Deep Attentive Sentence Ordering Network

Baiyun Cui, Yingming Li*, Ming Chen, and Zhongfei Zhang
College of Information Science and Electronic Engineering,
Zhejiang University, China
baiyunc@yahoo.com,{yingming,funkyblack,zhongfei}@zju.edu.cn

Abstract

In this paper, we propose a novel deep attentive sentence ordering network (referred as ATTOrderNet) which integrates self-attention mechanism with LSTMs in the encoding of input sentences. It enables us to capture global dependencies among sentences regardless of their input order and obtains a reliable representation of the sentence set. With this representation, a pointer network is exploited to generate an ordered sequence. The proposed model is evaluated on Sentence Ordering and Order Discrimination tasks. The extensive experimental results demonstrate its effectiveness and superiority to the state-of-the-art methods.

1 Introduction

Modeling a coherent text is one of the key problems in natural language processing. A well-organized text with a logical structure is much easier for people to read and understand. Sentence ordering task (Barzilay and Lapata, 2008) has been proposed to cope with this problem. It aims to organize a set of sentences into a coherent text with a logically consistent order and has wide applications in natural language generation such as concept-to-text generation (Konstas and Lapata, 2012a,b, 2013), retrieval-based question answering (Yu et al., 2018; Verberne, 2011), and extractive multi-document summarization (Barzilay and Elhadad, 2002; Galanis et al., 2012; Nallapati et al., 2017), where the improper ordering of sentences would introduce ambiguity and degrade readability. An example of this task is shown in Table 1.

A set of sentences		An ordered text	
4	Finally, the parser is evaluated.	1	We develop a useful parser.
1	We develop a useful parser.	2	We first describe the older one.
3	Then we present our parser.	3	Then we present our parser.
2	We first describe the older one.	4	Finally, the parser is evaluated.

Table 1: An example to illustrate the sentence ordering task. The set of sentences is confusing in its current order. We aim to reorganize them with a more coherent order.

Traditional methods developed for this task employ handcrafted linguistic features to model the document structure such as Entity Grid (Barzilay and Lapata, 2008), Content Model (Barzilay and Lee, 2004), and Probabilistic Model (Lapata, 2003). However, manual feature engineering heavily relies on linguistic knowledge and also limits these systems to be domain specific. Inspired by the success of deep learning, data-driven approaches based on neural networks have been proposed including Pairwise Ranking Model (Chen et al., 2016) which learns the relative order of sentence pairs to predict the pairwise ordering of sentences, and Window network (Li and Hovy, 2014) sliding a window over the text to evaluate the coherence.

Recently, hierarchical RNN-based approaches (Gong et al., 2016; Logeswaran et al., 2018) have been proposed to deal with this task. Such methods exploit LSTMs based paragraph encoder to compute a context representation for the whole sequential sentences and then adopt a pointer network (Vinyals et al., 2015) as the decoder to predict their order. However, since LSTM works sequentially, paragraph encoder only based on LSTMs suffers from the incorrect input sentence order and has difficulty in capturing a logically reliable representation through the recurrent connections, which makes trouble for the decoder to find the correct order.

To overcome the above limitation, in this work, we develop a novel deep attentive sentence ordering network (referred as ATTOrderNet) by inte-

*Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4340–4349
Brussels, Belgium, October 31 - November 4, 2018.

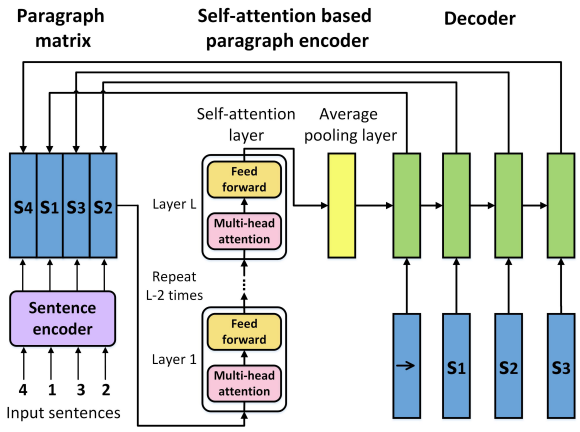

Figure 1: Architecture of deep attentive Sentence Ordering Network.

grating self-attention mechanism (Vaswani et al., 2017) with LSTMs to learn a relatively reliable paragraph representation for subsequent sentence ordering. In particular, the bidirectional LSTM is first adopted as a sentence encoder to map the input sentences to the corresponding distributed vectors, and then a self-attention based paragraph encoder is introduced to capture structural relationships across sentences and to obtain a hierarchical context representation of the entire set of sentences. Consequently, based on the learned paragraph vector, a pointer network is applied to perform sentence ordering by decoding an ordered sequence. Figure 1 shows the architecture of ATTOrderNet, where self-attention mechanism is introduced to capture the dependencies among sentences.

