SUPPLEMENTARY MATERIAL 1
Set Interdependence Transformer: Set-to-Sequence Neural Networks for Permutation Learning and Structure Prediction

Anonymous Author(s)

1 Introduction

This document constitutes the first of the two main supplementary sources for the paper. It contains complete information regarding each of the presented experimental results as well as a more detailed description of the synthetic structures dataset and its corresponding evaluation functions.

The other source of information is the code base enabling other researchers to reproduce the results. The code also contains full configuration files specifying the entirety of model architectures, layer dimensions and training hyperparameters. Full requirements regarding utilized libraries are provided to make it easier to recreate the original software environment. Hardware information can be found in the main paper. The reported Perfect Match Ratio (PMR) and Kendall's Rank Correlation Coefficient (τ) are scaled by a factor of a 100 for increased readability, following convention from Cui *et al.* (2018) and Yin *et al.* (2020).

2 Experiments

2.1 Travelling Salesman Problem

TSP is an NP-hard problem, which arises in many areas of theoretical computer science. Solutions to this problem comprise algorithms that find usage in DNA sequencing and microchip design [Vinyals *et al.*, 2015]. In this specific setting, the task is to take as input a set of n 2-dimensional points $\mathcal{X} \sim \mathbf{X} \in \mathbb{R}^{n\times 2}$, represented by their coordinates on a Euclidean plane, and output their predicted permutation $\mathbf{y}'_\pi \in \mathbb{N}^n$ such that the lines drawn between these points in the predicted order form the shortest possible circuit, exemplified by the target permutation $\mathbf{y}_\pi$. Point coordinates are sampled uniformly from the range of $[0, 1]$.

The proposed set-to-sequence model is compared with a number of joint set encoding and permutation learning methods, random solutions that consist of valid paths without repetitions and with the paths obtained via the Bellman–Held–Karp algorithm [Hansen and Krarup, 1974] as the target on which the models were trained. Three datasets were generated, each with a different cardinality $n \in (10, 15, 20)$ of the input sets of points. Models were only trained on the dataset consisting of sets of 10 elements, but their performance was also tested on the other cardinalities to gauge the ability to generalize to unseen lengths. The $n = 10$ training dataset consisted of 20K examples, the validation set of 5K and each test set, for all 3 cardinalities, consisted of 3K examples.

Method	TSP cardinality $n = 10$	$n = 15$*	$n = 20$*
Held–Karp	2.97	3.87	4.42
Random	4.48	7.32	8.96
DeepSets	3.27 ± 0.12	4.37 ± 0.19	5.12 ± 0.14
SetTrans	<u>3.00 ± 0.02</u>	3.99 ± 0.05	4.55 ± 0.08
AttSets	3.06 ± 0.06	4.02 ± 0.13	4.64 ± 0.19
RepSet	3.03 ± 0.05	4.13 ± 0.04	4.59 ± 0.11
PtrNet	3.18 ± 0.26	4.85 ± 0.69	5.83 ± 0.85
RPW	3.67 ± 0.10	4.92 ± 0.04	4.82 ± 0.11
ATTOrdNet	3.10 ± 0.06	4.53 ± 0.15	5.63 ± 0.39
PtrNet+	3.01 ± 0.04	<u>3.92 ± 0.06</u>	<u>4.52 ± 0.09</u>
Ours	**2.98 ± 0.01**	**3.90 ± 0.03**	**4.46 ± 0.06**

Table 1: Performance on the TSP in terms of average tour length.

Each tested model had approximately 5 million trainable parameters, corresponding dimensions of hidden layer weights and number of layers. For specific details we refer the reader to the provided code and configuration files. The proposed model consisted of 3 layers of SIT transformations over 4 attention heads. The AdamW [Loshchilov and Hutter, 2017] optimizer was used in all cases, with weight decay coefficient 1e-2, learning rate (α) 1e-4, dropout rate of 0.1 and batch size 32, for 50 epochs. The reported results are averaged over full training runs with reported standard deviation. In the case of TSP the only metric used is the average predicted tour length over the entire test set for each cardinality. As shown in table 1, our proposed model outperforms the baselines by predicting shorter average paths with lower standard deviation, the difference becoming more pronounced the more the input set cardinality differed from the ones seen during training. Specifically, the proposed model predicts paths that are shorter by 0.02 and 0.06 when $n = 15$ and $n = 20$ respectively, exhibiting increased ability to generalize to unseen lengths.

