Towards Domain-Independent Text Structuring Trainable on Large Discourse Treebanks

Grigorii Guz and Giuseppe Carenini
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{gguz, carenini}@cs.ubc.ca

Abstract

Text structuring is a fundamental step in NLG, especially when generating multi-sentential text. With the goal of fostering more general and data-driven approaches to text structuring, we propose the new and domain-independent NLG task of structuring and ordering a (possibly large) set of EDUs. We then present a solution for this task that combines neural dependency tree induction with pointer networks and can be trained on large discourse treebanks that have only recently become available. Further, we propose a new evaluation metric that is arguably more suitable for our new task compared to existing content ordering metrics. Finally, we empirically show that our approach outperforms competitive alternatives on the proposed measure and is equivalent in performance with respect to previously established measures.

1 Introduction

Natural Language Generation (NLG) plays a fundamental role in data-to-text tasks like automatically producing soccer, weather and financial reports (Chen and Mooney, 2008; Plachouras et al., 2016; Balakrishnan et al., 2019), as well as in text-to-text generation tasks like summarization (Nenkova and McKeown, 2012).

Generally speaking, NLG involves three key steps (Gatt and Krahmer, 2017): first there is content determination which selects what information units should be conveyed, secondly there is text structuring, which is responsible for properly structuring and ordering those units; and finally microplanning-realization that aggregates information units into sentences and paragraphs that are then verbalized.

The focus of this paper is on the text structuring step, which is critical for the overall performance of an NLG system, as it ensures that the communicative goals of the text are realized in the most structurally coherent and cohesive way possible, making the main ideas expressed by the text easy to follow for the target audience.

Aiming to develop very general computational methods for text structuring, we keep our study independent from particular ways in which the input information units are represented and from explicitly provided ordering constraints for the target application domain (Gatt and Krahmer, 2017). More specifically, we propose and attack, in a fully data-driven way, the novel and domain-independent task of simultaneously structuring and ordering a set of Elementary Discourse Units (EDUs), i.e., clause-like text fragments that the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) assumes to be the building blocks of any discourse structure (see Figure 1(a)(left)). In other words, we assume that the system is given a set of EDUs (with cardinality possibly > 100) as input and returns their ordering, as well as the unlabelled RST dependency discourse tree structure for a document consisting of this set of EDUs, as illustrated in Figure 1(a).

Our data-driven approach relies on the very recent availability of large treebanks containing hundreds of thousands of (silver-standard) discourse trees that can be automatically generated by distant supervision following the approach presented by Huber and Carenini (2020). We formulate the problem as one of the dependency tree induction, repurposing existing solutions (Ma and Hovy, 2017; Vinyals et al., 2015) to perform an RST-based text structuring where both EDU ordering and tree building are executed simultaneously (Reiter and Dale, 2000). The resulting structures can be highly useful for subsequent NLG pipeline stages such as aggregation, and for downstream tasks like text simplification (Zhong et al., 2019). Our approach is trainable end-to-end, but since the discourse trees in the training treebank are constituency trees (see

Figure 1(b)), we face the additional challenge of turning them into dependency trees (see Figure 1(a)) before the learning process can start (Hayashi et al., 2016).

In a comprehensive evaluation, we compare our solution to three baselines along with a competitive approach based on pointer networks (Vinyals et al., 2015), which is the established method of choice not only for sentence ordering (Cui et al., 2018), but also for basic domain-specific text structuring in data-to-text applications (Puduppully et al., 2019). In particular, the comparison involves training and testing the different models on the MEGA-DT treebank (Huber and Carenini, 2020), containing ≈250,000 discourse trees obtained by distant supervision from a the Yelp'13 corpus of customer reviews (Tang et al., 2015).

With respect to evaluation metrics, we found the current ways of measuring content ordering (e.g., Kendall's τ) to be inadequate to capture the quality of long sequences of relatively short information units (i.e., sequences of EDUs of long multi-sentential text). Thus, we propose a novel evaluation measure, Blocked Kendall's τ, that we argue should be used for our new NLG task of ordering and structuring a possibly large set of EDUs, because it critically measures how well semantically close units are clustered together in the correct order.

To summarize the contributions of this paper: **(i)** we propose the new and domain-independent NLG task involving the structuring and ordering a set of EDUs, which is intended to enable more general and data-driven approaches to text structuring; **(ii)** we present a strong benchmark solution for this task, trainable on large discourse treebank, that combines neural dependency tree induction with pointer networks; **(iii)** we propose a new evaluation metric that is arguably much more suitable for this task than existing ordering metrics; **(iv)** and on this new metric along with standard tree-quality metrics, we show empirically that our approach outperforms or is comparable to competitive alternatives. The code for our solution and the new metric, as well as the treebank for training, is publicly available.[1]

