Formulating Neural Sentence Ordering as the Asymmetric Traveling Salesman Problem

Vishal Keswani
Indian Institute of Technology, Kanpur
vkeswani@iitk.ac.in

Harsh Jhamtani
Carnegie Mellon University
jharsh@cs.cmu.edu

Abstract

The task of Sentence Ordering refers to rearranging a set of given sentences in a coherent ordering. Prior work (Prabhumoye et al., 2020) models this as an optimal graph traversal (with sentences as nodes, and edges as local constraints) using topological sorting. However, such an approach has major limitations – it cannot handle the presence of cycles in the resulting graphs and considers only the binary presence/absence of edges rather than a more granular score. In this work, we propose an alternate formulation of this task as a classic combinatorial optimization problem popular as the Traveling Salesman Problem (or TSP in short). Compared to the previous approach of using topological sorting, our proposed technique gracefully handles the presence of cycles and is more expressive since it takes into account real-valued constraint/edge scores rather than just the presence/absence of edges. Our experiments demonstrate improved handling of such cyclic cases in resulting graphs. Additionally, we highlight how model accuracy can be sensitive to the ordering of input sentences when using such graphs-based formulations. Finally, we note that our approach requires only lightweight fine-tuning of a classification layer built on pre-trained BERT sentence encoder to identify local relationships.

1 Introduction

A logical and coherent structure is an important characteristic of easily comprehensible text. As such, modeling the structure of coherent texts has been an important problem in NLP (Barzilay and Elhadad, 2002). In this paper, we work on the task of sentence ordering, wherein given an unordered set of sentences, the aim is to generate the most coherent ordering among them (Table 1). It essentially arises in situations where the information available in parts (sentences) is to be presented as a whole in a coherent and logical ordering. Correctly ordering sentences has applications for summarization (Barzilay and Elhadad, 2002; Nallapati et al., 2017; Narayan et al., 2018), constructing natural language explanations (Jhamtani and Clark, 2020), automatic scoring of an essay (Attali and Burstein, 2006; Farag et al., 2018; Amorim et al., 2018), and automatic generation and evaluation of a narrative (See et al., 2019; Jhamtani and Berg-Kirkpatrick, 2020; Gangal et al., 2021; Sakaguchi et al., 2021).

Input	s_1: Our son really likes his new bike.
	s_2: It's almost Christmas time.
	s_3: They really love what they got.
	s_4: We had a very fun time.
	s_5: The Children begin to open their presents.
Correct Output Sequence: $s_2 \rightarrow s_5 \rightarrow s_3 \rightarrow s_1 \rightarrow s_4$	

Table 1: An instance of the Sentence Order Generation Task. Given a set of incoherently arranged sentences as input, the goal is to output a coherent ordering. (Story from the SIND dataset)

Much prior work has formulated sentence ordering as a sequence prediction task (Gong et al., 2016; Cui et al., 2020), by first encoding the input sentences, and then predicting the ordering or numbering of the sentences. There have also been attempts to model the task as a ranking problem (Kumar et al., 2020). More recently, Prabhumoye et al. (2020) model this as a constraint-solving problem, wherein they first identify the relative ordering between pairs of sentences using a BERT-based (Devlin et al., 2019) classifier. Thereafter, they formulate it as performing a topological sort on a graph, wherein each sentence is a node in a graph, and each directed edge represents a pairwise constraint. Compared to prior work, it simplifies the decoding process by using a graph traversal formulation. However, the topological sort algorithm used for translating constraints into the final order

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 128–139,
Aberdeen, Scotland, UK, 20-24 September 2021.

leads to major limitations. Firstly, it cannot handle cycles in the resulting graph, and picks some arbitrary ordering in such cases (Figure 1). Our analysis shows that cycles were present in more than 54% of the graphs across the four datasets under consideration. Secondly, though the underlying classifier can provide more fine-grained scores, the scores are discretized/binarized to run the topological sorting algorithm. This leads to a loss of useful information in the decoding process.

In this work, we propose a reformulation of Neural Sentence Ordering as the classical Traveling Salesman Problem, while leveraging the recent progress in large pre-trained models such as BERT. We build upon the classification model used in Prabhumoye et al. (2020) and overcome the limitations in their approach by making better use of the information yielded by the classifier, taking into account global dependencies by employing a combinatorial minimization objective, and working with an overall framework that can handle cycles in the resulting graph.