In contrast to the previous paragraph encoders with LSTMs, self-attention mechanism is less sensitive to the input order of sentence sequence and is effective in modeling the accurate relationships across sentences, which reduces the influence of the original order of the input sentences and perfectly meets the requirement of our task. Further, unlike Transformer (Vaswani et al., 2017), we do not add any positional encodings in our model to minimize the influence of the unclear order information.

Extensive evaluations are conducted on the sentence ordering task and order discrimination task to investigate the performances of ATTOrderNet. The experimental results on seven public sentence ordering datasets show the superior performances of the framework to the competing models. Meanwhile, the visualization of the attention layer in paragraph encoder is provided for better understanding of the effectivity of self-attention mechanism. Besides, in the Order Discrimination task, our model also achieves the state-of-the-art performance with remarkable improvements on two benchmark datasets.

2 Deep Attentive Sentence Ordering Network

In this section, we first formulate the sentence ordering problem and then describe the proposed model ATTOrderNet, which is based on the encoder-decoder architecture applying self-attention mechanism as paragraph encoder and a pointer network as the decoder. This combination effectively captures the intrinsic relations across a set of sentences with the desirable property of being invariant to the sentence order, which directly helps address the difficulty of this task.

2.1 Problem formulation

The sentence ordering task aims to order a set of sentences as a coherent text. Specifically, a set of n sentences with the order $\mathbf{o} = [o_1, o_2, \cdots, o_n]$ can be described as $\mathbf{s} = [s_{o_1}, s_{o_2}, \cdots, s_{o_n}]$. The goal is to find the correct order $\mathbf{o}^*$ for them, $\mathbf{o}^* = [o_1^*, o_2^*, \cdots, o_n^*]$, with which the whole sentences have the highest coherence probability:

$$P(\mathbf{o}^*|\mathbf{s}) > P(\mathbf{o}|\mathbf{s}), \forall \mathbf{o} \in \psi \quad (1)$$

where $\mathbf{o}$ indicates any order of these sentences and ψ denotes the set of all possible orders. For instance, in Table 1, the current order $\mathbf{o}$ is $[4, 1, 3, 2]$ and $\mathbf{o}^* = [1, 2, 3, 4]$ is the correct order for these sentences.

2.2 Intuition and Model architecture

Given a set of sentences, the existing hierarchical RNN-based models first transform each sentence into a distributed vector with a sentence encoder and then these sentence embeddings are fed to a LSTMs-based paragraph encoder. Consequently, based on the learned paragraph vector, a pointer network is exploited to decode the order of the input sentences. However, since LSTM works sequentially while the order of these sentences is unknown and quite possibly wrong in this problem, LSTMs-based paragraph encoder has difficulty in capturing a convincing representation through the recurrent connections, which influences the performance of sentence ordering.

In this paper, we use self-attention mechanism for paragraph encoder instead. In particular, we employ this mechanism without encoding any positional information of the sentences. Ignoring the current order of the sentences, self-attention based paragraph encoder perfectly meets the requirement of the sentence ordering task and learns a logically reliable representation of the whole paragraph by globally capturing the relationships across sentences. With this learned representation vector, a decoder is then designed to generate a coherent order assignment for the input sentences.

In the following, we elaborate on the main building blocks of our ATTOrderNet in details: a sentence encoder, a self-attention mechanism based paragraph encoder, and a decoder.

2.3 Sentence Encoder

For a sentence, we first apply word embedding matrix to translate the raw words in the sentence into distributional representations, and then adopt bidirectional LSTMs to learn a sentence-level representation for summarizing its high level semantic concepts.

Specifically, assume that a sentence s_{o_i} containing n_w raw words as $s_{o_i} = [w_1, \cdots, w_{n_w}]$, these words are transformed to dense vectors through a word embedding matrix $\mathbf{W}_e$: $\mathbf{x}_t = \mathbf{W}_e w_t$, $t \in [1, n_w]$. The sequence of vectors $[\mathbf{x}_1, \cdots, \mathbf{x}_{n_w}]$ is then fed into bidirectional LSTMs sequentially to compute a semantic representation of the sentence.

Long Short-term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) is capable of learning long-term dependencies and alleviating the problems of gradient vanishment and exploding. Here, we adopt bidirectional LSTMs (Bi-LSTMs) to take full advantages of additional backward information and enhance the memory capability. In particular, the Bi-LSTMs contain the forward LSTMs which process the sentence s_{o_i} from w_1 to w_{n_w} and backward LSTMs which read s_{o_i} in the reversed direction:

$$\begin{aligned} \overrightarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{c}}_t &= \text{LSTM}(\overrightarrow{\mathbf{h}}_{t-1}, \overrightarrow{\mathbf{c}}_{t-1}, \mathbf{x}_t) \\ \overleftarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{c}}_t &= \text{LSTM}(\overleftarrow{\mathbf{h}}_{t+1}, \overleftarrow{\mathbf{c}}_{t+1}, \mathbf{x}_t) \\ \mathbf{h}_t &= [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t] \end{aligned} \quad (2)$$

where $\mathbf{h}_t$ denotes the representation of position t by concatenating the forward hidden state $\overrightarrow{\mathbf{h}}_t$ and backward hidden state $\overleftarrow{\mathbf{h}}_t$ together. The output of the last hidden state of the Bi-LSTMs is taken to be the sentence representation vector as $\mathbf{s}_{o_i} = \mathbf{h}_{n_w}$, which incorporates the contextual information from both directions in the sentence.

