sequence $\boldsymbol{Y}$ and those of the special token ⟨*null*⟩ of length $I - (J+1)$. Here, $\boldsymbol{y}_{J+1}$ is a one-hot vector of ⟨*eos*⟩, and $\boldsymbol{y}_j$ for each $j \in \{J+2, \ldots, I\}$ is a one-hot vector of ⟨*null*⟩. Note that the length of $\boldsymbol{Y}'$ is always no shorter than that of $\boldsymbol{Y}$, that is, $|\boldsymbol{Y}'| \geq |\boldsymbol{Y}|$ because headline generation always assumes $I > J$.

Based on this setting, we train the model in an unsupervised manner without gold alignment signals. To this end, we consider a sentence-wise loss function instead of token-wise loss. Specifically, we define the sentence-wise loss as the degree of difference between the sum of all one-hot vectors in the source sequence and the sum of UAM predictions. Namely, we take the difference of $\tilde{\boldsymbol{x}} = \sum_{i=1}^{I} \boldsymbol{x}_i$ and $\tilde{\boldsymbol{q}} = \sum_{j=1}^{I} \boldsymbol{q}_j$. Note that $\tilde{\boldsymbol{x}}$ is a vector representation of the occurrences (or bag-of-words representation) of the source-side tokens. To minimize sentence-wise loss, the model must predict the bag-of-words representation. Through this optimization, the model is expected to eventually find the token-wise alignment.

EncDec with UAM models the following conditional probability:

$$p(\boldsymbol{Y}', \tilde{\boldsymbol{x}} | \boldsymbol{X}) = p(\tilde{\boldsymbol{x}} | \boldsymbol{Y}', \boldsymbol{X}) p(\boldsymbol{Y}' | \boldsymbol{X}). \tag{14}$$

We define $p(\boldsymbol{Y}'|\boldsymbol{X})$ as follows:

$$p(\boldsymbol{Y}'|\boldsymbol{X}) = \prod_{j=1}^{I} p(\boldsymbol{y}_j | \boldsymbol{y}_{0:j-1}, \boldsymbol{X}), \tag{15}$$

which is identical to the conditional probability modeled by the base EncDec, except for considering ⟨*null*⟩ as part of the probability. Next, we define $p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}', \boldsymbol{X})$ as follows:

$$p(\tilde{\boldsymbol{x}} | \boldsymbol{Y}', \boldsymbol{X}) = \frac{1}{Z} \exp\left(\frac{-\|\tilde{\boldsymbol{q}} - \tilde{\boldsymbol{x}}\|_2^2}{C}\right), \tag{16}$$

where Z is a normalization term and C is a hyper-parameter that controls the sensitivity of the distribution.

4.2 Training of UAM

We optimize the UAM combined with EncDec by minimizing the negative log-likelihood of Equation 14 as a loss function. Let γ represent the parameter set of SPM Then, we define the UAM loss ℓ_{src} as follows;

$$\ell_{\mathrm{src}}(\tilde{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{Y}', \gamma, \theta) = -\log(p(\tilde{\boldsymbol{x}} | \boldsymbol{Y}', \boldsymbol{X}, \gamma, \theta)) \tag{17}$$

From Equation 16, we can derive ℓ_{src} as

$$\ell_{\mathrm{src}}(\tilde{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{Y}', \gamma, \theta) = \frac{1}{C}\|\tilde{\boldsymbol{q}} - \tilde{\boldsymbol{x}}\|_2^2 + \log(Z). \tag{18}$$

We can discard $\log(Z)$ from the RHS, since it is independent of γ and θ.

Here, we regard the sum of ℓ_{src} and ℓ_{trg} as an objective loss function of multi-task training. Formally, we train EncDec with the UAM by minimizing the following objective function G_1:

$$G_1(\theta, \gamma) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X}, \boldsymbol{Y}) \in \mathcal{D}} \Big(\ell_{\mathrm{trg}}(\boldsymbol{Y}', \boldsymbol{X}, \theta) + \ell_{\mathrm{src}}(\tilde{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{Y}', \gamma, \theta) \Big), \tag{19}$$

where $\mathcal{D}$ is training data for headline generation, which consists of source-headline sentence pairs.

