Topic-Guided Coherence Modeling for Sentence Ordering by Preserving Global and Local Information

Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo, Kyong-Ho Lee
Department of Computer Science, Yonsei University
Seoul, South Korea
{bkoh, smseo, chshin, ejjo}@icl.yonsei.ac.kr
khlee89@yonsei.ac.kr

Abstract

We propose a novel ***topic-guided coherence modeling (TGCM)*** for sentence ordering. Our attention based pointer decoder directly utilize sentence vectors in a permutation-invariant manner, without being compressed into a single fixed-length vector as the paragraph representation. Thus, TGCM can improve global dependencies among sentences and preserve relatively informative paragraph-level semantics. Moreover, to predict the next sentence, we capture topic-enhanced sentence-pair interactions between the current predicted sentence and each next-sentence candidate. With the coherent topical context matching, we promote local dependencies that help identify the tight semantic connections for sentence ordering. The experimental results show that TGCM outperforms state-of-the-art models from various perspectives.

1 Introduction

Modeling the coherence among sentences to compute their gold order is one of the fundamental tasks in Natural Language Processing (NLP) with many applications such as document modeling (Narayan et al., 2018a), extractive document summarization (Jadhav and Rajan, 2018; Nallapati et al., 2017), question answering (Yu et al., 2018; Liu et al., 2017), conversational analysis (Zeng et al., 2018), automated text generation (Guo et al., 2018), and image captioning (Anderson et al., 2018). The coherence helps readers to improve reading comprehension and better understand the intent of a document. Sentence ordering is a set-to-sequence problem, which aims to identify the correct order of a sentence set. To do this, various studies on sentence ordering typically combine the coherent features extracted from sentences.

In recent years, most of the traditional approaches to sentence ordering are designed based on a pairwise strategy (Chen et al., 2016; Agrawal et al., 2016; Li and Jurafsky, 2016). The sentence pair ordering (SPO) models determine the relative order within a sentence pair via neural networks based semantic matching, which computes the relevance between the two sentence vectors. However, such models cause combinatorial optimization problems because search algorithms (e.g., beam search) are necessary to find the most optimal permutation. Since the SPO models only focus on the sentence-pair interactions (i.e., *local dependency*), they have trouble in capturing the interactions among three or more input sentences (i.e., *global dependency*) in an entire paragraph.

More recently, state-of-the-art models aim to put randomly sorted sentences into a coherent paragraph with the correct order so that the whole sentences have the highest coherence probability (Vinyals et al., 2015a; Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018). Unlike the SPO models, these models can perform the sentence set ordering (SSO) task for an entire paragraph based on the pointer network (Vinyals et al., 2015b). Hierarchical RNN networks consisting of sentence and paragraph encoders take unordered sentences as input. They build a paragraph-level vector representation, which represents a semantic summary of the input sentences. Then, a pointer network based decoder fetches the paragraph vector and iteratively outputs sentences in the correct order. At this time, since the output sentences are taken from the input sentences, this can solve the combinatorial optimization problems, which the SPO models suffer from. Also, the SSO models can capture the global dependency among input sentences via the paragraph vector.

Despite the successes of the SSO models, there still exist severe limitations as shown in Figure 1.

- **[L1]** The conventional pointer decoders for the

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2273–2283,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

SSO task only utilize the last hidden state at the end of paragraph encoders. The encoder always pushes sentence information into a single fixed-length paragraph vector, no matter how many sentences are in the paragraph. Thus, they may struggle with the bottleneck problem where important information between the encoder and the decoder is shrunk. Especially, the more the number of sentences, the more difficult to preserve the **global information** of the paragraph.

- **[L2]** The attention layer repeatedly decides the next sentence from sentence-pair interactions between the current predicted sentence and each paragraph-independent candidate $\{s_i\}_1^5$. Thus, they do not elaborately utilize the context of previously predicted sentences $\{s_1, s_2, s_3\}$ and the context of s_4 conditioned on the context of the paragraph (i.e., the shuffled sentences $\{s_2, s_4, s_5, s_1, s_3\}$), where the coherent contexts indicate the **local information** that helps determine the tight s_3-s_4 interactions.

To address the above limitations, this paper proposes a novel **T**opic-**G**uided **C**oherence **M**odeling **(TGCM)** for sentence ordering by capturing local and global dependencies among sentences. Specifically, TGCM is composed of two major components: topic-sensitive sentence encoder and attentive pointer decoder.

To complement the structural limitations of existing RNN-based pointer decoders, which sequentially decode one paragraph vector, our attentive pointer decoder relies entirely on the attention mechanism (Vaswani et al., 2017) without any recurrent units or convolutions. Since our decoder directly receives the set of sentence vectors regardless of their input order via attention, our encoder is free from the constraint that all sentences must be compressed into a single fixed-length vector as a paragraph representation. As a result, the preservation of **global information** improves the global dependencies among sentences and provides a relatively informative paragraph-level semantics, which can deal with the bottleneck problem. Moreover, TGCM can better preserve the fine-grained word/sentence-level semantics from the encoder to the decoder.