Our contributions can be summarized as follows. Firstly, we provide a novel formulation of the sentence ordering task as the Traveling Salesman problem. Compared to the previous graph-based approach of using topological sorting, our proposed technique gracefully handles the presence of cyclic constraints. Moreover, it is more expressive since it admits real-valued soft constraints as opposed to hard binary constraints. Secondly, experiments with multiple datasets demonstrate improved results under some setups compared to the baselines using alternative graph-based formulation. Finally, we observe how certain choices in data pre-processing in graph-based approaches for sentence ordering can affect accuracy scores. We propose and use a more robust data processing and evaluation. The code is publicly available. [1]

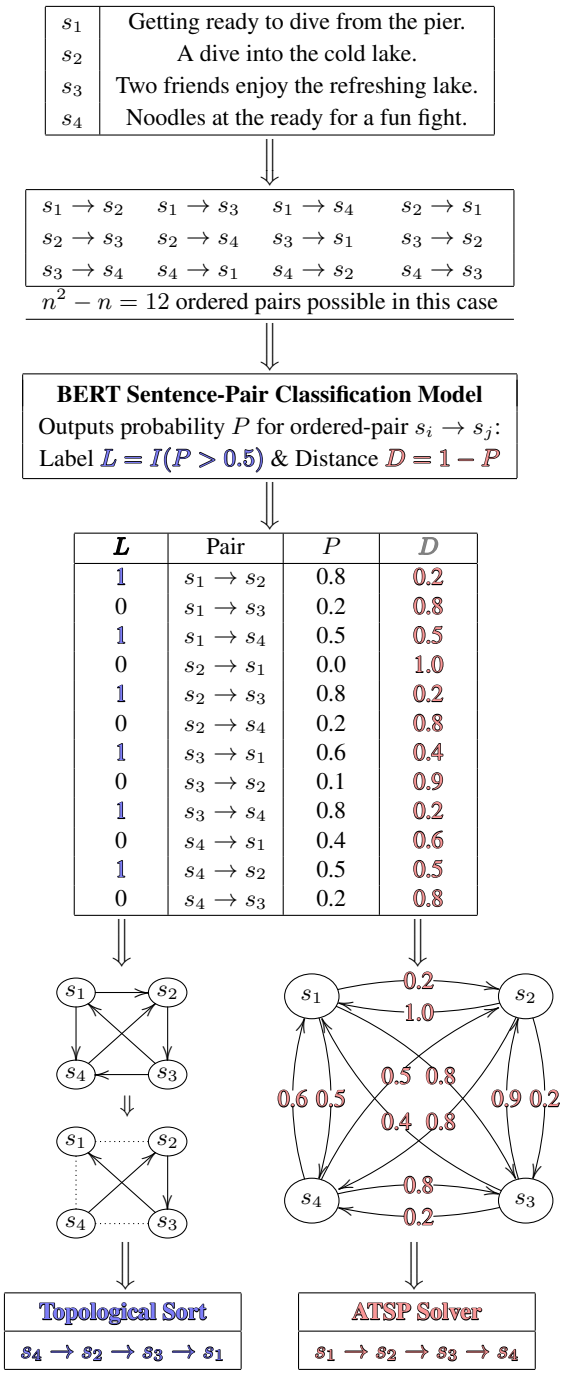

Figure 1: **Graph-traversal based formulation for sentence ordering task:** Such approaches first compute local pairwise constraints using next sentence prediction probability from a fine-tuned BERT classifier. **(a)** Topological Sort (Prabhumoye et al., 2020) discretizes the edges (0/1) and then runs topological sorting to get the final output sequence. However, such an approach is likely to pick an arbitrary ordering in the case of cycles. **(b)** In the proposed Traveling Salesman **ATSP** formulation, classifier probabilities are used to derive soft constraint scores between pairs of nodes, thus making use of more expressive fine-grained scores.

2 Background

In this section, we first formally describe the task of Sentence Ordering. Then we discuss its formulation as a constrained graph traversal, and discuss limitations of prior formulations of the task as a graph traversal problem.

2.1 Problem Formulation

Consider an 'unordered' set of n sentences: $S = \{s_1, s_2, ..., s_n\}$. Our aim is to find a permutation

[1] https://github.com/vkeswani/BerTSP

$P = \{i_1 i_2 ... i_n\}$ such that the resulting 'ordered' sequence is $S^* = \{s_{i_1}, s_{i_2}, ..., s_{i_n}\}$ is coherent.

Formulation as a Graph Traversal: Prabhumoye et al. (2020) propose to first identify binary constraints between all pairs of sentences. More specifically, given a pair of sentences $\{s_i, s_j\}$, they aim to extract whether s_i should follow s_j or the other way around. Thereafter, they decode the global ordering by treating the decoding as topological sorting in a graph, wherein sentences are treated as nodes, and pairwise constraints denote presence/absence of edges (i.e. there is an edge from s_i to s_j if a constraint states that s_j should follow s_i.). A high-level outline of this approach is shown in Figure 1.

2.2 Topological Sorting and its Limitations

To the best of our knowledge, (Prabhumoye et al., 2020) is the only work till now that utilizes a graph based formulation of Sentence Order Prediction on top of pairwise scores from a BERT-based classifier. Though it succeeds in achieving a light and efficient method for this task with minimal training, there are some inherent issues in this approach which we discuss below. Next, we will describe these limitations.

