Bidirectional Sentence Ordering with Interactive Decoding

GUIRONG BAI, SHIZHU HE, KANG LIU, and JUN ZHAO, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, and School of Artificial Intelligence, University of Chinese Academy of Sciences

Sentence ordering aims at restoring orders of shuffled sentences in a paragraph. Previous methods usually predict orders in a single direction, i.e., from head to tail. However, unidirectional prediction inevitably causes error accumulation, which restricts performance. In this article, we propose a bidirectional ordering method, which predicts orders in both head-to-tail and tail-to-head directions at the same time. In our bidirectional ordering method, two directions can interact with each other and help alleviate the error accumulation problem of ordering. Experiments demonstrate that our method can effectively improve performance of previous models.

CCS Concepts: • **Computing methodologies → Discourse, dialogue and pragmatics;**

Additional Key Words and Phrases: Sentence ordering, bidirectional, interactive

ACM Reference format:
Guirong Bai, Shizhu He, Kang Liu, and Jun Zhao. 2023. Bidirectional Sentence Ordering with Interactive Decoding. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 22, 2, Article 45 (March 2023), 15 pages.
https://doi.org/10.1145/3561510

1 INTRODUCTION

Sentence ordering [3] is an important task in **natural language processing** (**NLP**), which is to automatically organize shuffled sentences in a paragraph into the correct order. Besides better reading and understanding text, sentence ordering is useful for many other tasks in NLP, such as concept-to-text generation [16–18], retrieval-based question answering [36, 41], answer summarization [9], and extractive multi-document summarization [2, 10, 24, 26, 28, 38, 43].

Recently, with development of deep learning, neural models are proposed for sentence ordering and achieve significant improvements [7, 8, 13, 23, 27, 29, 40, 45]. Typically, these models employ a pointer network [37] as a decoder and predict orders in a single direction, i.e., from head to tail. However, unidirectional prediction inevitably causes error accumulation, which makes it difficult to correctly predict orders at farther timesteps. As shown in Figure 1, there are results predicted in a single direction by the typical model [7]. We can see it performs well at earlier timesteps

The work is supported by the National Natural Science Foundation of China under Grant Nos. 61533018, U1936207, 61976211, and 61702512, and the independent research project of National Laboratory of Pattern Recognition under Grant. This research work was also supported by Youth Innovation Promotion Association CAS.
Authors' address: G. Bai, S. He, K. Liu, and J. Zhao, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, and School of Artificial Intelligence, University of Chinese Academy of Sciences, No. 95, Zhongguancun East Road, Beijing 100190, China; emails: {guirong.bai, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn.

2375-4699/2023/03-ART45 $15.00
https://doi.org/10.1145/3561510

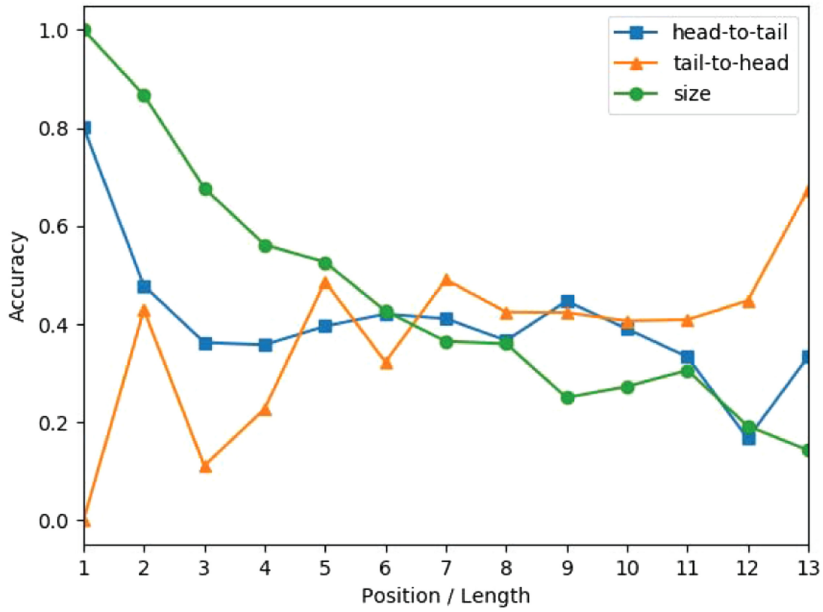

Fig. 1. The head-to-tail and tail-to-head are accuracy curves at different positions in head-to-tail and tail-to-head directions, respectively. The curve "size" shows the accuracy with different numbers of shuffled sentences (length of paragraph). The number of samples causes some fluctuation.

and performs poorly at farther timesteps. Specifically, the accuracy gradually decreases when the timestep increases. Moreover, when the number of shuffled sentences is larger, the accuracy is worse. These statistics demonstrate that the error accumulation problem does exist and restricts performance. Besides, we can see the accuracy is significantly higher at earlier timesteps. This is because both the beginning and ending of a paragraph have obvious features to identify. For example, in the paper abstract dataset [23], the first sentence usually contains "in this paper ..." to describe contribution, and the last sentence usually contains "experiments demonstrate ..." to talk about empirical results. Therefore, simultaneously using head-to-tail and tail-to-head prediction may take advantage of this feature and alleviate the error accumulation problem.

In this article, we propose a bidirectional ordering method for sentence ordering. Comparing with previous unidirectional methods, our method can predict orders in both head-to-tail and tail-to-head directions at the same time. Specifically, we bridge the connection between two different directions by their decoding history. At every timestep, the decoding history of both head-to-tail and tail-to-head directions is stored. When predicting, two different directions interact with each other through decoding history. With the interaction, we can simultaneously utilize information from different directions and alleviate the error accumulation problem. Our method has compatibility and is easy to apply in other models of sentence ordering. We conduct experiments on four datasets. The results demonstrate that our method can effectively alleviate the error accumulation problem and improve performance of previous models.