Instead of paragraph-independent next-sentence candidates, the topic-sensitive sentence encoder enriches each candidate sentence with its topical context conditioned on the paragraph context via

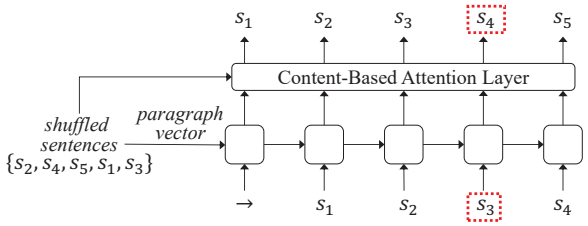

Figure 1: Pointer decoder for sentence ordering. Based on the paragraph vector encoded from the shuffled sentences, the attention layer selects the next sentence s_4 following s_3 from sentence-pair interactions between the previously predicted s_3 and each candidate $\{s_i\}_1^5$.

topic modeling based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003). For each position of the decoder, TGCM also incorporates the topical context flow of previously predicted sentence sequence into the current predicted sentence. It then predicts the next sentence from topic-enhanced sentence-pair interactions with the coherent topical context as local information. Consequently, the preservation of **local information** promotes the local dependencies between the current predicted sentence and each next-sentence candidate, resulting in identifying the strongest semantic connection for ordering a shuffled paragraph.

2 Related Work

The key idea of sentence modeling is to embed each sentence into continuous vector spaces by combining word vectors with recurrent neural networks (RNNs) (Mikolov et al., 2010) to capture long-term dependencies between words and convolution neural networks (CNNs) (Kim, 2014) to capture important local invariance context.

Sentence Pair Modeling: To identify the relationship between given two sentences, sentence pair modeling learns a semantic matching function based on neural networks, which extracts their task-specific features. As a famous example, question answering measures relevance between question-answer pairs with a matching function and ranks candidate answer sentences (Severyn and Moschitti, 2016; Liu and Huang, 2016). Recent models further measure the semantic similarity elaborately by using multi-attention mechanism (Tan et al., 2018; Tay et al., 2018) and a word-level similarity matrix (Pang et al., 2016; Shen et al., 2017).

Another example is the SPO task. Chen et al. (2016) compute a similarity score for each sen-

tence pair independently. After pairwisely scoring, they use a beam search algorithm to find an optimal order for input sentences. Agrawal et al. (2016) develop a pairwise scoring model to organize unordered image-caption pairs. However, such models have the combinatorial optimization problem of finding the most optimal permutation. Moreover, they cannot capture the paragraph-level contextual information, which indicates the global dependency of an entire paragraph.

Sentence Set Modeling: To solve the above issues, the pointer network (Vinyals et al., 2015b) is proposed based on the sequence-to-sequence model (Sutskever et al., 2014). The pointer network consists of encoding input tokens to a summary vector, decoding next token vectors iteratively via content-based attention over input token vectors, and producing the output token sequence from the output token vectors. Inspired by the pointer network, state-of-the-art SSO models (Vinyals et al., 2015a; Gong et al., 2016; Logeswaran et al., 2018) usually employ hierarchical RNN-based encoders to produce a paragraph vector from unordered sentences. Then, the pointer network based decoders predict the correct sentence sequence. However, the paragraph vector depends on the permutation of input sentences.

To address the issue, ATTOrderNet (Cui et al., 2018) employs self-attention at the encoder to capture global dependencies regardless of an input sentence order. However, similar to traditional models, they also compress sentence vectors into a single fixed-length vector via average pooling. While Our TGCM is also permutation-agnostic to input sentences, we feed the sentence vector set directly into the attentive pointer decoder and capture the coherent topical context in sentence-pair interactions via topic modeling. Thus, unlike ATTOrderNet, TGCM can simultaneously improve both local and global dependencies.

Topic-Aware Sentence Modeling: The commonly used topic model is based on LDA, which extracts the latent topic vectors (i.e., topic distributions) of words, sentences, and paragraphs from a training corpus. Given sentences in a paragraph, their topic latent vectors help in capturing global topical context for the paragraph and local topical context for each sentence. Therefore, topic modeling is used in various research fields requiring sentence modeling (Narayan et al., 2018b; Dieng et al., 2016; Gong et al., 2018). LTMF (Jin et al., 2018) is a context-aware recommender system, which combines LSTMs and topic modeling before applying matrix factorization. They extract the global context information related to words in a user review via topic distributions.

3 The Proposed Model

In this section, we present TGCM as a novel topic-guided coherence modeling for SSO. TGCM can address the above-mentioned limitations **[L1]** and **[L2]** simultaneously. We first build a topic-distribution generating function via topic modeling. Then, we describe two major components: topic-sensitive sentence encoder and attentive pointer decoder. The encoder leverages the topic distributions of a paragraph and its sentences. Then, the decoder directly utilizes them in a permutation-agnostic manner and determines the correct order of randomly sorted sentences.