So far, we have obtained a syntactic and semantic representation for a single sentence. In the following, a self-attention based paragraph encoder is proposed to obtain a high level representation for all given sentences by capturing sequential structures and logical relationships among them.

2.4 Paragraph Encoder

2.4.1 Self-attention mechanism

We start by introducing the scaled dot-product attention, which is the foundation of self-attention mechanism used in ATTOrderNet. Given a matrix of n query vectors $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d}$, and values $\mathbf{V} \in \mathbb{R}^{n \times d}$, the scaled dot-product attention computes the output matrix as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{\mathrm{d}}})\mathbf{V} \quad (3)$$

The multi-head attention with h parallel heads is employed, where each head is an independent scaled dot-product attention. The mathematical formulation is shown below:

$$\mathbf{M}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_{\mathrm{i}}^{\mathrm{Q}}, \mathbf{K}\mathbf{W}_{\mathrm{i}}^{\mathrm{K}}, \mathbf{V}\mathbf{W}_{\mathrm{i}}^{\mathrm{V}}) \quad (4)$$

$$\text{MH}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{M}_1, \cdots, \mathbf{M}_h)\mathbf{W} \quad (5)$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_a}$ with $d_a = d/h$ are the projection matrices for the i-th head and $\mathbf{W} \in \mathbb{R}^{hd_a \times d}$.

Self-attention (Vaswani et al., 2017; Tan et al., 2017; Shen et al., 2017) is a special case of attention mechanism that only requires a single sequence to compute its representation where queries, keys, and values are all from the same place.

2.4.2 Self-attention based Paragraph Encoder

The paragraph encoder is composed of multiple self-attention layers followed by an average pooling layer.

Sentence vectors encoded by the sentence encoder are first packed together into a paragraph matrix $\mathbf{S} = [\mathbf{s}_{o_1}, \mathbf{s}_{o_2}, \cdots, \mathbf{s}_{o_n}]$ as $\mathbf{E}^0$. This paragraph matrix $\mathbf{S} \in \mathbb{R}^{n \times d}$ is then fed forward to L self-attention layers, where each layer learns a representation $\mathbf{E}^{l+1} = \mathrm{U}(\mathbf{E}^l)$ by taking the output

from the previous layer l:

$$\mathrm{U}(\mathbf{E}^l) = \Phi(\mathrm{FN}(\mathrm{D}(\mathbf{E}^l)), \mathrm{D}(\mathbf{E}^l)) \quad (6)$$

$$\mathrm{D}(\mathbf{E}^l) = \Phi(\mathrm{MH}(\mathbf{E}^l, \mathbf{E}^l, \mathbf{E}^l), \mathbf{E}^l) \quad (7)$$

$$\Phi(\mathbf{v}, \mathbf{w}) = \mathrm{LayerNorm}(\mathbf{v} + \mathbf{w}) \quad (8)$$

$$\mathrm{FN}(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}\mathbf{W}_1^l + \mathbf{b}_1^l)\mathbf{W}_2^l + \mathbf{b}_2^l \quad (9)$$

where $\Phi(\cdot)$ performs layer normalization (Ba et al., 2016) on the residual output to preserve the auto-regressive property, and $\mathrm{FN}(\cdot)$ represents the fully connected feed-forward networks which consists of two linear layers with ReLU nonlinearity in the middle. $\mathbf{W}_1^l \in \mathbb{R}^{d \times d_f}$, $\mathbf{b}_1^l \in \mathbb{R}^{d_f}$, $\mathbf{W}_2^l \in \mathbb{R}^{d_f \times d}$, and $\mathbf{b}_2^l \in \mathbb{R}^d$ are trainable parameters. We set $d_f = 1024$ in all our experiments.

Self-attention mechanism adopted in the paragraph encoder directly relate sentences at different positions from the text by computing the attention score (relevance) between each pair of sentences. This allows each sentence to build links with all other sentences in the text, which enables the encoder to exploit latent dependency relationships among sentences without regarding to their input order. Then, attention mechanism uses weighted sum operation to establish a higher level representation for the entire sentence set. As we see, there is no order information used in the encoding process which prevents the model from being affected by the incorrect sentence order. Therefore, self-attention based paragraph encoder is efficient in modeling of dependencies while being invariant to the sentence order.