In brief, our main contributions are shown as follows:

— We find that (1) the error accumulation problem in sentence ordering, (2) the forward model does well in predicting the head (3) and the backward model does well in predicting the tail. Thus we propose to make use of backward prediction to enhance forward prediction (vice versa).

	Shuffled Sentences		*Order Sentecnes*
s_1	**During the landing, two vehicles were struck.**	s_2	**All engine power was lost.**
s_2	**All engine power was lost.**	s_3	**The pilots then performed a forced landing.**
s_3	**The pilots then performed a forced landing.**	s_1	**During the landing, two vehicles were struck.**

Fig. 2. An example of sentence ordering task.

— We propose a bidirectional ordering method for sentence ordering, which is able to fuse the forward prediction and the backward prediction to alleviate the error accumulation problem and reduce the difficulty of finding the correct order.
— We verify the effectiveness of the bidirectional ordering method for sentence ordering, and the experimental results demonstrate that our proposed method is useful to improve previous models.

2 PRELIMINARY

2.1 Task Definition

The sentence ordering task aims at ordering a set of shuffled sentences in a paragraph as a coherent text. An example is shown in Figure 2. There are three shuffled sentences s_1, s_2, and s_3. The ordering models need to obtain the correct order $s_2 \rightarrow s_3 \rightarrow s_1$.

Formally, L shuffled sentences are denoted by $\mathbf{x} = [s_{o_1}, s_{o_2}, \ldots, s_{o_L}]$, where $\mathbf{o} = [o_1, o_2, \ldots, o_L]$ is the shuffled order list. The goal is to find the correct order $\mathbf{o}^* = [o_1^*, o_2^*, \ldots, o_L^*]$ for them from the order space ($L!$). With the correct order $\mathbf{o}^*$, the whole sentences have the highest coherence probability:

$$P(\mathbf{o}^*|\mathbf{x}) > P(\mathbf{o}|\mathbf{x}), \forall \mathbf{o} \in \boldsymbol{\psi}, \tag{1}$$

where $\mathbf{o}$ indicates any order of input sentences, and $\boldsymbol{\psi}$ indicates all possible orders of these sentences.

2.2 Framework

For sentence ordering, the input is shuffled sentences in a paragraph and the output is their correct order. Recently, the encoder-decoder framework in Figure 3 obtains state-of-the-art results. In this article, we adopt this framework.

Sentence Encoder: First, a sentence encoder is used to obtain a sentence-level representation of each single sentence:

$$\mathbf{s}_{o_i} = \text{SentEnc}(w_1, w_2, \ldots, w_T), \tag{2}$$

where $\mathbf{s}_{o_i}$ denotes the representation of the sentence s_{o_i} with T words. Word embedding of every word can be obtained with a shared word embedding matrix $\mathbf{W}_e \in \mathbb{R}^{n_e \times d_e}$, where n_e denotes the vocabulary size and d_e denotes the embedding size. Then word embedding is sent to the sentence encoder SentEnc() to obtain the sentence-level representation. Typically, previous models adopt bidirectional LSTM [14] as the sentence encoder SentEnc() and regard the output of the last hidden state as the sentence representation.

Paragraph Encoder: Then, a paragraph encoder is used to obtain a paragraph-level representation of these sentences:

$$\mathbf{v}_{para} = \text{ParaEnc}(\mathbf{s}_{o_1}, \mathbf{s}_{o_2}, \ldots, \mathbf{s}_{o_L}), \tag{3}$$

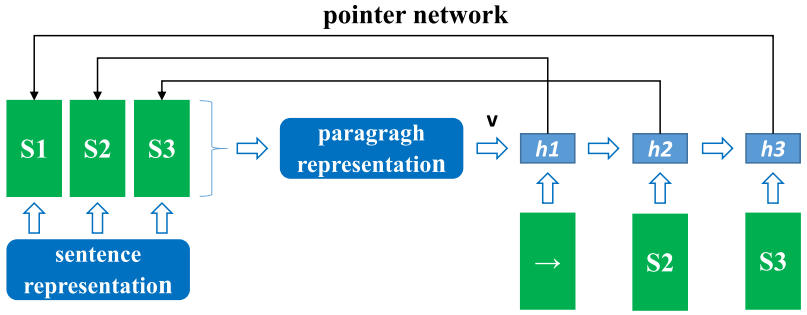

Fig. 3. The encoder-decoder framework with a pointer network for sentence ordering.

where the paragraph encoder ParaEnc() encodes the whole sentences and obtains the paragraph-level representation $\mathbf{v}_{para}$, which can capture global dependencies among sentences and help predict orders for the decoder. In recent studies, ParaEnc() is made up of multiple self-attention [32, 34, 35] layers. $\mathbf{v}_{para}$ can be obtained by average pooling for all sentences at the last self-attention layer.

Decoder: Finally, a pointer network [37] is employed as the decoder to predict the order of input sentences:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\text{Func}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{s}_{\overrightarrow{o}_{i-1}}), \tag{4}$$

$$\overrightarrow{\mathbf{u}}_j^i = \mathbf{g}^\top \tanh(\mathbf{W}_1 \mathbf{s}_{o_j} + \mathbf{W}_2 \overrightarrow{\mathbf{h}}_i), \tag{5}$$

$$P(\overrightarrow{o}_i | \overrightarrow{o}_{i-1}, \ldots, \overrightarrow{o}_1, \mathbf{x}) = \text{softmax}(\overrightarrow{\mathbf{u}}^i), \tag{6}$$

where $\rightarrow$ denotes the head-to-tail direction. $\overrightarrow{\text{Func}}()$ is a function for recurrent prediction. Both self-attention and LSTM can be employed as $\overrightarrow{\text{Func}}()$. $\overrightarrow{o}_{i-1}$ is the predicted order at last timestep and its sentence is sent to predict the next order. The initial state $\overrightarrow{\mathbf{h}}_0 \in \mathbb{R}^d$ is the paragraph-level representation $\mathbf{v}_{para}$ and the first input $\mathbf{s}_{\overrightarrow{o}_0}$ is zero vector. $\mathbf{g} \in \mathbb{R}^d$, $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are learnable parameters. $\overrightarrow{\mathbf{h}}_i$ is used to output the conditional probability of the order $P(\overrightarrow{o}_i | \overrightarrow{o}_{i-1}, \ldots, \overrightarrow{o}_1, \mathbf{s})$ with these parameters.

