Enhancing Pointer Network for Sentence Ordering with Pairwise Ordering Predictions

Yongjing Yin,[1*] **Fandong Meng,**[2] **Jinsong Su,**[1†] **Yubin Ge,**[3] **Linfeng Song,**[4] **Jie Zhou,**[2] **Jiebo Luo**[5]
[1]Xiamen University, China, [2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
[3]University of Illinois at Urbana-Champaign, USA, [4]TencentAI Lab, Bellevue, USA, [5]University of Rochester, USA
yinyongjing@stu.xmu.edu.cn, {fandongmeng, jiezhou}@tencent.com, jssu@xmu.edu.cn, yubinge2@illinois.edu,
freesunshine0316@gmail.com, jluo@cs.rochester.edu

Abstract

Dominant sentence ordering models use a pointer network decoder to generate ordering sequences in a left-to-right fashion. However, such a decoder only exploits the noisy left-side encoded context, which is insufficient to ensure correct sentence ordering. To address this deficiency, we propose to enhance the pointer network decoder by using two pairwise ordering prediction modules: The FUTURE module predicts the relative orientations of other unordered sentences with respect to the candidate sentence, and the HISTORY module measures the local coherence between several (e.g., 2) previously ordered sentences and the candidate sentence, without the influence of noisy left-side context. Using the pointer mechanism, we then incorporate this dynamically generated information into the decoder as a supplement to the left-side context for better predictions. On several commonly-used datasets, our model significantly outperforms other baselines, achieving the state-of-the-art performance. Further analyses verify that pairwise ordering predictions indeed provide extra useful context as expected, leading to better sentence ordering. We also evaluate our sentence ordering models on a downstream task, multi-document summarization, and the summaries reordered by our model achieve the best coherence scores. Our code is available at https://github.com/DeepLearnXMU/Pairwise.git.

Introduction

Modeling text coherence is an essential problem in natural language processing (NLP) as evidenced by its significance on several downstream NLP tasks (Barzilay, Elhadad, and McKeown 2002; Bollegala, Okazaki, and Ishizuka 2006; Konstas and Lapata 2012; Galanis, Lampouras, and Androutsopoulos 2012; Nallapati, Zhai, and Zhou 2012; Nayeem and Chali 2017). As one subtask of coherence modeling, sentence ordering (Barzilay and Lapata 2008) aims to learn such a coherence structure by reconstructing a coherent paragraph from an unordered set of sentences. By learning to order sentences, the model is able to identify crucial

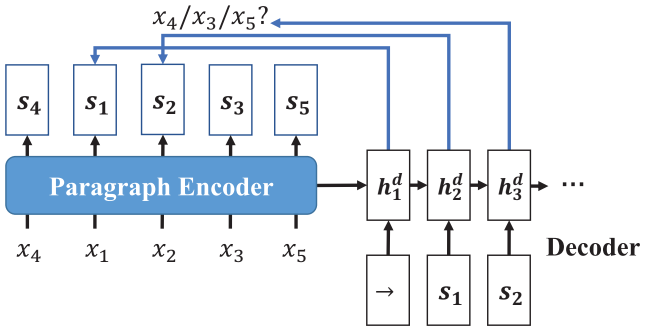


Figure 1: The model with a pointer network decoder. s_* denotes the semantic vector representation of sentence x_*, produced by the encoder.

properties that cause text coherence, which can be exploited to generate coherent texts for other tasks.

Recently, inspired by the success of deep learning, neural network based models have been proposed, where representative work includes Window network (Li and Hovy 2014), pairwise models (Agrawal et al. 2016; Li and Jurafsky 2017), and pointer network (Vinyals, Fortunato, and Jaitly 2015) decoder based models (Gong et al. 2016; Logeswaran, Lee, and Radev 2018; Cui et al. 2018; Wang and Wan 2019; Yin et al. 2019). Particularly, the last kind of models attracts much attention due to its state-of-the-art performance (Wang and Wan 2019; Yin et al. 2019). As shown in Figure 1, the pointer network decoder is based on a simplified attention model, which first updates the current hidden state with the previously pointed sentence representation as the input, and then applies this state to produce the attention distribution over the unordered input sentences. It has been proven suitable for dealing with sorting the elements of a given set (Vinyals, Fortunato, and Jaitly 2015; Vinyals, Bengio, and Kudlur 2015), and thus become a standard component in dominant sentence ordering models.

Despite its success, there still exists a serious drawback. Due to its autoregressive structure that produces the ordering sequence in a left-to-right fashion, it only exploits the noisy left-side encoded context while ignoring other useful information. Specifically, the ordering information between the

[*]This work was conducted at Pattern Recognition Center, WeChat AI, Tencent Inc, China
[†]Corresponding author

unsorted sentences are completely not considered, although it is intuitively beneficial to the current prediction. In addition, due to the negative effect of accumulated noisy context, the conditional probability modeled by the decoder is unable to accurately measure the local coherence between previously ordered sentences and the one being considered. Let us look at the example shown in Figure 1. When considering x_3 as the current candidate, the decoder can only leverage the left-side context x_1 and x_2, without considering relative orientations of x_4, x_5 with respect to x_3. Meanwhile, if the ordering of x_1 or x_2 is incorrect, the hidden state at the 3rd timestep is unable to accurately model the local coherence between x_2 and x_3. Such local coherence provides an important hint to determine whether x_3 should be placed or not. Therefore, we believe there is significant room for improvement beyond the standard pointer network decoder.