3.1 Problem Definition

The primary goal of sentence set ordering is to put an unordered set of sentences into a coherent paragraph in the correct order. Specifically, given a paragraph p, the correct sentence sequence and its order are denoted by $S_p = [s_1, s_2, ..., s_{|p|}]$ and $O_p = [o_{s_1}, o_{s_2}, ..., o_{s_{|p|}}]$, respectively, where $|p|$ denotes the number of sentences in p.

Given a shuffled sentence sequence, our TGCM outputs a sentence sequence $\hat{S}_p$ whose order is denoted by $\hat{O}_p$. The objective of coherence modeling is to make the coherence probability for the predicted order $\hat{O}_p$ approximate to the coherence probability for the correct order O_p as follows:

$$P(O_p|p) \approx P(\hat{O}_p|p), \tag{1}$$

where $P(O_p|p)$ and $P(\hat{O}_p|p)$ denote the coherence probabilities for O_p and $\hat{O}_p$, respectively.

3.2 Topic Latent Vectors

Given the number of hidden topics, which is denoted by d_t, the preprocessing of TGCM trains a topic model on a given corpus as shown in Figure 2. The topic model builds a generating function $topicDistribution(\cdot)$ based on the probability distribution over topics for each word. At test time, $topicDistribution(\cdot)$ infers the d_t-dimensional topic latent vectors $\mathbf{t}_{doc}$ (i.e., topic distribution) of a new, unseen document doc such as sentences and paragraphs.

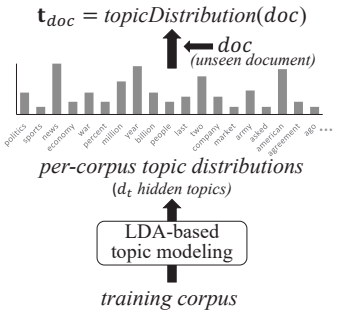

Figure 2: Topic modeling on a corpus as preprocessing of TGCM. $topicDistribution(\cdot)$ returns LDA-based features, a topic latent vector $\mathbf{t}_{doc}$ of a given document.

To do this, we utilize LDA (Blei et al., 2003), which is the most simple and popular algorithm for topic modeling. The topic latent vectors of sentences and paragraphs indicate the sentence-level (local) and the paragraph-level (global) topical context via hidden topics, respectively.

3.3 Topic-Sensitive Sentence Encoder

Attention-Based bi-LSTM Layer. We leverage an extended version of LSTMs (Hochreiter and Schmidhuber, 1997) as our base model in Figure 3, which overcomes the gradient vanishing of existing RNNs. Particularly, attention-based bidirectional LSTMs (Att-BLSTMs) (Zhou et al., 2016) are mostly used in sentence modeling.

In this layer, we aim to produce the sentence vector $\mathbf{s}$ from a given sentence s consisting of a word sequence $[w_1, w_2, ..., w_n]$. Each word w_i is encoded into a word embedding $\mathbf{w}_i \in \mathbb{R}^{d_w}$. Att-BLSTMs consist of two sub-networks with forward and backward LSTMs, which take the word embeddings $[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n]$ and output sequences of forward and backward hidden states $[\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_n]$ and $[\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_n]$, respectively as follows:

$$\begin{aligned} \overrightarrow{h}_t &= \overrightarrow{LSTM}(M_t, \overrightarrow{h}_{t-1}), \\ \overleftarrow{h}_t &= \overleftarrow{LSTM}(M_t, \overleftarrow{h}_{t+1}), \end{aligned} \tag{2}$$

where M_t is a set of learnable parameters. Then, we combine the forward and backward hidden states by an element-wise sum as follows:

$$\mathbf{h}_i = [\overrightarrow{h}_i \oplus \overleftarrow{h}_i]. \tag{3}$$

Thus, the hidden state vectors $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n] \in \mathbb{R}^{d_w \times n}$ are obtained from previous LSTM layers and fed into an attention layer as follows:

$$\begin{aligned} \alpha &= softmax(\mathbf{m}_1^T tanh([\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n])), \\ \mathbf{s} &= [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]\alpha^T, \end{aligned} \tag{4}$$

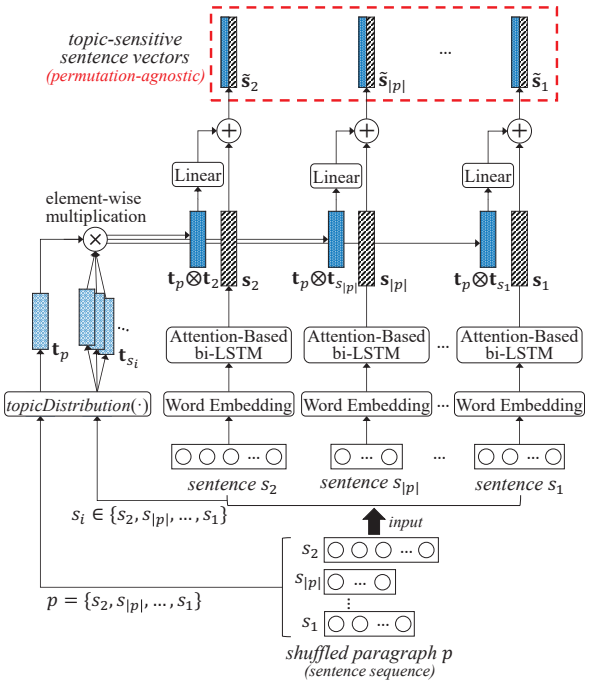

Figure 3: Architecture of topic-sensitive sentence encoder. The encoder takes a shuffled paragraph consisting of $\{s_2, s_{|p|}, ..., s_1\}$ and outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, ..., \tilde{\mathbf{s}}_1\}$ from their vectors and topic latent vectors of the paragraph and sentences. The outputs are directly utilized in a decoder without being compressed into a vector.