2.2 ROCStory

ROCStory is a popular sentence ordering dataset consisting of 98,162 common-sense stories with 50 words per story

Method	ROCStory		PROCAT		Synthetic $\|y\| = 30$	
	PMR	Kendall's τ	PMR	Kendall's τ	PMR	Kendall's τ
DeepSets (2017)	33.81 ± 4.55	62.41 ± 4.2	15.21 ± 2.69	35.15 ± 3.8	25.61 ± 1.90	40.94 ± 2.7
Set Transformer (2019)	41.94 ± 1.29	73.15 ± 1.9	21.03 ± 0.98	42.74 ± 2.6	30.23 ± 1.86	44.71 ± 2.9
AttSets (2020)	42.51 ± 1.45	74.81 ± 2.3	19.24 ± 1.30	38.44 ± 1.7	31.48 ± 2.04	46.20 ± 3.2
RepSet (2020)	42.47 ± 1.61	73.39 ± 1.7	22.72 ± 1.19	41.30 ± 2.5	34.84 ± 1.99	47.95 ± 3.4
PointerNet (2015)	28.73 ± 3.91	59.72 ± 6.2	02.90 ± 1.17	16.85 ± 3.4	17.33 ± 3.41	32.66 ± 3.1
Read-Process-Write (2016)	20.38 ± 7.22	51.78 ± 8.5	03.54 ± 1.06	21.11 ± 2.9	21.40 ± 2.07	36.28 ± 4.3
ATTOrderNet (2018)	41.14 ± 2.10	73.02 ± 2.0	17.33 ± 2.31	37.47 ± 3.2	28.02 ± 2.40	42.67 ± 3.4
Enhanced PointerNet+ (2020)	44.32 ± 1.25	76.43 ± 1.3	22.09 ± 1.57	42.53 ± 1.9	34.34 ± 1.39	48.14 ± 2.9
Ours (2021)	**47.00 ± 0.89**	**79.86 ± 0.9**	**25.61 ± 1.81**	**46.41 ± 1.3**	**37.16 ± 1.01**	**52.03 ± 2.2**

Table 2: Precise Match Ratio (PMR) and Kendall's Rank Correlation Coefficient (τ) results for ROCStory, PROCAT and the Synthetic task.

on average [Mostafazadeh and Chambers, 2016], publicly available at the following link: https://cs.rochester.edu/nlp/rocstories.

Each story contains exactly 5 sentences. Following common practice established by Wang and Wan (2019) and Yin *et al.* (2020), the provided dataset split is used to get the training, testing and validation datasets of 78K, 9.8K and 9.8K stories per set. The input is a set of five sentences in random order π, in English, with the canonical order $\mathbf{y}$ provided as the learning target. In our experiment, we use the concatenated and averaged output of the last 4 layers of the cased large version of BERT [Devlin *et al.*, 2019] to obtain fixed-length vector representations of each sentence. The language model is frozen during training to isolate the effect of different set encoding methods on the permutation task performance.

Each tested model had approximately 7 million trainable parameters, corresponding dimensions of hidden layer weights and number of layers. For specific details we refer the reader to the provided code and configuration files. The proposed model consisted of 5 layers of SIT transformations over 4 attention heads. The AdamW [Loshchilov and Hutter, 2017] optimizer was used, with weight decay coefficient 1e-2, learning rate (α) 1e-4, dropout rate of 0.1 and batch size 32, for 50 epochs. The reported results are averaged over 3 full training runs with reported standard deviation. The performance is reported in terms of the Perfect Match Ratio and the Kendall's τ Rank Correlation Coefficient, both of which are frequently used metrics for the sentence ordering challenge [Cui *et al.*, 2018; Wang and Wan, 2019; Yin *et al.*, 2020]. The overall results on the two main ROCStory and PROCAT datasets are shown in table 2. Our proposed model outperforms the state-of-the-art on both datasets. The PMR scores are increased by 2.68 and 2.89 and the τ by 3.43 and 3.67 over second best performances on the ROCStory and PROCAT datasets respectively.