The final paragraph representation $\mathbf{v}$ is obtained in the average pooling layer by averaging the output matrix $\mathbf{E}^L \in \mathbb{R}^{n \times d}$ from the last self-attention layer: $\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}^{L_i}$, where n is the number of sentences and $\mathbf{e}^{L_i}$ denotes the i-th row in $\mathbf{E}^L$. This learned representation vector can be viewed as a hierarchical encoding of the entire set of sentences which will then be used as the input of the decoder to perform sentence ordering.

2.5 Decoder

The aim of decoder is to predict a consistent order for the input set of sentences.

Following the previous approaches (Gong et al., 2016; Logeswaran et al., 2018), the coherence probability of given sentences $\mathbf{s}$ with the order $\mathbf{o}$ is formalized as:

$$P(\mathbf{o}|\mathbf{s}) = \prod_{i=1}^{n} P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s}) \quad (10)$$

The higher the probability, the more coherent sentences assignment is.

To calculate $P(\mathbf{o}|\mathbf{s})$, we employ the pointer network architecture (Vinyals et al., 2015) as our decoder which consists of LSTMs cells (Equation 11-13). The LSTM takes the embedding of the previous sentence as the input to decoder step. During training, the correct order of sentences $\mathbf{o}^*$ is known, so the input sequence $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] = [\mathbf{s}_{o_1^*}, \mathbf{s}_{o_2^*}, \cdots, \mathbf{s}_{o_n^*}]$. For step i, the input to the decoder is $\mathbf{x}_{i-1} = \mathbf{s}_{o_{i-1}^*}$. At test time, the predicted sentence assignment $\mathbf{s}_{\hat{o}_{i-1}}$ is used instead.

The initial state of the decoder LSTM is initialized with the final paragraph vector from the encoder: $\mathbf{h}_0 = \mathbf{v}^T$. And the input at the first step in decoder $\mathbf{x}_0 \in \mathbb{R}^d$ is a vector of zeros. The mathematical formulation for the i-th step in decoder is as follows:

$$\mathbf{h}_i, \mathbf{c}_i = \mathrm{LSTM}(\mathbf{h}_{i-1}, \mathbf{c}_{i-1}, \mathbf{x}_{i-1}) \quad (11)$$

$$\mathbf{u}_j^i = \mathbf{g}^T \tanh(\mathbf{W}_1 \mathbf{s}_{o_j} + \mathbf{W}_2 \mathbf{h}_i) \quad (12)$$

$$P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s}) = \mathrm{softmax}(\mathbf{u}^i) \quad (13)$$

where $\mathbf{g} \in \mathbb{R}^d$, $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are learnable parameters and $j \in (1, \cdots, n)$. The softmax function normalizes the vector $\mathbf{u}^i \in \mathbb{R}^n$ to produce an output distribution over all input sentences. And $P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s})$ can be interpreted as the coherence probability for the current output sequence when s_{o_i} being the sentence choice at position i conditioned on the previous sentences assignment.

Order Prediction: The predicted order $\hat{\mathbf{o}} = [\hat{o}_1, \hat{o}_2, \cdots, \hat{o}_n]$ is the one with the highest coherence probability:

$$\hat{\mathbf{o}} = \underset{\mathbf{o}}{\mathrm{argmax}}\ P(\mathbf{o}|\mathbf{s}) \quad (14)$$

In this work, we use beam search strategy to find a sub optimal result.

2.6 Training

For each ordered document, we use one random permutation of sentences as the input sample at each epoch during the training and testing process. Assume that there are K documents in the training set. We define $(q_j, y_j)_{j=1}^K$, where y_j is in the correct order $\mathbf{o}^*$ of original document j and q_j denotes the set of sentences with a specific permutation of y_j. $P(y_j|q_j) = P(\mathbf{o}^*|\mathbf{s} = q_j)$ can be interpreted as the probability that sentences are assigned in the correct order when given sentences q_j.

Dataset	Length statistics			Data split			Vocabulary
	min	mean	max	train	valid	test	
Accident	6	11.5	19	100	-	100	4501
Earthquake	3	10.4	32	100	-	99	3022
NIPS abstract	2	6	15	2248	409	402	16721
AAN abstract	1	5	20	8569	962	2626	34485
NSF abstract	2	8.9	40	96070	10185	21580	334090
arXiv abstract	2	5.38	35	884912	110614	110615	64557
SIND caption	5	5	5	40155	4990	5055	30861

Table 2: Statistics of seven datasets used in our experiments.

We aim to train the overall model to maximize this probability by minimizing the loss function:

$$L = -\frac{1}{K}\sum_{j=1}^{K} \log P(y_j|q_j;\theta) + \frac{\lambda}{2}||\theta||_2^2 \quad (15)$$

where θ represents all trainable parameters in the networks and λ is a regularization parameter.