4.3 Inference of UAM

In the inference time, the goal is only to search for the best target sequence. Thus, we do not need to compute the UAM during the inference. Similarly, it is unnecessary to produce $\langle null \rangle$ after generating $\langle eos \rangle$. Thus, we can utilize the beam search procedure described in Section 3.3, and as a result the actual computation cost for the inference remains unchanged from the base EncDec.

5 Experiment

5.1 Dataset

The origin of the headline generation dataset used in our experiments is identical to that used in Rush et al. (2015). The dataset consists of pairs of the first sentence of each article and its headline from the annotated English Gigaword corpus (Napoles et al., 2012). Rush et al. (2015) defined the training, validation and test splits, which contain approximately 3.8M, 200K and 400K source-headline pairs, respectively

We used the entire training split for training as in the previous studies. We randomly sampled 8K instances as validation data and 10K instances as test data from the validation split. Moreover, we experimented on the test data provided by Zhou et al. (2017) and Toutanova et al. (2016) for comparison with the reported state-of-the-art performance (Zhou et al., 2017). We refer to those test data sets as Test (Ours), Test (Zhou), and MSR-ATC respectively. Among these test sets, MSR-ATC is the only dataset created by a human worker.

5.2 Implementation Details

In the experiment, we selected the hyper-parameter settings commonly used in previous studies e.g., (Rush et al., 2015; Nallapati et al., 2016; Suzuki and Nagata, 2017) We constructed the vocabulary set using byte pair encoding[4] (BPE) (Sennrich et al., 2016) to handle low-frequency words, since this is now a common practice in the field of neural machine translation. The BPE merge operations are jointly learned from the source and the target. We set the number of BPE merge operations at $5,000$.

[4] https://github.com/rsennrich/subword-nmt

Source Vocab. Size V_s	5131
Target Vocab. Size V_t	5131
Word Embed. Size D	200
Hidden State Size H	400
RNN Cell	Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997)
Encoder RNN Unit	2-layer bidirectional-LSTM
Decoder RNN Unit	2-layer LSTM with attention (Luong et al., 2015)
Optimizer	Adam (Kingma and Ba, 2015)
Initial Learning Rate	0.001
Learning Rate Decay	0.5 for each epoch (after epoch 9)
Weight C of ℓ_{src}	10
Mini-batch Size	256 (shuffled at each epoch)
Gradient Clipping	5
Stopping Criterion	Max 15 epochs with early stopping
Regularization	Dropout (rate 0.3)
Beam Search	Beam size 20 with the length normalization

Table 1: Configurations used in our experiments

5.3 Evaluation Metric

We evaluated the performance by ROUGE-1 (RG-1), ROUGE-2 (RG-2) and ROUGE-L (RG-L)[5]. We report the F1 value as given in a previous study[6]. We computed the ROUGE scores using the official ROUGE script (version 1.5.5).

5.4 Results

To investigate the effectiveness of the UAM quantitatively, we chose a very strong baseline, SEASS (Zhou et al., 2017), which is the current state-of-the-art model. We reimplemented SEASS, hereafter EncDec+sGate, and compared EncDec+sGate with the model incorporating UAM into EncDec+sGate.

Table 2 summarizes ROUGE-F1 results for all test data. The table shows that EncDec+sGate+UAM achieved a huge gain particularly for MSR-ATC and a performance comparable to EncDec+sGate in Test (Ours) and Test (Zhou). Considering that the MSR-ATC dataset was created by a human worker, we believe that the improvement in MSR-ATC is the most remarkable result among the three test sets, since it indicates that our model most closely fits the human-generated summary.

6 Discussion

We investigated whether the UAM improves token-wise alignment between the source and target se-

[5] We restored sub-words to the standard token split for the evaluation.