3 THE PROPOSED METHOD

3.1 Bidirectional Ordering with Interactive Decoding

Previous unidirectional models inevitably cause error accumulation. Therefore, we propose a bidirectional ordering method, which predicts orders in both head-to-tail and tail-to-head directions at the same time. When decoding orders, different directions can interact with each other by decoding history and reduce the difficulty of ordering. The architecture of the method is illustrated in Figure 4. Similarly, studies in [42] also propose to use a bidirectional decoder for machine translation. However, our method has some obvious differences. In sentence order, the output to predict is fixed sentences that can be predicted once and only once. Thus, our model is not generative. Moreover, our model handles the symmetrical characteristic in sentence order.

In bidirectional ordering, we also adopt the encoder-decoder framework in Section 2.2. The difference is that there are two synchronous bidirectional decoders interacting with each other. Specifically, Equation (5) in Section 2.2 is changed as follows:

$$\overrightarrow{\mathbf{u}}_j^i = \mathbf{g}^\top \tanh(\mathbf{W}_1 \mathbf{s}_{o_j} + \mathbf{W}_2((1 - \overrightarrow{\lambda}_i)\overrightarrow{\mathbf{h}}_i + \overrightarrow{\lambda}_i \overleftarrow{\mathbf{m}}_i)), \tag{7}$$

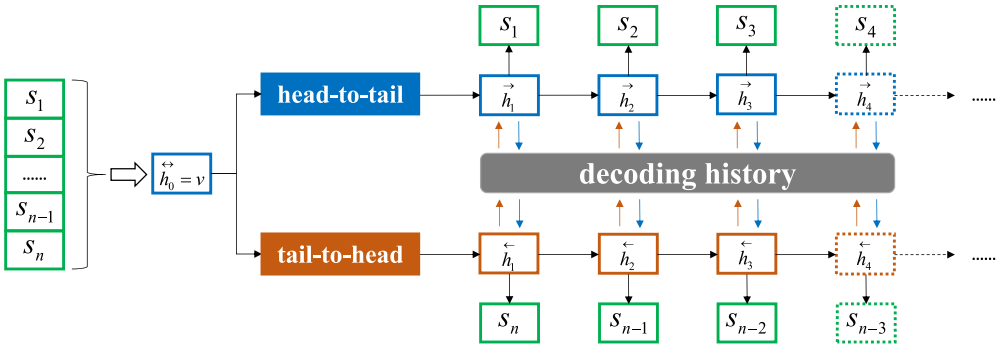

Fig. 4. Architecture of our bidirectional ordering method. Head-to-tail and tail-to-head directions interact with each other by the memory of decoding history.

$$\overrightarrow{\lambda}_i = \text{sigmoid}(\mathbf{W}_\lambda^\top[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{m}}_i] + b_\lambda), \tag{8}$$

where i denotes the timestep. $\overrightarrow{\mathbf{h}}_i$ denotes the hidden layer of head-to-tail decoding. $\overleftarrow{\mathbf{m}}_i$ denotes the memory of decoding history from reverse tail-to-head direction. Through $\overleftarrow{\mathbf{m}}_i$, the connection between two directions is bridged. $\overrightarrow{\lambda}_i$ is the weight to decide which direction is more reliable at the ith timestep. We expect that reverse direction has higher weight at farther timesteps and alleviate the error accumulation problem. $\mathbf{W}_\lambda \in \mathbb{R}^{2d}$ and b_λ are learnable parameters. [;] denotes concatenation.

Similarly, the tail-to-head order is synchronously predicted by the same framework and can also be boosted by the memory of decoding history from its reverse head-to-tail direction:

$$\overleftarrow{\mathbf{h}}_i = \overleftarrow{\text{Func}}(\overleftarrow{\mathbf{h}}_{i-1}, \mathbf{s}_{\overleftarrow{o}_{i-1}}), \tag{9}$$

$$\overleftarrow{\mathbf{u}}_j^i = \mathbf{g}^\top \tanh(\mathbf{W}_1 \mathbf{s}_{o_j} + \mathbf{W}_2((1 - \overleftarrow{\lambda}_i)\overleftarrow{\mathbf{h}}_i + \overleftarrow{\lambda}_i \overrightarrow{\mathbf{m}}_i)), \tag{10}$$

$$\overleftarrow{\lambda}_i = \text{sigmoid}(\mathbf{W}_\lambda[\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{m}}_i] + b_\lambda), \tag{11}$$

$$P(\overleftarrow{o}_i | \overleftarrow{o}_{i-1}, \ldots, \overleftarrow{o}_1, \mathbf{x}) = \text{softmax}(\overleftarrow{\mathbf{u}}^i), \tag{12}$$

where head-to-tail and tail-to-head directions share the same timestep i. $\overrightarrow{\mathbf{m}}_i$ denotes the memory of decoding history from head-to-tail direction.