2.3 PROCAT

PROCAT is a product catalog structure dataset consisting of over 1.5 million product offers composed into over 10,000 human-designed catalogs. It contains relative visual prominence information and the exact split of offers into complementary sections, which are then ordered into a full catalog [Jurewicz and Derczynski, 2021]. It is publicly available at the following link: https://doi.org/10.6084/m9.figshare.14709507.

Much like ROCStory, it lends itself to set-to-sequence challenges. A permutation learning objective is formed by the original order of offers and sections within a catalog. Each section can consist of a variable number of offers and each catalog of a variable number of sections, thus requiring greater flexibility than ROCStory. The input is formed by the text features of individual offers and their random order (along with section break markers) in the matrix representing the input set. The target is their actual order in the original catalog, in the proper section split defined by the section break markers. In our experiment, we used the concatenated and averaged output of the last 4 layers of the cased large version of BERT [Devlin *et al.*, 2019], publicly available under this link, to obtain fixed-length vector representations of each offer. The language model is frozen during training to isolate the effect of different set encoding methods on the permutation task performance. The split into training and test sets follows the one provided by the authors of the dataset (80/20).

Each tested model had approximately 7 million trainable parameters, corresponding dimensions of hidden layer weights and number of layers. For specific details we refer the reader to the provided code and configuration files. The proposed model consisted of 5 layers of SIT transformations over 4 attention heads. The AdamW [Loshchilov and Hutter, 2017] optimizer was used, with weight decay coefficient 1e-2, learning rate (α) 1e-4, dropout rate of 0.1 and batch size 32, for 50 epochs. The reported results are averaged over 3 full training runs with reported standard deviation. The performance is reported in terms of the Perfect Match Ratio and the Kendall's τ Rank Correlation Coefficient, both of which are frequently used metrics for permutation learning challenges [Cui *et al.*, 2018; Wang and Wan, 2019; Yin *et al.*, 2020]. The overall results are shown jointly in table 2. The proposed model outperforms the state-of-the-art on the PROCAT dataset. The PMR score is increased by 2.89 and the τ by 3.67 over second-best tested performance.

2.4 Formal Grammars

Formal grammars are a well-studied concept from the field of formal language theory, lending itself to the set-to-sequence framing as a machine learning task [Nakamura and Imada, 2011]. A grammar describes how to form strings from a language's alphabet that are valid according to this language's syntax. In our experiments we tackle two context-sensitive and one context-free grammar, as originally categorized by Chomsky (1956). The task is to take a set of terminal symbols that can be composed into a grammatical string, or in other words ordered into their original index sequence $\mathbf{y}$ which was shuffled to obtain the randomly ordered input set matrix $\mathbf{X}_\pi$.

We generate data from the first grammar, $a^n b^n c^n$, by sampling n uniformly from [1,100]. For the second grammar, $a^n b^k c^{nk}$, both n and k are sampled uniformly from [1,25], with the maximum sequence length being 675. For the Van Dyck we only form sequences consisting of 4 terminal symbols:'{', '}', '(' and ')', of random length sampled uniformly from [4,50]. The Van Dyck language requires that each pair of parenthesis and square brackets is properly closed. Each training set consisted of 5K randomly shuffled arrays representing $\mathbf{X}_\pi$, each test and validation set consisted of 1K examples.

Each tested model had approximately 5 million trainable parameters, corresponding dimensions of hidden layer weights and number of layers. For specific details we refer the reader to the provided code and configuration files. The progressive masking is removed from the permutation modules for this task. The proposed model consisted of 3 layers of SIT transformations over 4 attention heads. The AdamW [Loshchilov and Hutter, 2017] optimizer was used, with weight decay coefficient 1e-2, learning rate (α) 1e-4, dropout rate of 0.1 and batch size 32, for 50 epochs. The reported results are averaged over 3 full training runs with reported standard deviation. The performance is reported in terms of the percentage of predicted grammatical sequences.