Discretization of edges: Prior work utilizes a BERT based classifier to predict local ordering constraints between pairs of sentences. However, to run the topological sorting, the classifier probability is converted a $0/1$ prediction, which governs merely the direction of the edge i.e. the fine-grained probability scores are thereafter not used. This leads to loss of valuable information about the likelihood of the edge, thus making it a significant issue. Since only binary constraints are learned and no score is attached to any order, it misses out on learning a rich global structure.

Cyclic constraints: The Topological Sort algorithm inherently cannot deal with cycles as it operates only on DAGs (Directed Acyclic Graphs). To deal with such cases, one can pick an arbitrary ordering of the nodes and edges, and delete edges that result in any cycles. However, such an approach will pick random orderings at best. For example, in Figure 1, the resulting graph has cycles, and the final output is a random ordering among the nodes.

Neutral pairs: To determine the direction of the edge between a pair of nodes, prior work feeds the pair of sentences to the classifier either as $s_1 \rightarrow s_2$ or as $s_2 \rightarrow s_1$ (both are equally likely). For instance, if $s_1 \rightarrow s_2$ is selected, the classifier predicts $P(s_1 \rightarrow s_2)$. If $P(s_1 \rightarrow s_2) > 0.5$, then the edge is directed from s_1 to s_2 in the graph (i.e. $s_1 \rightarrow s_2$). Otherwise, it is directed from s_2 to s_1 (i.e. $s_2 \rightarrow s_1$). Due to the lack of information in some sentence pairs or the limited efficacy of the classifier, the left out possibility is also probable. In such cases, both $P(s_1 \rightarrow s_2) > 0.5$ and $P(s_2 \rightarrow s_1) > 0.5$ are possible. We call such pairs neutral. The baseline approach does not consider breaking ties for neutral pairs. Datasets contain up to 50% samples with one or more neutral pairs. Note that $s_1 \rightarrow s_2$ and $s_2 \rightarrow s_1$ are not complementary events. Both lead to different input representations being fed to the classifier giving rise to different outputs (Section 3.3).

3 Sentence Ordering as the Traveling Salesman Problem

As mentioned previously, our objective is to work with soft constraints rather than binary constraints. To enable the use of such soft constraints, we cast the sentence ordering task as a Traveling Salesman traversal with respect to the graph denoting sentences as nodes, and constraints as edges. In the rest of this section, we first briefly describe the Traveling Salesman Problem (TSP) (Section 3.1), then describe how we reduce the sentences ordering as TSP (Section 3.2). Thereafter, we discuss the procedure to identify soft constraints using a classifier built on BERT representations. Finally, we discuss the solutions to solve TSP given a graph (Section 3.4).

3.1 Traveling Salesman Problem

The Traveling Salesman Problem is one of the more well-known problems studied in combinatorial optimization. In terms of graph theory, given an undirected weighted graph, it aims to find the shortest Hamiltonian Cycle, i.e. the cycle with the least weight that visits each node of the graph exactly once. Additionally, for our purpose and other practical applications, the graph is complete. Formally, we are given a complete undirected weighted graph $G = (V, E, W)$ where V denotes the set of vertices, E denotes the set of edges and W denotes the matrix containing weights for every edge.

$V = \{v_i\}_{\forall i \in \{1,2,...,n\}}$
$E = \{e_{ij}\}_{\forall i,j \in \{1,2,...,n\}}$

$W = \{w_{ij}\}\ \forall i,j \in \{1,2,...,n\}$

For G and a source vertex v_s ($s \in \{1, 2, ..., n\}$), we need to find a cyclic permutation or a Hamiltonian cycle $P = \{si_1i_2...i_{n-1}\}$ for which the following summation is minimized:

$$w_{si_1} + w_{i_1i_2} + ... + w_{i_{n-2}i_{n-1}} + w_{i_{n-1}s}$$

Since the permutation is cyclic, the summation is independent of the choice of s. This is the formulation of symmetric TSP for which $w_{ij} = w_{ji}$, i.e. distance from v_i to v_j is exactly the same as the distance from v_j to v_i. However, for some practical applications including ours, the two distances may not be equal. This is the case of asymmetric TSP.

3.2 Asymmetric TSP and Sentence Ordering

In the Asymmetric formulation of the Traveling Salesman Problem, $w_{ij} \neq w_{ji}$. These cases may arise in case of one-way traffic, accidental blockage, etc. The graph is still complete but directed with two edges between each pair of vertices in either direction. For the purpose of Sentence Ordering, we train a classifier that learns the probability of sentence i being followed by sentence j in the correct order and vice-versa. These probabilities give us the weights for the two edges between nodes v_i and v_j. Since the traditional TSP is proposed as a distance/cost minimization problem, a high probability should correspond to low distance. Hence, we use the probability of the complement (i.e. $1 - P$) as the weight.

$$w_{ij} = 1 - P(s_i \rightarrow s_j)$$
$$w_{ji} = 1 - P(s_j \rightarrow s_i)$$

Since the ground truth for the classifier is 1 for a correct pair-order and 0 for an incorrect pair-order, it is reflected in the predicted probabilities and hence, $P(s_i \rightarrow s_j) \neq P(s_j \rightarrow s_i)$. Intuitively, if sentence i is followed by sentence j in the correct order, the distance from i to j should be less than the distance from j to i, i.e. $w_{ij} < w_{ji}$. TSP requires the source to be pre-specified, for which we introduce a dummy node (Additional description about dummy node can be found in Appendix).