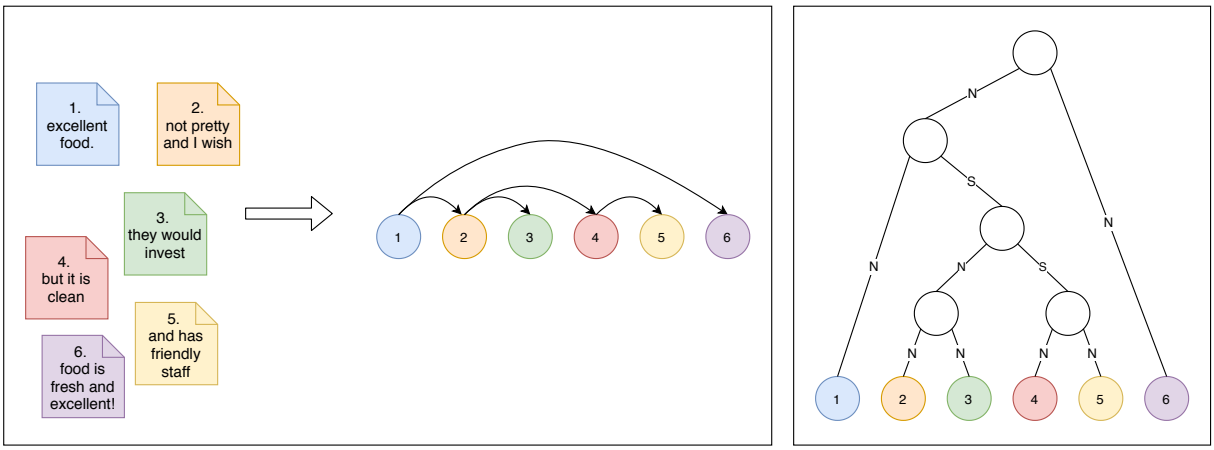
[1] http://www.cs.ubc.ca/cs-research/lci/research-groups/natural-language-processing/index.html

2 Related Work

(a) Text structuring is a key step in NLG, especially when generating long multi-sentential documents. Not surprisingly, this is also the case in recent neural approaches. Wiseman et al. (2017) presented the RotoWire corpus, targeting long-document data-to-text NLG. To generate the document, their model conditions on all records in the data table by weighting their embeddings with attention, in addition to using copying mechanism for out-of-vocabulary data entries. The follow-up work of Puduppully et al. (2019), instead of conditioning on all records, arguably performs better text structuring by first selecting and then ordering the entries of a data table using Pointer network architecture (Vinyals et al., 2015). That way, the surface realization module considers previously generated text and only one new data table entry at a time. Their model was extended by Iso et al. (2019), with an additional GRU for tracking the entities that the model already referred to in the past. Pursuing a rather different approach to improve text structuring, Shao et al. (2019) proposed a hierarchical latent-variable model where the problem is decomposed into dependent sub-tasks, aggregating groups of data table entries into sentences first and then generating the sentences sequentially, conditioned on the plan and already generated sentences. Overall, these last three models significantly outperform the initial approach of Wiseman et al. (2017) both in terms of fluency and coverage, with increasing sophistication of the text structuring module yielding bigger gains, confirming that text structuring is indeed crucial for generating coherent long documents.

The task we propose and investigate in this paper can be seen as pushing this line of research even further. We aim for a more ambitious text structuring module inspired by traditional NLG work, viewing the process as the construction of an RST discourse tree for the target document (Reiter and Dale, 2000), which critically includes assigning importance to each constituent. Tellingly, our task is also domain-independent and agnostic on the representation of the input information units.

(b) The goal of **sentence ordering** is to sort a given set of unordered sentences into a maximally coherent document. Most recent work on sentence ordering (Logeswaran et al., 2016; Cui et al., 2018; Wang and Wan, 2019) involves constructing contextualized order-agnostic representations of indi-



(a) (b)

Figure 1: (a) A simple example of the novel NLG task we propose in this paper: generating an ordered discourse dependency tree (right) for a given set of EDUs (left). (b) The constituency discourse tree corresponding to the dependency tree shown in (a). The RST-style discourse trees in the treebanks we use for our experiments are initially represented as constituency trees.

vidual sentences and full documents using architectures such as Transformer Encoder without positional embeddings (Vaswani et al., 2017), and then feeding those representations into a pointer-based decoder (Vinyals et al., 2015).

The new task we propose in this paper is similar, but more challenging than sentence ordering. Instead of ordering sentences, we need to order EDUs, which are often shorter sentence constituents, and therefore by expressing smaller semantic units they arguably require more fine-grained processing. Furthermore, our task goes beyond ordering by also requiring the synergistic and simultaneous step of generating the RST discourse structure for the EDUs. To address these challenges, more powerful techniques for tree induction are needed on top of pointer networks.

(c) Document discourse tree structure induction: The third related line of research involves the induction of latent tree structures over documents. Some of these works aim at obtaining better document representations for tasks such as text classification (Karimi and Tang, 2019) and single-document extractive summarization (Liu et al., 2019). In essence, a neural framework is designed so that a discourse tree for a document is induced while training on the target downstream task. However, even if these approaches demonstrated improvements over non-tree-based models, subsequent studies have shown that the resulting latent discourse dependency trees are often trivial and too shallow (Ferracane et al., 2019). In contrast, recent work on distant supervision from sentiment (Huber and Carenini, 2020) indicates that large treebanks of discourse trees can be generated by combining neural multiple-instance learning (Angelidis and Lapata, 2018) with a CKY-inspired algorithm (Jurafsky and Martin, 2014). Since a series of experiments in inter-domain discourse parsing have certified the high-quality of these treebanks, we use one of such treebaks, called MEGA-DT, for training and testing our data-driven text structuring approach.

3 Novel Task and Methods

Our novel task for text structuring receives as input a set of n EDUs and returns both an ordering and a discourse structure for that set. We first describe how the EDUs are encoded, as this is the initial step for all the approaches we consider. Then, after discussing a basic method for just ordering the input EDUs (Pointer Networks), which will serve as our main baseline, we present our solution for fully solving the task in detail, which combines tree induction with pointer networks. We will refer to our final approach as *DepStructurer*. We conclude the section with two simple baselines for EDU ordering and structuring, respectively.