where $\mathbf{m}_1 \in \mathbb{R}^{d_w}$ and $\alpha \in \mathbb{R}^n$ denote a weight vector and resulting attention weights, respectively. $\mathbf{s} \in \mathbb{R}^{d_w}$ is the final sentence vector of s, which is computed by a weighted sum between the hidden state vectors and the attention weights.

Topic-Sensitive Sentence Vectors. As shown in Figure 3, our topic-sensitive sentence encoder takes a shuffled paragraph p as input, which consists of unordered sentences $\{s_i\}_1^{|p|}$. We create sentence vectors $\mathbf{s}_i$ for all sentences s_i with the attention-based bi-LSTMs. In addition, topic latent vectors $\mathbf{t}_p \in \mathbb{R}^{d_t}$ and $\mathbf{t}_{s_i} \in \mathbb{R}^{d_t}$ are generated by $topicDistribution(\cdot)$ for p and s_i, where d_t denotes the number of latent topics.

During the encoding of TGCM, we multiply $\mathbf{t}_{s_i}$ and $\mathbf{t}_p$ for each sentence s_i to weight the coherent latent topics that appear simultaneously in both the sentence and the paragraph. The topic latent vector $\mathbf{t}_{s_i}$ captures how topical a sentence s_i is in itself (local context), whereas the topic latent vector $\mathbf{t}_p$ represents the overall theme of a paragraph p (global context). Thus, the encoder can extract the paragraph-level context relating to each sentence by enriching the context of the sentence with its

topical relevance to the paragraph as follows:

$$(\mathbf{t}_p \otimes \mathbf{t}_{s_i}) \in \mathbb{R}^{d_t}; \quad s_i \in \{s_1, s_2, ..., s_{|p|}\}, \quad (5)$$

where the operation $\otimes$ indicates an element-wise multiplication. To combine $\mathbf{t}_p \otimes \mathbf{t}_{s_i}$ with the sentence vectors $\mathbf{s}_i$, we apply two linear transformations with a ReLU activation to each $\mathbf{t}_p \otimes \mathbf{t}_{s_i}$ separately. Then, the transformed vectors are added with the corresponding sentence vectors as follows:

$$\tilde{\mathbf{s}}_i = max(0, (\mathbf{t}_p \otimes \mathbf{t}_{s_i})^\intercal W_1 + b_1) W_2 + b_2 + \mathbf{s}_i, \quad (6)$$

where $\tilde{\mathbf{s}}_i \in \mathbb{R}^{d_w}$ denotes a topic-sensitive sentence vector. Thus, the global topical context relating to the local topical context of each sentence is incorporated into the corresponding sentence vector. This helps our pointer decoder for next-sentence predictions to guide the semantic connection between a predicted sentence and its next sentence by considering coherent topical context flow.

3.4 Attentive Pointer Decoder

As shown in Figure 4, we build the attentive pointer decoder based on the decoder of Transformer (Zhou et al., 2016) and the pointer network (Narayan et al., 2018b). The pointer network determines next token vectors iteratively via content-based attention over input token vectors. For coherent topical context matching, our decoder is mainly composed of a stack of n attention modules relying entirely on attention which have been attracted as a promising technique in many sequence-based tasks (Bahdanau et al., 2014).

Existing SSO models utilized RNN-based decoders cannot pass encoded sentence vectors because their decoders sequentially decode one paragraph vector. Attention mechanisms allow each sentence in a different position to link other sentences regardless of the order and number of input sentences. Thus, our attention based pointer decoder directly utilize the topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ in a permutation-agnostic manner (global information). This allows our model to avoid bottleneck problems which dilute word/sentence semantics between encoder and decoder and fully utilize the semantics for sentence ordering.

