Sentence Ordering with a Coherence Verifier

Sainan Jia[1], **Wei Song**[1*], **Jiefu Gong**[2], **Shijin Wang**[2], **and Ting Liu**[3]

[1]Information Engineering College, Capital Normal University, Beijing, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Hefei, China
[3]Harbin Institute of Technology, Harbin, China
jiasainan0929@gmail.com, wsong@cnu.edu.cn
{jfgong, sjwang3}@iflytek.com, tliu@ir.hit.edu.cn

Abstract

This paper presents a novel sentence ordering method by plugging a coherence verifier (CoVer) into pair-wise ranking-based and sequence generation-based methods. It does not change the model parameters of the baseline, and only verifies the coherence of candidate (partial) orders produced by the baseline and reranks them in beam search. We also propose a coherence model as CoVer with a novel graph formulation and a novel data construction strategy for contrastive pre-training independently of the sentence ordering task. Experimental results on four benchmarks demonstrate the effectiveness of our method with topological sorting-based and pointer network-based methods as the baselines. Detailed analyses illustrate how CoVer improves the baselines and confirm the importance of its graph formulation and training strategy. Our code is available at `https://github.com/SN-Jia/SO_with_CoVer`.

1 Introduction

Coherence is essential for effective communication. The correct order of sentences is a necessary attribute of text coherence. Sentence ordering aims to organize a set of possibly unordered sentences into a coherent text. It is closely associated with coherence modeling. On one hand, it has been used as an objective for learning coherence models. On the other hand, it can be viewed as a follow-up module of coherence evaluation, e.g., for improving texts with low coherence scores. So sentence ordering has highly practical value in downstream tasks for evaluating and improving the quality of human writing (Amorim et al., 2018; Mim et al., 2019) or machine-generated content (Reiter and Dale, 1997; Fan et al., 2019; Hu et al., 2020; Guan et al., 2021).

[*]Corresponding author, supported by the National Natural Science Foundation of China (No. 61876113)

Recent sentence ordering studies can be classified into 2 categories: pair-wise ranking-based and sequence generation-based methods.

Pair-wise ranking-based methods first model the relative order of each sentence pair and then integrate all the predicted relative orders with some ranking methods to get the final order (Chen et al., 2016; Prabhumoye et al., 2020; Ghosal et al., 2021; Zhu et al., 2021). For example, B-TSort (Prabhumoye et al., 2020) uses BERT for pair-wise classification, builds a constraint graph to integrate pair-wise predictions, and adopts the topological sorting algorithm for sentence ranking.

Sequence generation-based methods are mainly based on the pointer networks (Vinyals et al., 2015). An encoder encodes all unordered sentences in various ways to capture the paragraph-level contextual information (Cui et al., 2018; Yin et al., 2019; Wang and Wan, 2019; Yin et al., 2021; Lai et al., 2021), then a decoder iteratively selects the next one from the set of unordered sentences conditioned on the states of the encoder and the already ordered sentence sequence.

However, both categories of methods have a shortcoming in that the coherence of ordered sentences is not directly optimized but is approximated by optimizing auxiliary tasks, e.g., pair-wise ordering and ranking algorithms, or optimizing a series of conditional decisions, e.g., iterative sentence selection by a pointer network. These sub-optimal objects have a misalignment with the purpose of finding an order with maximal global coherence.

In this paper, we propose a simple sentence ordering method by introducing a **Co**herence **Ver**ifier (CoVer). It can be plugged into the ranking-based and sequence generation-based models. Figure 1 shows an example of how CoVer works together with a sequence generation baseline. CoVer only intervenes in the generation process. At each inference step, we let the baseline provide top candidates as the next sentence (e.g., s_4 and s_3) and

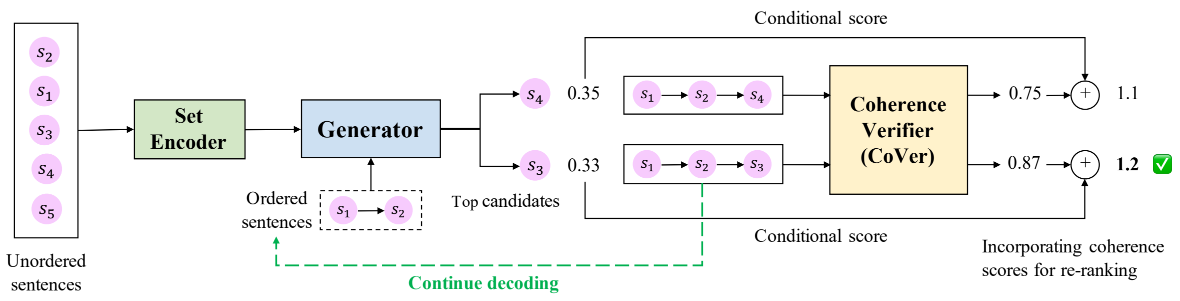

Figure 1: An example of how CoVer works together with a sequence generation model. The model's encoder can encode a set of sentences in various ways and its decoder iteratively generates sentence order based on (accumulated) conditional scores. CoVer is used as a coherence verifier to measure the coherence of a candidate order. It can be pre-trained and flexibly plugged into the decoding process through beam search.

use CoVer to verify the coherence of the sentence sequence candidates (e.g., s_1, s_2, s_4 and s_1, s_2, s_3) and re-rank the candidates for future generations. As a result, our method combines local conditional evidence and global coherence.