3.1 EDU Encoder

For a clear comparison of tree vs. non-tree based approaches, we encode EDUs in a very similar way to previous sentence ordering works (Cui et al., 2018; Wang and Wan, 2019). Given a document

with n EDUs $e_{1:n}$, with each EDU e_i containing a list of m_i words $w_{1:m_i}$, the final output of the EDU encoder is a set $v_{1:n}$, $v_i \in \mathbb{R}^d$ of embeddings of its EDUs. First, using the base version of the ALBERT language model (Lan et al., 2020), we construct individual EDU embeddings $b_i \in \mathbb{R}^{768}$ as the means of EDU word embeddings $\hat{w}_{1:m_i}$ from the last layer of ALBERT:

$$b_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \hat{w}_j \quad (1)$$

This language model was chosen because it uses a sentence-ordering objective during pre-training, see Lan et al. (2020). The EDU embeddings are then fed into a Transformer Encoder (Vaswani et al., 2017) without positional embeddings, yielding contextualized EDU representations $v_{1:n}$:

$$v_{1:n} = \text{TransformerEncoder}(b_{1:n}) \quad (2)$$

As Cui et al. (2018), we compute the final document representation z by averaging the document's EDU embeddings $v_{1:n}$.

3.2 Predicting Order Only: Pointer Networks

Pointer networks are commonly used for sentence ordering tasks (Cui et al., 2018) and have been recently applied to basic text structuring in data-to-text NLG (Puduppully et al., 2019). We train a pointer network to maximize the probability of correct ordering o^s of an unordered set of EDUs $v_{1:n}$ as a sequence prediction:

$$P(o^s|v_{1:n}) = \prod_{i=1}^{n} P(o_i^s|o_{i-1}^s, ..., o_1^s, v_{1:n}) \quad (3)$$

Here, each term in the product of probabilities is computed as:

$$h_j, c_j = LSTM(h_{j-1}, c_{j-1}, v_{i-1}) \quad (4)$$

$$u_i^j = k^T tanh(W_1 v_i + W_2 h_j) \quad (5)$$

$$p(o_i|o_{i-1}, .., o_1, s) = softmax(u_i) \quad (6)$$

where $k \in \mathbb{R}^d$ and $W_1, W_2 \in \mathbb{R}^{d \times d}$ are learnable parameters and $i, j \in (1, ..., n)$ index into input and output sequences respectively. Similarly to (Vinyals et al., 2015), we use the document embedding vector z as the initial hidden state and a vector of zeros as the first input to the pointer network. More specifically, during training, for each document s in our dataset D we feed in the EDU embeddings v_i according to the gold document order o^s and maximize the probability according to

$$\theta^* = \arg\max_{\theta} \sum_{s \in D} \log p(o^*|s, \theta) \quad (7)$$

During inference, since an exhaustive search over the most likely ordering is intractable, we use beam search for finding a suboptimal solution.

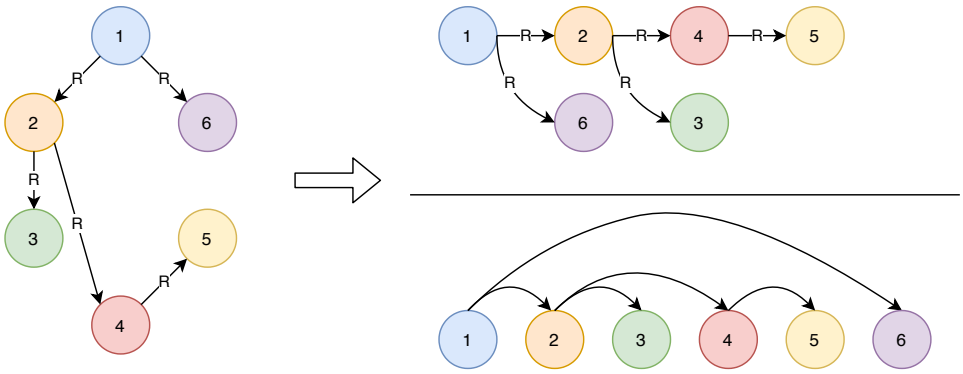
3.3 Performing the whole task: Our DepStructurer

The first design choice in addressing the task of simultaneously structuring and ordering a set of EDUs is whether the system should learn how to build dependency or constituency discourse trees (see Figure 1 (a)-(b) for corresponding examples). We decided to aim for dependency discourse structures for two key reasons. Not only have they been shown to be more general and expressive (Morey et al., 2018), but there are also readily available graph-based methods for learning and inference of dependency trees (Ma and Hovy, 2017) that when properly combined enable structure and ordering prediction to benefit from each other. However, since the only large-scale discourse treebank for training (MEGA-DT) contains constituency trees, we first convert them into dependency ones. For this, we follow the protocol of (Hayashi et al., 2016), which effectively resolves the ambiguity involved in converting multinuclear constituency units. Notice that we want dependency trees that fully specify the ordering for the EDUs, so our translation algorithm also labels each dependency arc with label - L or R, denoting whether the modifier node should be on the left (L) or on the right (R) of the head node in the linearized EDU sequence.