In this paper, we propose to enhance the pointer network decoder using two pairwise ordering prediction modules. The intuition behind is two-fold. First, any sentence order can be equivalently considered as several pairwise orderings, each of which is easier to model. Second, pairwise ordering predictions are lightweight since they only depend on their own semantic representations. Thus, they can be easily incorporated into a decoder, dynamically generating information complementary to the encoded context.

To this end, we **first** introduce two pairwise modules based on the learned sentence embeddings: the first one (FUTURE module) is employed to predict the relative orientations between the candidate sentence and other unordered sentences, and the second one (HISTORY module) is used to calculate the probabilities of two previously ordered sentences occurring before the candidate sentence at different relative distances, which measures their local coherence without the negative impact of the accumulated noisy context. **Next**, in each decoding step, all probability distributions and semantic features produced by these two modules are fused to generate a new vector representation for each candidate. **Finally**, the decoder predicts the ordering via the pointer mechanism. In this way, our decoder not only exploits the relative orientation information between unordered sentences but also examines the local coherence between the candidate and previously ordered sentences, and thus has the potential to produce better sentence ordering.

The main contributions of this paper can be summarized as follows:

- We first point out the drawback of the pointer network decoder for sentence ordering, and then propose a novel pointer network decoder enhanced by two specially designed pairwise ordering modules.
- Our model significantly outperforms competitive baselines and advances the state-of-the-art in this task.
- We conduct experiments on a downstream task, multi-document summarization, and the summaries reordered by our proposed model achieve the best coherence scores. Moreover, the analysis indicates that our model can alleviate the adverse effect of the noisy context.

Related Work

Previous work on coherence modeling mainly focused on the utilization of linguistic features and statistical models (Lapata 2003; Barzilay and Lee 2004; Barzilay and Lapata 2005; Guinaudeau and Strube 2013). Recently, neural network based models have shown powerful capability in sentence ordering, where the work most relevant to ours can be classified into two categories:

(1) **Pairwise models**. For example, Gong et al. (2016) investigated the effectiveness of various neural models on judging the order of each sentence pair. Agrawal et al. (2016) implemented multiple neural network models based on individual and pairwise element-based predictions (and their ensemble). Li and Jurafsky (2017) applied sequence-to-sequence based generative models (Sutskever, Vinyals, and Le 2014) to model pairwise coherence.

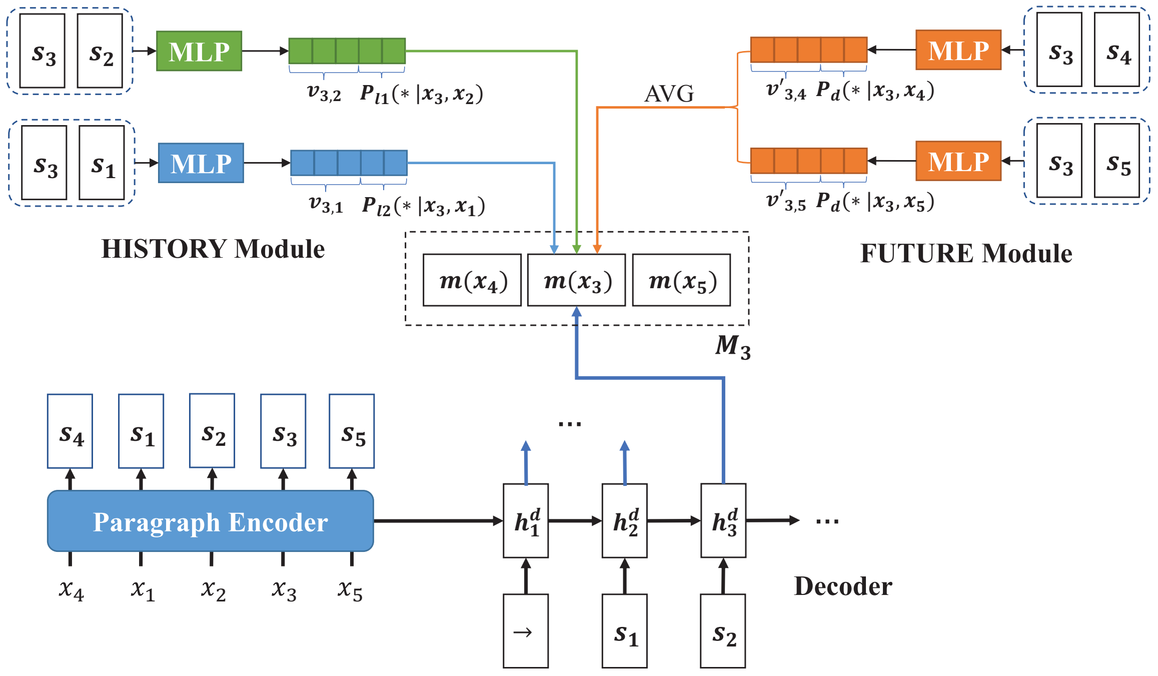
Our model is significantly different from the above. In previous models, order predictions for pairwise sentences are utilized to generate an ordered sentence sequence using other algorithms. In contrast, these predictions are incorporated into our decoder as auxiliary information.