3 Experiments

3.1 Datasets

Accident, Earthquake: two datasets obtained from (Barzilay and Lapata, 2008). The first one is a collection of aviation accident reports from the National Transportation Safety Board and the second one comprises Associated Press articles related to earthquake. Since the original datasets do not provide validation samples, we follow the setup in (Louis and Nenkova, 2012; Li and Hovy, 2014) and use 10-fold cross validation on the training data.

NIPS abstract, AAN abstract, NSF abstract: these three datasets are from (Logeswaran et al., 2018) containing abstracts from NIPS papers, ACL papers, and the NSF Research Award Abstracts dataset respectively.

arXiv abstract, SIND caption: we further consider two datasets used in (Gong et al., 2016). The former consists of abstracts from papers on arXiv website (Chen et al., 2016) and the other contains captions from SIND dataset (Huang et al., 2016).

Further statistics about seven datasets are illustrated in Table 2.

3.2 Training setup

We use pre-trained 100 dimensional GloVe word embeddings (Pennington et al., 2014). And all the out-of-vocabulary words are replaced with <UNK>, whose embeddings are updated during training process. The nltk sentence tokenizer is used for word tokenization.[1] Parameter optimization is performed using stochastic gradient descent. We adopt Adadelta (Zeiler, 2012) as the optimizer with $\epsilon = 10^6$ and $\rho = 0.95$. The learning rate is initialized to 1.0, the batch size is 16, and the beam size is set to 64. The hidden layer size of LSTMs in sentence encoder is 256, and is 512 in the decoder. The number of attention layers in the paragraph encoder is 6 for AAN abstract, 4 for NSF abstract and arXiv abstract, and 2 for the rest of datasets. We employ 8 parallel heads throughout all self-attention layers and use L2 weight decay on the trainable variables with regularization parameter $\lambda = 10^{-5}$. The model is implemented with TensorFlow[2]. Hyperparameters are chosen using the validation set.

3.3 Sentence Ordering

We first evaluate our model on the sentence ordering task, as proposed by Barzilay and Lapata (2008). Given a set of permuted sentences, our goal is to return the original order for them which is considered to be the most coherent.

3.3.1 Baselines

We compare ATTOrderNet against a random baseline and all the competing models. These baseline methods can be categorized into three classes and results are reported in (Soricut and Marcu, 2006; Gong et al., 2016; Logeswaran et al., 2018).

(1) Traditional approaches: Probabilistic Model (Lapata, 2003); Content Model (Barzilay and Lee, 2004); Utility-Trained model (Soricut and Marcu, 2006); Entity Grid (Barzilay and Lapata, 2008). These four methods employ handcrafted features in modeling the document structure.

(2) Data-driven methods: Window network (Li and Hovy, 2014); Seq2seq (Li and Jurafsky,

[1] NLTK implementation: http://www.nltk.org/

[2] https://www.tensorflow.org/

Models	Accident	Earthquake	NIPS abstract		AAN abstract		NSF abstract		arXiv abstract		SIND caption	
	τ	τ	Acc	τ	Acc	τ	Acc	τ	PMR	τ	PMR	τ
Random	0	0	15.59	0	19.36	0	9.46	0	8.07	0	6.05	0
Probabilistic Model	0.07	0.48	-	-	-	-	-	-	-	-	-	-
Content Model	0.44	0.81	-	-	-	-	-	-	-	-	-	-
Utility-Train Model	0.50	0.47	-	-	-	-	-	-	-	-	-	-
Entity Grid	0.12	0.19	20.10	0.09	21.82	0.10	-	-	-	-	-	-
Seq2seq	-	-	27.18	0.27	36.62	0.40	13.68	0.10	-	-	-	-
Window network	-	-	41.76	0.59	50.87	0.65	18.67	0.28	-	-	-	-
Pairwise Ranking Model	-	-	-	-	-	-	-	-	33.43	0.66	-	-
RNN Decoder	-	-	48.22	0.67	52.06	0.66	25.79	0.48	-	-	-	-
Varient-LSTM+PtrNet	-	-	51.55	**0.72**	58.06	**0.73**	28.33	0.51	-	-	-	-
CNN+PtrNet	0.58	0.84	48.64	0.66	58.21	0.69	33.22	0.52	39.28	0.71	12.32	0.48
LSTM+PtrNet	0.58	0.85	50.87	0.67	58.20	0.69	32.45	0.52	40.44	0.72	12.34	0.48
ATTOrderNet (ATT)	0.59	0.87	52.17	0.71	60.95	0.70	37.07	0.52	41.03	0.72	12.77	0.48
ATTOrderNet (CNN)	0.61	0.89	54.69	0.72	62.97	0.72	37.59	0.54	42.09	0.73	13.95	0.49
ATTOrderNet	**0.64**	**0.92**	**56.09**	**0.72**	**63.24**	**0.73**	**37.72**	**0.55**	**42.19**	**0.73**	**14.01**	**0.49**

Table 3: Experimental results for different methods on the Sentence Ordering task.