Once the training data is generated as a dependency treebank, our two-step solution for the task of structuring and ordering a set of EDUs can be applied. Notice that the same EDU embeddings $v_{1:n}$ are reused in both steps - for tree induction (Step 1 §3.3.1) and child ordering (Step 2 §3.3.2). These embeddings are generated by training a single EDU Encoder as described in §3.1.

3.3.1 Step 1: Compatibility Matrix and Initial Tree Induction

The first step of our solution learns how to build a discourse dependency tree for the input sequence of EDU embeddings $v_{1:n}$. Formally, this can be framed as learning a compatibility matrix (edge



a) Output of Step 1 b) Output of Step 2

Figure 2: Outputs of the two inference steps: (a) Initially induced Dependency Tree and (b) Final total ordering.

score tensor more precisely) $M \in \mathbb{R}^{n \times n \times 2}$, where the last dimension of l an entry i, j corresponds to the scores of the labels L and R for the edge from node i to node j. Similarly to (Ma and Hovy, 2017), each entry is computed as follows:

$$M_{i,j} = v_i^T W_1 v_j + W_2 v_i + W_3 v_j + b \quad (8)$$

where $W_1 \in \mathbb{R}^{d \times d \times 2}$, $W_2 \in \mathbb{R}^{d \times 2}$ and $W_3 \in \mathbb{R}^{d \times 2}, b \in \mathbb{R}^2$ are learnable bilinear, linear and bias terms. Once the tensor M is predicted, it is used for inferring an initial dependency structure.

Learning M: The objective is to maximize the probability of the correct tree structure $\mathbf{y}$:

$$P(\mathbf{y}|e_{1:n}, \theta) = \frac{\exp\left\{\sum_{(v_i,v_j,l)\in\mathbf{y}} M_{i,j,l}\right\}}{Z(e_{1:n}, \theta)} \quad (9)$$

where

$$Z(e_{1:n}, \theta) = \sum_{\mathbf{y}\in T(e_{1:n})} \exp\left\{\sum_{(v_i,v_j,l)\in\mathbf{y}} M_{i,j,l}\right\} \quad (10)$$

with $T(e_{1:n})$ denoting all possible trees from a set of EDUs $e_{1:n}$. Since the number $|T(e_{1:n})|$ of possible trees grows exponentially with the number of EDUs, we need an efficient way of computing $Z(e_{1:n}, \theta)$. Following (Koo et al., 2007), we achieve this goal by first computing the weighted adjacency matrix $A(M) \in \mathbb{R}^{n \times n \times 2}$ for left-child and right-child edges:

$$A_{i,j,l} = \begin{cases} 0, & \text{if } i = j \\ \exp\{M_{i,j,l}\} & \text{otherwise} \end{cases} \quad (11)$$

as well as the root scores for each node:

$$r_i(v) = \exp\{MLP(v_i)\} \quad (12)$$

Then, the weight of the correct dependency structure $\mathbf{y}$ is defined as

$$\psi(\mathbf{y}, \theta) = r_{root(\mathbf{y})}(v) \prod_{i,j,l\in\mathbf{y}} A_{i,j,l} \quad (13)$$

where $root(\mathbf{y})$ is the child of the root node in the dependency tree. We then compute the Laplacian matrix L of G:

$$L_{i,j} = \begin{cases} \sum_{i'=1}^{n} \sum_{l=1}^{2} A_{i',j,l}, & \text{if } i = j \\ \sum_{l=1}^{2} -A_{i,j,l} & \text{otherwise} \end{cases} \quad (14)$$

and replace its first row by $r(v)$:

$$\hat{L}_{i,j} = \begin{cases} r_i(v), & \text{if } i = 1 \\ L_{i,j} & \text{otherwise} \end{cases} \quad (15)$$

It can be shown (Koo et al., 2007) that the determinant of $\hat{L}$ is in fact equal to the normalizing constant that we need:

$$Z(e_{1:n}, \theta) = |\hat{L}| \quad (16)$$

which takes $O(n^3)$ time to compute. Hence, the loss for tree construction derived from eq. 9 can be computed efficiently:

$$l_{tree}(\theta) = -\log \psi(\mathbf{y}, \theta) + \log Z(e_{1:n}, \theta) \quad (17)$$

Inference of the initial tree structure: The learned model is applied to the input sequence of EDU embeddings $v_{1:n}$. Then, using the predicted compatibility matrix M, the highest-weighting tree structure can be constructed by the Chu-Liu-Edmonds algorithm (Edmonds, 1967), with the root being the node with highest root score r_i (eq. 12). Figure 2 (a) shows a sample output of this process.

3.3.2 Step 2: Ordering Children

The key limitation of Step 1 is that some nodes in the resulting dependency tree can have multiple left or right children, which makes their relative order unrecoverable from the basic tree structure. For instance, this is the case for nodes 1 and 2 in Figure 2 (a), both of which have two left children (outgoing edges labeled by L). To address this issue, in Step 2 for every node $s_i \in s_{1:n}$ that has $k > 1$ left or right children $s_{i_1}, ..., s_{i_k}$, we train a pointer network that predicts the correct order of children on each side - in the same way as described in § 3.2, but specifically trained on groups of children in MEGA-DT. Figure 2 (b)(top) illustrates an output of the Pointer network applied to plain dependency structure in Figure 2 (a), from which the final EDU ordering 2 (b)(bottom) is decoded as follows.