[6] ROUGE script option is: "`-n2 -m -w 1.2`"

Model	Test (Ours)			Test (Zhou)			MSR-ATC		
	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L
EncDec+sGate	46.80	24.48	43.74	46.79	24.75	43.62	29.73	15.45	27.30
EncDec+sGate+UAM†	**46.91**	**24.86**	**43.87**	**46.89**	**24.93**	**43.68**	**32.32**	**17.02**	**29.76**
SEASS (Zhou et al., 2017)	-	-	-	46.86	24.58	43.53	25.75	10.63	22.90

Table 2: Full-length ROUGE F1 evaluation results. † is the proposed model.

quences. We compared the alignments acquired by the UAM and the attention model, since attention has been implicitly considered as alignment (Bahdanau et al., 2015; Luong et al., 2015; Tu et al., 2016).

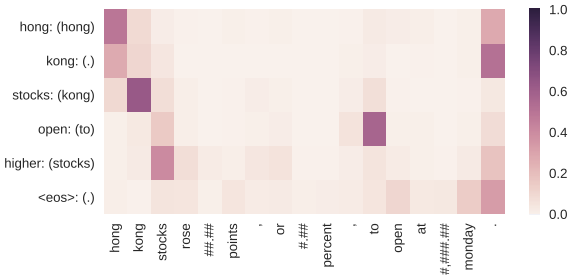
6.1 Visualizing UAM and Attention

We visualized UAM prediction and the attention matrix to see the acquired token-wise alignment between the source and target. Specifically, we fed the source-target pair $(\boldsymbol{X}, \boldsymbol{Y})$ to EncDec+sGate and EncDec+sGate+UAM, and then collected UAM predictions $(\boldsymbol{q}_{1:I})$ of EncDec+sGate+UAM and the attention vectors $(\boldsymbol{\alpha}_{1:J})$ of EncDec+sGate. We computed the attention vectors using Equation 7. For UAM prediction, we extracted the probability of each token $\boldsymbol{x}_i \in \boldsymbol{X}$ from $\boldsymbol{q}_{1:I}$.

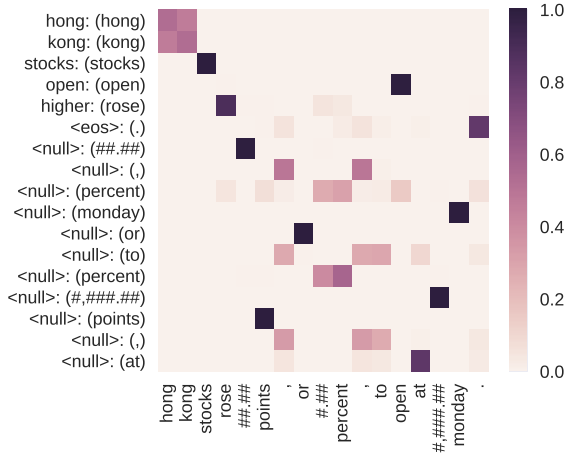
Figure 2 and 3 show an example of the heat map. We used Test (Ours) as the input. The brackets in the y-axis represent the source-side tokens that are aligned with target-side tokens. We obtained the aligned tokens as follows: For attention (Figure 2a, 3a), we select the token with the largest attention value. For the UAM (Figure 2b, 3b), we select the token with the largest probability over the vocabulary $\mathcal{V}_s$.

Figure 2a indicates that attention provides poor token-wise alignment. For example, the target-side token "kong" is incorrectly aligned with the source-side sentence period. In contrast, Figure 2b shows that the UAM captures reasonable alignments. Here, the source-side token "rose" is aligned with the target-side token "higher." UAM also correctly aligned unimportant tokens such as "##.##" with $\langle null \rangle$.

In Figure 3a, the attention model repeatedly pays attention to the source-side token "positive." As a result, the attention model aligned the target-side token "egyptian" to "positive." On the other hand, in 3b, the UAM correctly aligns "egyptian." In addition, the UAM aligned source-side token "foreign" with target-side token "fm."