2017); Pairwise Ranking Model (Chen et al., 2016). These three approaches capture the local coherence of text based on the neural networks.
(3) Hierarchical RNN-based models: Varient-LSTM+PtrNet, RNN Decoder (Logeswaran et al., 2018); CNN+PtrNet, LSTM+PtrNet (Gong et al., 2016). These architectures adopt RNN based approaches to obtain the representation for the input set of sentences and employ the pointer network as the decoder to predict order. The main difference between ATTOrderNet and these models lies in the design of paragraph encoder.

For thorough comparison, besides the models proposed in the existing literature, we further implement two variants of ATTOrderNet.

ATTOrderNet (ATT): The sentence encoder in this model is also entirely based on self-attention mechanism with 4 self-attention layers and 5 heads. Different from the paragraph encoder, the positional encoding method proposed by Vaswani et al. (2017) is applied here to encode temporal information of each input word.

ATTOrderNet (CNN): This model employs convolutional neural networks to model sentences. In experiment, the number of feature maps is set to 512 and the width of convolution filter is 4.

3.3.2 Evaluation Metrics

To provide assessments on the quality of the orderings we predict in this task, we use the following three metrics:

Kendall's tau (τ): Kendall's tau is one of the most frequently used metrics for the automatic evaluation of document coherence (Lapata, 2003; Logeswaran et al., 2018; Li and Jurafsky, 2017). It could be formalized as: $\tau = 1 - 2\times$ (number of inversions) $/\binom{n}{2}$, where n is the length of the sequence and the number of inversions denotes the number of pairs in the predicted sequence with incorrect relative order. This metric ranges from -1 (the worst) to 1 (the best).

Accuracy (Acc): We follow (Logeswaran et al., 2018) in employing Accuracy to measure how often the absolute position of a sentence was correctly predicted. Compared to τ, it penalizes correctly predicted subsequences that are shifted.

Perfect Match Ratio (PMR): Perfect match ratio (Gong et al., 2016) is the most stringent measurement in this task. It calculates the radio of exactly matching orders: PMR=$\frac{1}{K}\sum_{i=1}^{K}\mathbf{1}(\widehat{\mathbf{o}}^i = \mathbf{o}^{i*})$, where $\widehat{\mathbf{o}}^i$ and $\mathbf{o}^{i*}$ are predicted and correct orders of the i-th text respectively.

3.3.3 Results

The experimental results on all datasets are reported in Table 3. Results show that ATTOrderNet gives the best performance across most datasets and under most evaluation measurements.

The improvement is regardless of data sizes. In particular, for smaller datasets such as Accident and Earthquake datasets, ATTOrderNet outperforms the previous best baseline methods by 6% and 7% tau score respectively. As for medium size datasets including NIPS abstract and AAN

abstract, ATTOrderNet shows absolute improvements of 4.54% and 5.03% accuracy score over the previous state-of-the-art. Such finding is consistent across larger datasets. ATTOrderNet outperforms the previous state-of-the-art systems by 4.50% accuracy score with 3% tau score on NSF abstract, 1.75% PMR score with 1% tau score on arXiv abstract, and 1.67% PMR score with 1% tau score on SIND caption. Interestingly, ATTOrderNet reaches 42.19% PMR score on arXiv abstract, which means that more than 2/5 texts in the test set can be ordered exactly right. This performance clearly demonstrates the adaptability and flexibility of the proposed model.

As shown in Table 3, ATTOrderNet performs much better than data-driven methods by a significant margin on all corresponding datasets. It proves the importance of exploiting the context by self-attention mechanism as these competing models only consider the local coherence in the text. Among the traditional ordering approaches, Content Model (Barzilay and Lee, 2004) representing topics as states and capturing possible orderings for global coherence performs better than other methods with the tau score of 0.81 on Earthquake dataset, which also demonstrates that global context is important to sentence ordering. However, Content Model requires manual feature engineering that costs great human efforts. In contrast, the self-attention mechanism used in ATTOrderNet directly captures the global dependences for the whole text while requiring no linguistic knowledge anymore and enables ATTOrderNet to further improve tau score to 0.92 on the same dataset.

In addition, hierarchical RNN-based models capture the global coherence among sentences with LSTMs and outperform the traditional methods and data-driven approaches in most cases. However, these models still suffer from the permutation of sentences within the document since LSTM works sequentially. ATTOrderNet achieves superior performances to them by adopting the self-attention mechanism to reduce the influence of the permutation of sentences.