$\overrightarrow{\mathbf{m}}_i$ and $\overleftarrow{\mathbf{m}}_i$ are the memory of decoding history from different directions, which play an important role in bidirectional ordering. They contain the information of different directions and can send the information from a direction to the other direction. With $\overrightarrow{\mathbf{m}}_i$ and $\overleftarrow{\mathbf{m}}_i$, head-to-tail and tail-to-head directions can interact with each other at the same time and reduce the difficulty of ordering by decoding history of the other direction. Formally, $\overrightarrow{\mathbf{m}}_i$ can be obtained according to decoding history as follows:

$$\overrightarrow{\mathbf{m}}_i = \sum_{j=1}^{i-1} \overrightarrow{\alpha}_{ij} \overrightarrow{\mathbf{h}}_j, \tag{13}$$

$$\overrightarrow{\alpha}_{ij} = \frac{exp(\overrightarrow{e}_{ij})}{\sum_{j'=1}^{i} exp(\overrightarrow{e}_{ij'})}, \tag{14}$$

$$\overrightarrow{e}_{ij} = \mathbf{W}_m^\top \tanh(\mathbf{W}_4(\overrightarrow{\mathbf{s}}_{o_j} + \mathbf{t}_j) + \mathbf{W}_5(\overleftarrow{\mathbf{h}}_i + \mathbf{t}_{L-i+1})), \tag{15}$$

where $\overrightarrow{\mathbf{m}}_i$ consists of hidden layers in head-to-tail direction with attention mechanism [1], which contain information of decoding history. At the first timestep, $\overrightarrow{\mathbf{m}}_i$ is zero vector. $\overrightarrow{\alpha}_{ij}$ denotes the weight of attention mechanism. Intuitively, decoding history at different timesteps has different importance. $\mathbf{W}_4 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_5 \in \mathbb{R}^{d\times d}$ are learnable parameters. $\mathbf{t} \in \mathbb{R}^d$ denotes position embedding and the subscript of $\mathbf{t}$ denotes the position index. L denotes the number of timesteps (total number of shuffled sentences).

Similarly, the tail-to-head prediction is formulated as

$$\overleftarrow{\mathbf{m}}_i = \sum_{j=1}^{i-1} \overleftarrow{\alpha}_{ij} \overleftarrow{\mathbf{h}}_j, \tag{16}$$

$$\overleftarrow{\alpha}_{ij} = \frac{exp(\overleftarrow{e}_{ij})}{\sum_{j'=1}^{i} exp(\overleftarrow{e}_{ij'})}, \tag{17}$$

$$\overleftarrow{e}_{ij} = \mathbf{W}_m^\top \tanh(\mathbf{W}_4(\overleftarrow{\mathbf{s}}_{o_j} + \mathbf{t}_{L-j+1}) + \mathbf{W}_5(\overrightarrow{\mathbf{h}}_i + \mathbf{t}_i)), \tag{18}$$

where two different directions share the same position embedding $\mathbf{t}$. To obtain the final order, we use predictions of both head-to-tail and tail-to-head directions. It will be described in Section 3.3.

Specially, there is a symmetrical relation between head-to-tail and tail-to-head directions. For example, there are 10 sentences, the head-to-tail prediction at the 4th timestep corresponds to the tail-to-head prediction at the $7(10 - 4 + 1)$-th timestep. Therefore, in reverse direction, we use the symmetrical timestep $(L - i + 1)$ for $\mathbf{t}$. With the position embedding, we expect to capture information around symmetrical positions of the reverse direction, and reduce the difficulty of sentence ordering, especially at farther timesteps.

3.2 Training

We train the model with both the head-to-tail direction and the tail-to-head direction together by minimizing the following loss function:

$$\begin{aligned} L = &\frac{1}{N}\sum_{j=1}^{N}\frac{1}{L_j}\sum_{i=1}^{L_j}\log P(\overrightarrow{o}^*_i|\overrightarrow{o}^*_{i-1},\ldots,\overrightarrow{o}^*_1|\mathbf{x}) \\ &+\frac{1}{N}\sum_{j=1}^{N}\frac{1}{L_j}\sum_{i=1}^{L_j}\log P(\overleftarrow{o}^*_i|\overleftarrow{o}^*_{i-1},\ldots,\overleftarrow{o}^*_1|\mathbf{x}) \end{aligned} \tag{19}$$

where N denotes the number of shuffled paragraphs (batch size) and L_j denotes the number of sentences in the jth paragraph. $P(\overrightarrow{o}^*_i|\overrightarrow{o}^*_{i-1},\ldots,\overrightarrow{o}^*_1,\mathbf{x})$ denotes the probability of the correct head-to-tail order at the ith timestep. Similarly, $P(\overleftarrow{o}^*_i|\overleftarrow{o}^*_{i-1},\ldots,\overleftarrow{o}^*_1|\mathbf{x})$ denotes the correct tail-to-head order at the ith timestep.

3.3 Inference

Following previous methods, the coherence probability of the final output sentence order $\mathbf{o}$ is formalized as

$$P(\overrightarrow{\mathbf{o}}|\mathbf{x}) = \prod_{i=1}^{n} P(\overrightarrow{o}_i|\overrightarrow{o}_{i-1},\ldots,\overrightarrow{o}_1|\mathbf{x}), \tag{20}$$

$$P(\overleftarrow{\mathbf{o}}|\mathbf{x}) = \prod_{i=1}^{n} P(\overleftarrow{o}_i|\overleftarrow{o}_{i-1},\ldots,\overleftarrow{o}_1|\mathbf{x}), \tag{21}$$

ALGORITHM 1: Bidirectional Inference.