(a) Attention matrix of EncDec+sGate



(b) UAM prediction of EncDec+sGate+UAM

Figure 2: Visualization of models. The x-axis and y-axis correspond to the source and the target sequence respectively. Tokens in the brackets are source-side tokens aligned at that time step.

6.2 Alignment Accuracy

In order to quantitatively investigate the quality of the alignment, we evaluated the alignment accuracy of both EncDec+sGate and EncDec+sGate+UAM.

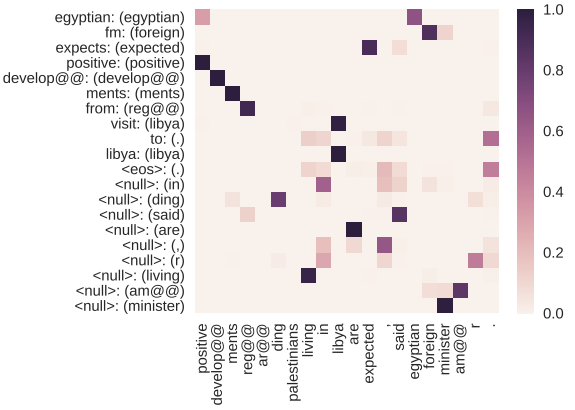
We randomly sampled 40, 30 and 30 instances from Test (Ours), Test (Zhou) and MSR-ATC respectively. We acquired the alignment of the data by applying the UAM and the attention model, in the same manner as described in Section 6.1. A single annotator then evaluated the accuracy of the alignment by hand.

Table 3 summarizes the results. The alignment of the UAM is significantly more accurate than

Model	Test (Ours)	Test (Zhou)	MSR-ATC	Overall
EncDec+sGate	8.60	5.97	6.11	6.71
EncDec+sGate+UAM	52.52	50.91	51.07	51.41

Table 3: Alignment Macro Accuracy (%)

(a) Attention matrix of EncDec+sGate



(b) UAM prediction of EncDec+sGate+UAM

Figure 3: Visualization of models. The x-axis and y-axis correspond to the source and the target sequence respectively. Tokens in the brackets are source-side tokens aligned at that time step.

that of the attention model. This result is consistent with the empirical results reported by Koehn and Knowles (2017) and Luong et al. (2015). The attention alignment mistakes are mostly due to paying attention to either the sentence period or to the token decoded in the previous time step. It is noteworthy that the accuracy of the UAM alignment exceeds 50% even though we trained the model in an unsupervised manner. The fact that the UAM prediction acquires reasonable alignment suggests that the UAM has the potential to provide us a better understanding of the model's behavior.

7 Conclusion

In this paper, we introduced the Unsupervised token-wise Alignment Module (UAM), which learns to predict the token-wise alignment of tokens in the source and the target. Experiments on the headline generation task showed that the UAM can achieve comparable performance to the current state-of-the-art sGate model. In addition, the UAM obtained token-wise alignment that is superior to that of the attention model. This finding suggests we can use the UAM as an alternative to the attention matrix to attain a better understanding of the token-wise alignment of EncDec-based model.

Acknowledgment

We are grateful to anonymous reviewers for their insightful comments. We thank Sosuke Kobayashi for providing helpful comments. We also thank Qingyu Zhou for providing a dataset and information for a fair comparison.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

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 (ACL 2017)*, pages 1150–1159.

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 (IJCNLP 2017)*, pages 30–39.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2124–2129.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2006)*, pages 104–111.

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*, pages 3093–3102.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

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 (ACL 2016)*, pages 1715–1725.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL & IJCNLP 2015)*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off Redundant Repeating Generations for Neural Abstractive Summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 291–297.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A Dataset and Evaluation Metrics for Abstractive Compression of Sentences and Short Paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 340–350.

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 (ACL 2016)*, pages 76–85.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective Encoding for Abstractive Sentence Summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104.