Algorithm 1: PredictEduOrder

```
   Data: Root
   Result: The ordering of elements of V
1  ordering = []
2  ordChildren = PtrNet(Root.leftChildren)
3  for child in ordChildren do
4  |   ordering.extend(PredictEduOrder(child))
5  end
6  ordering.append(Root)
7  ordChildren = PtrNet(Root.rightChildren)
8  for child in ordChildren do
9  |   ordering.extend(PredictEduOrder(child))
10 end
```

Inference of final ordering: The pseudocode for predicting the final ordering is provided in Algorithm 1. The ordering is built recursively bottom-up - at each step, given the ordering of all left and right subtrees (recursive calls in lines 4, 9), the ordering is obtained by concatenating, in the order predicted by Pointer network (lines 2, 7), the orderings of those subtrees, together with the current root node (line 6). Specifically, the children are ordered according to their root node; for example in Figure 2(b)(top), when deciding the order for child subtrees rooted at nodes 2,6 for the node 1, the pointer network orders them using the embeddings for those nodes.

3.4 Baselines for Ordering and Full Task

Language model decoding (LMD): greedily predicts the linear EDU ordering. The next EDU at each timestep is the one maximizing the length normalized language modelling objective from ALBERT.

Unsupervised tree induction (UTI): computes the compatibility matrix M using cosine similarity between the means of ALBERT embeddings for each EDU. The label for dependency (left vs. right child) is chosen randomly, while dependent orders for nodes with multiple children are chosen according to above ordering baseline LMD.

Tree Induction (TI+LMD): being an ablation for our main model, this baseline only learns to induce the tree structure in the same way as DepStructurer, but orders the children as in LMD, without performing supervised leaf ordering.

4 Experiments

4.1 The MEGA-DT Dataset

Our evaluation relies on MEGA-DT, a discourse treebank generated by distant supervision from the Yelp'13 corpus of customer reviews (Tang et al., 2015), according to the method presented by Huber and Carenini (2020). The high-quality of MEGA-DT trees has been certified in experiments in inter-domain discourse parsing similar to the ones described in (Huber and Carenini, 2020). In practice, their approach for generating the discourse trees for a set of documents can be applied to any other genre. If the required sentiment annotation is not naturally available (like star ratings for customer reviews), it can be obtained from an off-the-shelf sentiment analyzer. We train all models on 100k and 215k subsets of MEGA-DT, and use 7.5k documents for development and 15k for testing. Due to memory requirements induced by finetuning ALBERT, the training splits only contain documents with less than 35 EDUs, whereas to evaluate the performance on longer documents, the development and test sets contain respectively 2.5k and 5k of longer documents. The project GitHub repository provides the exact splits.

4.2 Evaluation Metrics

In all experiments, we assess the quality of the EDUs ordering and of their tree structure independently with two sets of corresponding metrics.

4.2.1 Information Ordering Metrics

Measuring the quality of information ordering is a challenging task because different metrics can be more or less appropriate depending on the num-

ber and the nature/granularity of the information units that are ordered. In accord with previous works, we first consider a set of simple metrics that essentially penalize the distance of an information unit from its correct position. These include Kendall's τ, Position Accuracy (POS) and Perfect Match Ratio (PMR). Then, we propose a new, more sophisticated metric, which is arguably much more appropriate for longer sequences of relatively short information units (i.e., sequences of EDUs of long multisentential text). This metric, that we call Blocked Kendall'τ rewards a correctly ordered sub-sequence even if its location is shifted as a single block.

Kendall's τ: a metric of rank correlation, widely used for information ordering evaluation; found to correlate with human judgement (Lapata, 2006). It is computed as follows:

$$\frac{1}{|D|}\sum_{o_i \in D} \tau_{\hat{o}_i} \tag{18}$$

where

$$\tau_{\hat{o}_i} = 1 - 2 * \frac{\text{\# of transpositions}}{\binom{n}{2}} \tag{19}$$

Position Accuracy (POS) computes the average proportion of EDUs that are in their correct absolute position according to the gold ordering.

$$\frac{1}{|D|}\sum_{o_i \in D} \frac{count(\hat{o}_i = o_i)}{length(o_i)} \tag{20}$$

Perfect Match Ratio (PMR) is the strictest metric, measuring the proportion of documents where positions of all EDUs are predicted correctly.

$$\frac{1}{|D|}\sum_{o_i \in D} 1(\hat{o}_i = o_i) \tag{21}$$

The new metric Blocked Kendall's τ: All metrics from previous work simply penalize the distance of an information unit from its correct position. However, ideally, a good metric for information ordering should also capture how well semantically close units are clustered together. This aspect is even more critical when ordering discourse units of long documents - oftentimes, paragraphs or groups of sentences are largely independent in their meaning from other parts of text, so as long as a paragraph's subset of EDUs is ordered correctly, placing it in a different position should not be penalized harshly. As a short example, given the correct ordering o_c [1, 2, 3, 4, 5], all aforementioned metrics would give a low score to the predicted ordering o_p [3, 4, 5, 1, 2] - zero for PMR and POS, and -0.2 for Kendall's τ. Yet, since the blocks [1, 2] and [3, 4, 5] are preserved in o_p, it makes sense to penalize this ordering for only one transposition, and not for twelve like Kendall's τ does. Arguably, these blocks of EDUs are likely to be much more coherent and interpretable than random sequences.