Input: sentences $\mathbf{x} = [s_{o_1}, s_{o_2}, \ldots, s_{o_L}]$, beam size K, number of sentences L
Output: order $\mathbf{o}^* = [o_1^*, o_2^*, \ldots, o_L^*]$

```
Initialize head-to-tail/tail-to-head order o_h2t/o_t2h and probability p_h2t/p_t2h;
for i = 1; i ≤ L do
    o^i_h2t, p^i_h2t = BeamSearch([o^{i-1}_h2t, o^{i-1}_t2h], K);//descending
    o^i_t2h, p^i_t2h = BeamSearch([o^{i-1}_t2h, o^{i-1}_h2t], K);//descending
end for
if p^i_h2t[0] > p^i_t2h[0] then
    return o* = o^i_h2t[0];
else
    return o* = o^i_t2h[0];
end if
```

$$P(\mathbf{o}|\mathbf{x}) = \max(P(\overrightarrow{\mathbf{o}}|\mathbf{x}), P(\overleftarrow{\mathbf{o}}|\mathbf{x})), \tag{22}$$

where $P(\mathbf{o}|\mathbf{x})$ denotes the distribution of the final output order. $\mathbf{x}$ denotes the input shuffled paragraph. $\rightarrow$ denotes head-to-tail direction and $\leftarrow$ denotes tail-to-head direction.

For bidirectional inference, we design a special beam search in Algorithm 1, where each candidate prediction in a direction is incorporated into every path of beam search in the other direction. Specially, each candidate prediction information in backward prediction is incorporated into every path of beam search in forward prediction at one timestep (vice versa). Comparing with standard beam search, function BeamSearch() has an extra input from the other direction, which means we select top K paths from K^2 candidates at each timestep. Finally, we select the order with the highest probability between head-to-tail and tail-to-head predictions.

4 EXPERIMENTS

4.1 Data

Following previous studies, we adopt four datasets to conduct the experiments. The statistics are shown in Table 1. Specifically, the four datasets are as follows:

NIPS, AAN: They are made up of abstracts from NIPS papers, ACL papers dataset, respectively [23].

arXiv: It consists of abstracts from papers on arXiv website [6].

SIND: It contains photos and corresponding captions [15].

4.2 Evaluation Metrics

There are three evaluation metrics for sentence ordering.

Kendall's tau (τ): It is one of the most popular metrics for the automatic evaluation of text coherence. It is formalized as $\tau = 1 - 2 \times n_{inversion} / \binom{n}{2}$, where $n_{inversion}$ denotes the number of pairs of incorrect relative order in the predicted sequence, n denotes the length of the sequence. It ranges from -1 (the worst) to 1 (the best).

Accuracy (Acc): It measures how many absolute positions of sentences are correctly predicted. It ranges from 0 (the worst) to 1 (the best).

Perfect Match Ratio (PMR): It calculates the ratio of exactly matching orders, which the most stringent measurement in this task. It could only be 0 (not exactly match) or 1 (exactly match) for a paragraph.

Table 1. Statistics of the Data Size, Average Number of Sentences in a Paragraph and Vocabulary Size

	Train	Valid	Test	Len	Vocab
NIPS	2,248	409	402	6.0	16,721
AAN	8,569	962	2,626	4.9	34,485
arXiv	884,912	110,614	110,615	5.4	64,557
SIND	40,155	4,990	5,055	5.0	30,861

4.3 Comparisons

The methods to compare with are as follows:

(1) Sentence-level Neural Models: Pairwise Model [6]; Seq2Seq [22]; SIM [31]. These methods employ neural networks to model sentence representation and predict orders.
(2) Paragraph-level Neural Models: CNN+PtrNet, LSTM+PtrNet [13]; RNN Decoder, V-LSTM+PtrNet [23]; ATTNet [7]; TGCM [27]; SLM [12]; BERT4SO [44]; BERSON [8]. Besides sentence representation, these methods model paragraph representation. They obtain state-of-the-art results.
(3) **B**idirectional **O**rdering with **I**nteractive **D**ecoding: It's our proposed method. We use BOID to indicate it. Paragraph-level models (such as ATTNet, TGCM, and BERSON) are recent sentence ordering models and all of them use a pointer-network type decoder framework. Thus, our method has compatibility and is easy to apply in other models of sentence ordering. Specifically, we can get pointer vectors, candidate vectors, decoding history vectors from these pointer-network based methods. Then, we add BOID into these models with the related vectors.

Models based on manual feature engineering perform significantly worse and thus we don't show their results. Besides, our method is used to enhance previous unidirectional models and we use the same settings of them in BOID. Moreover, all baseline models are evaluated with using beam search.

4.4 Results

The experimental results are shown in Table 2. First, we can see that the results of paragraph-level neural models perform better than sentence-level neural models (e.g, Pairwise Model and Seq2Seq). With employing a pointer network, these models can model the coherence of a paragraph. Therefore, paragraph-level neural models can capture global dependencies among sentences and have better ability to order sentences. Besides, we can see three recent models (ATTNet, TGCM, BERSON) perform better and BERSON performs best with pre-trained parameters.

Moreover, we combine the three best models (ATTNet, TGCM, BERSON) with our proposed method BOID. We can see the three best models have further improvements after combining BOID. It demonstrates that the proposed BOID can enhance previous unidirectional models and obtain better performance. Previous models predict orders in a single direction and ignore the information of the other reverse direction. It inevitably causes error accumulation and makes it difficult to predict orders at farther timesteps. BOID predicts orders in both head-to-tail and tail-to-head directions at the same time, and prediction in two directions can interact with each other. Therefore, BOID can alleviate the error accumulation problem and improve performance.

5 DISCUSSION

In discussion, we employ the typical model ATTNet [7] on NIPS abstract dataset [23] to conduct experiments.