Further, ATTOrderNet (CNN) has better performances than ATTOrderNet (ATT) on most of the datasets. We conjecture that this is due to the limitation of data size. Since ATTOrderNet (ATT) applies self-attention mechanism in both sentence and paragraph encoders requiring more data to train the model, however the size of the datasets used in this task is smaller than those in other tasks such as document classification (Yang et al., 2016). Given larger datasets in the future, we believe ATTOrderNet (ATT) would perform much better. Among three sentence encoders, ATTOrderNet presents a superior performance across the board. This indicates that LSTM is more efficient in learning semantic representation for sentence level in this task. ATTOrderNet becomes more competitive through combining both advantages of LSTMs and self-attention mechanism.

Models	arXiv abstract		SIND caption	
	head	tail	head	tail
Random	23.06	23.16	22.78	22.56
Pairwise Ranking Model	84.85	62.37	-	-
CNN+PtrNet	89.43	65.36	73.53	53.26
LSTM+PtrNet	90.47	66.49	74.66	53.30
ATTOrderNet (ATT)	89.68	65.75	75.88	54.30
ATTOrderNet (CNN)	90.86	67.85	75.95	54.37
ATTOrderNet	**91.00**	**68.08**	**76.00**	**54.42**

Table 4: The performance of correctly predicting the first and the last sentences on arXiv abstract and SIND caption datasets.

Since the first and the last sentences of the text are more special to discern (Chen et al., 2016; Gong et al., 2016), we also evaluate the ratio of correctly predicting the first and the last sentences. Table 4 summarizes our performances on arXiv abstract and SIND caption. As we see, all models show fair well in predicting the first sentence, and the prediction accuracy declines for the last one. It is observed that ATTOrderNet still achieves a boost in predicting two positions compared to the previous state-of-the-art system on both datasets.

3.3.4 Visualization of attention

The aim of this section is to visualize the relationship between sentences captured by self-attention mechanism and understand how it helps perform the sentence ordering task. A technique for visualizing attention mechanism in neural networks is proposed by Vaswani et al. (2017) [3]. Inspired by this work, we select a text from AAN abstract dataset to visualize the hierarchical attention layer from the paragraph encoder of ATTOrderNet in Figure 2. Different from visualizing the dependencies of one word with the other words in the sentence (Vaswani et al., 2017), our visualization

[3]https://github.com/tensorflow/tensor2tensor

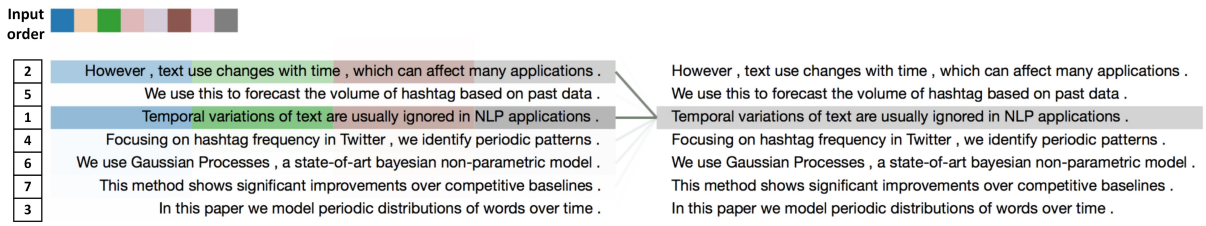

Figure 2: An example of the attention mechanism describing sentence dependencies in the encoder self-attention in layer 5 of 6 for the text from AAN abstract. Four attention heads attend to a dependency of the selected sentence *"Temporal variations of text are ..."*, providing important clues for the correct order prediction. Attention here shown is only for this sentence. Different color indicate different heads. Best viewed in color.

shows the dependencies between each sentence and all other sentences in the text.

For the example in Figure 2, the left text is the input sample to our model which contains a set of permuted sentences with the correct order besides it. The right side is a copy of the input text, which is presented for showing the relevance between each pair of sentences more clearly. The line with grey color on the right text is an example sentence chosen to visualize the attention weights with other sentences. On the top of the text are 8 colored squares representing 8 different attention heads used in the paragraph encoder. Colored columns on the left text show the performance of their corresponding heads. The darkness of the color in column denotes the normalized distribution of the attention weight for the example sentence in the head. Sentences in darker shades show more attention weight which reflects stronger links they have with the example sentence.

In particular, Figure 2 shows the attention distribution for the first sentence in the original document. We present the weight distribution in four heads as an instance. It is interesting to see that all of them showing significant higher attention weights on the true second sentence *"However, text use changes ..."* than other sentences in the text. This indicates that these heads are able to learn the latent dependency relationships from sentences and can successfully distinguish which one is the true next following among all sentence candidates. These heads build much stronger links between this sentence with the chosen one in order to keep structural information for higher level representation, such as paragraph representation.

3.4 Order Discrimination

In this section, we assess ATTOrderNet on another common evaluation task which is usually adopted in the existing literature: order discrimination task.