CoVer is trained to measure coherence independently of the sentence ordering task. This is reasonable and important since the input of a coherence model is an ordered sentence sequence rather than a set of unordered sentences, and the model can be pre-trained with multi-domain datasets. We propose a novel coherence model, with a new graph formulation to model sentence pair orders, sequence order, and paragraph-to-sentence relations, and a novel gradual permutation-based data construction strategy for effective contrastive pre-training from pairs of sentence orders with different coherence degrees.

We evaluate the effectiveness of CoVer by letting it work with a topological sorting-based baseline B-TSort (Prabhumoye et al., 2020) and a pointer network-based sequence generation baseline BERSON (Cui et al., 2020). Experimental results on four benchmarks demonstrate that our method improves both baselines and especially, obtains a large gain for the topological sorting-based baseline. It also outperforms other recent methods.

We conduct a series of in-depth analyses showing that our method can correct a large ratio of sentence pair classification errors made by B-TSort and improve ordering accuracy at the early decoding stage for BERSON, which alleviates the gap between training and inference, and reduces error propagation. These effects come from the key designs of our coherence model. Moreover, the CoVer pre-trained with larger cross-domain datasets obtains better performance than the models trained with domain-specific datasets. The results verify the importance of pre-training the independent coherence model and also indicate that sentence ordering and coherence modeling can cooperate and interact well.

2 Background

2.1 Coherence Modeling

The main coherence modeling methods can be classified into the following categories.

Entity grid-based Methods measure local coherence by tracking the transitions of the grammatical roles of entities between sentences (Barzilay and Lapata, 2008; Lin et al., 2011). Tien Nguyen and Joty (2017) proposed the first neural entity model based on convolutional neural networks (CNNs). Jeon and Strube (2022) proposed to compute coherence by constraining the input to noun phrases and proper names since they explicitly lead to the notion of focus in sentences.

Graph-based Methods are another framework for modeling local coherence. Guinaudeau and Strube (2013) described relations between sentences and entities with graphs and measured local coherence by computing the average out-degree of graphs. Mesgar et al. (2021) adopted graph convolutional networks (GCNs) for encoding entity graphs to model local coherence.

Data-driven Methods focus on learning domain-independent neural models of discourse coherence (Li and Jurafsky, 2017; Farag and Yannakoudakis, 2019). The key is to define proper learning objects, including discriminative models to distinguish coherent from incoherent discourse, generative models to produce coherent texts (Li and Jurafsky, 2017), and multi-task learning with auxiliary tasks (Farag and Yannakoudakis, 2019).

2.2 Sentence Ordering

Sentence ordering task takes possibly out-of-order sentences $s = s_1, s_2, ..., s_n$ as input, and aims to find the best order $o^* = o_1, o_2, ..., o_n$ to make the sentence sequence $s_{o_1}, s_{o_2}, ..., s_{o_n}$ with maximal global coherence.

Recent sentence ordering methods are mainly based on neural networks and can be classified into the following two categories.

2.2.1 Pair-wise Ranking based Methods

The main procedure of this category of methods is:

Step 1: Learn a pair-wise classifier to determine the relative order of each sentence pair. The classifier can be trained based on BERT (Prabhumoye et al., 2020; Zhu et al., 2021) or GCNs (Ghosal et al., 2021).

Step 2: Integrate the relative orders to build relations between sentences. A common way is to build a constraint graph based on the relative orders.

Step 3: Rank the sentences based on the graph with a ranking algorithm like topological sorting (Prabhumoye et al., 2020; Ghosal et al., 2021), or using a neural network to score sentences (Zhu et al., 2021), or modeling it as the asymmetric traveling salesman problem (Keswani and Jhamtani, 2021).

2.2.2 Sequence Generation based Methods

Sequence generation-based models mainly depend on the pointer networks (Vinyals et al., 2015). The encoder maps a set of sentences into a fixed-length vector representation in various ways (Cui et al., 2018; Yin et al., 2019, 2021; Lai et al., 2021; Basu Roy Chowdhury et al., 2021; Cui et al., 2020). The decoder iteratively generates the sentence sequence based on the attention scores over input sentences.

Formally, the decoders focus on modeling an autoregressive factorization of the joint coherence probability of a predicted order $\hat{\mathbf{o}}$,

$$\begin{aligned} p(\hat{\mathbf{o}}|s) &= \prod_{i=1}^{n} \underbrace{p(\hat{o}_i|\hat{\mathbf{o}}_{<i}, s)}_{\text{conditional probability}} \\ &\propto \prod_{i=1}^{n} \underbrace{a_{U_i}(\hat{o}_i|\hat{\mathbf{o}}_{<i}, s)}_{\text{attention score}} \end{aligned} \tag{1}$$

where $\hat{\mathbf{o}}_{<i}$ is the sequence of already ordered sentences, U_i is the set of unselected sentences at step i, $\hat{o}_i$ is the i-th sentence in $\hat{\mathbf{o}}$ and $a_{U_i}(s_i|\hat{\mathbf{o}}_{<i}, s)$ is the attention score for a candidate sentence $s_i \in U_i$.

Beam search can be used for enlarging the search space and ranking partially generated hypotheses during decoding. But the ranking in beam search is still based on conditional evidence (Equation 1).

3 The Proposed Framework

3.1 The Motivation

The existing sentence ordering methods make the best decisions based on conditional evidence or local constraints, but do not directly optimize global coherence. This is natural because the model cannot see the complete global information before generating the final ordering.

When people do the same task, we also start from incomplete information. However, once we have a partial or final ordering, we often revisit the already-ordered sentences to verify whether the current text is coherent or needs to be revised. The verification step is intuitive and important since we can see more complete information.

Motivated by the above observations, we propose a simple sentence ordering framework by incorporating an independent coherence verifier. We call it CoVer. CoVer reads an ordered sentence sequence and gives a coherence score. We expect CoVer can verify the predicted results of a baseline model and rerank the candidates to get a more coherent one.