Table 2. Sentence Ordering Results of Different Methods on Four Datasets

Models	NIPS abstract			Models	AAN abstract		
	τ	PMR	Acc		τ	PMR	Acc
Pairwise Model	0.47	19.72	26.63	Pairwise Model	0.58	21.54	41.82
Seq2Seq	0.27	14.39	21.18	Seq2Seq	0.40	18.09	36.62
SIM	0.85	42.29	68.06	SIM	0.86	59.79	75.96
RNN Decoder	0.67	23.31	48.22	RNN Decoder	0.66	21.31	52.06
V-LSTM+PtrNet	0.72	27.87	51.55	V-LSTM+PtrNet	0.73	29.79	58.06
CNN+PtrNet	0.66	26.79	48.64	CNN+PtrNet	0.69	26.65	58.21
LSTM+PtrNet	0.67	28.20	50.87	LSTM+PtrNet	0.69	30.41	58.20
SLM	0.67	29.45	51.02	SLM	0.71	31.85	59.97
BERT4SO	0.75	24.13	-	BERT4SO	0.80	44.42	-
ATTNet	0.72	29.87	56.09	ATTNet	0.73	32.11	63.24
ATTNet+BOID(Ours)	0.73	30.98	57.32	ATTNet+BOID(Ours)	0.75	34.04	64.72
TGCM	0.72	29.02	57.67	TGCM	0.74	34.14	64.65
TGCM+BOID(Ours)	0.74	31.61	59.22	TGCM+BOID(Ours)	0.75	35.79	64.95
BERSON	0.85	48.01	73.87	BERSON	0.85	59.79	78.03
BERSON+BOID(Ours)	**0.85**	**49.37**	**74.72**	BERSON+BOID(Ours)	**0.86**	**61.08**	**79.16**

Models	arXiv abstract			Models	SIND caption		
	τ	PMR	Acc		τ	PMR	Acc
Pairwise Model	0.66	33.43	50.79	Pairwise Model	0.32	10.43	30.75
Seq2Seq	0.52	29.43	45.67	Seq2Seq	0.21	8.64	26.18
SIM	0.82	51.34	66.79	SIM	0.61	25.42	41.89
RNN Decoder	0.66	35.53	48.31	RNN Decoder	0.38	10.68	31.53
V-LSTM+PtrNet	0.72	41.74	55.90	V-LSTM+PtrNet	0.45	13.44	35.26
CNN+PtrNet	0.71	39.28	52.92	CNN+PtrNet	0.48	12.32	35.52
LSTM+PtrNet	0.72	40.44	54.31	LSTM+PtrNet	0.48	12.34	34.45
SLM	0.73	42.75	55.63	SLM	0.50	16.22	37.41
BERT4SO	0.78	49.97	65.41	BERT4SO	0.60	18.83	-
ATTNet	0.73	42.19	56.11	ATTNet	0.49	14.01	36.24
ATTNet+BOID(Ours)	0.75	43.37	57.28	ATTNet+BOID(Ours)	0.51	15.77	37.89
TGCM	0.73	42.51	55.16	TGCM	0.51	14.41	36.75
TGCM+BOID(Ours)	0.75	44.06	56.98	TGCM+BOID(Ours)	0.53	16.12	38.25
BERSON	0.83	56.06	75.08	BERSON	0.65	31.69	58.91
BERSON+BOID(Ours)	**0.84**	**58.82**	**75.45**	BERSON+BOID(Ours)	**0.67**	**34.22**	**59.26**

"-" indicates the number was not reported in the original paper.

5.1 Effectiveness of Direct Hard Bidirectional Decoding

We employ hard-bidirection decoding to show whether bidirectional ordering in BOID can be directly replaced by a simple bidirectional ordering algorithm. This simple method directly puts the prediction of head-to-tail decoder and tail-to-head decoder into the final order, where predicted ones of a decoder will become hard masks for the other and avoid predicting it again. In the method, head-to-tail decoder and tail-to-head decoder don't share the same timestep, and the total timesteps of both the head-to-tail decoder and the tail-to-head decoder will be cut into a half. For example, from the timestep 1 to 5, the prediction order is: [1 (head-to-tail), 5 (tail-to-head), 2 (head-to-tail), 4 (tail-to-head), 3 (head-to-tail)].

Table 3. Results of Direct Hard Bidirectional Ordering

	τ	PMR	Acc
base	56.09	29.87	0.72
hard bidirection	49.45	25.32	0.64
BOID	**57.32**	**30.98**	**0.73**

Table 4. Results at the Head and the Tail with/without BOID

	unidirectional	bidirectional
head	0.796	**0.824**
tail	0.687	**0.725**

Table 5. Results of Different Directions

	τ	PMR	Acc
head-to-tail	56.09	29.87	0.72
tail-to-head	55.73	29.43	0.71
bidirectional	**57.32**	**30.98**	**0.73**

Table 3 shows the results. It is shown that direct hard bidirectional ordering performs worse than ours. Through the results, we can know soft limit with vectors from BOID is better than hard limit with masks. This is because vectors can be smoothly and selectively merged into the model, but masks may bring error accumulation. So the soft limit of BOID is more reliable.

5.2 Performance at the Head and the Tail

We want to know whether BOID can help improve the performance at the head and the tail. Besides, the performance at the head and the tail can reflect whether BOID with bidirectional prediction can reduce the difficulty of ordering at farther timesteps and alleviate the error accumulation problem.

Table 4 shows the results. We can see BOID improves the performance at both the head and the tail. Previous unidirectional models predict the tail at the last timestep and the error accumulation problem is serious at the end. BOID can employ the other direction and help alleviate the error accumulation problem. Therefore, the performance at both the head and the tail has improvements.

5.3 Reliability of Reverse Direction

Here, we show whether the reverse tail-to-head prediction is reliable. Table 5 shows the results of head-to-tail direction and its reverse tail-to-head direction. We can observe that the model also has comparable results when trains and predicts in reverse tail-to-head direction. It can prove that the information of tail-to-head direction is reliable and can be used to model the coherence of sentences. Meanwhile, we find that head-to-tail direction performs better than tail-to-head direction, which shows that the head-to-tail direction is a little easier for modeling coherence. It is intuitive that the first sentence is easier than the last one to identify in most cases.