This way, each sentence serves as a vertex of graph G and the probabilities serve as entries of the weight matrix W. We use W to find the exact or heuristic solutions of the asymmetric TSP which eventually results in the correct Sentence Ordering.

In the TSP formulation, all of the issues with topological sorting are overcome. Firstly, the solution for TSP is independent of the order in which the input sentences are fed, hence preventing any sensitivity to the order in which sentences are processed. The graph constructed for every sample is cyclic and it does not have an inherent issue with cycles. This is reflected in its performance on samples with cycle(s) (table 2). Secondly, in the case of asymmetric TSP, weighted edges in both directions are required. This way both $P(s_1 \rightarrow s_2)$ and $P(s_2 \rightarrow s_1)$ make it to the weight matrix and the ambiguity of neutral pairs (local relationship) is resolved via their dependencies with other nodes (global relationship). Thirdly, since exact probability values are used as weights to arrive at a final order, there is no loss of information via discretization. Lastly, the order with the minimum cost arrangement is chosen which takes care of the global context. We refer to our method as **BerTSP**.

3.3 Learning Soft Constraints

We leverage BERT (Devlin et al., 2019) base-uncased configuration to obtain sentence pair representations. Specifically, we train a multi-layer feed-forward neural network that operates on sentence representation from pre-trained BERT, and makes a binary prediction about the relative ordering of the sentence pair. For every pair, it gives the probability of the first sentence (s_i) being followed by the second sentence (s_j). This way we obtain $P(s_i \rightarrow s_j)$ and consequently w_{ij}. It differs from (Prabhumoye et al., 2020) as it makes a prediction for either direction for a sentence pair ($s_i \rightarrow s_j$ and $s_j \rightarrow s_i$) while Prabhumoye et al. (2020) pick any one direction with a probability of 0.5 and use the binary label to define a constraint (unweighted edge). This way, for a set of n sentences, we obtain $2 \times C_2^n = n^2 - n$ scores which serve as off-diagonal elements of the distance matrix (diagonal elements are set to 0 as $P(s_i \rightarrow s_i) = 0$). This matrix is augmented with a row and a column of 0s to account for the dummy node discussed in the previous section and appendix. This augmented matrix serves as input for TSP.

In practice, we use the 'BertForSequenceClassification' module (Wolf et al., 2019) which takes the following sequence as input $[t_1^i, t_2^i, ..., t_n^i, \text{'SEP'}, t_1^j, t_2^j, ..., t_m^j]$. Here, the t^is represent tokens of sentence i and t^js represent tokens of sentence j. 'SEP' represents the separator which is a special token used by BERT.

3.4 Solution of ATSP

To produce an ordered sequence from the learned weight matrices, we delve into exact and approxi-

mate solutions to the Asymmetric TSP. The exact solution involves starting from the source node and calculating the cost of all permutations for the remaining nodes. This method is straightforward but expensive with a runtime complexity of $O(n!)$, where n is the number of sentences in the paragraph. Note that the above-mentioned complexity is only for the decoding process – the soft constraint computation involves running BERT only once per sentence i.e. number of forward passes on BERT scales linearly with the number of sentences ($O(n)$).

Approximation Algorithm for solving TSP: The popular approximate solvers available for TSP pose some limitations for our problem. They mainly focus on minimizing cost without regard to order. Their efficacy relies on the fraction of cost added to the optimal cost. They require assumptions such as triangle inequality. Hence, we consider a lightweight heuristic solution for the TSP problem where we simply sort the nodes (sentences) in the increasing order of the sum of soft-constraint scores arising at a given node. For example, for i^{th} node, the computed score is $\sum_{j}^{j!=i} P(s_i \rightarrow s_j)$. (Additional details about the employed approximation are provided in the appendix.) Since this process involves simple sorting, its runtime complexity for n sentences is $O(nlogn)$ (Same as the worst-case complexity for the topological sorting). We refer to this version of our method as **BerTSP-Approx**.

Ensemble Approach: Since the exact solution is expensive but generally more accurate while the above-mentioned approximation is computationally much cheaper but slightly less accurate in practice, we use a combination of these two as per the following criteria: We use the exact solution for samples with up to 10 sentences and the heuristic approximation for samples with more than 10 sentences. This gives us a practical solution that is feasible and almost optimal. We refer to this version of our method as **BerTSP-Ensemble**.

4 Experiments

4.1 Datasets

Stories: We use the textual portion of the Sequential Image Narrative Dataset or SIND (Huang et al.) which has been used in most of the previous studies on Sentence Ordering. It contains stories (image captions) with 5 sentences each.