Moreover, Our attentive pointer decoder can capture topic-enhanced sentence-pair interactions between the current predicted sentence and

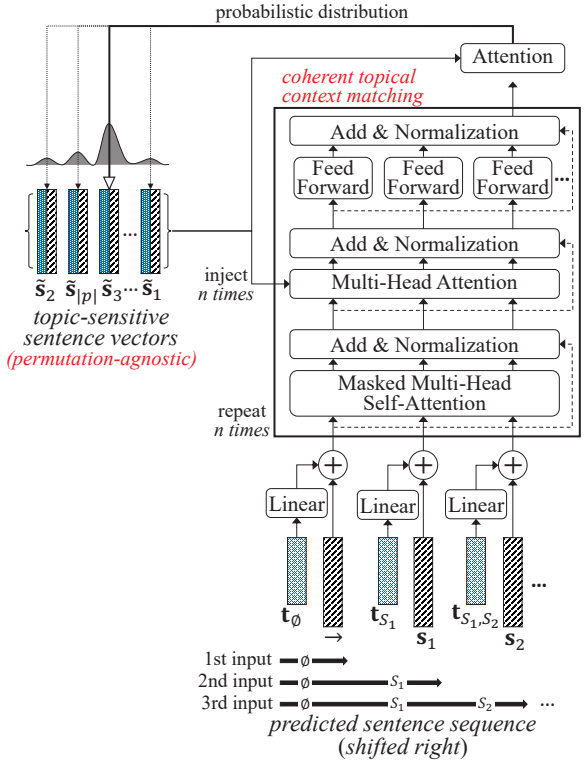

Figure 4: Architecture of attentive pointer decoder. The decoder fetches permutation-agnostic topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ from the encoder. For coherent topical context modeling, at the 3rd step, a stack of n attention modules position-wisely takes $\{\rightarrow, \mathbf{s}_1, \mathbf{s}_2\}$ and newly created topic latent vectors $\{\mathbf{t}_\varnothing, \mathbf{t}_{s_1}, \mathbf{t}_{s_1,s_2}\}$, where $\mathbf{t}_\varnothing$ denotes a zero vector. The topic vectors are generated by incrementally including previously predicted $\{s_1, s_2\}$, The attention layer outputs a probabilistic distribution over encoder outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ and select the next sentence s_3.

encoder outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$) as next-sentence candidates. With the coherent topical context (local information) matching, we promote local dependencies for identifying the tight semantic connections for sentence ordering.

Coherent Topical Context Matching. Our attentive pointer decoder employs a stack of attention modules identical with the decoder of Transformer (Zhou et al., 2016) for the coherent topical context matching. Given query $\mathbf{Q} \in \mathbb{R}^{n \times d_m}$, key $\mathbf{K} \in \mathbb{R}^{n \times d_m}$, and value $\mathbf{V} \in \mathbb{R}^{n \times d_m}$, the attention mechanism computes the output matrix $\mathbf{Out}_{att} \in \mathbb{R}^{n \times d_m}$ obtained from the value matrix $\mathbf{V}$ with an attention weight α as follows:

$$\alpha = softmax(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}), \quad \mathbf{Out}_{att} = \alpha \mathbf{V}, \quad (7)$$

where α is calculated with the query-key pair by the scaled dot product. Note that self-attention is

the case when all query, key, and value matrices are the same. Masked attention, a variant of attention, masks out all positions after the current position by arbitrarily setting a large value ($-\infty$) in the softmax function.

Firstly, for the coherent topical context matching, we position-wisely feed previously predicted sentence vectors and their newly created topic latent vectors to the masked multi-head self-attention sub-layer. In that, each topic latent vector is generated by $topicDistribution(\cdot)$ with predicted sentences in all positions before the current position. Thus, we incorporate the topical context flow of the previously predicted sentence sequence into the corresponding predicted sentence vector.

After a residual connection (He et al., 2016) and layer normalization (Lei Ba et al., 2016), the resulting vectors are injected as the query matrix $\mathbf{Q}$ into the multi-head attention sub-layer. The sub-layer also takes permutation-agnostic topic-sensitive sentence vectors obtained from our encoder as the key and value matrices $\mathbf{K}$ and $\mathbf{V}$. We follow Transformer for the remaining coherent topical context matching process, including multi-head attention strategy. As a result, with the multiple attention modules, TGCM draws global dependencies among sentences by attending over the topic-sensitive sentence vectors repeatedly.

Probabilistic Distribution for Ordering. At the ith step, the attention module of the decoder takes previously predicted sentences $\{\tilde{s}_1, ..., \tilde{s}_{i-1}\}$ and $\tilde{s}_0$ for the token "$\rightarrow$" as inputs. After repeating the attention module n times, we then utilize the d_w-dimensional output vector $\mathbf{c}_{i-1}$ corresponding to the $(i-1)$th decoder position. As shown in Figure 4, at the 3rd step, the output vector $(\mathbf{c}_3)$ of the 3rd position is fed into the final attention layer as like in conventional pointer decoders.

In the final attention layer, for the output vector $\mathbf{c}_i$ of the attention modules, TGCM produces an output distribution over topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_{|p|}\}$ obtained from encoder via content-based attention as follows:

$$\mathbf{u}_i^j = \mathbf{v}_a^\intercal \, tanh(W_a \begin{bmatrix} \tilde{\mathbf{s}}_j \\ \mathbf{c}_i \end{bmatrix}), \ (1 < j < |p|), \\ P(\hat{o}_i | \hat{o}_{<i}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_{|p|}) = softmax(\mathbf{u}_i), \quad (8)$$

where $\hat{o}_i$ denotes the order of the ith position. $W_a \in \mathbb{R}^{d \times 2d}$ $\mathbf{v}_a^\intercal \in \mathbb{R}^d$ denote the weight matrix and vector, respectively, which are shared in all positions. Thus, we can select the correct next sentence s_i that yields the highest probability from the output distribution.