(2) **Pointer network** (Vinyals, Fortunato, and Jaitly 2015). In this aspect, although contrary to human intuition, Logeswaran, Lee, and Radev (2018) used a hierarchical RNN based encoder to model the input sentences. Furthermore, Cui et al. (2018) introduced a self-attention mechanism (Vaswani et al. 2017) to refine encoder modeling. Very recently, Wang and Wan (2019) proposed a hierarchical attention based encoder and a masked self-attention based decoder, and Yin et al. (2019) leveraged a graph recurrent network (Zhang, Liu, and Song 2018; Song et al. 2019) to model the co-occurrence between sentences and entities.

Different from these work, we enhance the pointer network decoder using pairwise ordering predictions which can be an effective supplement to the left-side context. Particularly, we leverage pairwise orderings between unordered sentences to guide our decoder, which is essentially consistent with the future information modeling in other sequence generation tasks (Bahdanau et al. 2017; Serdyuk et al. 2018; Zheng et al. 2018). Experimental results demonstrate the superiority of our decoder over these models.

Our Model

Given an out-of-order set of N sentences $\boldsymbol{x} = [x_1, \ldots, x_N]$ as input, our model aims to recover its correct order $\boldsymbol{o} = [o_1, \ldots, o_N]$. Essentially, our model is an extension of ATTOrderNet (Cui et al. 2018). As shown in Figure 2, our model mainly consists of two components: 1) a paragraph encoder based on the multi-head self-attention mechanism (Vaswani et al. 2017) for encoding each sentence into a distributed representation; and 2) a pointer network decoder (Vinyals, Fortunato, and Jaitly 2015) enhanced by two pairwise ordering prediction modules which generate a new vector representation for each candidate.



s_3 s_2
MLP
$v_{3,2}$ $P_{l1}(*\,|x_3,x_2)$
s_3 s_1
MLP
$v_{3,1}$ $P_{l2}(*\,|x_3,x_1)$
HISTORY Module
AVG
$v'_{3,4}$ $P_d(*\,|x_3,x_4)$
MLP
s_3 s_4
$v'_{3,5}$ $P_d(*\,|x_3,x_5)$
MLP
s_3 s_5
FUTURE Module
$m(x_4)$ $m(x_3)$ $m(x_5)$
M_3
s_4 s_1 s_2 s_3 s_5
Paragraph Encoder
x_4 x_1 x_2 x_3 x_5
h_1^d h_2^d h_3^d
→ s_1 s_2
Decoder

Figure 2: The architecture of the proposed model. x_1, x_2 are previously ordered sentences, and x_4, x_3, x_5 are unsorted. The matrix M_3 is a pack of three vectors $m(x_4)$, $m(x_3)$ and $m(x_5)$, each of which encodes the ordering information for its corresponding candidate at the 3rd timestep. The generation process of $m(x_3)$ is illustrated in detail and $m(x_4)$ and $m(x_5)$ are generated in the same way.

Paragraph Encoder

To build our paragraph encoder, we first apply a sentence encoder to learn semantic representations of sentences. This encoder is a bidirectional Long Short-Term Memory (Hochreiter and Schmidhuber 1997) (Bi-LSTM), which recurrently produces context-aware semantic representations of all words in sentence x_i from both directions. For the j-th word $w_{i,j}$, its hidden states in two directions ($\overrightarrow{h}_{i,j}$ and $\overleftarrow{h}_{i,j}$) are updated as follows:

$$e_{i,j} = \text{one\_hot}(w_{i,j})W_e, \tag{1}$$

$$\overrightarrow{h}_{i,j} = \text{LSTM}(\overrightarrow{h}_{i,j-1}, e_{i,j}), \tag{2}$$

$$\overleftarrow{h}_{i,j} = \text{LSTM}(\overleftarrow{h}_{i,j+1}, e_{i,j}), \tag{3}$$

where W_e is the word embedding matrix. In this way, we can obtain the sentence embedding s_i for x_i by concatenating the last states of the Bi-LSTM in two directions.

Then, we pack all learned sentence embeddings together into a matrix S, and feed it into a self-attention module, where these sentence representations can be updated into paragraph-aware ones. This self-attention module contains a stack of L identical layers, each of which consists of two sub-layers: a multi-head self-attention layer (MultiHead) and a fully-connected feed-forward network (FFN). For the l-th layer, the output matrix $S^{(l)}$ are produced as follows:

$$A^{(l)} = \text{MultiHead}(S^{(l-1)}, S^{(l-1)}, S^{(l-1)}), \tag{4}$$

$$S^{(l)} = \text{FFN}(A^{(l)}), \tag{5}$$

where MultiHead(Q,K,V) is a multi-head self-attention function with a query matrix Q, a key matrix K, and a value matrix V as inputs, generating the temporary hidden state matrix $A^{(l)}$. Here, we omit the descriptions of residual connection and layer normalization in each sub-layer for simplicity. Please refer to (Vaswani et al. 2017; Cui et al. 2018) for more details.

Finally, we obtain the global paragraph representation g by averaging the output matrix from the last layer $g = \frac{1}{N}\sum_{i=1}^{N} S_i^{(L)}$, where $S_i^{(L)}$ denotes the i-th row in $S^{(L)}$. This vector g will then be used as the initial state of the decoder.