Abstracts: We experiment on three research paper abstracts datasets, namely NIPS, AAN, and NSF, commonly used for this task. These are derived from NIPS conference papers, ACL Anthology Network corpus and NSF research award abstract dataset respectively (Logeswaran et al., 2018). (See appendix for data-split information.)

4.2 Evaluation

We employ the following five metrics to evaluate our approach. For all these metrics, a higher value corresponds to better performance.

Perfect Match Ratio (PMR) (Chen et al., 2016) measures the percentage of predicted orders that exactly match the correct order. It does not discount for minor differences.

Sentence Accuracy (ACC) (Logeswaran et al., 2018) measures the number of sentences having correct absolute positions in the predicted order. It is less strict than PMR.

Kendall's Tau (TAU) (Lapata, 2003) or Tau accounts for the pairwise relative order of sentences. Tau=1-2I/T where I represents the number of incorrect pair-orders in the predicted order and T represents total number of pair-orders in the correct order.

Rouge-S (R-S) (Chen et al., 2016) measures the number of skip-bigrams with the correct relative ordering in the predicted order as a percentage of the total number of skip-bigrams in the correct order. Here, skip-bigrams are pairs of sentences which may or may not be consecutive in the respective orders.

Longest Common Subsequence (LCS) (Gong et al., 2016) calculates the percentage of the longest common subsequence (not necessarily consecutive) between the predicted and the correct orders.

4.3 Call for Careful Data Pre-processing

We observe a potential risk of ground truth label leakage when employing graph-based methods. More specifically, if indexing of nodes (sentences) is performed in the same ordering as ground truth sequence, then the graph algorithms can inadvertently exploit this information. We demonstrate this through an illustrated example in Figure 2, wherein we show how the results of topological sorting change when indexing order of nodes is changed. In other words, if the input sentences

Figure 2: **Need for careful data pre-processing and evaluation:** When dealing with cycles in a graph, consider an implementation to handle cycles that always traverses nodes in the order $t = \{0, 1, 2, 3, 4\}$. For every node $j \in t$, cycle detection begins from j and if an edge $s_k \rightarrow s_j$ is encountered, it is deleted. As the traversal is in ascending order, for every deleted edge $s_k \rightarrow s_j$ $j < k$. Hence, edges that oppose the ascending order (t) are deleted and t is always favored. **(A)** In this case, the correct order is the same as t. Since t is favored, the predicted order is correct. **(B)** In this case, the correct order is the reverse of t (the rest of graph e is equivalent to a). Since t is favored, the predicted order is incorrect.

are indexed/fed as per the ground truth ordering, there is an information leak from the index numbers. To analyze the potential impact of such a pre-processing issue, we feed the input in the exact reverse of the correct order $\{n-1, n-2, ..., 0\}$, and present corresponding results for B-TSort (Prabhumoye et al., 2020) in Table 2. We observe that the accuracy values for methods like B-TSort can vary a lot based on the index ordering. From a practical use perspective, we are interested in analyzing the worst-case results across orderings. So from this point onward, we report results for B-TSort considering worst performance across orderings.

4.4 Results

We experiment with both variants of the proposed method. Additionally, we consider B-TSort method (Prabhumoye et al., 2020) as a baseline. (To the best of our knowledge, Prabhumoye et al. (2020) is the only work till now that delves into Graph Theory for Sentence Order Prediction like us.). We do not compare against non-graph based formulations since our aim is to improve upon the limitations of prior graph traversal based formulations.

We present the performance of B-TSort and BerTSP on the subsets of data that have cycles (as per discrete edge-based graphs used in B-TSort) in Table 2. The results demonstrate that the output of B-TSort varies as per the index ordering. Since the variation in the 3 presented outputs is very high, we prefer to use the worst-case output with BerTSP instead of the average case, henceforth. As shown in the table, we obtain significant worst-case improvements for both BerTSP-Approx and BerTSP-Ensemble, particularly the Perfect Match Ratio.

In Table 3, we present the results for whole

		PMR	ACC	TAU	R-S	LCS	PMR	ACC	TAU	R-S	LCS
		SIND - 12.74%					**AAN - 12.84%**				
B-TSort	Correct	26.71	55.37	0.64	81.88	78.97	28.87	57.42	0.79	87.54	81.18
	Shuffled	12.08	39.39	0.42	70.94	70.91	11.79	43.66	0.65	81.98	74.87
	Reverse	0.00	27.42	0.18	59.05	63.63	0.00	33.01	0.52	76.35	70.22
BerTSP	Approx	7.14	38.70	0.41	70.78	68.82	6.84	42.74	0.63	80.56	71.79
	Ensemble	12.42	39.19	0.41	70.62	71.06	12.50	43.26	0.64	80.89	74.50
		NIPS - 17.41%					**NSF - 70.47%**				
B-TSort	Correct	25.71	54.64	0.78	87.45	80.18	4.44	31.72	0.65	82.02	66.71
	Shuffled	12.00	43.50	0.67	82.97	74.71	1.61	22.99	0.51	76.18	60.64
	Reverse	0.00	33.93	0.56	78.01	70.54	0.00	19.18	0.37	70.30	58.05
BerTSP	Approx	10.00	44.11	0.64	82.24	71.96	0.75	20.98	0.45	71.76	54.23
	Ensemble	14.29	43.39	0.64	81.48	72.32	1.65	20.92	0.44	71.49	54.76