Finally, the selected sentence vector $\mathbf{s}_i$ is fed into the decoder along with previously selected sentences $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{i-1}\}$ and newly created topic latent vectors $\{\mathbf{t}_\varnothing, \mathbf{t}_{s_1}, \mathbf{t}_{s_1,s_2}, ..., \mathbf{t}_{s_1,...,s_i}\}$ as inputs at the $(i+1)$th step.

Training. For each correct sentence sequence, we sample 5 shuffled sentence sequences at training phase. Following the typical sentence ordering models, we train parameters of our model to maximize the coherence probability by minimizing the loss function as follows.

$$L = -\frac{1}{|D|} logP(\hat{O}_p | p, \Theta) + \frac{\lambda}{2} ||\Theta||_2^2, \quad (9)$$

where Θ and λ denote a set of trainable parameters and a regularization parameter, respectively.

	Train	Valid	Test	#Voca.
NIPS abstract	2,248 (5.91)	409 (5.95)	402 (5.86)	16,721
ANN abstract	8,569 (4.83)	962 (4.84)	2626 (4.97)	34,485
NSF abstract	96,070 (8.68)	10,185 (8.73)	21580 (8.63)	334,090
arXiv abstract	884,912 (5.38)	110,614 (5.39)	110,615 (5.37)	64,557
SIND	40,155 (5)	4,990 (5)	5,055 (5)	30,861

Table 1: The number of paragraphs, average sentences (in parentheses), and vocabulary size of 5 datasets.

4 Experimental Setup

The parameters of TGCM were tuned by the RMSProp[1] optimizer, which adaptively adjusts the learning rate for each parameter and resolves the problem of Adagrad (Duchi et al., 2011) where the learning rate radically decreases or increases.

All experiments were implemented in Python using TensorFlow (Abadi et al., 2016), which supports the GPU-accelerated deep learning. We also utilized Natural Language Toolkit[2] (NLTK) (Loper and Bird, 2002) for sentence tokenization and data preprocessing, and Gensim[3] for LDA-based topic modeling. The word embeddings were initialized with pre-trained GloVe vectors of dimension $d_w = 200$, trained by GloVe (Pennington et al., 2014). We trained our LDA-based topic model for each dataset with its training, valid, and test corpora.

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
[2] https://www.nltk.org/
[3] https://radimrehurek.com/gensim/

4.1 Datasets

We first conducted our experiments on commonly used 4 abstract datasets: NIPS, ANN (Wang et al., 2018), NSF, and arXiv (Chen et al., 2016), which contain abstracts of research papers. The abstracts datasets collected from NIPS papers, ACL Anthology Network (AAN) corpus, NSF Research Award abstract, and arXiv website, respectively. Abstracts have the logical consistency of high quality, which helps coherence modeling for sentence ordering. We further considered the Sequential Image Narrative Dataset (SIND) (Huang et al., 2016), which consists of personal multimodal stories consisting of five sentences per story. We did not utilize accident and earthquake datasets (Barzilay and Lapata, 2008) because the datasets are too small and do not provide the validation set. Following the setup in (Logeswaran et al., 2018), we split undivided datasets for training, validation, and test. Table 1 shows the details of the 5 datasets used in our experiments.

4.2 Evaluation Metrics

Following the evaluation metrics widely used in previous SPO and SSO models, we adopted three metrics to assess sentence ordering performance: Kendall's tau, perfect match ratio, and positional accuracy.

Kendall's tau (τ): Kendall's tau (Lapata, 2003, 2006) measures the ordinal association between two sequences to automatically evaluate coherence modeling as follows:

$$\tau = \frac{1}{|D|} \sum_{i \in |D|} \frac{2 \times inversions(O_{p_i}, \hat{O}_{p_i})}{|p_i|(|p_i|-1)/2}, \quad (10)$$

where $|D|$ and $|p_i|$ denote the total number of paragraphs in a test dataset D and the number of sentences in a paragraph p_i, respectively. The function $inversions(O_{p_i}, \hat{O}_{p_i})$ returns the number of sentence-pair interchanges for reconstructing the correct order O_{p_i} from the predicted order $\hat{O}_{p_i}$. The value ranges from -1 to 1. A higher value indicates better performance. The evaluation metric closely correlates with user ratings and reading times, which are related with the readability.

Perfect Match Ratio (PMR in %): The perfect match ratio is the ratio of exactly matching orders across all predicted paragraphs as follows:

$$\text{PMR} = \frac{1}{|D|} \sum_{i \in |D|} \mathbb{I}(O_{p_i} = \hat{O}_{p_i}), \quad (11)$$

where $\mathbb{I}(O_{p_i} = \hat{O}_{p_i})$ denotes the indicator function, which returns 1 if the correct sentence order O_{p_i} and the predicted sentence order $\hat{O}_{p_i}$ are identical and 0 otherwise.