Order discrimination (Barzilay and Lapata, 2008; Elsner and Charniak, 2011, 2008) aims to compare a document to a randomly permuted version of it. Models are evaluated with Pairwise Accuracy: the ratio of correctly identifying the original document with higher coherence probability (defined in Equation 10) than the probability of its permutation.

Among seven datasets mentioned above, we use two of them to assess the performance of ATTOrderNet on the order discrimination task: Accident and Earthquake datasets. These two have been widely used for this task in the previous literature (Li and Hovy, 2014; Logeswaran et al., 2018). This gives us the convenience of directly comparing the result of the proposed model against the reported results. Following the setup in (Barzilay and Lapata, 2008), a maximum of 20 random permutations were generated for each training and testing article to create the pairwise data. There are 1986 and 1956 test pairs in Accident and Earthquake datasets respectively.

3.4.1 Baselines

To demonstrate that ATTOrderNet truly improves the order discrimination performance, we compare ATTOrderNet with the following representative models: Graph from (Guinaudeau and Strube, 2013), HMM and HMM+Entity from (Louis and Nenkova, 2012), Entity Grid from (Barzilay and Lapata, 2008), Recurrent and Recursive from (Li and Hovy, 2014), Discriminative model from (Li and Jurafsky, 2017), Varient-LSTM+PtrNet from (Logeswaran et al., 2018), CNN+PtrNet and LSTM+PtrNet from (Gong et al., 2016). The results of the last two methods were obtained by training their models on two datasets.

Models	Accident	Earthquake
Random	50.0	50.0
Graph	84.6	63.5
HMM+Entity	84.2	91.1
HMM	82.2	93.8
Entity Grid	90.4	87.2
Recurrent	84.0	95.1
Recursive	86.4	97.6
Discriminative Model	93.0	99.2
Varient-LSTM+PtrNet	94.4	99.7
CNN+PtrNet	93.5	99.4
LSTM+PtrNet	93.7	99.5
ATTOrderNet (ATT)	95.4	99.6
ATTOrderNet (CNN)	95.8	99.7
ATTOrderNet	**96.2**	**99.8**

Table 5: Experimental results of Pairwise Accuracy for different approaches on two datasets in the Order Discrimination task.

3.4.2 Results

Table 5 reports the results of ATTOrderNet and currently competing architectures in this evaluation task. ATTOrderNet also achieves the state-of-the-art performance, showing a remarkable advancement of about 1.8% gain on Accident dataset and further improving the pairwise accuracy to 99.8 on Earthquake dataset.

LSTM+PtrNet and CNN+ PtrNet (Gong et al., 2016) fall short of Varient-LSTM+PtrNet (Logeswaran et al., 2018) in performance. This could also be blamed for their paragraph encoder. Documents in both datasets are much longer than those in others, which brings more trouble for LSTMs in paragraph encoder to build logical representations. Compared to the result in the sentence ordering task, Entity Grid (Barzilay and Lapata, 2008) achieves a good performance in this task and even outperforms Recurrent neural networks and Recursive neural networks (Li and Hovy, 2014) on Accident dataset. However, Entity Grid requires hand-engineered features and heavily relies on linguistic knowledge which restrain the model to be adapted to other tasks.

4 Conclusion

In this paper, we develop a novel deep attentive sentence ordering model (referred as ATTOrderNet) integrating self-attention mechanism with LSTMs. It enables us to directly capture logical relationships among sentences regardless of their input order and obtain a reliable representation of the sentence set. With this representation, a pointer network is applied to generate an ordered sequence. ATTOrderNet is evaluated on Sentence Ordering and Order Discrimination tasks. The experimental results demonstrate its effectiveness and show promising improvements over existing models across most datasets.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61702448, 61672456) and the Fundamental Research Funds for the Central Universities (No. 2017QNA5008, 2017FZA5007). We thank all reviewers for their valuable comments.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Regina Barzilay and Noemie Elhadad. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, pages 113–120.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *ACL*, pages 41–44.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *ACL*, pages 125–129.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. *COLING*, pages 911–926.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *ACL*, volume 1, pages 93–103.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *NAACL*, pages 1233–1239.

Ioannis Konstas and Mirella Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *ACL*, pages 369–378. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2012b. Unsupervised concept-to-text generation with hypergraphs. In *NAACL*, pages 752–761. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *EMNLP*, pages 1503–1514.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *ACL*, pages 545–552. Association for Computational Linguistics.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *EMNLP*, pages 198–209.

Lajanugen Logeswaran, Honglak Lee, and Dragomir R. Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI*.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *EMNLP-CONLL*, pages 1157–1168. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *COLING/ACL*, pages 803–810. Association for Computational Linguistics.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586*.

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 *NIPS*, pages 6000–6010.

Suzan Verberne. 2011. Retrieval-based question answering for machine reading evaluation. In *CLEF (Notebook Papers/Labs/Workshop)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489.

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. *arXiv preprint arXiv:1804.09541*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.