We will introduce the details of CoVer in §4. In this section, we focus on demonstrating that CoVer can be flexibly incorporated with sequence generation-based (§3.2) and topological sorting-based (§3.3) models through beam search.

3.2 CoVer for Sequence Generation-based Models

As Figure 1 shows, CoVer can be easily incorporated into a pointer network-based baseline model. It only intervenes in the decoding process. At each decoding step, we compute the score of a candidate sentence s_i as

$$g(s_i) = \alpha\, \underbrace{a_{U_i}(s_i|\hat{\mathbf{o}}_{<i}, s)}_{\text{attention score}} + \underbrace{\text{CoVer}(\hat{\mathbf{o}}_{<i}, s_i)}_{\text{coherence verifier}} \tag{2}$$

where $a_{U_i}(s_i|\hat{\mathbf{o}}_{<i}, s)$ is the attention score. We put s_i at the end of $\hat{\mathbf{o}}_{<i}$ and $\text{CoVer}(\hat{\mathbf{o}}_{<i}, s_i)$ returns a coherence score for the resulted sentence sequence.

CoVer can be incorporated through beam search and $g(s_i)$ in Equation 2 becomes $g(\hat{\mathbf{o}}_{<i}, s_i)$.

A beam $B = \{\hat{\mathbf{o}}_{<i}\}$ stores the top k preceding orders where k is the beam size and each candidate $s_i \in U_i$ is combined with the items in B. We score each combination $(\hat{\mathbf{o}}_{<i}, s_i)$ based on $g(\hat{\mathbf{o}}_{<i}, s_i)$ and store the top k combinations in B.

3.3 CoVer for Pair-wise Ranking-based Methods

For a pair-wise model, CoVer does not affect the pair-wise classifier and only affects the ranking part as long as the model can provide multiple ordering candidates. In this paper, we focus on improving topological sorting-based methods.

The topological sorting algorithm reads a constraint graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where an edge from $v_i \in \mathcal{V}$ to $v_j \in \mathcal{V}$ indicates the sentence s_i is predicted to be preceding s_j in the document. At each time, the node without any incoming edges would be selected. Then the algorithm removes this node and its associated edges from $\mathcal{G}$ and repeats the above process until all nodes are processed.

We can see that the ordering process is also a generation process. As a result, we slightly modify the generation process and describe it in Algorithm 1.

Algorithm 1: CoVer for Topological Sorting through Beam Search

```
Input: Directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, beam size $k$,
       steps $t$ to look ahead, start returns the start
       node in a graph based on the topological
       sorting algorithm, top_k returns the top $k$
       ranked items in a list
Output: beam $B = [o_1, o_2, ..., o_k]$ , each $o_i$ is an
        ordered list
$B \leftarrow [\emptyset], iter = 0$
while $iter < |V|$ do
    $b \leftarrow []$
    for $o \in B$ do
        $c \leftarrow o$
        for $j = 0; j < t; j++$ do
            $next = \texttt{start}(G \backslash set(c))$
            $c.append(next)$
            $b.append(o + next)$
        end
    end
    CoVer scores items in $b$
    $B = \texttt{top\_k}(b)$
    $iter = iter + 1$
end
return $B$
```

We introduce a beam B to store the top k partial orderings (line 1). A key operation is letting the topological sorting algorithm look ahead t steps to have more and longer partial ordering candidates and store them in a temporary list b (line 3 to line 11). CoVer scores the partial ordering candidates in b and the top k ones are stored in the beam B for future generation (line 12 to line 13).

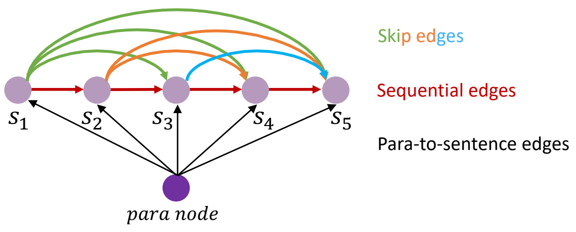

Figure 2: A tournament digraph to encode the order and topic of sentences in a paragraph. Each sentence has a node and the paragraph as a whole has a node.

In this way, CoVer plays a role in the whole generation process and corrects the errors made by the pair-wise classifier in time by measuring coherence, which is ignored by topological sorting.

4 CoVer: The Coherence Model

We propose a new graph-based coherence model as CoVer. Specially, we propose a new graph formulation and model it with GNNs for coherence evaluation (§4.1). We also propose a new data construction strategy for contrastive pre-training of the coherence model (§4.2).

4.1 Graph Formulation and Modeling

Given ordered sentences in a paragraph d, we construct a graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E}, \mathcal{R})$. $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a set of directed edges connecting nodes and $\mathcal{R}$ is the set of edge types. Figure 2 shows an example of the graph for a paragraph with 5 sentences. The graph is a tournament digraph, in which every pair of distinct nodes is connected by a directed edge.

We consider two types of nodes $\mathcal{V} = \{v_d\} \cup \mathcal{V}_s$:

- **Sentence nodes** $\mathcal{V}_s$: Each sentence s_i with an ordered index i has a node $v_i \in \mathcal{V}_s$.
- **Paragraph node** v_d: The paragraph has a node to represent the general topic of the paragraph.

We also consider three types of directed edges and the edge types are $\mathcal{R} = \{r_d, r_s, r_k\}$:

- **Paragraph-to-sentence edges**: We build a directed labeled edge (v_d, r_d, v_i) from the paragraph node (para-node) to each sentence node, where r_d indicates the edge type.
- **Sequential edges**: We build a directed labeled edge (v_i, r_s, v_{i+1}) with a type r_s between sentence s_i and s_{i+1}.