In table 3 we present the average percentage of predicted sequences that adhered to the rules underlying two context sensitive grammars and one context free grammar, in the form of the Van Dyck language. The simplest grammar ($a^n b^n c^n$) was able to be fully learned by most models utilizing the enhanced pointer network as their permutation module, our proposed method among them. With regards to the more challenging $a^n b^k c^{nk}$ context-sensitive grammar our method outperformed the second best by 1.24 percentage points and by 1.56 in the case of the Van Dyck language.

2.5 Synthetic Structures

The synthetic structure datasets are generated via an expanded version of the library provided by Jurewicz and Derczynski (2021), which is publicly available under the following link: https://github.com/mateuszjurewicz/procat.

Each 1-dimensional structure is generated through a set of customizable rulesets, which define how the atomic tokens that comprise each structure can be ordered. In the reported experiments we generate structures consisting of 30 instances of the 5 atomic token types, represented by five colours: blue, yellow, red, green and purple. And additional token type represents section breaks.

	Grammars % valid		
Method	$a^n b^n c^n$	$a^n b^k c^{nk}$	Van Dyck
DeepSets	97.17 ± 2.9	94.52 ± 2.4	79.84 ± 2.3
SetTrans	**100.0 ± 0.0**	96.62 ± 0.8	92.16 ± 1.5
AttSets	98.24 ± 1.3	96.12 ± 1.1	92.31 ± 1.2
RepSet	**100.0 ± 0.0**	97.64 ± 0.9	93.22 ± 1.4
PtrNet	79.31 ± 5.3	75.85 ± 4.7	58.85 ± 6.7
RPW	86.41 ± 1.5	81.74 ± 1.2	61.13 ± 5.3
ATTOrdNet	97.75 ± 1.1	94.17 ± 1.0	84.31 ± 3.8
PtrNet+	**100.0 ± 0.0**	96.82 ± 0.7	93.51 ± 1.2
Ours	**100.0 ± 0.0**	**98.88 ± 0.6**	**95.07 ± 1.0**

Table 3: Results for 2 context-sensitive and 1 context-free grammar, the Van Dyck language, consisting of {} and () pairs. On a scale of 0-100, reflecting the proportion of predicted grammatical sequences.

	n-th Order Relation Ruleset		
Method	$n = 3$	$n = 4$	$n = 5$
DeepSets	53.37 ± 6.1	40.12 ± 4.0	21.13 ± 0.2
SetTrans (4)	92.41 ± 1.6	92.90 ± 1.4	91.74 ± 1.7
SetTrans (5)	94.66 ± 1.9	93.38 ± 1.3	92.93 ± 1.5
AttSets	93.84 ± 2.8	92.93 ± 1.1	86.44 ± 1.3
RepSet	91.29 ± 3.4	90.84 ± 2.4	89.73 ± 2.9
PtrNet	41.26 ± 5.3	31.96 ± 4.8	15.23 ± 4.2
RPW	45.11 ± 2.0	36.31 ± 1.6	16.12 ± 2.5
ATTOrdNet	82.31 ± 4.4	67.07 ± 2.0	0.12 ± 1.4
PtrNet+	89.58 ± 3.9	87.22 ± 3.2	86.94 ± 2.8
Ours (2 layers)	93.83 ± 3.6	89.01 ± 2.6	86.79 ± 1.9
Ours (3 layers)	**98.73 ± 0.8**	93.44 ± 2.3	92.72 ± 1.8
Ours (4 layers)	98.48 ± 1.2	**97.52 ± 0.9**	**96.10 ± 1.6**

Table 4: Synthetic structure prediction scores per ruleset type, split by order of interaction required. On a scale of 0-100, reflecting the proportion of valid predicted structures with regards to each ruleset.

Rulesets define the valid composition of sections and their order. An example of a valid section composition might be one that consists only of 3 blue, yellow or red tokens. An example of a valid section order might be always starting the sequential structure with an all-red section or always ending it with an all-blue one. These are intended to represent a more abstract simplification of the rules that govern actual product catalog structures.