Therefore, we propose a modification for Kendall's τ that treats the correctly ordered blocks as single units. For the example above with $n = 5$, we first merge the correct blocks into single units (indexed by the first EDU in the block), so $[3, 4, 5, 1, 2] \rightarrow [3, 1]$, and compute the Kendall's τ on the resulting reduced sequence:

$$\textit{Block } \tau_{\hat{o}_i} = 1 - 2\frac{\text{\# block transpositions}}{\binom{n}{2}} \tag{22}$$

The number of transpositions can be at least zero (if the sequence is perfectly ordered) and at most $\binom{n}{2}$, if the sequence is in reversed order. Thus, Blocked Kendall's τ has the same range $[-1, 1]$ and is lower bounded by the standard Kendall's τ, with the key advantage of rewarding correct blocks of EDUs. We also note that our proposed measure and the standard Kendall's τ are not metrics in mathematical sense, as they both give a score of 1 to perfectly ordered sequences.

4.2.2 Tree Structure Metrics

UAS and LAS: Unlabelled and labelled attachment scores are the most commonly used measures for evaluation of dependency parsers:

$$\text{UAS} = \frac{\{e | e \in E_G \cap E_P\}}{|V|} \tag{23}$$

$$\text{LAS} = \frac{\{e | l_G(e) = l_P(e), e \in E_G \cap E_P\}}{|V|} \tag{24}$$

where V is the set of EDUs, E_G, E_P are the sets of gold and predicted edges, and $l_G(e)$ is the label of edge e in G.

5 Quantitative and Qualitative Results

Results are presented in Table 1 for the full test set (upper sub-table) and its longer (> 35 EDUs) document subset (lower-sub-table). Remarkably, the DepStructurer (§3.3) dominates other approaches on the new ordering metric (Blocked Kendall's τ),

Approach	New ordering metric	Tree structure		Previous ordering metrics		
	Blocked τ	UAS	LAS	Kendall's τ	POS	PMR
Full test set						
LM Decoding	8.7	×	×	-1.3	8.4	1.86
Unsup Tree Induction (UTI)	10.7	13.1	9.27	0.3	9.3	2.61
Tree Induction (TI+LMD) (100k)	41.7	24.5	22.9	20.0	16.9	7.36
Tree Induction (TI+LMD) (215k)	45.6	**25.9**	**24.3**	21.2	17.5	7.76
Pointer Network (100K)	38.2	×	×	29.4	19.9	6.89
Pointer Network (215K)	40.4	×	×	**31.3**	20.7	7.22
DepStructurer (100K)	48.7	24.3	22.7	28.8	20.0	8.90
DepStructurer (215K)	**52.7**	25.8	24.2	30.7	**21.0**	**9.35**
Long documents only (> 35 EDUs)						
LM Decoding	2.4	×	×	-1.7	2.0	0
Unsup Tree Induction (UTI)	4.5	3.41	2.22	0.0	2.07	0
Tree Induction (TI+LMD) (100k)	21.2	12.4	11.5	5.0	2.8	0
Tree Induction (TI+LMD) (215k)	25.1	**13.6**	**12.7**	5.5	3.0	0
Pointer Network (100K)	21.9	×	×	16.6	4.5	0
Pointer Network (215K)	24.1	×	×	**18.3**	**4.84**	0
DepStructurer (100K)	27.5	12.0	11.1	11.7	3.55	0
DepStructurer (215K)	**31.5**	13.4	12.5	12.3	3.51	0

Table 1: Evaluation results on full test set (15k documents) and its long-document subset (5k documents), with best results per subtable highlighted in **bold**. The entries marked as (×) signify that these metrics cannot be computed for the corresponding models, since they do not induce document tree structures.

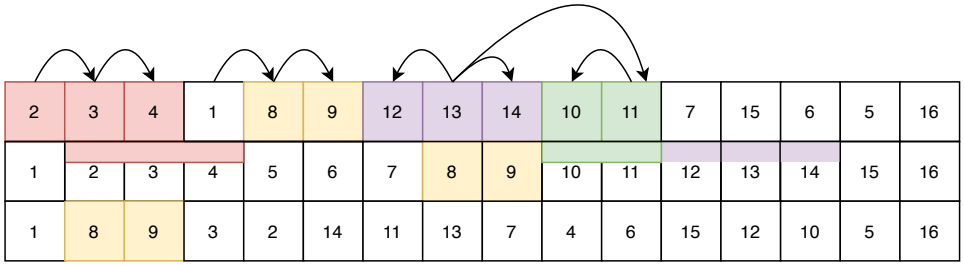


Figure 3: Ordering produced by DepStructurer (top row) and Pointer (bottom row); Gold ordering in middle row.

and surprisingly, our TI+LMD baseline also outperforms the Pointer Network on the full test set and has the performance similar to it on the long-document subset. In contrast, results are mixed for ordering metrics from previous work (last column), which as we have argued in §4.2.1 are however less appropriate for our text structuring task. Interestingly, all trainable models (Pointer Networks §3.2, our DepStructurer §3.3 and TI+LMD §3.4) benefit from more training data (100K → 215K), with equal or even bigger absolute gains for the DepStructurer, especially on the new metric. This validates the quality of the MEGA-DT treebank and suggests that training on larger corpora could increase the performance even further.

Focusing on the performance of tree induction systems, our DepStructurer outperforms the unsupervised model (UTI) by a wide margin and has nearly identical performance with TI+LMD model, indicating that a trainable tree induction model is essential to obtain much more accurate trees.

Lastly, among the unsupervised models, UTI outperforms LM across all metrics. This suggests that even without training, forcing a model to generate a tree structure is by itself a useful inductive bias.