- **Skip edges**: We build a directed labeled edge (v_i, r_k, v_j) with a type r_k between sentence s_i and s_j, if $j > i + 1$.

Sequential edges are the most natural choice for describing local coherence (Mesgar et al., 2021). We further use densely connected skip edges to describe long-distance ordering information so that every sentence s_j can directly receive information from all preceding sentences in the same paragraph rather than only receiving summarized information from s_{j-1}. The formulation is rarely explored in previous coherence modeling work.

Node Representations We map the nodes to dense vectors. Specifically, we use DeBERTa (He et al., 2021) to get the representation of each sentence node. Each sentence is fed to DeBERTa independently and the hidden state of the [CLS] token is used as the node representation. For the paragraph node, we let DeBERTa read the entire paragraph to get the representation of the para-node. So the positional embeddings naturally encode the ordering information.

Graph Modeling Following previous work (Mesgar et al., 2021; Ghosal et al., 2021), we use Relational Graph Convolutional Networks (RGCN) (Schlichtkrull et al., 2018) to further encode the relations between nodes, which is a natural choice for the modeling of edges between nodes.

The RGCN model can accumulate relational evidence from the neighborhood around a given node v_i in multiple inference steps, i.e.,

$$h_i^{(l+1)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{W_r^{(l)}h_j^{(l)}}{|\mathcal{N}_i^r|} + W_0^{(l)}h_i^{(l)}\Big) \quad (3)$$

where $h_i^{(l)}$ represents the hidden state of node v_i in the l-th layer of the neural network. We use the representation of node v_i from DeBERTa as $h_i^{(0)}$. $r \in \mathcal{R}$ is one of the edge types and $\mathcal{N}_i^r$ represents the set of nodes connected to v_i through edge type r. W_r is the parameter matrix for r and W_0 is the parameter matrix for the self-connection edge, which is an extra type in addition to $\mathcal{R}$. $\sigma(\cdot)$ is set as $\texttt{ReLU}(\cdot)$. RGCN stacks L layers and we set $L = 2$, the same as (Ghosal et al., 2021).

Coherence Evaluation After getting the final representations of all nodes, we get the representation of the graph $\mathcal{G}$ via $h_\mathcal{G} = \sum_{v\in\mathcal{V}} h_v$ and map it to a coherence score $Coh(\mathcal{G})$, i.e.,

$$Coh(\mathcal{G}) = \texttt{sigmod}(\texttt{FFN}(h_\mathcal{G})) \quad (4)$$

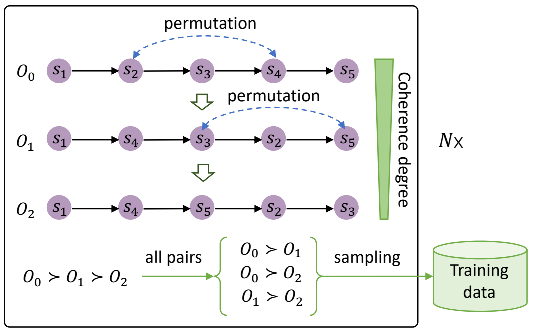

Figure 3: An example for constructing training instances with the gradual permutation strategy.

where FFN is a single-layer feed-forward neural network.

4.2 Model Training

Training Objective We train our model based on a pair-wise ranking manner. Given a text d^+ with a higher coherence degree than a text d^-, we use the following loss function for updating model parameters, $\mathcal{L} = \max(0, \tau - Coh(\mathcal{G}_{d^+}) + Coh(\mathcal{G}_{d^-}))$, where $\mathcal{G}_{d^+}$ and $\mathcal{G}_{d^-}$ are corresponding graphs for d^+ and d^-, and $\tau = 0.1$ is the margin.

Training Instance Construction The model can be trained using documents with manually annotated coherence degrees. However, the data scale is very limited. Another common way is distinguishing a coherent document from its permutations, where a coherent document and one of its random sentence permutations form a training instance. We call this way **random permutation**.

We propose a **gradual permutation** strategy by *gradually* corrupting a coherence document through pair-wise sentence permutation. Figure 3 illustrates an example of gradual permutation. A pair-wise permutation operation is to randomly select a pair of sentences that are not selected before in the current order and exchange them to get a new order. We assume the new order is less coherent than the previous one. By repeating this process, we can get a sequence of order samples $o_1, o_2, ...$ with descending coherence degrees. Finally, we sample pairs of orders in the final sequence to form pair-wise training instances according to their relative coherence degrees. For one document, gradual permutation can be done multiple times.

Compared with random permutation, gradual permutation pays more attention to evaluating relative coherence between imperfect orders with different coherence degrees, instead of only distinguishing a perfect order from imperfect ones.