Decoder

As illustrated in Figure 2, our decoder is an LSTM-based pointer network, enhanced by two modules for pairwise ordering predictions. Formally, using this decoder, we calculate the conditional probability of a predicted order $\boldsymbol{o'}$ of the input out-of-order sentence set $\boldsymbol{x}$ as follows:

$$P(\boldsymbol{o'}|\boldsymbol{x}) = \prod_{i=1}^{N} P(o'_i|\boldsymbol{o'}_{<i}, \boldsymbol{x}), \tag{6}$$

$$P(o'_i|\boldsymbol{o'}_{<i}, \boldsymbol{x}) = \text{softmax}(v^T \tanh(W h_i^d + U M_i)), \tag{7}$$

$$h_i^d = \text{LSTM}(h_{i-1}^d, s_{o'_{i-1}}), \tag{8}$$

where W, U and v are model parameters, $s_{o'_{i-1}}$ is the embedding of the previous sentence, h_i^d is the hidden state of the decoder, and M_i is a matrix indicating two kinds of information for all unordered sentences: One is global orientation information of other unsorted sentences with respect to x_u, and the other is local coherence between previously ordered sentences and x_u, where x_u is a candidate sentence. Next, we will give detailed descriptions of the two new introduced modules.

FUTURE Module. In this module, one probability distribution $P_d(ori|x_u, x_{u'})$, where $ori \in$\{*before*,*after*\}, is modeled to calculate the probability of x_u appearing before/after another unordered sentence $x_{u'}$ [1]:

$$p'_{u,u'} = \text{ReLU}(W'_1 s_u + W'_2 s_{u'}), \tag{9}$$

$$v'_{u,u'} = \text{ReLU}(W'_v p'_{u,u'}), \tag{10}$$

$$P_d(*|x_u, x_{u'}) = \text{softmax}(W'_s v'_{u,u'}), \tag{11}$$

where W'_* are weight matrices, s_u and $s_{u'}$ are the vector representations of x_u and x'_u, respectively. In similar ways, we consider all other unsorted sentences and then generate a vector $m_d(x_u)$ as

$$m_d(x_u) = \frac{1}{|X_u|}(\sum_{x_{u'} \in X_u} [v'_{u,u'}; P_d(*|x_u, x_{u'})]), \tag{12}$$

where X_u denotes the set of unordered sentences except x_u. Via this distribution, our model is capable of exploiting global relative orientation of other unsorted sentences to x_u.

HISTORY Module. We model two Bernoulli distributions $P_{l1}(b|x_u, x_{o'_{i-1}})$ and $P_{l2}(b|x_u, x_{o'_{i-2}})$, where $b \in$\{*true*,*false*\}. $P_{l1}(true|x_u, x_{o'_{i-1}})$ measures the probability of previously predicted sentence $x_{o'_{i-1}}$ occurring before x_u with a relative distance 1. It is defined as follows:

$$p_{u,i-1} = \text{ReLU}(W_1 s_u + W_2 s_{o'_{i-1}}), \tag{13}$$

$$v_{u,i-1} = \text{ReLU}(W_v p_{u,i-1}), \tag{14}$$

$$P_{l1}(*|x_u, x_{o'_{i-1}}) = \text{softmax}(W_s v_{u,i-1}), \tag{15}$$

where W_* are model parameters. The definition of $P_{l2}(*|x_u, x_{o'_{i-2}})$ is similar to that of $P_{l1}(*|x_u, x_{o'_{i-1}})$, with a different relative distance 2. To model this distribution, we employ the same equations as $P_{l1}(*|x_u, x_{o'_{i-1}})$, however, with different parameters. Due to the space limitation, we omit the specific equation descriptions of $P_{l2}(*|x_u, x_{o'_{i-2}})$. Then, $m_{l1}(x_u)$ and $m_{l2}(x_u)$ implying local coherence are generated as

$$m_{l1}(x_u) = [v_{u,i-1}; P_{l1}(*|x_u, x_{o'_{i-1}})], \tag{16}$$

$$m_{l2}(x_u) = [v_{u,i-2}; P_{l2}(*|x_u, x_{o'_{i-2}})]. \tag{17}$$

By introducing these two distributions, we expect our model is able to accurately measure the local coherence between each candidate and its previously ordered sentences, without the influence of noisy context.

Finally, we concatenate the above three vectors to form a new vector $m(x_u) = [m_{l1}(x_u); m_{l2}(x_u); m_d(x_u)]$, which not only encodes relative orientations of other unsorted sentence with respect to x_u, but also measures local coherence between previously ordered sentence and x_u. Please note that we combine both the semantic vectors and the probability distributions for providing richer information for x_u. Likewise, we generate such vectors for all unordered sentences, which are then packed into a matrix M_i.

Our decoder has two advantages over the standard pointer network decoder. First, our decoder is capable of exploiting pairwise relative orientations between unsorted sentences, which is encoded by the distribution $P_d(*|x_u, x_{u'})$. Essentially, this information plays an important role to provide future hints for the current prediction. Second, without the effect of the left-side encoded noisy context, our decoder reviews the local coherence between previously ordered sentences and each candidate $P_{l1}(*|x_u, x_{o'_{i-1}})$ and $P_{l2}(*|x_u, x_{o'_{i-2}})$ to inspect the rationality of selecting the candidate x_u.