Table 2: **Results for data subset containing cycles as per graph representations used in B-TSort**: Percentage of cyclic cases and comparison of performance when using topological sorting on cyclic cases for three orderings in which inputs are fed: first is the correct order $\{0, 1, ..., n-1\}$, second is randomly shuffled order, and the third is the reverse of the correct order, i.e. $\{n-1, n-2, ..., 0\}$. The results demonstrate that if not properly handled, method outputs can depend highly on the indexing order of nodes. The two variants of BerTSP improve significantly on the reverse ordering output as depicted in the table.

datasets, comparing B-TSort and BerTSP on the worst case. We observe that BerTSP outperforms B-TSort on all metrics across all datasets (except LCS on NSF dataset). The improvements are also statistically significant with $p < 0.05$. It achieves up to 20% improvement on Perfect Match Ratio, 11% on Kendall Tau and 6% on position-wise Sentence Accuracy. It also improves on Rouge-S and LCS. Since these two metrics do not penalize gaps, their values are on the higher end and consequently, the improvements are less as compared to the other metrics. We also provide some qualitative examples in the appendix.

Model	PMR	ACC	TAU	R-S	LCS
SIND					
B-TSort	16.28	48.29	0.54	77.04	75.08
BerTSP-Approx	17.01	49.96	0.57	78.65	75.60
BerTSP-Ensemble	19.54	49.75	0.56	78.11	76.35
NIPS					
B-TSort	27.36	56.42	0.77	86.91	81.01
BerTSP-Approx	32.09	59.86	0.79	88.27	81.75
BerTSP-Ensemble	32.59	59.36	0.79	87.75	81.79
AAN					
B-TSort	46.67	64.54	0.79	86.61	83.81
BerTSP-Approx	48.09	66.44	0.81	87.81	84.19
BerTSP-Ensemble	48.81	66.41	0.81	87.59	84.57
NSF					
B-TSort	6.84	24.22	0.45	71.32	61.05
BerTSP-Approx	6.99	25.57	0.50	72.61	57.69
BerTSP-Ensemble	7.79	25.11	0.48	72.17	58.02

Table 3: Result for B-TSort and BerTSP on all 4 datasets. Both BerTSP-Approx and BerTSP-Ensemble outperform the baseline B-TSort for overall dataset results. (Results for BTSort are different from Prabhumoye et al. (2020) since we analyze the worst-case results considering all possible orderings for preprocessing (Section 4.3)).

We also note that both the variants of the proposed method outperform B-TSort. Among the two, BerTSP-Ensemble is consistently better as per PMR. For all the other metrics, both show competitive performance and BerTSP-Approx even beats BerTSP-Ensemble for some metrics/datasets. This shows that our simple heuristic approach works pretty well even though it is computationally inexpensive. This could be attributed to the fact that averages (or sums, equivalent in this case) tend to discount the bad predictions. They regress the overall representation of a sentence towards the good predictions (correctly predicted pair-orders) if they are in majority. This way we get a single-valued representation for a sentence that has information from both the local and global contexts.

5 Analysis and Discussions

In this section, we analyze our model on additional parameters including scalability, end-point performance, and displacement. Note that in this section, we use the BerTSP-Approx method for comparison

Dataset	Model	PMR	ACC	TAU	R-S	LCS
NIPS	**B-TSort**	0.00	28.30	0.59	78.67	63.52
	BerTSP	0.00	34.59	0.67	83.11	64.15
AAN	**B-TSort**	0.00	28.48	0.53	74.63	65.63
	BerTSP	0.00	29.41	0.55	76.06	64.09
NSF	**B-TSort**	0.04	17.29	0.41	70.96	56.43
	BerTSP	0.01	17.96	0.45	71.69	51.31

Table 4: Results for cases with more than 10 sentences.

Model	First	Last	First	Last
Dataset	**SIND**		**NIPS**	
B-TSort	77.19	57.03	89.55	74.38
BerTSP	78.85	59.33	92.54	74.63
Dataset	**AAN**		**NSF**	
B-TSort	88.76	78.78	61.13	40.55
BerTSP	90.10	79.55	66.95	47.13

Table 5: Prediction accuracy for first and last sentences.

with B-TSort as the BerTSP-Approx variant can be uniformly applied across all the datasets under consideration, as opposed to BerTSP-Ensemble which is a mixture of two methods.

5.1 Longer Sequences

To compare the scalability of models, we present the results for samples with more than 10 sentences. NIPS, AAN, and NSF qualify for this analysis while SIND doesn't as all samples in SIND have exactly 5 sentences. Table 4 shows the results on all metrics. BerTSP is dominant over B-TSort. For NIPS, we obtain a 22% improvement in position-wise sentence accuracy and 14% in Kendall Tau as compared to 6% and 3% respectively for all samples showing that the relative improvement is more for longer sequences.