Dataset→	NIPS			AAN			SIND			ROCStory		
Model	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR
HAN	-	66.71	14.06	-	69.03	31.29	-	50.21	15.01	-	73.22	39.62
DARN	-	74.62	24.13	-	77.48	39.18	-	56.52	15.48	-	76.02	38.02
SEK-Graph	58.25	76.49	-	65.06	78.60	-	17.17	53.16	-	-	-	-
Con-Graph	-	80.29	32.84	-	82.36	49.81	-	58.56	19.07	-	81.22	49.52
STaCK	63.60	81.66	37.31	71.60	85.56	54.01	54.20	61.94	20.79	76.70	85.34	55.96
B-TSORT	61.07	79.91	32.63	65.05	79.39	47.29	45.43	47.34	13.16	65.02	72.58	39.14
+ COVER$_{dom}$	69.55	84.39	46.42	74.06	86.00	61.43	55.95	61.22	28.55	81.63	86.06	68.44
+ COVER	70.77	85.92	48.54	74.15	**86.06**	**62.04**	54.85	60.06	27.00	81.04	85.57	67.33
BERSON	69.08	82.32	38.73	77.73	85.04	58.56	59.74	65.53	31.89	83.51	88.35	69.17
+ COVER$_{dom}$	72.42	84.59	46.42	78.11	85.26	59.14	60.07	65.92	32.50	84.76	**89.22**	71.82
+ COVER	**74.86**	**86.10**	**50.93**	**78.13**	85.21	59.17	**60.31**	**66.01**	**32.96**	**84.80**	89.16	**72.27**

Table 1: The general comparison results against two baselines and other recent methods on four datasets.

Pre-Training The training of the coherence model can be independent of the sentence ordering task. As a result, COVER can be pre-trained with domain-independent resources and be maintained as a verifier for sentence ordering in specific domains.

5 Experiment

5.1 Experimental Settings

Datasets We conduct experiments on four widely used benchmarks. **NIPS** and **AAN** contains abstracts from NIPS and ACL anthology network papers (Logeswaran et al., 2018). **SIND** is originally used for visual storytelling (Huang et al., 2016), where natural language descriptions are provided for five images of each story. **ROCStory** is a dataset of short stories, each of which has five sentences (Mostafazadeh et al., 2016). We follow the original papers to split each dataset into training, validation, and test sets. Detailed statistics are shown in Table 2.

Dataset	Length statistics		Data split		
	mean	max	train	valid	test
NIPS	6	14	2427	408	377
AAN	5	20	8568	962	2626
SIND	5	5	40155	4990	5055
ROCStory	5	5	78529	9816	9816

Table 2: Statistics of datasets used in our experiments.

Evaluation Metrics We adopt the following three commonly used metrics for evaluation.

Perfect Match Ratio (PMR): PMR measures the percentage of documents for which the entire sequence is correctly predicted (Chen et al., 2016).

Kendall's τ: It measures the difference between the predicted order and the gold order of sentences based on the number of inversions (Lapata, 2003).

Accuracy (ACC): It measures the percentage of sentences, whose absolute positions are correctly predicted (Logeswaran et al., 2018).

Baselines and Settings We use B-TSort (Prabhumoye et al., 2020)[*] and BERSON (Cui et al., 2020)[†] as our main baselines. We choose them because they are recent representative pair-wise ranking-based and pointer network-based methods, with top performance and almost reproducible results with publicly released codes. We use optimized parameters provided by the original papers and re-run the source codes in our machine. We run these baselines for three times with different random seeds and use the baseline models with the best performance in our experiments.

Our method lets COVER work together with B-TSort and BRESON, utilizing and adjusting their predictions with beam search. The same as the setting of BERSON, we set the beam size as 16 for both baselines. The looking ahead steps t in Algorithm 1 is 2. The hyper-parameter α in Equation 2 is 0.1, which is chosen from $\{0.01, 0.1, 0.5, 1\}$ based on the validation performance.

We use the AdamW optimizer for training the coherence model. The learning rate for the parameters of DeBERTa, which is used for getting node representations, is 1e-6 and the learning rate for the parameters of the RGCN model is 1e-4.

We pre-train COVER using the combination of the training sets of the four benchmarks with an A100 GPU for 40 hours and train a domain-specific COVER$_{dom}$ for each dataset using the corresponding training set. For one document, we sample two sentence permutations as negative instances.

[*]https://github.com/shrimai/Topological-Sort-for-Sentence-Ordering/

[†]https://github.com/hwxcby/BERSON

5.2 General Results on Sentence Ordering

Table 1 shows the performance of our method, two main baselines, and other recent methods.

First of all, we can see that both CoVER and CoVER$_{dom}$ improve the two baselines on all benchmarks. The pre-trained CoVER outperforms the domain-specific CoVER$_{dom}$ in most cases, indicating pre-training the coherence model is feasible and useful. We can maintain a single coherence model instead of domain-specific ones and even have a boost in overall performance.

Based on the beam search algorithm for topological sorting, CoVER obtain 11.1%, 9.6%, and 18.1% average absolute improvements in Acc, Kendall's τ, and PMR compared with B-TSort.

Based on adding coherence verification in the beam search-based decoding, CoVER achieves 2.0%, 1.3%, and 4.2% average absolute improvements in Acc, Kendall's τ, and PMR compared with BERSON. The improvements are smaller but still significant. Especially, our method has significant performance improvement in PMR.

We also conduct comparisons with other recent methods, including hierarchical attention network (HAN) (Wang and Wan, 2019), deep attentive ranking network (DARN) (Kumar et al., 2020), constraint graph (ConGraph) (Zhu et al., 2021), knowledge-enhanced GNN (SEK-Graph) (Yin et al., 2021) and STaCK (Ghosal et al., 2021). Our method based on either B-TSort or BERSON outperforms all these methods.

5.3 Effect of CoVER for B-TSort

Our method gets large improvements for B-TSort. We hope to analyze the improvements more deeply and conduct investigations on the NIPS dataset.

We start by analyzing the predictions made by B-TSort's pair-wise classifier. Specifically, we group sentence pairs according to the distance between two sentences. We investigate the *error ratio* for different distance d, where

$$\text{error ratio} = \frac{\#\text{incorrect pair-wise prediction}}{\#\text{all pairs within distance } d},$$

and analyze the *confidence* of the pair-wise classifier, using its prediction probability as a confidence measure.