Positional Accuracy (PAcc in %): The positional accuracy is defined as the ratio of the matched sentence-level absolute positions between the predicted and correct orders as follows:

$$\text{PAcc} = \frac{1}{|D|} \sum_{i \in |D|} \sum_{j \in |p_i|} \mathbb{I}(o_{s_j} = \hat{o}_{s_j}) \quad (12)$$

where o_{s_j} denotes the absolute position of sentence s_j in the correct sequence S_j. Likewise, $\hat{o}_{s_j}$ denotes the absolute position of sentence s_j in the predicted sequence $\hat{S}_j$. $\mathbb{I}(o_{s_j} = \hat{o}_{s_j})$ denotes the indicator function equal to 1 if $o_{s_j} = \hat{o}_{s_j}$ and 0 otherwise.

4.3 Hyperparameters

For LDA-based topic modeling, we decided its hyperparameters β and d_t with a grid search on each dataset. Following Griffiths and Steyvers (2004), we kept $\alpha = 0.1$ and $\beta = 50/d_t$ constantly and obtained the best results with $d_t = 300$ for all abstract datasets and $d_t = 200$ for SIND. We initially configured $d_w = 200$ with the Glove vectors and updated the word vectors during training. For our attentive pointer decoder, we followed the same hyperparameters in Transformer and used a learning rate of 0.01. Other parameters were initialized randomly based on He et al. (2015).

5 Experimental Results

This section reports experimental results on the sentence ordering task for determining a coherent order of a given sentence. The proposed TGCM was compared with state-of-the-art methods as baselines such as Pairwise Model (Chen et al., 2016), Seq2seq (Logeswaran et al., 2018), RNN Decoder (Logeswaran et al., 2018), V-LSTM+PtrNet (Logeswaran et al., 2018), CNN+PtrNet (Gong et al., 2016), LSTM+PtrNet (Gong et al., 2016), and ATTOrderNet (Cui et al., 2018). Here, except the random model, all of the baselines are based on neural networks, which are typically more competitive than traditional approaches (e.g., utilizing handcraft features). For the LDA-based topic model in our preprocessing, we obtained the topic distributions of words by learning their relative importance for each topic.

	NIPS abstract			ANN abstract			NSF abstract			arXiv abstract			SIND		
	τ	PMR	PAcc	τ	PMR	PAcc	τ	PMR	PAcc	τ	PMR	PAcc	τ	PMR	PAcc
Random Model	0	7.53	15.59	0	8.07	19.36	0	5.42	9.46	0	8.07	14.11	0	6.05	9.32
Pairwise Model	0.47	19.72	26.63	0.58	21.54	41.82	0.25	13.56	15.51	0.66	33.43	50.79	0.32	10.43	30.75
Seq2seq	0.27	14.39	21.18	0.40	18.09	36.62	0.10	11.63	13.68	0.52	29.43	45.67	0.21	8.64	26.18
RNN Decoder	0.67	23.31	48.22	0.66	21.31	52.06	0.48	14.87	25.79	0.66	35.53	48.31	0.38	10.68	31.53
V-LSTM+PtrNet	0.72	27.87	51.55	0.73	29.79	58.06	0.51	18.53	28.33	0.72	41.74	55.90	0.45	13.44	35.26
CNN+PtrNet	0.66	26.79	48.64	0.69	26.65	58.21	0.52	16.73	33.22	0.71	39.28	52.92	0.48	12.32	35.52
LSTM+PtrNet	0.67	28.20	50.87	0.69	30.41	58.20	0.52	19.53	32.45	0.72	40.44	54.31	0.48	12.34	34.45
ATTOrderNet	0.72	29.87	56.09	0.73	32.11	63.24	**0.55**	21.35	37.72	0.73	42.19	56.11	0.49	14.01	36.24
TGCM-S	0.72	29.02	57.67	0.74	34.14-	64.65	0.53	21.09	39.15	0.73	42.51	55.16	0.51	14.41	36.75
TGCM	**0.75**	**31.44**	**59.43**	**0.75**	**36.69**	**65.16**	**0.55**	**22.35**	**42.67**	**0.75**	**44.28**	**58.31**	**0.53**	**15.18**	**38.71**

Table 2: Performance of neural networks based models on the sentence ordering task. Best results are in boldface.

	arXiv		SIND	
	Head	Tail	Head	Tail
Random Model	23.06	23.16	22.78	22.56
Pairwise 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	91.00	68.08	76.00	54.42
TGCM-S	91.15	67.93	77.34	54.64
TGCM	**92.46**	**69.45**	**78.98**	**56.24**

Table 3: Performance of predicting the correct head and tail sentences on arXiv and SIND. The results are directly taken from (Cui et al., 2018).

We could infer the topic latent vectors of new documents at the test phases.

Given a shuffled sentence sequence, the main goal is to find the most coherent sentence sequence. The coherence probability of a predicted sentence sequence is approximated to the coherence probability of a correct sentence sequence. In Table 2-3, the several values of some models are directly taken from (Cui et al., 2018), while we implemented the rest of models using their public code with the same experimental setup. Some methods do not release their implemented codes and so were implemented. In the case of the latest model ATTOderNet, we only utilized the high-performance model among three versions.