5.4 Performance on Different Paragraph Lengths

We explore the performance on different paragraph lengths (number of sentences). The results are shown in Figure 5. We can see the accuracy gradually decreases when the length of paragraphs is longer. BOID decreases more slowly and performs better. This is because BOID can synchronously

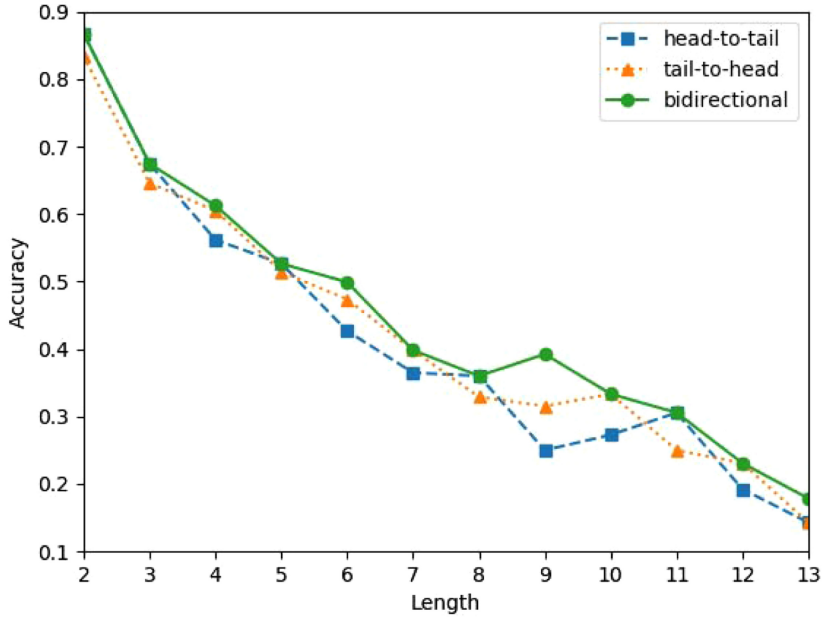

Fig. 5. Changing curves of accuracy with different lengths of paragraphs. There are three results, which belong to the single head-to-tail model, the single tail-to-head model, and BOID, respectively. BOID with bidirectional prediction can perform better for longer lengths.

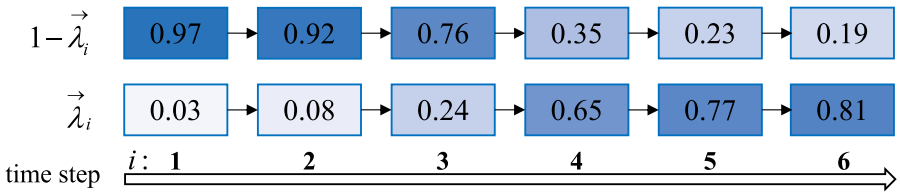

Fig. 6. Visualization for the weight of different directions. The number and color depth denote the weight value. We can see that our method can adaptively adjust the weight of different directions with timestep increasing.

utilize bidirectional information and alleviate the error accumulation problem, especially at farther timesteps.

Although our method alleviates error propagation to a certain extent, we can observe that the performance still drops significantly with longer lengths. Longer distance means more predictions to output and brings more possible errors, which is natural and inevitable. This problem limits our proposed model and is difficult to handle.

5.5 Visualization for the Weight of Different Directions

In Section 3.1, we design the weight $\overrightarrow{\lambda}_i$ and $\overleftarrow{\lambda}_i$ to decide which direction is more reliable at the ith timestep. When predicting, we expect that reverse direction has higher weight at farther timesteps and alleviate the error accumulation problem. To show how the weight changes when ordering, here we sample a real example of the head-to-tail decoder ($\overrightarrow{\lambda}_i$). The visualization is shown in Figure 6.

We can see that the weight of the reverse direction becomes higher when the timestep increases. It is intuitive that reverse direction is more reliable at farther timesteps. The results demonstrate that the weight can adaptively change to alleviate the error accumulation problem. Besides, the weight significantly fluctuates around the middle timestep. This is because we introduce position embedding to capture the symmetrical positions of bidirectional ordering. After timestep crossing the middle timestep, a head-to-tail position can correspond to a symmetrical tail-to-head position and thus position embedding can effectively help adjust the weight.

6 RELATED WORK

The methods for sentence ordering can be divided into three categories. The first is based on manual feature engineering, such as Probabilistic Model [19], Content Model [4], Entity Grid [3], and Utility-Trained Model [33]. However, these methods rely heavily on expensive handcrafted features. The second is based on sentence-level neural models, which utilize neural networks to model sentence representation. For example, Window Network [21] considers the coherence in a window of text, Pairwise Ranking Model [6] orders sentences in a pairwise way, and Seq2Seq [22] models employ an end-to-end framework for sentence ordering. However, they can't capture global dependencies among sentences in a paragraph. The third is based on paragraph-level neural models, which employs LSTM [14] or self-attention [35] as the encoder to model paragraph representation and then a pointer network [37] is used as the decoder to predict orders, such as [7, 8, 13, 23, 27, 40]. However, these methods always order sentences in a single direction, which may aggravate the error accumulation problem, especially at farther timesteps. To this end, we design two interactive decoders to alleviate the problem.

Multiple decoders are proved effective in some other tasks such as machine translation [39, 42] and dialogue generation [25]. However, our method has some obvious differences from them. First, tasks in above methods are generative and their models can't be directly used for sentence ordering. Then, there is an error accumulation problem in sentence ordering, and thus the reverse direction in sentence ordering is expected to have higher weight at farther timesteps. To adaptively adjust the weight, our method models the reliability of different decoders. In addition, candidates in sentence ordering must be predicted once and only once. Therefore, sentence ordering has a symmetrical characteristic. To make use of this characteristic, our method introduces special position embedding.

The easy-first decoding method is also related. For NLP, easy-first decoding was first applied to transition-based parsing in [11]. Similar to curriculum learning [5], easy-first decoding proposes to output sequences with an easy-to-hard direction as a human. Recently, non-autoregressive sequence generation models such as [20] appear, which is not limit to a directional decoding order. However, some studies [30] demonstrate that the difficulty of non-autoregressive generation correlates on the target token dependency, and knowledge distillation as well as alignment constraint reduces the dependency of target tokens and encourages the model to rely more on source context for target token prediction.