Figure 4 illustrates the error ratio and averaged prediction confidence for different values of d. B-TSort's classifier is more confident and accurate for determining the relative order of sentence pairs

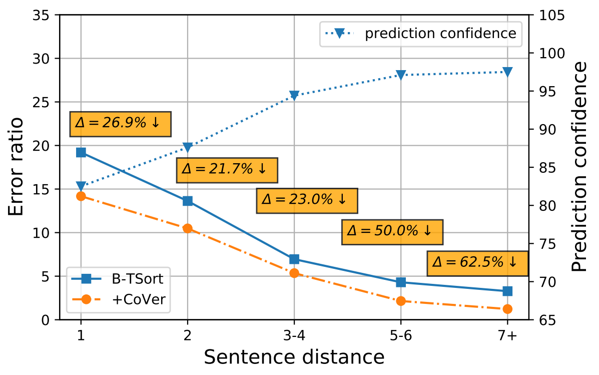

Figure 4: The error ratio of predicted pair-wise relative orders w/ and w/o CoVER for B-TSort and the average prediction confidence of B-TSort's pair-wise classifier.

with larger distances but is less confident and struggles in handling the relative order of nearby sentences. This is reasonable since nearby sentences share similar topics in content so it is hard to determine the relative order without a larger context. The topological sorting algorithm does not consider content information and cannot deal with low-confidence predictions as well.

Figure 4 also shows the error ratio of B-TSort plus CoVER. Our method reduces 21% to 27% errors for sentence pairs with distance $d \leq 4$ and reduces more than 50% errors for long-distance sentence pairs. This indicates that based on Algorithm 1, CoVER overcomes the limitations of the original topological sorting algorithm and gradually improves the predictions.

5.4 Effect of CoVER for BERSON

We infer that one of the reasons that CoVER improves BERSON is alleviating the gap between the training and inference. We conduct a controlled experiment to verify this assumption.

During inference, we experiment with different input orders to the decoder of BERSON: 1) perfect: the input order is the right-shift of the gold order, which is the same as the training phase; 2) predicted: the input order is according to the predicted order, which is the normal way for inference. In either case, we evaluate the outputs of the decoder.

Method	Input Order	Acc	τ	PMR
BERSON	Perfect	79.84	83.47	57.55
BERSON	Predicted	72.51	80.31	49.58
+ CoVER	Predicted	75.53	81.62	53.83

Table 3: Average performance with perfect and predicted order as the input of BERSON's decoder.

Method	Acc	τ	PMR
B-TSort+Mesgar et al. (2021)	63.19	73.91	34.51
B-TSort+CoVer	70.20	79.40	51.23
- Skip edges	69.19	78.87	49.46
- Para node	58.90	71.23	29.94
BERSON+Mesgar et al. (2021)	72.60	80.41	50.23
BERSON+CoVer	74.53	81.62	53.83
- Skip edges	73.17	80.91	51.02
- Para node	72.39	80.11	49.58

Table 4: Ablation study of the graph formulation.

B-Sort	Acc	τ	PMR
Random permutation	69.19	78.23	49.76
Gradual permutation	70.20	79.40	51.23
BERSON	**Acc**	**τ**	**PMR**
Random permutation	73.75	81.15	51.77
Gradual permutation	74.53	81.62	53.83

Table 5: Average performance with two strategies.

	Error Ratio		
position	BERSON	Random	Gradual
1	7.69	5.57	3.98
2	28.38	21.22	17.51
3	40.69	32.18	26.60
4-5	43.22	40.69	36.07
6-7	40.76	39.67	40.49
8+	40.71	39.29	42.86

Table 6: Error ratio for sentences at different positions with random and gradual permutation strategies.

Table 3 shows the average performance over four datasets. BERSON with perfect input orders sets an upper bound. In contrast, in the normal way for inference, BERSON's performance drops a lot because there are likely errors during order generation and the imperfect preceding order would affect the future generation as well. With the help of CoVer, BERSON can get a performance closer to that with perfect input order.

A natural assumption about the effect is that CoVer improves the predictions in the early decoding stage so that future generation is based on a closer-to-perfect preceding order.

5.5 Ablation Study of CoVer

We further investigate the effectiveness of key designs of CoVer, mainly from two aspects: the graph formulation and the training strategy.

Graph Formulation We focus on analyzing the importance of the skip edges and the paragraph node, which mainly encode ordering information.

Table 4 shows the results. For B-TSort, removing skip edges leads to a small performance decrease, while removing the paragraph node leads to a large decrease. The reason may be that the topological sorting algorithm depends on the predicted pair-wise relative orders but does not consider any content information. So encoding the content of a paragraph is more important. For BERSON, the paragraph node and the skip edges are both important. The skip edges explicitly connect preceding sentences and the candidate sentence, which may help deal with imperfect partial orders.

A state-of-the-art coherence model Mesgar et al. (2021) is also used as the verifier. It improves B-TSort and BERSON, but has a certain gap with CoVer, indicating the advantage of coherence verification and the designs of CoVer.

Training Strategies We compare the random permutation and gradual permutation strategies. Table 5 shows the average performance over four datasets. Gradual permutation consistently gets better performance than random permutation.

We further analyze the error ratio for sentences at different positions in the gold orders on the NIPS dataset. Table 6 shows that using either strategy, our method can obviously reduce the error ratio for almost all positions. Random permutation outperforms BERSON at all positions, while gradual permutation has the lowest error ratio for sentences at the front and middle of the documents. This is because, with the training instances constructed by gradual permutation, the model can better compare relative coherence between imperfect orders so it can correct more errors in preceding sentences, making the decoding more robust. But gradual permutation has a slightly worse error ratio at the end of the documents. The reason may be that the training instances containing perfect orders are less, affecting the judgment for sentences at the end. In the future, we will investigate better sampling strategies that can keep a trade-off between random permutation and gradual permutation.