To highlight the strengths and potential weaknesses of our solution and new metric, we analyze the output of the DepStructurer and Pointer models for two medium-length illustrative sample documents with 16 and 14 EDUs respectively (see Figures 3 and 4). In each figure, the top row indicates the ordering output of the DepStructurer, the middle row is the gold (i.e., correct) ordering, and the bottom is the Pointer's output. We color-coded the blocks that each model predicted correctly, with the highlights in the middle gold ordering denoting whether the top or bottom model predicted that block correctly. Additionally, for both exam-

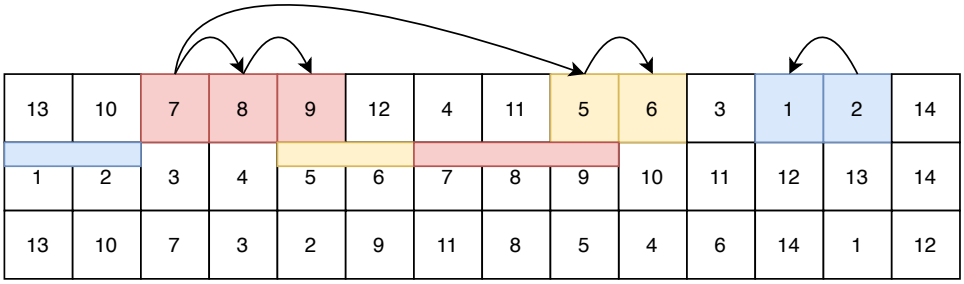


Figure 4: Example illustrating benefits of new metric.

ples, on the top of the DepStructurer ordering, we show the predicted tree dependency edges within the blocks. The main structural benefit of the DepStructurer can be clearly seen in the Figure 3 - the adjacent EDUs tend to form subtrees, the nodes of which the model learns to put close together. In the case of the Pointer model, however, even though it was able to infer a reasonable approximate ordering - with EDUs 1, 3, 2 and EDUs 15, 12, 16 being placed respectively at the beginning/end of the sequence, it failed to arrange them properly in coherent blocks. In Figure 4, we can see an example where the DepStructurer scores in the standard and Blocked Kendall's τ are very different: -36.3 vs. 34.1; while they are the same for the pointer model -9.9. This example clearly illustrates the benefit of our new metric for text structuring. While both models made poor predictions with respect to the distance of each EDU to its correct position, our DepStructurer arguably learned a much more coherent document structure by better grouping related information, which is reflected in the Blocked metric, but is ignored by the standard Kendall's τ.

6 Conclusions and Future work

By proposing the domain-independent task of structuring and ordering a set of EDUs, we aim to stimulate more general and data-driven approaches for text structuring. The solution we have developed for such task combines neural dependency tree induction with pointer networks, which are both trainable on large discourse treebanks. Since existing text ordering metrics are not capturing key aspects of text structuring, we have also proposed a new metric that is arguably much more suitable for the task. In a series of experiments, complemented by qualitative error analysis, we have shown that our solution delivers top performance and represents a promising initial framework for further developments. Fruitful directions for future work include: **(1)** Exploring more recent techniques for tree induction, such as pointer-based and higher-order dependency parsing. **(2)** Integrating our approach into existing long-document data-to-text NLG pipelines such as Puduppully et al. (2019), to explore the benefits of content structuring pre-training for data-to-text applications. **(3)** Verifying the validity of our proposed measure for ordering textual units of long documents (i.e. correlation with human judgement), as well as exploring further metrics for text structuring. **(4)** Extending our approach to fully-labelled RST discourse trees involving nuclearity and relation annotations, which can be obtained from state-of-the-art RST discourse parsers.

References

Stefanos Angelidis and Mirella Lapata. 2018. Multiple instance learning networks for fine-grained sentiment analysis. *Transactions of the Association for Computational Linguistics*, 6:17–31.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 128–135, New York, NY, USA. Association for Computing Machinery.

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.

Jack Edmonds. 1967. Optimum branchings. *JOURNAL OF RESEARCH of the National Bureau of Stan-*

dards - B. Mathematics and Mathematical Physics, 71B:233–240.

Elisa Ferracane, Greg Durrett, Junyi Jessy Li, and Katrin Erk. 2019. Evaluating discourse in structured text representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 646–653, Florence, Italy. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for RST discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136, Los Angeles. Association for Computational Linguistics.

Patrick Huber and Giuseppe Carenini. 2020. Mega rst discourse treebanks with structure and nuclearity from scalable distant sentiment supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2102–2113, Florence, Italy. Association for Computational Linguistics.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

Hamid Karimi and Jiliang Tang. 2019. Learning hierarchical discourse-level structure for fake news detection. 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 3432–3442, Minneapolis, Minnesota. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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

Yang Liu, Ivan Titov, and Mirella Lapata. 2019. Single document summarization as tree induction. 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 1745–1755, Minneapolis, Minnesota. Association for Computational Linguistics.

Lajanugen Logeswaran, Honglak Lee, and Dragomir R. Radev. 2016. Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective MST parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on RST discourse parsing and evaluation. *Computational Linguistics*, 44(2):197–235.

Ani Nenkova and Kathleen McKeown. 2012. *A Survey of Text Summarization Techniques*, pages 43–76. Springer US, Boston, MA.

Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L. Leidner, Dezhao Song, and Frank Schilder. 2016. Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 1121–1124, New York, NY, USA. Association for Computing Machinery.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *AAAI*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, USA.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. 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 3257–3268, Hong Kong, China. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

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

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yang Zhong, Chao Jiang, Wei Xu, and Junyi Jessy Li. 2019. Discourse level factors for sentence deletion in text simplification.

A Hyperparameters and training setup

For the Pointer Model §3.2, similarly to (Cui et al., 2018), the hidden state size in the decoder and transformer EDU encoder is 512, and beam size is 64. Also, as in Cui et al. (2018), the 4-layer Transformer has 8 attention heads. For the Dependency Model §3.3, the edge prediction weights have $d = 512$, and we choose the highest-scoring tree among the top-5 root classifier predictions during inference. The 768-dimensional outputs of ALBERT are transformed with a dense layer to match the dimensionality of EDU encoder. We use AdamW optimizer (Loshchilov and Hutter, 2019) with default weight decay 0.01 and learning rate 0.001, and clip gradient norm at 0.2. The learning rate scheduling rule as in (Vaswani et al., 2017) has 4000 warm-up steps. We apply word dropout (Srivastava et al., 2014) to outputs of ALBERT and of the contextual EDU encoder. We tune dropout value using 15k training subset, selecting among [0, 0.05, 0.15, 0.3], with best values 0.15 for Pointer and 0 for the Dependency Model. All models are trained using early stopping if validation loss did not decrease for three epochs. As only 1% of EDUs have length $>$ 20 word tokens, we clip each EDU's size at 50 ALBERT tokenizer tokens (since it keeps spaces). Batch size for all models is 2 - the highest that could fit into a single GTX 1080 Ti GPU with 11 GB of memory.

B EDU Ordering Examples

See the next page.

Dependency:

2: the lechon special on saturdays tasted 3: like it was premade. 4: the ``crispy`` part of the pork belly was almost gooey. 1: i would actually go for 2 1/2 stars. 8: for $2.00, you get 5 mini half, 9: that are great! 12: being a true filipino, i like my lumpia with a vinegar sauce. 13: if you ask the cashier, for a vinegar sauce, 14: they have a white vinegar, with some onions in it. 10: they give you a sweet and sour sauce on the side, 11: which i do n't think goes well with it. 7: the gem was the shanghai. 15: it was ok, better then then the sweet and sour. 6: the pancit was good, but heavy on the vegetables. 5: the meat itself tasted good, although better with some kikkoman shoyu. 16: overall, a descent find.

Gold:

1: i would actually go for 2 1/2 stars. 2: the lechon special on saturdays tasted 3: like it was premade . 4: the ``crispy`` part of the pork belly was almost gooey. 5: the meat itself tasted good , although better with some kikkoman shoyu. 6: the pancit was good, but heavy on the vegetables. 7: the gem was the shanghai. 8: for $2.00 , you get 5 mini half, 9: that are great! 10: they give you a sweet and sour sauce on the side, 11: which i don't think goes well with it. 12: being a true filipino , i like my lumpia with a vinegar sauce. 13: if you ask the cashier, for a vinegar sauce, 14: they have a white vinegar, with some onions in it. 15: it was ok , better then then the sweet and sour. 16: overall, a descent find.

Pointer:

1: i would actually go for 2 1/2 stars. 8: for $2.00 , you get 5 mini half , 9: that are great! 3: like it was premade. 2: the lechon special on saturdays tasted 14: they have a white vinegar, with some onions in it. 11: which i don't think goes well with it. 13: if you ask the cashier, for a vinegar sauce, 7: the gem was the shanghai. 4: the ``crispy`` part of the pork belly was almost gooey. 6: the pancit was good, but heavy on the vegetables. 15: it was ok, better then then the sweet and sour. 12: being a true filipino, i like my lumpia with a vinegar sauce. 10: they give you a sweet and sour sauce on the side, 5: the meat itself tasted good, although better with some kikkoman shoyu. 16: overall, a descent find.

Figure 5: Example from Figure 3 in the paper

Dependency:

13: i simply love their gyros! 10: it is set up like sauce 7: the food is cooked fresh 8: for you 9: so there will be a short wait. 12: and they bring the food to you. 4: the interior is cutesy and bright 11: where you order at the cashier area 5: while upbeat music is playing. 6: they have a small outdoor seating area and some booths and tables inside. 3: it's tucked away in a strip plaza shockingly! 1: i hope more people are frequenting this place 2: since i was last there. 14: it's relatively quick but always fresh and inexpensive!

Gold:

1: i hope more people are frequenting this place 2: since i was last there. 3: it's tucked away in a strip plaza shockingly! 4: the interior is cutesy and bright 5: while upbeat music is playing. 6: they have a small outdoor seating area and some booths and tables inside. 7: the food is cooked fresh 8: for you 9: so there will be a short wait. 10: it is set up like sauce 11: where you order at the cashier area, 12: and they bring the food to you. 13: i simply love their gyros! 14: it's relatively quick but always fresh and inexpensive!

Pointer:

13: i simply love their gyros! 10: it is set up like sauce 7: the food is cooked fresh 3: it's tucked away in a strip plaza shockingly! 2: since i was last there. 9: so there will be a short wait. 11: where you order at the cashier area 8: for you, 5: while upbeat music is playing. 4: the interior is cutesy and bright 6: they have a small outdoor seating area and some booths and tables inside. 14: it's relatively quick but always fresh and inexpensive! 1: i hope more people are frequenting this place 12: and they bring the food to you.

Figure 6: Example from Figure 4 in the paper