7 CONCLUSION

Through statistics of sentence ordering, we find that there is an error accumulation problem in previous unidirectional models. To alleviate the problem, we propose a bidirectional ordering method. It predicts orders in both head-to-tail and tail-to-head directions at the same time. Besides, two different directions are interactive and can enhance each other. Our method has compatibility and is easy to apply in other models of sentence ordering. We conduct experiments on four datasets. Experimental results demonstrate that our method can alleviate the error accumulation problem and improve performance of previous unidirectional models.

REFERENCES

[1] Dzmitry Bahdanau, Kyung Hyun 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*. arXiv:1409.0473. Retrieved from https://arxiv.org/abs/1409.0473.

[2] Regina Barzilay, Noemie Elhadad, and Kathleen R. Mckeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research* 17, 1 (09 2002), 35–55. DOI: https://doi.org/10.1613/jair.991

[3] Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. 141–148. Retrieved from http://aclweb.org/anthology/P05-1018.

[4] Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 113–120.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 41–48.

[6] Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. arXiv:1607.06952. Retrieved from https://arxiv.org/abs/1607.06952.

[7] Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4340–4349. Retrieved from http://aclweb.org/anthology/D18-1465.

[8] Baiyun Cui, Yingming Li, and Zhongfei Zhang. 2020. BERT-enhanced relational sentence ordering network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6310–6320. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.511

[9] Yang Deng, Wenxuan Zhang, Yaliang Li, Min Yang, Wai Lam, and Ying Shen. 2020. Bridging hierarchical and sequential context modeling for question-driven extractive answer summarization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1693–1696.

[10] Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *Proceedings of the COLING*. 911–926. Retrieved from http://aclweb.org/anthology/C12-1056.

[11] Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 742–750. Retrieved from https://aclanthology.org/N10-1115.

[12] Melika Golestani, Seyedeh Zahra Razavi, Zeinab Borhanifard, Farnaz Tahmasebian, and Hesham Faili. 2021. Using BERT encoding and sentence-level language model for sentence ordering. In *Proceedings of the International Conference on Text, Speech, and Dialogue*. Springer, 318–330.

[13] Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. arXiv:1611.04953. Retrieved from https://arxiv.org/abs/1611.04953.

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[15] Ting Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*. 1233–1239.

[16] Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. 369–378. Retrieved from http://aclweb.org/anthology/P12-1039.

[17] Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*. 752–761. Retrieved from http://aclweb.org/anthology/N12-1093.

[18] Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1503–1514. Retrieved from http://aclweb.org/anthology/D13-1157.

[19] Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*.

[20] Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1173–1182. DOI: https://doi.org/10.18653/v1/D18-1149

[21] Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 2039–2048. DOI: https://doi.org/10.3115/v1/D14-1218

[22] Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 198–209. DOI: https://doi.org/10.18653/v1/D17-1019

[23] Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[24] Tatsunori Mori, Masanori Nozawa, and Yoshiaki Asada. 2005. Multi-answer-focused multi-document summarization using a question-answering engine. *ACM Transactions on Asian Language Information Processing* 4, 3 (2005), 305–320.

[25] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of the 26th International Conference on Computational Linguistics*. 3349–3358.

[26] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[27] Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo, and Kyong-Ho Lee. 2019. Topic-guided coherence modeling for sentence ordering by preserving global and local information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 2273–2283. DOI: https://doi.org/10.18653/v1/D19-1232

[28] Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2005. Improving chronological ordering of sentences extracted from multiple newspaper articles. *ACM Transactions on Asian Language Information Processing* 4, 3 (2005), 321–339.

[29] Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W. Black. 2020. Topological sort for sentence ordering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2783–2792. DOI: https://doi.org/10.18653/v1/2020.acl-main.248

[30] Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 149–159. DOI: https://doi.org/10.18653/v1/2020.acl-main.15

[31] Aili Shen and Timothy Baldwin. 2021. A simple yet effective method for sentence ordering. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 154–160.

[32] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[33] Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*. 803–810. Retrieved from http://aclweb.org/anthology/P06-2103.

[34] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems*. 5998–6008.

[36] Suzan Verberne. 2011. Retrieval-based question answering for machine reading evaluation. In *Proceedings of the CLEF (Notebook Papers/Labs/Workshop)*.

[37] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of the International Conference on Neural Information Processing Systems*. 2692–2700.

[38] Hongling Wang and Guodong Zhou. 2012. Toward a unified framework for standard and update multi-document summarization. *ACM Transactions on Asian Language Information Processing* 11, 2 (2012), 1–18.

[39] Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proceedings of the International Conference on Neural Information Processing Systems*. 1784–1794.

[40] Yongjing Yin, Fandong Meng, Jinsong Su, Yubin Ge, Lingeng Song, Jie Zhou, and Jiebo Luo. 2020. Enhancing pointer network for sentence ordering with pairwise ordering predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9482–9489.

[41] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of the 5th International Conference on Learning Representations*.

[42] Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics* 7 (2019), 91–105.

[43] Junnan Zhu, Lu Xiang, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2021. Graph-based multimodal ranking models for multimodal summarization. *Transactions on Asian and Low-Resource Language Information Processing* 20, 4 (2021), 1–21.

[44] Yutao Zhu, Jian-Yun Nie, Kun Zhou, Shengchao Liu, Yabo Ling, and Pan Du. 2021. BERT4SO: Neural sentence ordering by fine-tuning BERT. arXiv:2103.13584. Retrieved from https://arxiv.org/abs/2103.13584.

[45] Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural sentence ordering based on constraint graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 14656–14664.

Received 27 July 2021; revised 2 January 2022; accepted 4 July 2022