Which ruleset should be applied when predicting a synthetic structure depends on higher order relational interactions between the elements of the input set of tokens. A pairwise interaction rule would trigger if the input set contains a blue and a yellow token. A third-order interaction rule, by comparison, would have to depend on the presence or absence of 3 atomic token types and so forth. A real life example of a 3rd order interaction between product offers comes in the form of an input set including beef and French cheese offers. With just the pairwise interaction between the two, they don't necessarily form a good pairing to go together in the same

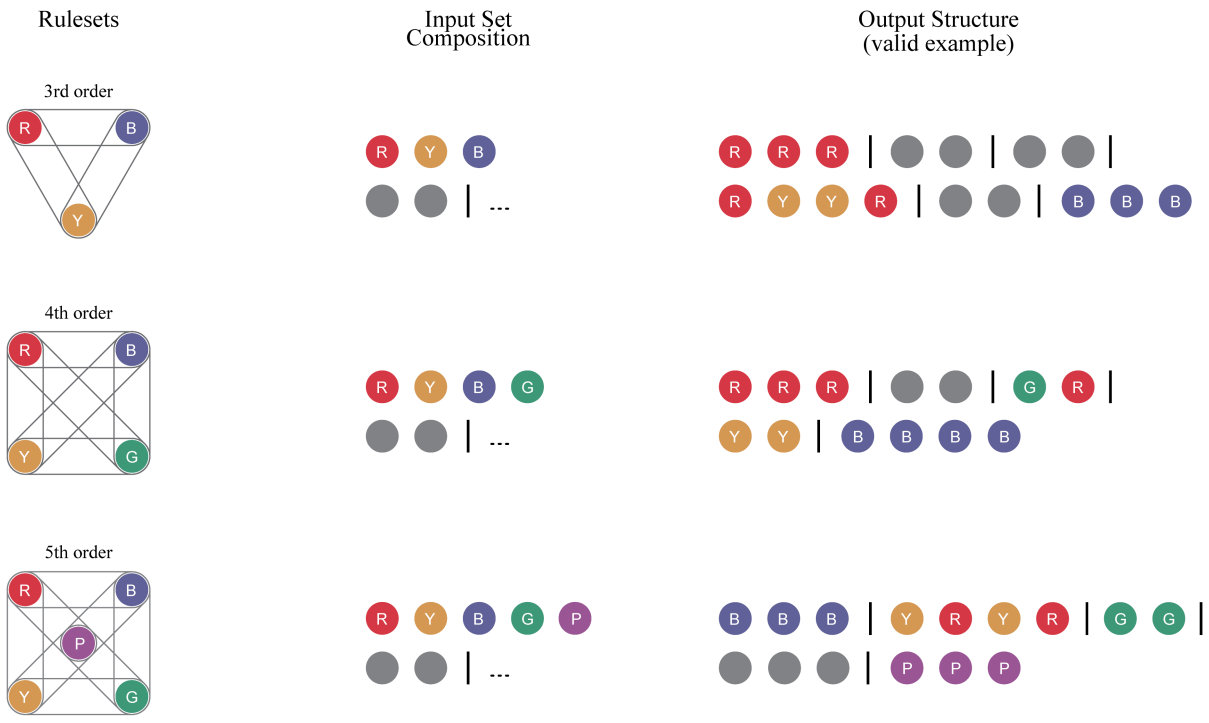

Figure 1: Synthetic structures and n-th order interaction rulesets. Rulesets (left) represent higher-order interactions that determine the validity of an output structure. Which ruleset is applicable is dependent on the input set composition (middle). For example, if a red, yellow and blue token are present in the input set, the valid output structure (right) has to start with an all-red section, end on an all-blue section and can only combine red and yellow tokens together in a 50/50 split. Inclusion of a 4th special token (second row) introduces a 4th-order interaction which changes the underlying ruleset and, in consequence, the target output structure.

section in a complementary way. However, if an offer for red wine is also present in the input set, the 3rd order interaction between these three products ties them together into a popular catalog section.

Given a predicted structure, based on a specific input set, we can easily check whether it is valid in relation to the appropriate ruleset. Thus we can report what percentage of predicted structures were valid as per rulesets requiring the model to learn n-th order interactions. This allows us to approximate a model's relational reasoning capability. For a visual explanation see figure 1.