5.2 First and Last Sentences

Table 5 shows the accuracy of prediction of first and last sentences of the sequence. This analysis is important as the end-points are crucial positions of the sequence and the first prediction is often decisive to the prediction of the rest of the sequence. BerTSP clearly overtakes B-TSort across all the datasets. For the NSF dataset which primarily has longer sequences (mean length $\sim$ 9), it achieves improvements of 10% and 16% on the prediction of first and last sentences respectively showing its efficacy for longer sequences.

5.3 Sentence Displacement

We perform sentence displacement analysis where we find the percentage of sentences for which the predicted position is within a window W (forward or backward) of its position in the correct order. For instance, if the correct position is 5 and $W = 1$, the predicted position should be within 4 to 6 to be included in the percentage. Naturally, a larger window allows for more displacement and hence the percentage is higher compared to smaller windows. Table 6 shows the results for $W = 1, 2, 3$. BerTSP clearly outperforms B-TSort across all datasets and window sizes. The improvement is generally more pronounced for smaller window sizes. Note that this metric is essentially a generalization of position-wise Sentence Accuracy for which $W = 0$.

5.4 Qualitative Examples

We also present three examples from the SIND captions dataset where BerTSP improves on B-TSort. The predicted orders for each example by B-TSort and BerTSP are also shown (Table 7).

6 Related Work

In the recent past, numerous neural approaches have been proposed for Sentence Ordering. Chen et al. (2016) used a pairwise ranking model (Zheng et al., 2007) that assigns a score to the relative ordering of every pair of sentences. Prabhumoye et al. (2020) train a classifier to predict an order constraint between any two sentences and use sorting based on these constraints to predict the final order. Zhu et al. (2021) construct multiple constraint graphs which are integrated into sentence representations by Graph Isomorphism Networks and ranked via ListMLE. The use of a pointer decoder for sequential prediction (Gong et al., 2016; Logeswaran et al., 2018; Wang and Wan, 2019; Oh et al., 2019) along with an intermediate paragraph encoder (Cui et al., 2018; Yin et al., 2019, 2020; Cui et al., 2020) for better capturing the global dependencies has been proposed in many variants. Kumar et al. (2020) replace the pointer decoder with a feed-forward neural network and use ranking loss to enable simultaneous prediction of scores for all sentences.

The pairwise approaches generally suffer from lack of global interactions while pointer-based ap-

Dataset	SIND			NIPS			AAN			NSF		
Model	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**
B-TSort	79.44	93.20	98.69	85.19	93.35	97.02	87.64	92.82	97.77	49.29	64.55	74.59
BerTSP	81.28	94.37	99.11	86.08	94.35	97.72	88.81	95.78	98.19	49.57	64.62	74.48

Table 6: Results of displacement analysis of sentences on all datasets (W=Window size)

Example 1
s_1: We went to the park and a deer followed us.
s_2: We pet it and took pictures with it.
s_3: Then we traveled to the marvelous hotel.
s_4: The grounds were so immaculate.
s_5: We also got great pictures of the outside decor.
B-TSort: $s_2 \rightarrow s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$
Example 2
s_1: I went on vacation last year.
s_2: It was a beautiful place.
s_3: There were a lot of flower stores.
s_4: The buildings were very old.
s_5: There were a lot of other tourists there too.
B-TSort: $s_1 \rightarrow s_3 \rightarrow s_2 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
Example 3
s_1: There were many people at the protest.
s_2: They had many signs.
s_3: And flags as well.
s_4: They did not like the war.
s_5: And made this known before leaving.
B-TSort: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$

Table 7: Qualitative comparison of B-TSort & BerTSP

proaches lag in utilizing the local pairwise context. The more recent works try to overcome this by incorporating the local relative ordering information into the pointer decoder (Yin et al., 2020; Cui et al., 2020). Also, likelihood-based decoding is prone to degeneration (Holtzman et al., 2020) especially for longer sequences of sentences and paragraph encoders can only capture limited information from every sentence. Lastly, the large memory requirement is the underlying issue with common neural approaches. In particular, (Cui et al., 2020) show better results compared to our approach (BerTSP) but methods such as BerTSP are much more memory efficient (Prabhumoye et al., 2020).

We have proposed a Traveling Salesman Problem based formulation for sentence ordering. The use of such graph-based optimizations have been explored in past work in NLP such as ARM-to-text generation (Song et al., 2016), opinion summarization (Nishikawa et al., 2010), multi-document summarization (Al-Saleh and Menai, 2018), etc.

7 Conclusion

We demonstrate the potential of a simpler, cheaper yet effective approach for the task of Sentence Ordering. Our reformulation of the task as an asymmetric TSP allows for the application of exact and heuristic graphical algorithms which are lighter and more transparent as opposed to heavier neural approaches.