Training and Testing

Given a training corpus $D = \{(\boldsymbol{x}, \boldsymbol{o})\}$, we train the proposed model by minimizing the loss function:

$$\begin{aligned}\mathcal{L}(\theta) = -\frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{o}) \in D} \{logP(\boldsymbol{o}|\boldsymbol{x}) \\ + \lambda(\mathcal{L}_{l1} + \mathcal{L}_{l2} + \mathcal{L}_d)\},\end{aligned} \tag{18}$$

where θ denotes the set of all trainable parameters, $\mathcal{L}_{l1}$ and $\mathcal{L}_{l2}$, and $\mathcal{L}_d$ are cross-entropy loss functions of HISTORY and FUTURE modules, respectively, and λ is a hyperparameter used to balance the preference between two terms. During testing, we employ beam search to select sentences sequentially.

To train the two distributions in the HISTORY module, we need to sample negative instances to balance the numbers of positive and negative instances. Take training the classifier $P_{l1}(*)$ as an example. For a sentence in the input set, we choose it and its previously ordered sentence at a relative distance 1 to form a positive sample, and pair it with those at other relative distances as negative samples. This inevitably causes imbalance between positive and negative instances. Therefore, we sample the negative instances to make the number of negative instances to that of positive instances in a ratio of 1 to 2.

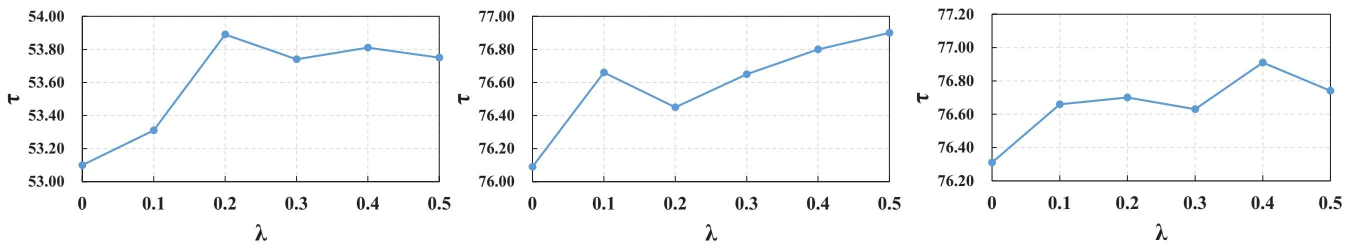
Experiments

Datasets

We carry out experiments on three benchmark datasets:

- **SIND** (Huang et al. 2016). It is a visual storytelling dataset, which includes 40,155 training stories, 4,990 validation stories and 5,055 testing stories. Here we use each story text as a paragraph that is composed of 5 sentences.
- **ROCStory** (Mostafazadeh et al. 2016). This dataset is a commonsense story one, which contains 98,162 stories with 50 words per story on average. Each story is composed of 5 sentences. Following (Wang and Wan 2019), we make an 8:1:1 random split on the dataset to get the training, validation and testing datasets of 78,529, 9,816 and 9,817 stories, respectively.

[1] In this paper, all bias terms in neural network functions are omitted for readability.



(a) SIND (b) ROCStory (c) arXiv

Figure 3: Experiment results on the validation sets using different λs.

- **arXiv Abstract** (Chen, Qiu, and Huang 2016). This dataset is collected from arXiv website. It consists of 884,912 training abstracts, 110,614 validation abstracts and 110,615 testing abstracts, and thus is obviously larger than the above two. Each abstract is composed of 2 to 20 sentences and the average word count per abstract is around 135.

Baseline Models and Metrics

We have three most related baselines:

- **Baseline**. It is our re-implemented ATTOrderNet (Cui et al. 2018).
- **Baseline(MSA)**. It is a variant of our **Baseline**, of which decoder is replaced by masked self-attention mechanism based on (Wang and Wan 2019). Here, we compare our model with this variant because such mechanism is analogous to our HISTORY module, which enables the decoder to capture semantic relation between sentences predicted.
- **MTL**. Another variant of our model, it uses multi-task learning based on the shared encoder to jointly model pointer network based sentence ordering and pairwise ordering predictions. Note that its decoder is not enhanced by pairwise ordering predictions.

In addition, we compare with several commonly-used contrast models:

- **LSTM+Pairwise** (Chen, Qiu, and Huang 2016). It is a pairwise ranking model with an LSTM encoder.
- **SkipThought+Pairwise** (Agrawal et al. 2016). This is a pairwise model which takes a pair of SkipThought sentence embeddings as input.
- **Seq2Seq+Pairwise** (Li and Jurafsky 2017). This model predicts the next sentence given the current sentence. Please note that this model is also a pairwise one.
- **LSTM+Ptr** (Chen, Qiu, and Huang 2016). It is an end-to-end approach based on pointer network. It treats the out-of-order set of sentences as a sequential input for encoder and predicts sentence orders recurrently.
- **LSTM+Set2Seq** (Logeswaran, Lee, and Radev 2018). This model is based on set-to-sequence framework and also adopts pointer network. The set encoder learns a context representation by iteratively attending to input sentence embeddings.
- **ATTOrderNet** (Cui et al. 2018). Self-attention mechanism is first introduced into this task. Compared to previous models, it is less sensitive for the permutation of input sentences.
- **HAN** (Wang and Wan 2019). Both encoder and decoder of this model are equipped with self-attention mechanism. Besides, the hierarchical attention network of its encoder captures both word clues and dependencies between sentences.
- **SE-Graph** (Yin et al. 2019). The encoder exploits both semantic relevance between coherent sentences and co-occurrence between sentences and entities to accurately learn semantic representations of sentences.