Connecting the above observations, CoVer significantly improves the accuracy at the front of documents and can gradually improve the partial orderings. These factors can reasonably explain the effects of CoVer for B-TSort and BERSON.

5.6 Predicting the First and Last Sentences

The first and last sentences are important to documents. Following previous studies, we report the

Method	NIPS	AAN	SIND	ROCStory
		First Sentence		
B-TSort	91.51	89.68	74.28	88.30
+ CoVer	94.43	92.74	80.24	94.50
BERSON	92.31	93.18	85.44	96.34
+ CoVer	96.02	93.54	85.64	97.13
		Last Sentence		
B-TSort	79.05	78.59	52.58	72.10
+ CoVer	79.84	82.07	61.17	83.38
BERSON	80.37	81.54	66.25	85.85
+ CoVer	79.58	81.54	66.41	84.90

Table 7: Accuracy of predicting the first and last sentences on four benchmarks.

performance of our model against two baselines in correctly predicting these two sentences on four benchmarks.

As displayed in Table 7, our method obtains significant improvements in predicting the first sentences across four benchmarks for both B-TSort and BERSON. However, it performs better than B-TSort but slightly worse than BERSON in predicting the last sentences. This observation is consistent with the analysis in §5.5.

5.7 Performance on Short and Long Documents

We conduct experiments on NIPS and AAN datasets to analyze the effects of CoVer for short and long texts. The documents in the test set are divided into short ones (with less than 8 sentences) and long ones (with 8 or more sentences). There are 298 short and 79 long documents in NIPS, and 2358 short and 268 long documents in AAN.

Table 8 shows the results. For two baselines, our method has great improvements in both short and long documents.

5.8 Performance on Coherence Rating

We also evaluate our model for the summary coherence rating (SCR) task. We use the dataset proposed by Barzilay and Lapata (2008), which contains English summaries produced by human experts and an extractive summarization system. Each instance in the dataset is a pair of two summaries with different ratings of the same text.

We compare our model with *EntGraph* (Guinaudeau and Strube, 2013), *Neural EntGrid* (Tien Nguyen and Joty, 2017) and Mesgar et al. (2021). Table 9 shows that our model performs very close to the best performance

Dataset	NIPS		AAN	
Method	τ	PMR	τ	PMR
		Short Text		
B-TSort	81.77	40.27	81.32	52.13
+ CoVer	87.20	56.71	87.47	67.46
BERSON	83.39	46.31	86.27	63.12
+ CoVer	86.47	56.71	86.45	63.67
		Long Text		
B-TSort	72.86	3.80	62.49	4.85
+ CoVer	81.08	17.72	73.76	14.55
BERSON	78.29	10.13	74.26	18.66
+ CoVer	84.72	29.11	74.31	19.78

Table 8: Results on short and long texts in the NIPS and AAN datasets.

Model	Acc
EntGraph	80.0
Neural EntGrid	86.3
Mesgar et al. (2021)	87.5
CoVer	87.4

Table 9: Results for summary coherence rating in DUC2003 dataset.

system, indicating that our method is effective for multiple coherence evaluation tasks.

6 Conclusion

This paper has presented a novel sentence ordering method by incorporating a coherence verifier (CoVer). We show that CoVer works well with pair-wise ranking-based and sequence generation-based baselines. Our framework combines local evidence from the baselines and larger context coherence from CoVer and can gradually improve partial orderings. The coherence verifier is independent of the sentence ordering task but can be optimized for sentence ordering (e.g., via gradual permutation), and can be pre-trained with multi-domain datasets, obtaining superior performance compared with domain-specific models. So it is effective and easy to maintain and transfer.

Sentence ordering is often used as a training task for coherence modeling. This paper, however, suggests that coherence models can also support sentence ordering methods to correct incoherent texts. Coherence models are able to identify sentences that are not well-connected. Sentence ordering models can then be used to reorder these sentences to improves the coherence of the text with the assistance of the coherence models.

Limitations

While the proposed method performs well on four benchmarks, we discuss some of its limitations.

On one hand, as discussed in §5.5, our method is not accurate enough to predict sentences at the end of the documents. There may be some better strategies to construct training samples so that the model can better take into account each part of the documents and make more accurate predictions.

On the other hand, our model is not pre-trained with more diverse domains and larger scale data. Our datasets are limited to two types, i.e., paper abstracts and short stories, both of which have comparatively obvious order characteristics. In addition, we do not use some larger scale datasets, such as NSF abstracts and arXiv abstracts, because of computation and time constraints. With more diverse and larger data, the performance of our model should be further improved.

References

Evelin Amorim, Marcia Cançado, and Adriano Veloso. 2018. Automated essay scoring in the presence of biased ratings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 229–237, New Orleans, Louisiana. Association for Computational Linguistics.

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

Somnath Basu Roy Chowdhury, Faeze Brahman, and Snigdha Chaturvedi. 2021. Is everything in order? a simple way to order sentences. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10769–10779, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *ArXiv*, abs/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, Brussels, Belgium. Association for Computational Linguistics.

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 (EMNLP)*, pages 6310–6320, Online. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy. Association for Computational Linguistics.

Youmna Farag and Helen Yannakoudakis. 2019. Multi-task learning for coherence modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 629–639. Association for Computational Linguistics (ACL).