Future Work: In place of BERT, use of alternative model architectures like Albert, XLNet and BART may provide better sentence representations for sentence ordering as their language modeling objectives better align with this task. In the decoding part, more heuristic-based or neural-based approaches to combinatorial optimization may provide better alternatives.

Acknowledgments

We thank anonymous INLG reviewers for providing valuable feedback.

Ethics Statement

We do not see any major ethical concerns arising out of our work. Our work focuses only on finding the coherent ordering of sentences.

References

Asma Al-Saleh and Mohamed El Bachir Menai. 2018. Solving multi-document summarization as an orienteering problem. *Algorithms*, 11(7):96.

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.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

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

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, 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Youmna Farag, Helen Yannakoudakis, and Ted Briscoe. 2018. Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv preprint arXiv:1804.06898*.

Varun Gangal, Steven Y. Feng, Eduard H. Hovy, and Teruko Mitamura. 2021. NAREOR: the narrative reordering problem. *CoRR*, abs/2104.06669.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, and Dhruv Batra. et almbox. 2016. visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2020. Narrative text generation with a latent discrete plan. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3637–3650.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 137–150.

Pawan Kumar, Dhanajit Brahma, Harish Karnick, and Piyush Rai. 2020. Deep attentive ranking networks for learning to order sentences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8115–8122.

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.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Coling 2010: Posters*, pages 910–918.

Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo, and Kyong-Ho Lee. 2019. Topic-guided coherence modeling for sentence ordering by preserving global and local information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2273–2283, Hong Kong, China. 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*. Association for Computational Linguistics.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proscript: Partially ordered scripts generation via pre-trained language models. *CoRR*, abs/2104.08251.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers?

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. *arXiv preprint arXiv:1609.07451*.

Tianming Wang and Xiaojun Wan. 2019. Hierarchical attention networks for sentence ordering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7184–7191.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yongjing Yin, Fandong Meng, Jinsong Su, Yubin Ge, Lingeng Song, Jie Zhou, and Jiebo Luo. 2020. Enhancing pointer network for sentence ordering with pairwise ordering predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9482–9489.

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-19*, pages 5387–5393. International Joint Conferences on Artificial Intelligence Organization.

Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294.

Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural sentence ordering based on constraint graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14656–14664.

A Additional Method Details

A.1 Dummy Node for ATSP:

TSP requires the source to be pre-specified. Since it finds a cycle, the choice of the source s is irrelevant. But for our purpose, we need to find an order which is inherently not cyclic. To deal with this, we introduce an additional node in the graph, v_{n+1}, which is at 0 distance from all other vertices in either direction, i.e. $w_{(n+1)i} = w_{i(n+1)} = 0, \forall\ i$. This dummy node serves as the source from which the cycle begins and consequently terminates. Since it is at 0 distance from all nodes, its addition does not affect the minimization process.

A.2 Heuristic solution for TSP:

We propose a heuristic solution that is cheap and accounts for the order along with the cost. Consider the unaugmented weight matrix: $W = \{w_{ij}\}_{\forall i,j \in \{1,2,...,n\}}$ where $w_{ij} = 1 - P(s_i \rightarrow s_j)$. Consider the following sum of probabilities:

$$P(s_i \rightarrow s_2) + P(s_i \rightarrow s_3) + ... + P(s_i \rightarrow s_n)$$

If this sum is high, s_i has a high probability of occurring before $\{1, 2, ..., i-1, i+1, ..., n\}$ in the predicted sequence. Consequently, if we take the sum of complement probabilities, the lower the sum, the higher is the probability of s_i occurring before $\{1, 2, ..., i-1, i+1, ..., n\}$ in the predicted sequence. This complementary sum is nothing but the row-sum of row i of matrix W:

$$w_{i1} + w_{i1} + ... + w_{in}\ (w_{ii} = 0)$$

Hence, we take the row-sums of all rows in W and sort them in ascending order. This gives us the predicted order. Since it involves simple sorting, its runtime complexity is $O(nlogn)$.

B Implementation details

The code for the sentence-pair encoder and the evaluation metrics is derived from (Prabhumoye et al., 2020)[2]. The hyperparameters values are also taken from this work. The experiments are conducted on GeForce RTX 2080 Ti GPU. Our code[3] is written using Pytorch deep learning framework.

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

[3] https://github.com/vkeswani/BerTSP

Dataset	Mean Length	Train	Dev	Test
SIND	5	40155	4990	5055
NIPS	6	2448	409	402
AAN	5	8569	962	2626
NSF	9	96070	10185	21580

Table 8: Descriptive statistics of the four datasets considered

C Data

C.1 Access

The SIND captions dataset is available online[4]. Note that the Stories of Images-in-Sequence (SIS) portion is the one relevant to our task. The abstract datasets, NIPS, AAN, and NSF, were obtained from Logeswaran et al. (2018).

C.1.1 Descriptive Statistics

We present the mean length and the split of the four datasets into train, development, and test sets in table 8.

[4] https://visionandlanguage.net/VIST/dataset.html