Finally, we use Kendall's τ and Perfect Match Ratio (PMR) as metrics, both of which have been frequently used in previous work (Gong et al. 2016; Cui et al. 2018; Wang and Wan 2019).

Setting

We use *Adadelta* (Zeiler 2012) as the optimizer with $\epsilon = 10^{-6}$, $\rho = 0.95$, where the initial learning rate is set as 1.0. The used batch size is 64 and the beam size is 8. We use pre-trained 100-dimensional GloVe word embeddings (Pennington, Socher, and Manning 2014). The sizes of LSTM hidden states in encoder and decoder are set to 512, and the hidden size of self-attention layers is also 512. We employ 2 self-attention layers, each of which has 4 parallel attention heads. The hidden size of pairwise modules is set to 256. We apply *dropout* (Srivastava et al. 2014) to word embedding layer and self-attention layers with the probability 0.1.

Effect of λ

As shown in Equation (18), the hyper-parameter λ is an important hyper-parameter, which directly reflects impacts of pairwise prediction modules on our model. Following common practices to determine the optimal hyper-parameters on each validation set (Cui et al. 2018; Wang and Wan 2019), we investigate the performance of our model with different λs. To this end, we gradually vary λ from 0.0 to 0.5 with an increment of 0.1 in each step. From Figure 3, we observe that our model achieves the best performance when λ=0.2, 0.5, 0.4 on SIND, ROCStory, and arXiv datasets. Therefore, we set λ=0.2, 0.5, 0.4 in all experiments thereafter, respectively.

Model	SIND		ROCStory		arXiv	
	τ	PMR	τ	PMR	τ	PMR
LSTM+Pairwise (Chen, Qiu, and Huang 2016) †	-	-	-	-	65.94	33.43
SkipThought+Pairwise (Agrawal et al. 2016) †	-	-	46.40	-	-	-
Seq2Seq+Pairwise (Li and Jurafsky 2017) †	18.92	12.50	34.19	17.93	5.93	13.70
LSTM+Ptr (Gong et al. 2016) †	48.42	12.34	71.58	40.44	-	-
LSTM+Set2Seq (Logeswaran, Lee, and Radev 2018) †	49.19	13.80	71.12	35.81	72.81	41.57
ATTOrderNet (Cui et al. 2018) †	49.00	14.01	-	-	73.00	42.19
HAN (Wang and Wan 2019) †	50.21	15.01	73.22	39.62	75.36	44.55
SE-Graph (Yin et al. 2019) †	52.00	16.22	-	-	75.00	44.33
Baseline	51.67	15.56	73.59	40.01	74.75	43.76
Baseline(MSA)	52.37	15.43	74.38	40.76	75.48	44.55
MTL	52.28	16.22	74.60	41.29	74.86	43.93
Ours	**53.19***	**17.37***	**76.81***	**46.00***	**76.65***	**46.58***

Table 1: Main results on the sentence ordering task. The marker † indicates previously reported scores, and * means significant at $p < 0.01$ over the best one among all three baselines on each test set. Here we conduct 1,000 bootstrap tests (Efron and Tibshirani 1994; Koehn 2004) to measure the significance in metric score differences.

Model	SIND		ROCStory		arXiv	
	τ	PMR	τ	PMR	τ	PMR
Ours	**53.19**	**17.37**	**76.81**	**46.00**	**76.65**	**46.58**
-$P_d(*)$	52.50	15.73	74.14	40.92	75.13	44.23
-$P_{l1}(*)$	52.79	16.44	76.24	44.66	76.06	45.53
-$P_{l2}(*)$	52.93	17.10	76.43	45.74	76.46	46.22
Baseline	51.67	15.56	73.59	40.01	74.75	43.76

Table 2: Ablation studies on three datasets.

Main Results

Table 1 reports the overall experimental results. The proposed model significantly outperforms both previous state-of-the-art models and all three baselines, demonstrating the effectiveness of our model. Moreover, we draw the following conclusions:

(1) *Ours* exhibits much better performance than *Baseline*, indicating that the exploration of pairwise ordering predictions are indeed complementary to the left-side encoded context utilized by the standard pointer network decoder.

(2) On all datasets, *Ours* outperforms *Baseline(MSA)*. The underlying reason is that compared with the masked self-attention mechanism that only captures the semantic relation between predicted sentences, our model fully exploits global orientation between and local coherence between other sentences and each candidate, both of which play positive roles in sentence ordering.

(3) *MTL* also performs sightly better than *Baseline*, indicating that adding pairwise ordering prediction loss functions is beneficial to the model training. Besides, the performance of *Ours* is significantly better than *MTL*. This demonstrates the effectiveness of incorporating the global orientation and local coherence generated from the pairwise ordering modules into the decoder as the complementary context.

Ablation Study

In this section, we conduct an ablation study to investigate the impacts of different modules on our model. All the results are reported in Table 2. Here, we can draw some interesting conclusions.

First, the variant of our model without the FUTURE module is obviously inferior to our model. The result is intuitive and indicates pairwise ordering prediction between unsorted sentences is the most effective among all introduced distributions.