Deepanway Ghosal, Navonil Majumder, Rada Mihalcea, and Soujanya Poria. 2021. STaCK: Sentence ordering with temporal commonsense knowledge. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8676–8686, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. 2021. Long text generation by modeling sentence-level and discourse-level coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6379–6393, Online. Association for Computational Linguistics.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103, Sofia, Bulgaria. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2020. What makes a good story? designing composite rewards for visual storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7969–7976.

Ting-Hao Kenneth 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: Human Language Technologies*, pages 1233–1239, San Diego, California. Association for Computational Linguistics.

Sungho Jeon and Michael Strube. 2022. Entity-based neural local coherence modeling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7787–7805, Dublin, Ireland. Association for Computational Linguistics.

Vishal Keswani and Harsh Jhamtani. 2021. Formulating neural sentence ordering as the asymmetric traveling salesman problem. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 128–139, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Pawan Kumar, Dhanajit Brahma, Harish Karnick, and Piyush Rai. 2020. Deep attentive ranking networks for learning to order sentences. 34(05):8115–8122.

Shaopeng Lai, Ante Wang, Fandong Meng, Jie Zhou, Yubin Ge, Jiali Zeng, Junfeng Yao, Degen Huang, and Jinsong Su. 2021. Improving graph-based sentence ordering with iteratively predicted pairwise orderings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2407–2417, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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*, pages 198–209, Copenhagen, Denmark. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 997–1006, USA. Association for Computational Linguistics.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Mohsen Mesgar, Leonardo F. R. Ribeiro, and Iryna Gurevych. 2021. A neural graph-based local coherence model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2316–2321, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Farjana Sultana Mim, Naoya Inoue, Paul Reisert, Hiroki Ouchi, and Kentaro Inui. 2019. Unsupervised learning of discourse-aware text representation for essay scoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 378–385, Florence, Italy. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

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*, pages 2783–2792, Online. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

Tianming Wang and Xiaojun Wan. 2019. Hierarchical attention networks for sentence ordering. 33(01):7184–7191.

Yongjing Yin, Shaopeng Lai, Linfeng Song, Chulun Zhou, Xianpei Han, Junfeng Yao, and Jinsong Su. 2021. An external knowledge enhanced graph-based neural network for sentence ordering. *Journal of Artificial Intelligence Research*, 70:545–566.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5387–5393. ijcai.org.

Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural sentence ordering based on constraint graphs. 35(16):14656–14664.

A Detailed Experimental Results

Table 10 lists the detailed error ratio data in Figure 4 for reference. We also report the detailed results of four benchmarks about the controlled experiment for analyzing the gap between training and inference in §5.4 (Table 11) , ablation study of the graph formulation in §5.5 (Table 12) and the performance with random and gradual strategies in §5.5 (Table 13).

Distance	1	2	3-4	5-6	7+
B-TSort	19.41	13.38	6.95	4.30	3.28
+ CoVer	14.18	10.48	5.35	2.15	1.23

Table 10: The detailed error ratio of predicted pair-wise relative orders w/ and w/o COVER for B-TSort

Dataset→		NIPS			AAN			SIND			ROCStory		
Method	Order	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR
BERSON	Perfect	75.32	86.96	44.03	85.52	87.71	68.04	69.54	68.79	42.04	88.99	90.42	76.10
BERSON	Predicted	69.08	82.32	38.73	77.73	85.04	58.56	59.74	65.53	31.89	83.51	88.35	69.17
+ CoVer		74.86	86.10	50.93	78.13	85.21	59.17	60.31	66.01	32.96	84.80	89.16	72.27

Table 11: The detailed results for Table 3 in §5.4.

Dataset→	NIPS			AAN			SIND			ROCStory		
Method	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR
B-TSort + Mesgar et al. (2021)	61.73	80.54	28.38	65.42	80.96	45.10	51.71	56.08	20.99	73.88	78.06	43.57
B-TSort + CoVer	70.77	85.92	48.54	74.15	86.06	62.04	54.85	60.06	27.00	81.04	85.57	67.33
- SkipEdges	68.77	84.51	43.50	71.38	84.56	58.60	54.37	59.29	26.55	80.86	85.23	67.01
- ParaNode	58.03	76.71	25.46	60.80	77.58	39.22	49.02	54.92	15.71	67.74	75.69	39.36
BERSON + Mesgar et al. (2021)	69.11	82.43	40.27	77.83	85.09	58.55	59.81	65.73	32.03	83.65	88.39	70.08
BERSON + CoVer	74.86	86.10	50.93	78.13	85.21	59.17	60.31	66.01	32.96	84.80	89.16	72.27
- SkipEdges	70.67	83.61	41.91	77.88	85.13	58.72	60.05	65.91	32.50	84.08	88.97	70.94
- ParaNode	68.87	81.77	38.99	77.79	85.09	58.49	59.73	65.50	31.93	83.18	88.07	68.89

Table 12: The detailed results for Table 4 §5.5.

Dataset→	NIPS			AAN			SIND			ROCStory		
B-TSort	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR
Random Permutation	69.73	84.46	46.15	73.10	85.15	60.59	54.76	59.53	27.75	79.15	83.78	64.53
Gradual Permutation	70.77	85.92	48.54	74.15	86.06	62.04	54.85	60.06	27.00	81.04	85.57	67.33
BERSON	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR	Acc	τ	PMR
Random Permutation	72.32	84.35	44.30	77.88	85.11	58.79	60.08	65.89	32.56	84.73	89.23	71.43
Gradual Permutation	74.86	86.10	50.93	78.13	85.21	59.17	60.31	66.01	32.96	84.80	89.16	72.27

Table 13: The detailed results for Table 5 in §5.5.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Limitations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
5.1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
5.1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5.1 and A

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
5.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.