Reported results are split between tables 2 and 4. In the former, we compare the PMR and τ on a synthetically generated structures dataset of length 30, following the default rulesets, where our model outperformed the best benchmarks by 2.32 and 1.89 respectively. In all reported cases the training dataset consists of 10K examples generated via the default rulesets including 3rd, 4th and 5th order interactions, and the test dataset consists of 5K examples.

The results presented in table 4 require further explanation. In order to assess the proposed model's ability to learn n-th order relational rules without performing n pairwise attention operations through a stack of n attention layers, we construct versions of our set-to-sequence model with $< n$ set encoding layers. Specifically, the tested versions of the proposed architecture consist of a single permutation equivariant, transformer-style layer for obtaining element representations followed by m SIT layers. Here, $m \in (1, \ldots, n-2)$ so that the proposed models always consists of at least one less layers $(m+1)$ than the number (n) required assuming only pairwise interactions per layer.

For specific dimensions of hidden layers we refer the reader to the provided code and configuration files. The proposed models consisted of m layers of SIT transformations over 4 attention heads. The AdamW [Loshchilov and Hutter, 2017] optimizer was used, with weight decay coefficient 1e-2, learning rate (α) 1e-4, dropout rate of 0.1 and batch size 32, for 50 epochs. The reported results are averaged over 3 full training runs with reported standard deviation.

Table 4 shows the percentage of predicted synthetic structures that were valid under the relevant ruleset. Each column refers to rulesets depending on n-th order relational interactions, where $n \in (3, 4, 5)$. The baselines in this case are specifically versions of the original models consisting of exactly 5 layers (with one exception) responsible for pairwise attention transformations. These are contrasted with three versions of the proposed model, with 1, 2 or 3 layers of the proposed SIT set encoder, plus 1 layer of the initial permutation equivariant attention, for a total of 2, 3 and 4 layers (see bottom rows of table 4). Thus the proposed models are always at a disadvantage in terms of the number of stacked attention operations, requiring the SIT encoder to learn higher

than pairwise interactions.

As seen in table 4, the 3-layer version of the proposed model has the highest score on the 3rd order ruleset by a margin of 4.07 compared to the best benchmark. However, even the 2-layer version outperforms all but two benchmark methods, each consisting of 5 stacked layers. On the 4th order ruleset the best result is obtained through the proposed model's 4-layer version, but the second best performance is attained by the 3-layer version, showcasing its ability to learn higher-than-pairwise interactions in a single SIT layer. Similarly on the 5th order ruleset ($n = 5$ in the table) the 4-layer version outperforms the best benchmark by 3.07 percentage points, with a 5-layer Set Transformer followed by the same permutation module as our proposed set-to-sequence model achieving second best performance. As an ablation study, we additionally include results for a 4-layer version of the Set Transformer (4), where the only difference between our 4-layer set-to-sequence model is the removal of the SIT matrix augmentation. Its presence accounts for a 4.36 performance increase.

References

[Chomsky, 1956] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[Cui *et al.*, 2018] Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. Deep attentive sentence ordering network. In *Proc. EMNLP*, pages 4340–4349, 2018.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, 2019.

[Hansen and Krarup, 1974] Keld Helbig Hansen and Jakob Krarup. Improvements of the held—karp algorithm for the symmetric traveling-salesman problem. *Mathematical Programming*, 7(1):87–96, 1974.

[Jurewicz and Derczynski, 2021] Mateusz Jurewicz and Leon Derczynski. Procat: Product catalogue dataset for implicit clustering. *Proc. NeurIPS: Track on Datasets and Benchmarks*, 35, 2021.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. In *CoRR, abs/1711.05101, 2017.*, 2017.

[Mostafazadeh and Chambers, 2016] Nasrin Mostafazadeh and Nathan Chambers. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proc. NAACL*, pages 839–849, 2016.

[Nakamura and Imada, 2011] Katsuhiko Nakamura and Keita Imada. Towards incremental learning of mildly context-sensitive grammars. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 1, pages 223–228. IEEE, 2011.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proc. NeurIPS*, volume 2015-January, pages 2692–2700, 2015.

[Wang and Wan, 2019] Tianming Wang and Xiaojun Wan. Hierarchical attention networks for sentence ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7184–7191, 2019.

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