Second, removing the HISTORY module leads to the performance degradation of our model. Moreover, the impact of removing $P_{l1}(*|x_u, x_{o'_{i-1}})$ is greater than that of $P_{l2}(*|x_u, x_{o'_{i-2}})$. This is reasonable, since it is more difficult to accurately model $P_{l2}(*|x_u, x_{o'_{i-2}})$ than $P_{l1}(*|x_u, x_{o'_{i-1}})$, as reported in the following classification experiments.

We investigate the classification performance of the pairwise ordering prediction modules on the three test sets. As implemented in classifier training, we employ the same approach to extract test instances from the test sets without sampling. The classifiers for $P_{l1}(*|x_u, x_{o'_{i-1}})$ and $P_{l2}(*|x_u, x_{o'_{i-2}})$ achieve 60.27% and 55.72%, 67.13% and 63.56%, 62.82% and 59.51% accuracies on SIND, ROCStory, and arXiv datasets, respectively. Note that the performances of the classifier predicting $P_{l2}(*|x_u, x_{o'_{i-2}})$ on SIND and arXiv seem unsatisfactory. The underlying reason is that the classifiers are difficult to be trained sufficiently due to the small corpus size of SIND and more diverse paragraphs in arXiv dataset. This echoes the results reported in Table 2 where the performance of our model drops slightly when removing this distribution. Besides, the classifier for $P_d(ori|x_u, x'_u)$ achieves 75.98%, 85.45%, 83.18% accuracies on the three test sets, respectively.

Table 3 shows an example. In this example, among all models, only *Ours* is able to produce the correct sequence

Input Sentences	(1) He never really got tan just really sunburnt. (2) Rick had very pale skin. (3) He did his best to always come in when his skin felt hot. (4) He fell asleep next to the pool one morning. (5) He was not only red but had blisters all over from the burns.
Ground Truth	(2) (1) (3) (4) (5)
Baseline	(2) **(3) (4) (5) (1)**
Baseline(MSA)	(2) (1) **(4) (5) (3)**
MTL	(2) **(3) (4) (5) (1)**
Ours	(2) (1) (3) (4) (5)

Table 3: Sentence ordering results produced by different models. Texts highlighted in bold are incorrect ordering sequences.

Model	arXiv		SIND	
	head	tail	head	tail
LSTM+Pairwise†	84.85	62.37	-	-
LSTM+Ptr†	90.47	66.49	74.66	53.30
ATTOrderNet†	91.00	68.08	76.00	54.42
Baseline	92.05	69.05	77.45	56.50
Baseline(MSA)	92.29	70.42	77.53	57.07
MTL	92.25	69.45	77.57	56.87
Ours	**92.76**	**71.49**	**78.08**	**57.32**

Table 4: Ratios of correctly predicting the first and last sentences on arXiv and SIND datasets. † indicates previously reported scores.

of ordered sentences. Particularly, although both *Ours* and *Baseline(MSA)* have the same correct previously ordered sentences (2) (1), only *Ours* can make a correct prediction at the 3rd timestep. This is because the FUTURE module gives high probabilities of sentence (3) appearing before sentences (4) and (5) (0.96 and 0.87). Meanwhile, the HISTORY module provides supportive probabilities of sentence (1) and (2) appearing before sentence (3) (0.83 and 0.65) at relative distances 1 and 2, respectively. In contrast, *Ours* gives low probabilities to another candidate sentence (4).

Prediction of the First and Last Sentences

As mentioned in previous work (Chen, Qiu, and Huang 2016; Gong et al. 2016; Cui et al. 2018), the first and the last sentences of the text should be paid more attention. Thus, we also conduct experiment to predict the first and last sentences of a paragraph. As shown in Table 4, *Ours* also outperforms all listed contrast models, reaching the best performance on arXiv and SIND datasets.

Summary Coherence Evaluation

As discussed in previous studies on text coherence (Barzilay and Lapata 2005; 2008), sentence ordering models can help generate a coherent text in downstream tasks such as multi-document summarization (Bollegala, Okazaki, and Ishizuka 2006; Nayeem and Chali 2017). Here we apply our model to extractive multi-document summarization and evaluate its effect in this task.

Model	Coherence
LexRank	41.95
Baseline	45.92
Baseline(MSA)	46.35
MTL	46.45
Ours	48.67

Table 5: Coherence probabilities of reordered summaries.

Specifically, we use a large-scale summarization dataset (Fabbri et al. 2019), where the average number of sentences in each summary is 9.97. We first train various neural sentence reordering models using summaries in this dataset. Then, following Nayeem and Chali (2017), we apply LexRank (Erkan and Radev 2004) on DUC 2004 (Task-2) to extract summaries and utilize these models to reorder the extracted summaries. Finally, since ROUGE scores focus on content similarity between system outputs and references, and insensitive to summary coherence, we compute the coherence probability (Lapata and Barzilay 2005; Nayeem and Chali 2017) of the summary reordered by different models:

$$\begin{aligned}\text{coherence}(\boldsymbol{x}) &= \frac{\sum_{i=1}^{n-1}\text{Sim}(x_i, x_{i+1})}{n-1},\\ \text{Sim}(x_i, x_{i+1}) &= \lambda * \text{NESim}(x_i, x_{i+1})\\ &\quad + (1-\lambda) * \text{CosSim}(x_i, x_{i+1}),\end{aligned} \tag{19}$$

where n is the number of sentences in a summary, NESim(x_i, x_{i+1}) calculates overlap of named entities in adjacent sentences and CosSim(x_i, x_{i+1}) calculates cosine similarity between the sentence vectors that is the weighted sum of word embeddings. We choose $\lambda = 0.8$, giving more preference to the named entities.