We discuss the reason for our highest performance among previous models by classifying the factors affecting performance into four categories: sentence set modeling, topic latent vectors, permutation invariance, and attention at decoder.

5.1 Impact of Sentence Set Modeling

This is related to the global dependencies of an entire paragraph. We observed that the performances of the random model and the pairwise model (Chen et al., 2016) were the worst. The pairwise model only learns the relative order from sentence-pair interactions. Since the two models do not consider all sentences at a time, they cannot leverage paragraph-level information. In other words, they cannot capture the global dependencies that help sentence ordering.

5.2 Impact of Topic Latent Vectors

Furthermore, we evaluated the variant version of TGCM, referred to TGCM-S, where the encoder of TGCM-S do not combine topic latent vectors and original sentence vectors. Also, the decoder of TGCM-S do not take newly created topic latent vectors. This allows us to explore whether the topic latent vectors actually help the sentence ordering task. Since TGCM-S cannot perform coherent topical context matching, TGCM-S do not elaborately capture the local dependencies, which help to identify tight semantic connections between the current predicted sentence and each next-sentence candidate.

As shown in the Tables 2-3, full TGCM outperformed TGCM-S which employs only attention-based bi-LSTMs without topic modeling. Although TGCM-S do not utilize topical context features at encoder and decoder (such as $\mathbf{t}_p$ and $\mathbf{t}_s$), TGCM-S also employs the transformer-based attentive pointer decoder. Thus, we found that topic modeling additionally contributes to modeling sentences through global topical context relating to local topical context of each sentence, resulting in improving the sentence ordering performance.

5.3 Impact of Permutaion Invariance

Compared to ATTOrderNet and our TGCM, the performance of other models was relatively low.

The conventional models typically adopt hierarchical RNN-based encoders, which combine sentence vectors sequentially and generate paragraph-level representation as a vector. Since they are dependent on the permutation of the input sentences, their paragraph-level representations are not reliable.

On the other hand, self-attention-based ATTOrderNet (encoder side) and attention-based our TGCM/TGCM-S (decoder side) use a set of sentence representations in a permutation-invariant manner rather than a single vector to represent a paragraph. Thus, these models can capture global dependencies regardless of the order of input sentences. The outputs of their encoder are more informative and reliable for improving the sentence ordering performance.

5.4 Impact of Attention at Decoder

This is related to solving the bottleneck problem by enhancing the expressiveness of the decoder's input and capturing the global and local dependencies among input sentences simultaneously.

Table 2-3 show that ATTOrderNet and TGCM, which utilize an attention mechanism, perform much better than all the other models. With an encoder employing self-attention mechanism, ATTOrderNet can capture some global dependencies with the reliable paragraph-level representation regardless of their input order. However, since ATTOrderNet also compresses sentence vectors into a single fixed-length paragraph vector via an average pooling layer, the word/sentence-level semantics from the encoder to the decoder are diluted.

In contrast to ATTOrderNet, TGCM feeds the sentence vectors directly into a decoder without compressing them into a single vector. Then, the attention mechanism allows our decoder to receive and utilize the sentence vectors, regardless of the order and number of sentences. As a result, since our TGCM could elaborately capture both local and global dependencies simultaneously, TGCM showed the best sentence ordering performance among all comparative models on all datasets.

6 Conclusions

We propose TGCM which better preserves local and global information for sentence ordering. Our attentive pointer decoder fully utilizes the semantics of sentence vectors without being compressed into a paragraph vector. Our sentence encoder produces topic-sensitive sentence vectors via topic modeling. With the coherent topical context matching between the current predicted sentence and each next-sentence candidate, we promote local dependencies that help identify the tight semantic connections. The empirical results on sentence ordering demonstrate that TGCM outperforms state-of-the-art models.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP; Ministry of Science, ICT & Future Planning) (No. NRF-2019R1A2B5B01070555).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. *arXiv preprint arXiv:1606.07493*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

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

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*, pages 4340–4349.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and Jinjun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2341–2351.

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

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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 *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 142–151.

Mingmin Jin, Xin Luo, Huiling Zhu, and Hankz Hankui Zhuo. 2018. Combining deep learning and topic modeling for review understanding in context-aware recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1605–1614.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 545–552. Association for Computational Linguistics.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

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

Jiwei Li and Dan Jurafsky. 2016. Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Biao Liu and Minlie Huang. 2016. A sentence interaction network for modeling dependence between sentences. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 558–567.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2020–2030.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *IJCAI*, pages 4411–4417.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-cast attention networks for retrieval-based question answering and response prediction. *arXiv preprint arXiv:1806.00778*.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015b. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Paper abstract writing through editing mechanism. *arXiv preprint arXiv:1805.06064*.

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*.

Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars, and Kam-Fai Wong. 2018. Microblog conversation recommendation via joint modeling of topics and discourse. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 375–385.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.