As shown in Table 5, compared with other baselines, the better performance of our model verifies the benefit of pairwise ordering predictions to pointer network.

Conclusion and Future Work

In this paper, we have thoroughly analyzed the drawback of the pointer network decoder for sentence ordering, and have presented a method to enhance the decoder with pairwise ordering predictions. Experimental results and in-depth analyses strongly demonstrate the effectiveness of our decoder.

In the future, we will investigate how to jointly leverage training corpora of different domains. Besides, we plan to introduce bidirectional decoding (Zhang et al. 2018; Su et al. 2019) to refine sentence ordering.

Acknowledgments

The authors were supported by Beijing Advanced Innovation Center for Language Resources, National Natural Science Foundation of China (No. 61672440), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Scientific Research Project of National Language Committee of China (Grant No. YB135-49).

References

Agrawal, H.; Chandrasekaran, A.; Batra, D.; Parikh, D.; and Bansal, M. 2016. Sort story: Sorting jumbled images and captions into stories. In *EMNLP*.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A. C.; and Bengio, Y. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Barzilay, R., and Lapata, M. 2005. Modeling local coherence: An entity-based approach. In *ACL*.

Barzilay, R., and Lapata, M. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*.

Barzilay, R., and Lee, L. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *NAACL*.

Barzilay, R.; Elhadad, N.; and McKeown, K. R. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *JAIR*.

Bollegala, D.; Okazaki, N.; and Ishizuka, M. 2006. A bottom-up approach to sentence ordering for multi-document summarization. In *ACL*.

Chen, X.; Qiu, X.; and Huang, X. 2016. Neural sentence ordering. *CoRR*.

Cui, B.; Li, Y.; Chen, M.; and Zhang, Z. 2018. Deep attentive sentence ordering network. In *EMNLP*.

Efron, B., and Tibshirani, R. J. 1994. *An introduction to the bootstrap*. CRC press.

Erkan, G., and Radev, D. R. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*.

Fabbri, A. R.; Li, I.; She, T.; Li, S.; and Radev, D. R. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *ACL*.

Galanis, D.; Lampouras, G.; and Androutsopoulos, I. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *COLING*.

Gong, J.; Chen, X.; Qiu, X.; and Huang, X. 2016. End-to-end neural sentence ordering using pointer network. *CoRR*.

Guinaudeau, C., and Strube, M. 2013. Graph-based local coherence modeling. In *ACL*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*.

Huang, T.-H. K.; Ferraro, F.; Mostafazadeh, N.; Misra, I.; Agrawal, A.; Devlin, J.; Girshick, R.; He, X.; Kohli, P.; Batra, D.; Zitnick, C. L.; Parikh, D.; Vanderwende, L.; Galley, M.; and Mitchell, M. 2016. Visual storytelling. In *NAACL*.

Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Konstas, I., and Lapata, M. 2012. Unsupervised concept-to-text generation with hypergraphs. In *NAACL*.

Lapata, M., and Barzilay, R. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*.

Lapata, M. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *ACL*.

Li, J., and Hovy, E. H. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*.

Li, J., and Jurafsky, D. 2017. Neural net models of open-domain discourse coherence. In *EMNLP*.

Logeswaran, L.; Lee, H.; and Radev, D. R. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI*.

Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. F. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*.

Nallapati, R.; Zhai, F.; and Zhou, B. 2012. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Nayeem, M. T., and Chali, Y. 2017. Extract with order for coherent multi-document summarization. In *ACL Workshop*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Serdyuk, D.; Ke, N. R.; Sordoni, A.; Trischler, A.; Pal, C.; and Bengio, Y. 2018. Twin networks: Matching the future for sequence generation. In *ICLR*.

Song, L.; Gildea, D.; Zhang, Y.; Wang, Z.; and Su, J. 2019. Semantic neural machine translation using AMR. *TACL* 7:19–31.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*.

Su, J.; Zhang, X.; Lin, Q.; Qin, Y.; Yao, J.; and Liu, Y. 2019. Exploiting reverse target-side contexts for neural machine translation via asynchronous bidirectional decoding. *Artif. Intell.*

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. In *ICLR*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *NIPS*.

Wang, T., and Wan, X. 2019. Hierarchical attention networks for sentence ordering. In *AAAI*.

Yin, Y.; Song, L.; Su, J.; Zeng, J.; Zhou, C.; and Luo, J. 2019. Graph-based neural sentence ordering. In *IJCAI*.

Zeiler, M. D. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701.

Zhang, X.; Su, J.; Qin, Y.; Liu, Y.; Ji, R.; and Wang, H. 2018. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*.

Zhang, Y.; Liu, Q.; and Song, L. 2018. Sentence-state LSTM for text representation. In *ACL*.

Zheng, Z.; Zhou, H.; Huang, S.; Mou, L.; Dai, X.; Chen, J.; and Tu, Z. 2018. Modeling past and future for neural machine translation. *TACL*.